

Spis treści

Wstęp	9
Część I Podstawy języka UML 2.0	13
Rozdział 1. Język UML — rozwój, struktura, pojęcia	15
Znaczenie obiektowości w modelowaniu systemów informatycznych	15
Geneza i ewolucja języka UML	17
Diagramy UML 2.0	21
Perspektywy w opisie architektury systemu	25
Mechanizmy rozszerzalności	27
Stereotyp	27
Ograniczenie	28
Metka	28
Podstawowe pojęcia	29
Pytania i zadania	30
Rozdział 2. Diagram przypadków użycia	33
Znaczenie diagramów przypadków użycia	33
Podstawowe kategorie pojęciowe oraz notacja graficzna	34
Przypadek użycia	34
Aktor	35
Związek	36
Zaawansowane składniki diagramu	39
Rozbudowa DPU poprzez różnicowanie związków	40
Zależności zawierania	40
Zależności rozszerzania	41
Uogólnienia	43
Rodzaje aktorów	45
Liczebność	46
Nawigacja	47
Realizacja	49
Przypadki użycia typu CRUD	49
Stosowanie nazw ścieżkowych	50
Diagram kontekstowy	50
Dokumentacja przypadków użycia	51
Proces tworzenia diagramu przypadków użycia	54
Studium diagramu przypadków użycia	55
Podstawowe pojęcia	56
Pytania i zadania	57

Rozdział 3. Diagram klas	61
Znaczenie diagramów klas	61
Podstawowe kategorie pojęciowe oraz notacja graficzna	61
Asocjacja	64
Nazwy asocjacji	65
Role	66
Nawigacja	66
Liczebność	66
Agregacja	67
Zaawansowane składniki diagramu	70
Rodzaje diagramów klas	71
Zobowiązania	71
Widoczność	72
Atrybuty i operacje statyczne	73
Nazwy klas, atrybutów i operacji	73
Notacja atrybutów i składnia operacji	74
Klasy asocjacyjne	77
Asocjacje zwrotne i wielokrotne	79
Kwalifikacja	79
Uogólnienia, klasy abstrakcyjne oraz konkretne	80
Zależność	83
Realizacja	83
Diagram obiektów	85
Proces tworzenia diagramu klas	86
Studium diagramu klas	87
Podstawowe pojęcia	89
Pytania i zadania	91
Rozdział 4. Diagram czynności	95
Znaczenie diagramów czynności	95
Podstawowe kategorie pojęciowe oraz notacja graficzna	96
Czynności a akcje	96
Zaawansowane składniki diagramu	99
Przepływy sterowania	99
Znacznik sterowania	99
Przepływy decyzyjne	100
Decyzja	100
Łącznik	104
Złączenie	104
Przepływy współbieżne	106
Akcje	107
Przepływy danych	111
Przełączniki danych	112
Parametr czynności	116
Wagi	117
Sygnały	118
Bufor centralny	118
Składnica danych	120
Partycje diagramów czynności	121
Obszar rozszerzenia	127
Obszar przerwania	130
Manipulator wyjątków	131

Proces tworzenia diagramu czynności	132
Studium diagramu czynności	133
Podstawowe pojęcia	135
Pytania i zadania	136
Rozdział 5. Diagram maszyny stanowej	143
Znaczenie diagramów maszyny stanowej	143
Podstawowe kategorie pojęciowe oraz notacja graficzna	144
Zaawansowane składniki diagramu	146
Sekcje symbolu graficznego stanu	146
Klasyfikacja stanów	148
Obszary współbieżne	150
Pseudostany	152
Rodzaje przejść	157
Protokołowe maszyny stanowe	158
Maszyny stanowe zachowania	161
Zdarzenia	163
Proces tworzenia diagramu maszyny stanowej	164
Studium diagramu maszyny stanowej	165
Podstawowe pojęcia	168
Pytania i zadania	169
Rozdział 6. Diagramy interakcji	173
Interakcje	173
Diagramy interakcji	173
Składnia komunikatu	174
Podstawowe pojęcia	177
Pytania i zadania	177
Rozdział 7. Diagram sekwencji	179
Znaczenie diagramów sekwencji	179
Podstawowe kategorie pojęciowe oraz notacja graficzna	180
Rodzaje diagramów sekwencji	180
Klasyfikator, komunikat i linia życia	181
Rodzaje klasyfikatorów	183
Ośrodek sterowania	184
Zaawansowane składniki diagramu	185
Rodzaje komunikatów	187
Tworzenie i niszczenie obiektów	190
Warunki	191
Samowywołanie	192
Iteracja	192
Rozgałęzienie	194
Fragmenty wyodrębnione i operatory interakcji	196
Przywoływane wystąpienia interakcji	213
Bramy	216
Proces tworzenia diagramu sekwencji	218
Studium diagramu sekwencji	219
Podstawowe pojęcia	223
Pytania i zadania	224

Rozdział 8. Diagram komunikacji	231
Znaczenie diagramów komunikacji	231
Podstawowe kategorie pojęciowe oraz notacja graficzna	232
Numerowanie komunikatów	233
Zasady wprowadzania komunikatów do diagramu	234
Zaawansowane składniki diagramu	235
Izomorfizm	235
Zagnieżdżenie	235
Poprzednik	239
Współbieżność	239
Obiekty wielokrotne	240
Klasy aktywne	241
Inne kategorie zaawansowane	242
Proces tworzenia diagramu komunikacji	244
Studium diagramu komunikacji	245
Podstawowe pojęcia	247
Pytania i zadania	247
Rozdział 9. Diagram harmonogramowania	251
Znaczenie diagramów harmonogramowania	251
Podstawowe kategorie pojęciowe oraz notacja graficzna	252
Zaawansowane składniki diagramu	253
Zdarzenia i ograniczenia czasowe	253
Alternatywne sposoby prezentacji stanów	254
Harmonizacja linii zmiany stanów	254
Przesyłanie komunikatów	256
Diagramy sekwencji a harmonogramowanie	258
Proces tworzenia diagramu harmonogramowania	259
Podstawowe pojęcia	260
Pytania i zadania	260
Rozdział 10. Diagram sterowania interakcją	263
Znaczenie diagramów sterowania interakcją	263
Podstawowe kategorie pojęciowe oraz notacja graficzna	264
Zaawansowane składniki diagramu	266
Proces tworzenia diagramu sterowania interakcją	268
Studium diagramu sterowania interakcją	268
Podstawowe pojęcia	270
Pytania i zadania	271
Rozdział 11. Diagramy wdrożeniowe	273
Znaczenie diagramów wdrożeniowych	273
Diagram komponentów	273
Interfejsy	277
Specyfikacja komponentów	278
Porty	280
Konektory	280
Implementacyjny diagram komponentów	281
Diagram rozlokowania	282
Węzły	284
Ścieżki komunikowania	285
Osadzone artefakty i komponenty	286
Manifestowanie	287
Specyfikacja rozlokowania	289
Diagramy rozlokowania na poziomie fizycznym	290

Proces tworzenia diagramów wdrożeniowych	291
Podstawowe pojęcia	292
Pytania i zadania	293
Rozdział 12. Diagram struktur połączonych	299
Znaczenie diagramów struktur połączonych	299
Podstawowe kategorie pojęciowe oraz notacja graficzna	300
Proces tworzenia diagramu struktur połączonych	302
Podstawowe pojęcia	303
Pytania i zadania	303
Rozdział 13. Diagram pakietów	305
Znaczenie diagramów pakietów	305
Podstawowe kategorie pojęciowe oraz notacja graficzna	305
Pakiet	306
Zależność	307
Zagnieżdżenie pakietów	308
Zaawansowane składniki diagramu	309
Stereotypowanie pakietów	310
Stereotypowanie zależności	311
Proces tworzenia diagramu pakietów	312
Studium diagramu pakietów	312
Podstawowe pojęcia	314
Pytania i zadania	314
Część 2. Metodyki, modele i narzędzia wspomagające UML 2	317
Rozdział 14. Metodyka RUP	319
Znaczenie iteracyjno-przyrostowego procesu projektowania systemów	319
Struktura RUP	321
Dyscypliny	323
Fazy	325
Podstawowe pojęcia	327
Pytania i zadania	328
Rozdział 15. Modelowanie biznesowe	329
Znaczenie modelowania biznesowego	329
Podstawowe kategorie pojęciowe oraz notacja graficzna	330
Studium modelu biznesowego	331
Podstawowe pojęcia	341
Pytania i zadania	341
Rozdział 16. Modelowanie analityczne	343
Znaczenie modelowania analitycznego	343
Podstawowe kategorie pojęciowe oraz notacja graficzna	344
Proces tworzenia modelu analitycznego	346
Studium modelu analitycznego	349
Podstawowe pojęcia	350
Pytania i zadania	350
Rozdział 17. Komputerowe wspomaganie modelowania systemu	353
Pakiety CASE wspomagające UML i RUP	353
Zakres wspomagania diagramów UML	355
Generowanie szkieletowego kodu źródłowego	356
Inżynieria zwrotna	357
Obsługiwane platformy	358

Sparx Systems Enterprise Architect	359
Narzędzia CASE firmy IBM pod marką Rational	362
IBM Software Development Platform	362
Zastosowanie IBM Rational Software Modeler	363
Poseidon for UML	366
Platforma Eclipse	366
UML i RUP w zasobach Internetu	368
Podstawowe pojęcia	369
Pytania i zadania	369

Dodatki	371
Dodatek A Definicje i pojęcia	373
Dodatek B Notacja graficzna	381
Dodatek C Diagramy	389
Dodatek D UML 2.0 w liczbach	399
Dodatek E Słownik polsko-angielski	401
Dodatek F Słownik angielsko-polski	409
Dodatek G Spis rysunków	417
Dodatek H Literatura	423
Dodatek I Contents	429
Skorowidz	435

Contents

Preface	9
Part I Background of UML	13
Chapter 1. Unified Modeling Language — development, structure, terms	15
Significance of object-oriented approach in information systems modeling	15
Evolution of UML	17
UML 2.0 diagram types	21
Views of system architecture	25
Extension mechanisms	27
Stereotype	27
Constraint	28
Tagged value	28
Index of basic concepts	29
Questions and exercises	30
Chapter 2. Use case diagram	33
Relevance of use case diagrams	33
Elementary modeling categories and notation	34
Use case	34
Actor	35
Relationship	36
Advanced elements of diagram	39
Structuring use case model	40
«include» dependency	40
«extend» dependency	41
Generalizations	43
Types of actors	45
Multiplicity	46
Navigability	47
Realization	49
CRUD use cases	49
Extended naming scheme	50
Context diagram	50
Documenting use case scenarios	51
Use case diagram development process	54
Advanced study of use case diagram	55
Index of basic concepts	56
Questions and exercises	57

Chapter 3. Class diagram	61
Relevance of class diagrams	61
Elementary modeling categories and notation	61
Association	64
Naming associations	65
Roles	66
Navigability	66
Multiplicity	66
Aggregation	67
Advanced elements of diagram	70
Types of class diagrams	71
Responsibilities	71
Visibility	72
Static attributes and associations	73
Naming classes, attributes and operations	73
Syntax of attributes and operations	74
Association classes	77
Multiple and reflexive associations	79
Qualification	79
Generalizations, abstract and concrete classes	80
Dependency	83
Realization	83
Object diagram	85
Class diagram development process	86
Advanced study of class diagram	87
Index of basic concepts	89
Questions and exercises	91
Chapter 4. Activity diagram	95
Relevance of activity diagrams	95
Elementary modeling categories and notation	96
Activities and actions	96
Advanced elements of diagram	99
Control flows	99
Token	99
Alternative flows	100
Decision	100
Activity edge connector	104
Merge node	104
Concurrent flows	106
Actions	107
Data flows	111
Pin	112
Activity parameter node	116
Weight	117
Signal	118
Central buffer	118
Data store	120
Activity partition	121
Expansion region	127
Interruptible activity region	130
Exception handler	131

Activity diagram development process	132
Advanced study of activity diagram	133
Index of basic concepts	135
Questions and exercises	136
Chapter 5. State machine diagram	143
Relevance of state machine diagrams	143
Elementary modeling categories and notation	144
Advanced elements of diagram	146
State compartments	146
Classification of states	148
Orthogonal regions	150
Pseudostates	152
Types of transitions	157
Protocol state machine	158
Behavioral state machine	161
Events	163
State machine diagram development process	164
Advanced study of state machine diagram	165
Index of basic concepts	168
Questions and exercises	169
Chapter 6. Interaction diagrams	173
Interactions	173
Interaction diagrams	173
Syntax of message	174
Index of basic concepts	177
Questions and exercises	177
Chapter 7. Sequence diagram	179
Relevance of sequence diagrams	179
Elementary modeling categories and notation	180
Types of sequence diagrams	180
Classifier, message and lifeline	181
Types of classifiers	183
Execution specification	184
Advanced elements of diagram	185
Types of messages	187
Object creation and destruction	190
Guard condition	191
Message to self	192
Iteration	192
Branching	194
Interaction fragments with interaction operators	196
Interaction occurrences	213
Gates	216
Sequence diagram development process	218
Advanced study of sequence diagram	219
Index of basic concepts	223
Questions and exercises	224

Chapter 8. Communication diagram	231
Relevance of communication diagrams	231
Elementary modeling categories and notation	232
Message numbering	233
Message insertion rules	234
Advanced elements of diagram	235
Isomorphism	235
Message nesting	235
Predecessor	239
Concurrency	239
Multiple objects	240
Active classes	241
Other advanced elements	242
Communication diagram development process	244
Advanced study of communication diagram	245
Index of basic concepts	247
Questions and exercises	247
Chapter 9. Timing diagram	251
Relevance of timing diagrams	251
Elementary modeling categories and notation	252
Advanced elements of diagram	253
Events and time constraints	253
Alternative ways of presenting states	254
Harmonizing timelines	254
Sending messages	256
Timing on sequence diagrams	258
Timing diagram development process	259
Index of basic concepts	260
Questions and exercises	260
Chapter 10. Interaction overview diagram	263
Relevance of interaction overview diagrams	263
Elementary modeling categories and notation	264
Advanced elements of diagram	266
Interaction overview diagram development process	268
Advanced study of interaction overview diagram	268
Index of basic concepts	270
Questions and exercises	271
Chapter 11. Implementation diagrams	273
Relevance of implementation diagrams	273
Component diagram	273
Interfaces	277
Component specification	278
Ports	280
Connectors	280
Generic component diagram	281
Deployment diagram	282
Nodes	284
Communication paths	285
Embedded artifacts and components	286
Manifesting	287
Deployment specification	289
Physical deployment diagrams	290

Implementation diagrams development process	291
Index of basic concepts	292
Questions and exercises	293
Chapter 12. Composite structure diagram	299
Relevance of composite structure diagrams	299
Elementary modeling categories and notation	300
Composite structure diagram development process	302
Index of basic concepts	303
Questions and exercises	303
Chapter 13. Package diagram	305
Relevance of package diagrams	305
Elementary modeling categories and notation	305
Package	306
Dependency	307
Package nesting	308
Advanced elements of diagram	309
Stereotyping packages	310
Stereotyping dependencies	311
Package diagram development process	312
Advanced study of package diagram	312
Index of basic concepts	314
Questions and exercises	314
Part II System development with UML and RUP	317
Chapter 14. Rational Unified Process	319
Relevance of iterative-incremental development process	319
Structure of Rational Unified Process	321
Disciplines	323
Phases	325
Index of basic concepts	327
Questions and exercises	328
Chapter 15. Business modeling	329
Relevance of business modeling	329
Elementary modeling categories and notation	330
Advanced study of business model	331
Index of basic concepts	341
Questions and exercises	341
Chapter 16. Robustness diagram	343
Relevance of robustness diagrams	343
Elementary modeling categories and notation	344
Robustness diagram development process	346
Advanced study of robustness diagram	349
Index of basic concepts	350
Questions and exercises	350
Chapter 17. Computer-aided software engineering	353
CASE tools supporting UML and RUP	353
The scope of UML diagrams support	355
Source code generation	356
Reverse engineering	357
Supported platforms	358

Sparx Systems Enterprise Architect	359
IBM Rational CASE tools	362
IBM Software Development Platform	362
IBM Rational Software Modeler in practice	363
Poseidon for UML	366
Eclipse platform	366
UML and RUP in the Internet	368
Index of basic concepts	369
Questions and exercises	369
Appendices	371
Dodatek A Glossary	373
Dodatek B Notation	381
Dodatek C Diagrams	389
Dodatek D UML in numbers	399
Dodatek E Polish-English dictionary	401
Dodatek F English-Polish dictionary	409
Dodatek G List of figures	417
Dodatek H References	423
Dodatek I Contents	429
Index	435

Wstęp

W pierwszej dekadzie XXI wieku dziedzina **modelowania systemów informatycznych** została zdominowana przez paradygmat modelowania obiektowego. Podejście to jest praktycznie wdrażane w postaci języka UML (*Unified Modeling Language*) oraz metodykę RUP (*Rational Unified Process*). Język UML jest graficznym językiem wizualizacji, specyfikowania, tworzenia i dokumentowania składników systemów informatycznych. Budzi on wielkie zainteresowanie środowisk akademickich i zawodowych, w tym korporacji informatycznych oraz instytucji standaryzacyjnych. Zainteresowanie to jest stymulowane rosnącą liczbą aplikacji pracujących w czasie rzeczywistym — telekomunikacyjnych, internetowych, bankowych, edukacyjnych i innych.

Język UML jest nieustannie, intensywnie rozwijany, co również było poważnym wyzwaniem autorów niniejszej książki. Pierwszą wersję tego języka przedstawiono w 1995 roku. Była ona nie tylko konsekwencją połączenia **metodyk** G. Boocha, I. Jacobsona oraz J. Rumbaugh, lecz przede wszystkim istotnym krokiem w rozwoju obiektowej analizy i projektowania. Szczególnie wiele zmian w zakresie standardu wprowadzono w 2003 roku, czego rezultatem jest obowiązująca obecnie wersja UML 2.0. Największy wkład w prace standaryzacyjne wnosi organizacja OMG (*Object Management Group*). Zmiany te są odpowiedzią na problemy występujące w aplikacjach praktycznych oraz pracach metodologicznych prowadzonych w ośrodkach akademickich na całym świecie. UML pozostaje stale otwartym wyzwaniem dla tych środowisk. Wnoszą one swój wkład poprzez bieżące zgłaszanie uwag i propozycji do OMG.

Ponieważ UML jest przedmiotem znacznego zainteresowania, powstało i wciąż powstaje wiele elektronicznych i drukowanych opracowań na ten temat. Tendencja ta wynika z potrzeby doskonalenia umiejętności zawodowych przez projektantów i analityków systemów, a także modernizacji programów dydaktycznych na studiach informatycznych na całym świecie. Kursy języka UML są składnikiem programów studiów większości uznanych uczelni. Autorzy niniejszej książki postanowili wypełnić lukę w postaci braku pełnego i przystępnego wyjaśnienia zasad efektywnego użytkowania języka UML w wersji 2.0. Czynnikiem ten był podstawowym motywem i inspiracją opracowania niniejszej książki. Główne jej **cechy**, odróżniające ją od innych, to:

- ♦ praktyczne zaprezentowanie i wyjaśnienie, poprzez liczne autorskie przykłady, zasad użytkowania języka UML 2.0;
- ♦ precyzyjne zdefiniowanie bazy terminologicznej i metodycznej standardu UML i metodyki RUP;
- ♦ zweryfikowanie, obok diagramów UML 2.0, słabo opracowanych w literaturze diagramów modelowania biznesowego i modelowania analitycznego;
- ♦ dydaktyczne opracowanie obszernego kanonu wiedzy o języku UML dla sprawnego posługiwania się nim w praktyce;
- ♦ wszechstronna egzemplifikacja modelowania systemów z zakresu biznesu i administracji.

Zagadnienia aktualnej prezentacji języka UML 2.0 oraz związanych z nim diagramów modelowania biznesowego i analitycznego są podejmowane po raz pierwszy w polskiej literaturze przedmiotu.

Książkę podzielono na dwie części:

- ♦ podstawy języka UML 2 (rozdziały 1. – 13.);
- ♦ metodyki, modele i narzędzia wspomagające UML 2 (rozdziały 14. – 17.).

Autorzy starali się potraktować tematykę w sposób uniwersalny, wychodząc poza oficjalny standard UML. Konsekwencją takiego podejścia jest wprowadzenie koncepcji wynikających ze zbliżonych tematycznie podejść i narzędzi oraz „dobrych praktyk” tworzenia oprogramowania.

Dla wielu pojęć związanych z językiem UML autorzy zaproponowali oryginalne nazwy polskie. Definicje kluczowych kategorii pojęciowych tego języka wyróżniono poprzez umieszczenie w ramkach. Niektóre z tych definicji mają autorski charakter, inne opierają się bezpośrednio na standardach OMG i aktualnych propozycjach twórców języka UML — G. Boocha, I. Jacobsona oraz J. Rumbaugh — razem tworząc spójną, a zarazem przystępną podstawę terminologiczną. Poszczególne źródła definicji oznaczono odpowiednio ikonami:



— definicje autorskie;



— terminologia oficjalnie stosowana przez Object Management Group;



— definicje zaproponowane przez twórców języka UML.

Część pierwsza książki zawiera prezentację i weryfikację treści dotyczącej poszczególnych **diagramów UML**. Kolejne rozdziały przedstawiają zasady modelowania systemów na podstawie trzynastu rodzajów diagramów stanowiących aktualny standard języka UML 2.0. Są to:

- ◆ diagramy przypadków użycia,
- ◆ diagramy klas,
- ◆ diagramy obiektów,
- ◆ diagramy czynności,
- ◆ diagramy maszyny stanowej,
- ◆ diagramy sterowania interakcją,
- ◆ diagramy sekwencji,
- ◆ diagramy komunikacji,
- ◆ diagramy harmonogramowania,
- ◆ diagramy struktur połączonych,
- ◆ diagramy komponentów,
- ◆ diagramy rozlokowania,
- ◆ diagramy pakietów.

Diagramy te, z nielicznymi wyjątkami wynikającymi z charakterystyki danego diagramu, prezentowane są w każdym rozdziale w **standardowym układzie**:

- ◆ znaczenie danego diagramu,
- ◆ podstawowe kategorie pojęciowe oraz notacja graficzna,
- ◆ zaawansowane składniki diagramu,
- ◆ proces tworzenia diagramu,
- ◆ studium diagramu,
- ◆ podstawowe pojęcia,
- ◆ pytania i zadania.

Ostatnie dwie części każdego rozdziału mają charakter **dydaktyczny**. Ich celem jest zwrócić uwagę czytelnika na najistotniejsze informacje zaprezentowane w danym rozdziale oraz umożliwić sprawdzenie wiedzy we własnym zakresie. Ćwiczenia niejednokrotnie wykraczają poza zakres pojedynczego rozdziału, stymulując do wypracowywania bardziej kompleksowego spojrzenia na problem oraz poszukiwania wiedzy w innych źródłach.

Wiodącą tematyką drugiej części książki jest charakterystyka **metodologicznych podstaw** modelowania systemów na podstawie metodyki RUP. Część drugą kończy analiza porównawcza i prezentacja funkcjonalności uznanych narzędzi CASE, wspomagających użytkowanie diagramów UML.

Książkę podsumowuje obszerny materiał pomocniczy w postaci dodatków, literatura, spisy oraz skorowidz. Dodatki obejmują:

- ◆ zestawienie definicji i pojęć,
- ◆ podsumowanie notacji graficznej poszczególnych kategorii pojęciowych,
- ◆ podsumowanie notacji diagramów,
- ◆ UML w liczbach,
- ◆ słownik polsko-angielski,
- ◆ słownik angielsko-polski.

Materiał pomocniczy obejmuje ponadto załączony na płycie CD obszerny zestaw darmowych narzędzi CASE wspierających język UML oraz plakat, ujmujący syntetycznie notację wersji 2.0 tego języka. Autorzy stworzyli możliwość studiowania UML 2.0 w trybie e-learningu na platformie edukacyjnej Uniwersytetu Gdańskiego (<http://peug.univ.gda.pl>).

Podstawowymi **źródłami** służącymi do przygotowania niniejszej publikacji były:

- ◆ dokumentacja standardu UML 2.0 opracowana przez OMG,
- ◆ hipertekstowa baza wiedzy RUP,
- ◆ publikacje J. Rumbaugh, G. Boocha oraz I. Jacobsona (*Three Amigos*),
- ◆ aktualna literatura przedmiotu,
- ◆ źródła internetowe licznie wymienione w rozdziale 17.,
- ◆ obszerne i dogłębne prace oraz doświadczenia własne związane z aplikacjami biznesowymi języka UML i jego diagramów.

Autorzy wyrażają nadzieję, że niniejsza książka przyczyni się do lepszego zrozumienia i opanowania języka UML przez absolwentów studiów informatycznych i ekonomicznych w Polsce oraz stanie się ważnym materiałem w praktyce zawodowej twórców systemów informatycznych.

Część I

Podstawy języka UML 2.0

Rozdział 1.

Język UML — rozwój, struktura, pojęcia

Znaczenie obiektowości w modelowaniu systemów informatycznych

Modelowanie systemów informatycznych to nieustające i trudne wyzwanie dla twórców systemów, począwszy od pierwszych biznesowych zastosowań systemów z lat 50. Pierwsze, stosunkowo proste podejścia do **modelowania systemów** zmieniały się i ewoluowały w zaawansowane teoretycznie paradygmaty tworzenia systemów informatycznych (TSI). Szczególnie dwa podejścia zdominowały analizę oraz projektowanie systemów — podejście strukturalne oraz obiektowe. Zaskakuje mnogość i różnorodność związanych z nimi metodyk i notacji. Różnorodność ta doprowadziła w latach 80. do swoistych „wojen metodologicznych”, w ramach których różne grupy twórców współzawodniczyły z sobą, akcentując odrębność własnych koncepcji. Ósma dekada XX wieku to okres szczególnej dominacji podejścia strukturalnego, które zaowocowało wieloma metodykami wypracowanymi przez środowiska naukowe i biznesowe. Mimo istotnych różnic pomiędzy nimi, istniały punkty, kategorie i mechanizmy wspólne, będące podstawą dalszego rozwoju. Zaliczyć do nich należy pojęcie i składniki metodyki tworzenia systemów informatycznych, cykl życia systemu, prototypowanie, diagramy związków encji, modele relacyjnych baz danych czy też narzędzia CASE (*Computer-Aided Software Engineering*).

Naturalnie różnorodność ta wprowadzała spory wśród specjalistów z dziedziny inżynierii oprogramowania co do wartości i użyteczności oferowanych, odmiennych podejść. Również pewne obietnice podejścia strukturalnego — chociażby automatyzacja procesu tworzenia systemów informatycznych od analizy do generowania kodu — nie zostały spełnione. W celu uporządkowania wiedzy i procedur modelowania systemów informatycznych podjęto dwie znaczące inicjatywy **unifikacyjne**:

- ♦ podjętą przez IFIP (*International Federation of Information Processing*) w postaci grupy roboczej CRIS (*Comparative Review of Information Systems Methodologies*),
- ♦ podjętą przez Unię Europejską w ramach zespołu reprezentującego autorów metodyk używanych w różnych krajach członkowskich pod nazwą *EuroMethod*.

Wymienione inicjatywy unifikacji procesu projektowania i wdrażania systemów informatycznych nie zyskały aprobaty środowiska informatyków. Niemniej ogólna, zbiorowa wiedza na temat różnorodnych metodyk strukturalnych i związane z nimi narzędzia CASE, stały się w latach 80. i później bardzo przydatnym, rutynowo wykorzystywanym, klasycznym narzędziem analizy oraz projektowania systemów informatycznych.

Problemy związane z podejściem strukturalnym sprzyjały zainteresowaniu oraz rozwojowi prac naukowo-badawczych i wdrożeniowych, opierających się na innych podstawach teoretycznych. Wymienić tu należy przede wszystkim społeczne i obiektowe modelowanie systemów. Od początku lat 90. szczególnie modele obiektowe znalazły się w centrum zainteresowania twórców i użytkowników systemów. Wzrost znaczenia **obiektowości** stymulowany był przez:

- ♦ wyzwania postępu technologicznego w informatyce, polegające na coraz bardziej powszechnym użytkowaniu systemów czasu rzeczywistego i systemów wbudowanych;
- ♦ różnorodność i powszechność aplikacji internetowych w gospodarce opartej na wiedzy, czyli w takich dziedzinach jak e-business, e-health, e-government, e-learning;
- ♦ multimedialny charakter aplikacji, wykorzystujących w coraz większym stopniu dźwięk, głos, grafikę, film;
- ♦ globalizację gospodarki, a zatem integrację systemów informatycznych — w telekomunikacji, bankowości, edukacji, turystyce, transporcie;
- ♦ powszechność i dostępność aplikacji, zwłaszcza internetowych, dla potrzeb społeczeństwa informacyjnego.

Wyzwaniom tym w najwyższym stopniu sprostało podejście obiektowe. Rozwijane od lat 50. XX wieku, w latach 90. znalazło się w centrum zainteresowania badaczy i praktyków. Pojęcie obiektowości początkowo wykorzystywane było w obiektowych językach programowania, takich jak *SIMULA* czy *SmallTalk*. Podejście obiektowe ma obecnie największe znaczenie w następujących **zastosowaniach**:

- ♦ metodykach tworzenia oprogramowania (przede wszystkim *Rational Unified Process*);
- ♦ graficznych językach ogólnego przeznaczenia (*UML*);
- ♦ obiektowych językach programowania (*JAVA*, platforma *.NET*);
- ♦ bazach danych (np. *ObjectStore*).

Każdy model obiektowy opiera się na pewnych elementarnych pojęciach i kategoriach, które pod względem metodologicznym stanowią tzw. podstawowy model obiektowy. Model ten jest szczegółowo przedstawiony w wielu opracowaniach naukowych na temat podejścia naukowego. **Główne pojęcia** podstawowego modelu obiektowego — obiekt, klasę, komunikat, hermetyzację, polimorfizm i dziedziczenie — charakteryzuje tabela 1.1.

Tabela 1.1. Podstawowe pojęcia obiektowości

Pojęcie	Interpretacja
obiekt (ang. <i>object</i>)	Każdy byt — pojęcie lub rzecz — mający znaczenie w kontekście rozwiązywania problemu w danej dziedzinie przedmiotowej.
klasa (ang. <i>class</i>)	Uogólnienie zbioru obiektów, które mają takie same atrybuty, operacje, związki i znaczenie.
komunikat (ang. <i>message</i>)	Specyfikacja wymiany informacji między obiektami, zawierająca zlecenia wykonania określonej operacji.
hermetyzacja (ang. <i>encapsulation</i>)	Różnicowanie dostępu do obiektu poprzez ujawnienie otoczeniu tylko tych informacji o jego atrybutach lub operacjach, które są niezbędne do efektywnego odwoływania się do obiektu w systemie za pośrednictwem komunikatów.
polimorfizm (ang. <i>polymorphism</i>)	Możliwość nadawania tej samej nazwy różnym atrybutom i operacjom oraz wykonywania różnych procedur i akcji poprzez operacje o tych samych nazwach; pozwala na redukcję liczby nazw atrybutów i operacji.
dziedziczenie (ang. <i>inheritance</i>)	Przyporządkowanie atrybutów i operacji klasom obiektów na podstawie hierarchicznej zależności między nimi. Możliwe jest wielokrotne dziedziczenie, co oznacza, że dana klasa dziedziczy atrybuty i operacje z dowolnej liczby klas nadrzędnych.

Obiektowość stanowi podstawę teoretyczną języka UML (*Unified Modeling Language*). W dalszych punktach niniejszego rozdziału przedstawiono ogólne zagadnienia związane z tym językiem, a więc:

- ♦ jego genezę i ewolucję,
- ♦ klasyfikację oraz krótką charakterystykę diagramów UML,
- ♦ perspektywy architektury systemu,
- ♦ mechanizmy rozszerzalności.

Geneza i ewolucja języka UML

W przypadku modelowania obiektowego również nie udało się uniknąć metodycznego rozproszenia. Szczególną dynamiką charakteryzowały się lata 1989 – 1994, kiedy to liczba identyfikowalnych rozwiązań wzrosła z kilku do ponad 50. Jednak w odróżnieniu od metodyk strukturalnych, spory nad kształtem metodyk obiektowych zakończyły się powszechnie uznawanym konsensusem. Autorzy dominujących w omawianym okresie rozwiązań, które ewoluowały wówczas niezależnie od siebie, zaproponowali **ujednolicony język modelowania systemów informatycznych** — UML. Nie należy

przy tym utożsamiać języka UML z pełną metodyką tworzenia systemów informatycznych, która jest znacznie szerszą kategorią (por. rozdział 14.). Podstawowe trzy autorskie źródła języka UML przedstawia tabela 1.2.

Tabela 1.2. Wkład dominujących podejść w standard UML

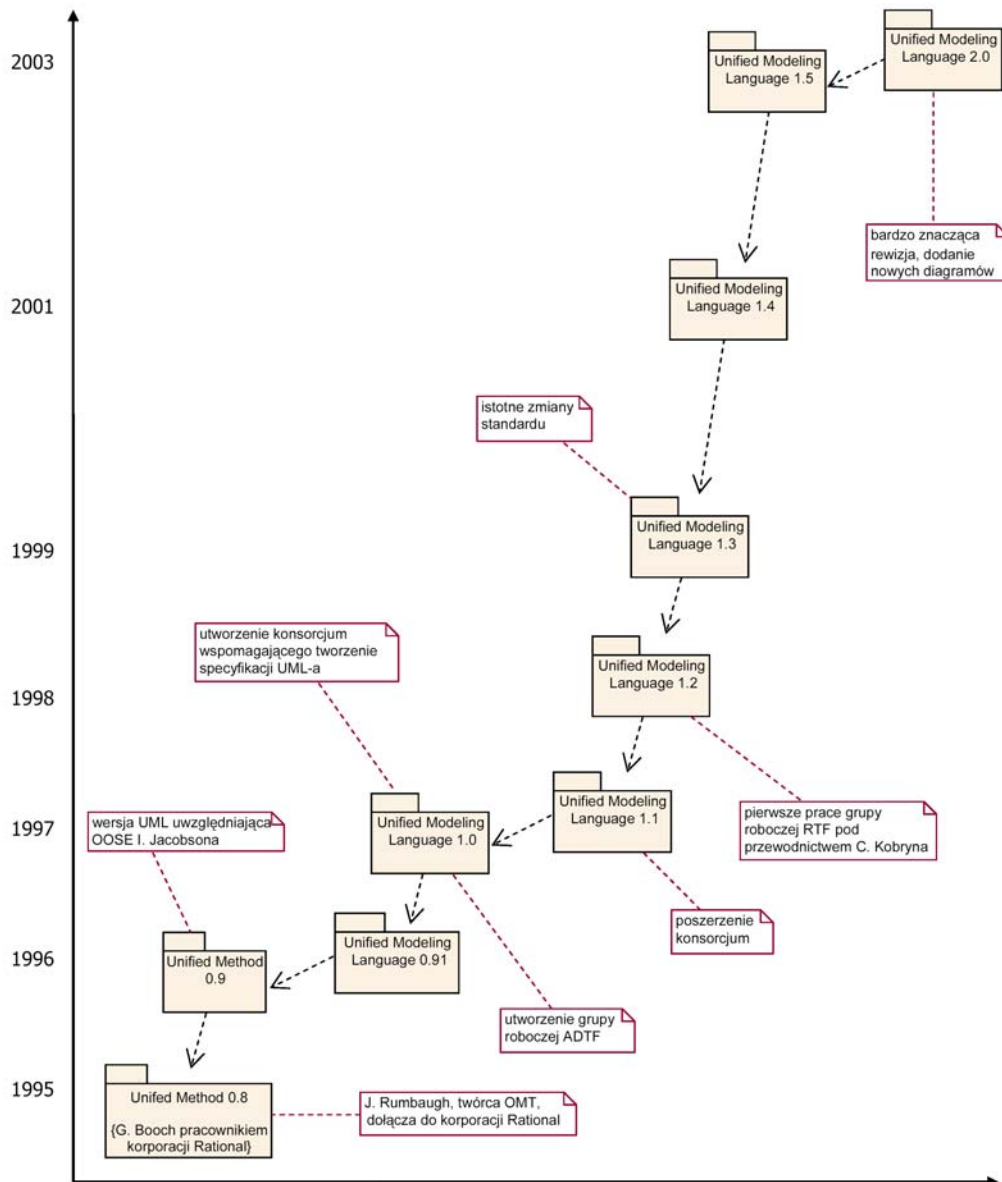
Autor	Nazwa metodyki	Skrót nazwy	Podstawowy wkład	Publikacje
Rumbaugh	<i>Object Modeling Technique</i>	OMT	Notacja diagramów UML, analiza i projektowanie	[Rumbaugh-1991]
Booch	<i>Object Oriented Analysis and Design</i>	OOAD	Analiza i projektowanie	[Booch-1991]
Jacobson	<i>Object Oriented Software Engineering</i>	OOSE	Modelowanie biznesowe, przypadki użycia, diagramy modelowania analitycznego	[Jacobson-1992]

Modelowanie systemów informatycznych obejmuje aspekty funkcjonalne oraz нефункциональные. Obie kategorie dynamicznie rozwijały się w związku z wymaganiami obiektowego wytwarzania oprogramowania. W praktyce analitycy, projektanci czy integratorzy wymagali adekwatnych metod i technik umożliwiających modelowanie złożonych współbieżności, interakcji, maszyn stanowych itd. Zatem możliwość modelowania funkcjonalnych i нефункциональных aspektów systemów informatycznych była i nadal jest motywem ewolucji języka UML, którego kolejne wersje są wzbogacane o nowe kategorie, będące odpowiedzią na wymagania twórców systemów informatycznych stawiane metodom i technikom modelowania.

W stworzeniu ujednoliconego podejścia upatrywano szansy nie tylko na osiągnięcie większej przejrzystości oferty rynkowej, lecz również na uzyskanie efektu skali poprzez wymianę doświadczeń. Założenia autorów języka UML wykraczały poza dokonanie prostej kompilacji. Przede wszystkim zdawano sobie sprawę z tego, że język modelowania pośredniczy pomiędzy ludzkim rozumieniem funkcjonowania programów komputerowych a ich fizyczną realizacją w postaci kodu źródłowego. Stąd język taki powinien jednocześnie:

- ♦ w sposób ścisły definiować podstawowe i zaawansowane kategorie oraz zasady modelowania obiektowego;
- ♦ umożliwiać dostosowywanie wykorzystywanej semantyki i notacji do rozwiązywania szerokiego spektrum problemów;
- ♦ wykazywać elastyczność wystarczającą do modelowania, obok systemów oprogramowania, także systemów biznesowych;
- ♦ wykazywać daleko posuniętą niezależność od konkretnych języków programowania oraz metodyk tworzenia systemów informatycznych;
- ♦ uwzględniać skalę realizowanych współcześnie projektów, związanych z bardzo rozbudowanymi systemami o kluczowym znaczeniu dla funkcjonowania przedsiębiorstw.

Ujednolicone podejście, powstałe z uwzględnieniem metodyk źródłowych zaprezentowanych w tabeli 1.1 oraz wymienionych założeń, dynamicznie rozwijało się na przestrzeni ostatnich lat. Historię oraz ścieżkę ewolucyjną języka UML przedstawia rysunek 1.1.



Rysunek 1.1. Ewolucja języka UML

Prace nad językiem UML zapoczątkowano w październiku 1994 roku, kiedy J. Rumbaugh dołączył do firmy Rational Software Corporation, gdzie wraz z G. Boochem rozpoczęli pracę nad ujednoliceniem metod OMT i OOAD. Efektem współpracy stał

się zaprezentowany przez firmę Rational na konferencji OOPSLA '95 pierwowzór języka UML. Wersję tę zaprezentowano w październiku 1995 roku. Ten dokument roboczy znany był jako UM (*Unified Method*) 0.8. W tym samym roku do zespołu dołączył I. Jacobson — autor metodyki OOSE. W czerwcu 1996 roku opublikowano wersję 0.9, uwzględniającą już wkład I. Jacobsona, natomiast pod koniec tego samego roku wersję 0.91. Od tego momentu język określa się jako UML.



UML (*Unified Modeling Language*) jest graficznym językiem wizualizacji, specyfikowania, tworzenia i dokumentowania systemów informatycznych.

Partnerami projektu rozwoju UML w korporacji Rational stały się czołowe korporacje świata. Należały do nich: IBM, Digital Equipment Corporation, Hewlett-Packard, Intellicorp, I-Logix, Icon Computing, Oracle, Unisys czy też Microsoft. Ich wsparcie w istotny sposób przyczyniło się do powstania wersji 1.0 w styczniu 1997 roku. Dzięki tej współpracy wersja 1.0 była już silnym i precyzyjnie zdefiniowanym językiem modelowania. Wersja ta została przekazana jako propozycja **standardu** organizacji OMG (*Object Management Group*) — międzynarodowej organizacji, której misją jest promowanie praktyki i teorii technologii obiektowych w tworzeniu oprogramowania. W OMG powołano wówczas grupę ADTF (*Analysis and Design Task Force*), która skoncentrowała się na rozwijaniu standardu.

W lipcu 1997 roku korporacja Rational przedstawia kolejną propozycję UML-a w wersji 1.1, również przekazaną do OMG. Dwa miesiące po jej przekazaniu, w listopadzie 1997 roku, OMG ADTF ostatecznie przyjmuje kolejny standard w wersji 1.1. W tym momencie powoływany jest nowy zespół do spraw zmian standardu UML — OMG RTF (*Revision Task Force*). Na jego czele staje Cris Kobryn, pod przewodnictwem którego w czerwcu 1998 roku powstaje wersja UML 1.2. Zmiany wprowadzone w wersji 1.2 były nieznaczne, koncentrowały się na poprawkach o charakterze redakcyjnym. Wersja 1.2 oraz następująca po niej wersja 1.3 to publikacje wewnętrzne OMG — oficjalnym standardem wciąż pozostawała wersja 1.1. W wersji 1.3 wprowadzono znaczące zmiany w zakresie diagramów przypadków użycia i diagramów czynności. W kwietniu 1999 roku zespół OMG RTF publikuje kolejną wersję — UML 1.4. Jej nieoficjalna dokumentacja była dostępna w połowie 2000 roku, natomiast oficjalną specyfikację wydano we wrześniu 2001 roku. Podobnie jak w przypadku przejścia z wersji 1.1 do 1.2, nie były to daleko idące zmiany — skoncentrowano się na poprawie spójności dokumentacji.

W tym okresie OMG wyznacza wersję 2.0 jako oficjalny kierunek zmian, wyodrębniając z wówczas obowiązującej dokumentacji cztery podlegające różnym zespołom specyfikacji:

- ♦ **infrastrukturę UML**, będącą podstawą architektoniczną składni języka UML 2.0, jego metamodeliem, pozwalającą na definiowanie i rozwijanie superstruktury UML [OMG-2003b];
- ♦ **superstrukturę UML**, specyfikującą ukierunkowane na użytkownika kategorie modelowania — elementarne składniki wykorzystywane w tworzeniu poszczególnych diagramów [OMG-2003a];

- ♦ **OCL** — język opisu ograniczeń [Warmer-2003];
- ♦ **wymienność diagramów UML**, czyli mechanizm przenoszenia dokumentów zgodnych ze standardem UML pomiędzy różnymi narzędziami [OMG-2003c].

We wrześniu 2002 roku pojawia się jednak nieoficjalnie wersja 1.5, która wprowadza bardzo szczegółowy model akcji. W marcu 2003 roku staje się ona wersją oficjalną.

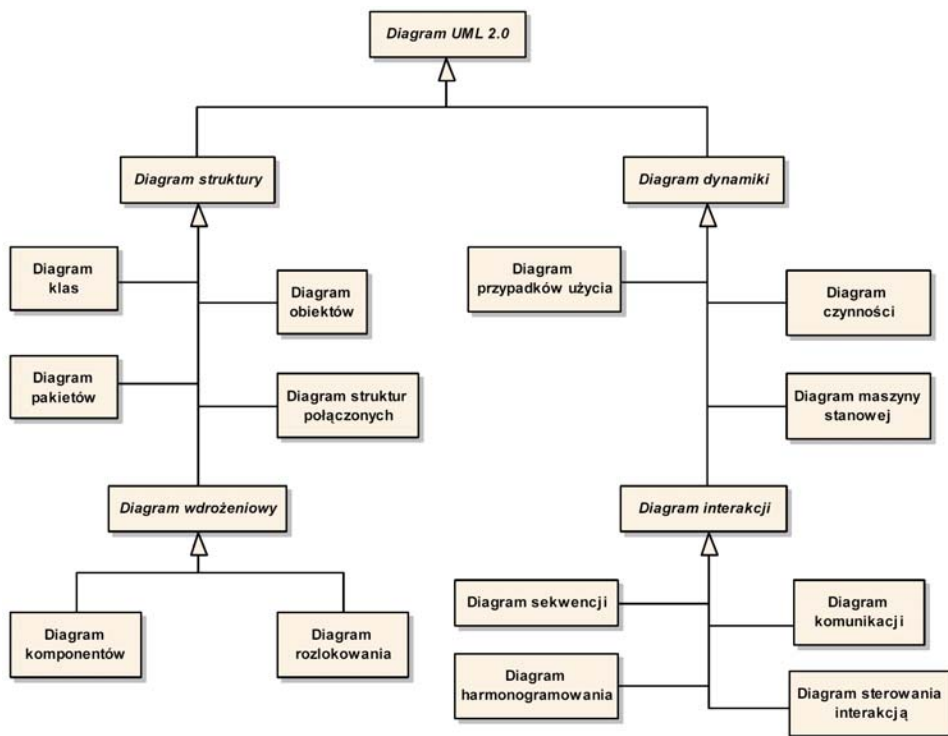
Wersja 2.0 zaprezentowana zostaje w sierpniu 2003 roku. Wersja ta znacząco różni się od poprzednich. Rozszerza ona zakres dostępnych diagramów do trzynastu oraz wprowadza szereg nowych kategorii modelowania w istniejących diagramach. Składnię i klasyfikację wielu istniejących kategorii istotnie zmodyfikowano, opierając je na opracowanej również przez OMG metaskładni **MOF** (*Meta Object Facility*). Na podstawie metaskładni MOF, OMG zdefiniowała mechanizm wymiany dokumentów zgodnych ze standardem UML, wprowadzając standardy **XMI** (*XML Metadata Interchange*) oraz **CWM** (*Common Warehouse Metamodel*). **XMI** umożliwia przejście ze specyfikacji UML do dokumentów opisanych w języku XML. Z kolei **CWM** opisuje wymianę metadanych pomiędzy hurtowniami danych, systemami klasy Business Intelligence, systemami zarządzania wiedzą oraz technologiami internetowymi.

Standard UML na bieżąco podlega aktualizacji, modyfikacjom i wzbogacaniu. W 2004 roku dokumentacja UML 2.0 ulega przeredagowaniu, co skutkuje opublikowaniem dokumentu *UML 2.0 Superstructure FTF Convenience Document* [OMG-2004]. Obecnie w pracach OMG współuczestniczy ponad 800 informatycznych organizacji biznesowych i badawczych, wprowadzających kolejne innowacje i rozszerzających zakres dokumentacji. Tym samym, ze względu na charakter i przeznaczenie niniejszej książki, mniejszą wagę przypisano kategoriom modelowania ściśle związanym z programowaniem.

Diagramy UML 2.0

Język UML przyjmuje w praktyce postać graficznej reprezentacji tworzonego systemu, składającej się z logicznie powiązanych z sobą **diagramów**. Pozwalają one na opisanie systemu od modeli ogólnych do bardzo szczegółowych. W standardzie UML w wersji 2.0 występuje trzynaście rodzajów diagramów, które charakteryzują statykę i dynamikę tworzonego systemu. Klasyfikację diagramów UML przedstawia rysunek 1.2.

Na rysunku 1.2 występują kategorie diagramów abstrakcyjnych i konkretnych. Diagramy **abstrakcyjne** są jedynie nazwami uogólniającymi grupy diagramów konkretnych. A zatem do kategorii diagramów abstrakcyjnych zalicza się diagramy struktury, dynamiki, interakcji, wdrożeniowe oraz samą kategorię diagramów UML. Abstrakcyjne kategorie pojęciowe zwyczajowo oznacza się kursywą. Pełny zestaw diagramów **konkretnych**, stanowiących standard języka UML 2.0, syntetycznie scharakteryzowano w tabeli 1.3, a szczegółowo zaprezentowano w kolejnych rozdziałach pierwszej części niniejszego podręcznika. Tabela 1.3 zawiera również wyróżniki rodzaju diagramu, które w unikalny sposób wskazują na jego specyficzną zawartość. Poza wymienionymi na rysunku 1.2 diagramami, środowiska akademickie i profesjonalne użytkujące język UML oraz metodykę RUP zaproponowały inne techniki diagramowe ściśle powiązane ze standardem UML. Są to m.in. diagramy modelowania analitycznego oraz diagramy modelowania biznesowego. Techniki te opisano w części drugiej.



Rysunek 1.2. Diagramy w języku UML 2.0

Tabela 1.3. Proponowane wyróżniki i charakterystyka rodzajów diagramów

LP	Rodzaj diagramu	Charakterystyka	Wyróżnik rodzaju diagramu		Rozdział
			polski	angielski	
1	Diagram klas (ang. <i>Class Diagram</i>)	Diagram klas to graficzne przedstawienie statycznych, deklaratywnych elementów dziedziny przedmiotowej oraz związków między nimi	kls	cld	3
2	Diagram obiektów (ang. <i>Object Diagram</i>)	Diagram obiektów to wystąpienie diagramu klas, odwzorowujące strukturę systemu w wybranym momencie jego działania	obk	od	3
3	Diagram pakietów (ang. <i>Package Diagram</i>)	Diagram pakietów to graficzne przedstawienie logicznej struktury systemu w postaci zestawu pakietów połączonych zależnościami i zagnieżdżeniami	pkt	pd	13
4	Diagram struktur połączonych (ang. <i>Composite Structure Diagram</i>)	Diagram struktur połączonych to graficzne przedstawienie wzajemnie współdziałających części dla osiągnięcia pożądanej funkcjonalności współdziałania	spl	csd	12

Tabela 1.3. Proponowane wyróżniki i charakterystyka rodzajów diagramów (ciąg dalszy)

LP	Rodzaj diagramu	Charakterystyka	Wyróżnik rodzaju diagramu		Rozdział
5	Diagram komponentów (ang. <i>Component Diagram</i>)	Diagram komponentów to rodzaj diagramu wdrożeniowego, który wskazuje organizację i zależności między komponentami	kmp	cod	11
6	Diagram rozlokowania (ang. <i>Deployment Diagram</i>)	Diagram rozlokowania to rodzaj diagramu wdrożeniowego, który przedstawia sieć połączonych ścieżkami komunikowania węzłów z ulokowanymi na nich artefaktami	rzk	dd	11
7	Diagram przypadków użycia (ang. <i>Use Case Diagram</i>)	Diagram przypadków użycia to graficzne przedstawienie przypadków użycia, aktorów oraz związków między nimi, występujących w danej dziedzinie przedmiotowej	uzc	ud	2
8	Diagram czynności (ang. <i>Activity Diagram</i>)	Diagram czynności to graficzne przedstawienie sekwencyjnych i (lub) współbieżnych przepływów sterowania oraz danych pomiędzy uporządkowanymi ciągami czynności, akcji i obiektów	czn	ad	4
9	Diagram maszyny stanowej (ang. <i>State Machine Diagram</i>)	Diagram maszyny stanowej to graficzne odzwierciedlenie dyskretnego, skokowego zachowania skończonych systemów stan-przejsie	stn	sm	5
10	Diagram sekwencji (ang. <i>Sequence Diagram</i>)	Diagram sekwencji jest rodzajem diagramu interakcji, opisującym interakcje pomiędzy instancjami klasyfikatorów systemu w postaci sekwencji komunikatów wymienianych między nimi	skw	sd	7
11	Diagram komunikacji (ang. <i>Communication Diagram</i>)	Diagram komunikacji jest rodzajem diagramu interakcji, specyfikującym strukturalne związki pomiędzy instancjami klasyfikatorów biorącymi udział w interakcji oraz wymianę komunikatów pomiędzy tymi instancjami	kmn	cd	8
12	Diagram harmonogramowania (ang. <i>Timing Diagram</i>)	Diagram harmonogramowania jest rodzajem diagramu interakcji, reprezentującym na osi czasu zmiany dopuszczalnych stanów, jakie może przyjmować instancja klasyfikatora uczestnicząca w interakcji	hrm	td	9
13	Diagram sterowania interakcją (ang. <i>Interaction Overview Diagram</i>)	Diagram sterowania interakcją jest rodzajem diagramu interakcji, dokumentującym przepływ sterowania pomiędzy logicznie powiązаныmi diagramami i fragmentami interakcji z wykorzystaniem kategorii modelowania diagramów czynności	sin	iod	10

W publikacjach polskich dotyczących języka UML, będących najczęściej tłumaczeniami literatury anglojęzycznej, używano synonimów nazw diagramów. Najistotniejsze z nich uwzględniono w słowniku angielsko-polskim (dodatek F). Specyfikacja języka UML wprowadza pojęcie **klasyfikatora** (ang. *classifier*), które ma zastosowanie w odniesieniu do praktycznie każdego rodzaju diagramu języka UML 2.0.



Klasyfikator to abstrakcyjna kategoria modelowania systemu w języku UML, która uogólnia kolekcję instancji o tych samych cechach.

Na konkretnych diagramach języka UML zamieszcza się **instancje** poszczególnych rodzajów klasyfikatorów. W świetle powyższej definicji instancję rozumieć należy następująco:



Instancja jest wystąpieniem, egzemplarzem klasyfikatora.

Tak więc klasyfikatorem jest interfejs, a jego wystąpieniem *IRejestracjaArtykułu*. Za podstawowy rodzaj klasyfikatora uznać należy klasę. Klasa ma instancje w postaci obiektów. Inne omawiane w niniejszej książce rodzaje klasyfikatorów obejmują m.in. węzeł, komponent, aktora, przypadek użycia i pakiet. Ze względu na logikę prezentacji materiału dydaktycznego wybrane rodzaje klasyfikatorów języka UML 2 podsumowano w tabeli 7.3. Obecnie w języku UML istnieje ponad 30 klasyfikatorów. Pełną taksonomię klasyfikatorów UML 2 zawiera dokumentacja OMG [OMG-2004, załącznik F].

W języku UML 2 funkcjonuje również pojęcie klasyfikatora ustrukturyzowanego (ang. *structured classifier*). Klasyfikator ustrukturyzowany zawiera specyfikację wewnętrznej struktury. Pojęcie to odnosi się do diagramów struktur połączonych (por. rozdział 12.).

Dowolny z wymienionych w tabeli 1.3 diagramów może być prezentowany w postaci:

- ♦ obramowanej,
- ♦ nieobramowanej.

Diagram w postaci obramowanej otoczony jest prostokątną **ramą** (ang. *frame*) zawierającą nagłówek. Przyjmuje on postać pentagamu umieszczonego w lewym górnym rogu diagramu. Typowy nagłówek diagramu UML 2.0 charakteryzuje się formalną, zaprezentowaną poniżej składnią:

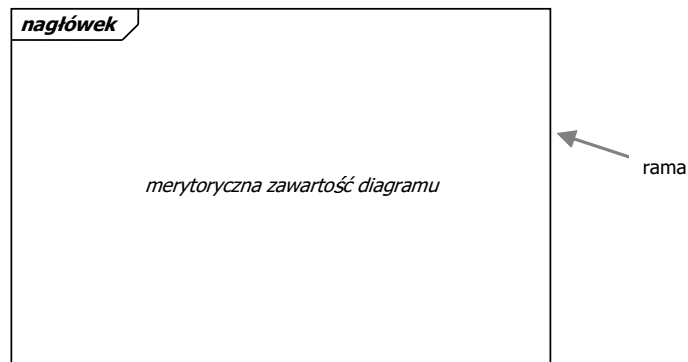
$$\langle \text{nagłówek-diagramu} \rangle ::= [\langle \text{rodzaj} \rangle] \langle \text{nazwa} \rangle [\langle \text{parametry} \rangle]$$

przy czym:

- ♦ *rodzaj* — pełny lub skrótowy **wyróżnik diagramu** zawartego w ramie;
- ♦ *nazwa* — syntetyczna **nazwa** odzwierciedlająca merytoryczną zawartość diagramu;
- ♦ *parametry* — szczegółowe **parametry** kluczowe dla danego diagramu, np. nazwy instancji klasyfikatorów, wartości zwrotne, operatory interakcji.

Notacja ta oznacza, że rodzaj diagramu oraz jego parametry są elementami umieszczanymi fakultatywnie, natomiast nazwa jest obligatoryjna. Elementy diagramu UML w postaci obramowanej przedstawia rysunek 1.3.

Rysunek 1.3.
*Format diagramu
obramowanego*



W formie obramowanej przedstawić można każdy z 13 rodzajów diagramów języka UML. Podczas tworzenia nagłówków diagramów można skorzystać z polsko- bądź anglojęzycznych wyróżników rodzajów diagramów (tabela 1.3). Kwestia wyróżników nie jest zagadnieniem ujednoliconym. I tak autorzy dokumentacji OMG oznaczają na przykład wszystkie diagramy interakcji wyróżnikiem **sd** (ang. *sequence diagram*). Propozycje i dostępność wyróżników w pakietach CASE wspierających język UML są zróżnicowane.

Konwencja tworzenia diagramów obramowanych, chociaż zalecana od momentu wprowadzenia wersji 2.0 języka UML, nie musi być bezwarunkowo realizowana. Dokumentacja systemu w języku UML, sporządzona z uwzględnieniem informacji zawartych w nagłówkach diagramów obramowanych, w przypadku rozbudowanych systemów poprawia przejrzystość dokumentacji.

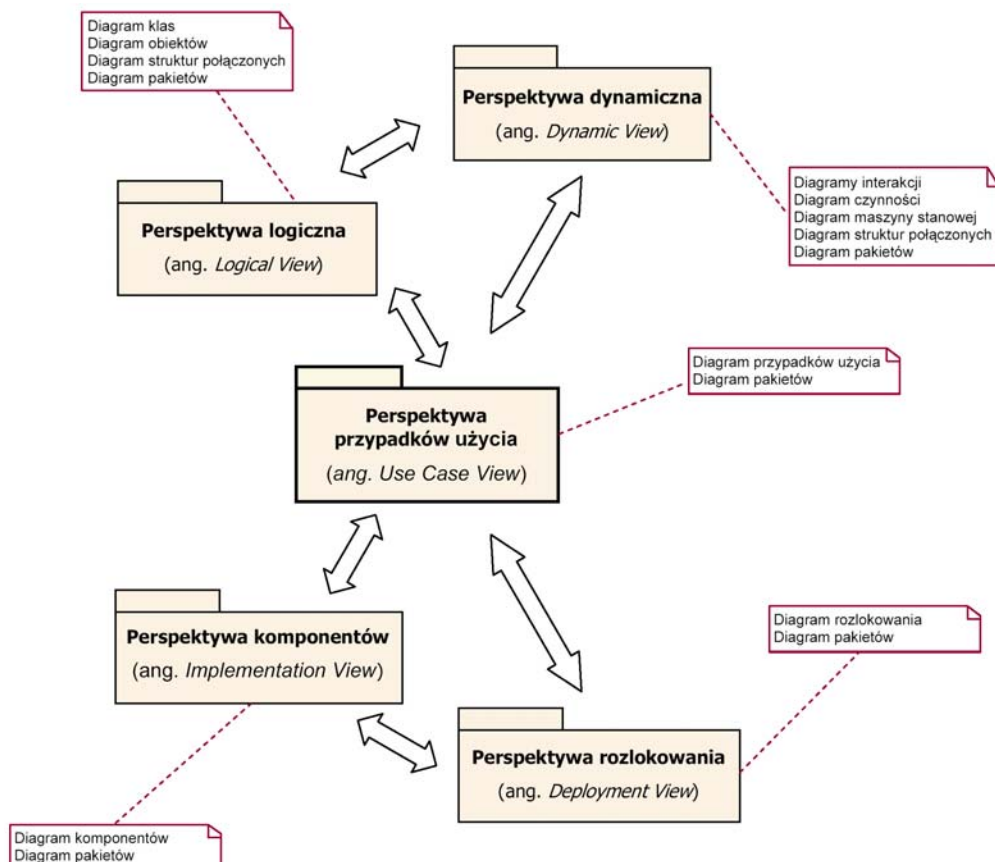
Stosowanie formy obramowanej można rozszerzyć na poszczególne instancje klasyfikatorów języka UML. I tak przedstawienie pakietu systemu w formie ramy może wiązać się z wykorzystaniem nieskrótowego wyróżnika jednoznacznie identyfikującego zawartość ramy jako pakiet — np. wyróżnika **package**. Z kolei zastosowanie ramy do opisu pojedynczego przypadku użycia opiera się na wyróżniku **useCase**.

Perspektywy w opisie architektury systemu

W skutecznym tworzeniu i użytkowaniu systemów informatycznych bierze udział wiele osób o różnych kompetencjach zawodowych i organizacyjnych. Są to właściciele, menedżerowie, analitycy, projektanci, programiści, testujący itd. Każda z tych grup zawodowych ma własny pogląd na system informatyczny, toteż poprzez swoje decyzje powinna mieć wpływ na jego architekturę. Ta różnorodność spojrzeń była zasadniczo nieuwzględniana w podejściu strukturalnym do TSI, a akcentowana w podejściu społecznym. W kontekście języka UML zróżnicowanie punktów widzenia znajduje swój wyraz w postaci powiązanej z nim metodyki RUP, proponującej pięć **perspektyw** (ang. *views*) architektury systemu informatycznego. Są to:

- ♦ perspektywa przypadków użycia — kluczowa i dominująca wobec pozostałych, definiuje zakres i oczekiwaną funkcjonalność tworzonego systemu;
- ♦ perspektywa dynamiczna — wskazuje, w jaki sposób jest realizowane zachowanie instancji klasyfikatorów w systemie;
- ♦ perspektywa logiczna — dokumentuje statykę systemu;
- ♦ perspektywa komponentów — przeznaczona głównie dla programistów, specyfikuje oprogramowanie na poziomie komponentów;
- ♦ perspektywa rozlokowania — specyfikuje sprzęt informatyczny niezbędny do funkcjonowania konkretnych komponentów systemu.

Każda z perspektyw uwypukla zatem inne aspekty architektury systemu, pomijając jednocześnie szczegóły nieistotne z punktu widzenia danej grupy zawodowej. Perspektywy wspierane są poprzez określone **zestawy diagramów** (rysunek 1.4).



Rysunek 1.4. Modelowanie perspektyw architektury systemu

Mechanizmy rozszerzalności

Język UML zapewnia bogaty zestaw pojęć związanych z modelowaniem systemów informatycznych oraz elementów notacji przydatnych w zapisywaniu typowych projektów systemów, niemniej odbiorcy mogą wymagać dodatkowych cech wykraczających poza te zdefiniowane w specyfikacji języka. Z tego powodu autorzy zdecydowali się na włączenie do języka UML zestawu **mechanizmów rozszerzalności** (ang. *extension mechanisms*), pozwalającego na używanie przez twórców systemu nowych kategorii modelowania specyficznych dla danej dziedziny zastosowania. Umożliwiają one wzbogacanie już istniejących kategorii pojęciowych o dodatkowe informacje, jak również modyfikację znaczeniową tych kategorii. W języku UML wyróżnia się trzy rodzaje mechanizmów rozszerzalności:

- ♦ stereotyp,
- ♦ ograniczenie,
- ♦ metka.

Stereotyp

Stereotypów (ang. *stereotypes*) używa się do klasyfikowania bądź oznaczania istniejących elementów modelu obiektowego oraz wprowadzania nowych kategorii modelowania, wywodzących się z już istniejących. I tak:



Stereotyp to mechanizm rozszerzalności, który wzbogaca zbiorowość standardowych kategorii modelowania języka UML.

Wzbogacenie, rozszerzenie o stereotypy w praktyce oznacza wprowadzenie nowych kategorii modelowania w poszczególnych diagramach języka UML, przy czym kategorie te bazują na semantyce standardowych, istniejących kategorii modelowania.

Wyróżnia się stereotypy **tekstowe** oraz **graficzne**. Nazwa stereotypu tekstowego zwyczajowo ujmowana jest w podwójny cudzysłów ostrokątny i umieszczana na stereotypowanym elemencie. Przykładowym stereotypem tekstowym odnoszącym się do:

- ♦ pakietów jest `<<subsystem>>`,
- ♦ związku zależności — `<<include>>` czy też `<<extend>>`,
- ♦ komponentu — `<<file>>`.

Z kolei stereotypy graficzne mają domyślnie postać specyficznych symboli graficznych umieszczanych na stereotypowanych elementach. Zastosowanie stereotypów nie ogranicza w ten sposób twórcy systemu, który ma daleko posuniętą swobodę w używaniu i proponowaniu różnorodnych symboli, specyficznych dla danej aplikacji.

Twórca systemu ma do dyspozycji znaczną liczbę stereotypów standardowych, rekomendowanych przez autorów języka UML oraz OMG — i w konsekwencji powszechnie stosowanych. Zostały one wymienione i scharakteryzowane w dokumentacji standardu [OMG-2004, załącznik C].

Ograniczenie

Kolejnym mechanizmem rozszerzenia jest **ograniczenie** (ang. *constraint*).



Ograniczenie to wyrażenie semantyczne określające warunek bądź zastrzeżenie związane z ograniczanym elementem modelowania bądź grupą elementów.

Ograniczenie jest w istocie tekstem wyrażonym w języku naturalnym, formułą matematyczną, predykatem logiki formalnej bądź instrukcją pseudokodu. W standardzie UML definiować je można w dedykowanym języku ograniczeń OCL (*Object Constraint Language*), zawierającym formalną składnię ograniczeń obiektowych. Ograniczenia umieszczane są w nawiasach klamrowych w bezpośrednim sąsiedztwie elementu lub elementów, których znaczenie jest precyzowane;

np. $\{disjoint\}$
 $\{czas < 15\ min\}$.

Metka

Każda kategoria modelowania języka UML charakteryzuje się specyficznym zestawem właściwości. Taki zbiór właściwości może być rozszerzany z wykorzystaniem **metek** (ang. *tagged values*).



Metka stanowi jawne zdefiniowanie właściwości w postaci przyporządkowania nazwa-wartość.

Konwencja wprowadzania metek do diagramów jest podobna jak w przypadku ograniczeń — jest to zapis w nawiasach klamrowych, składający się z nazwy, separatora oraz wartości (z zastrzeżeniem, że jest on umieszczany bezpośrednio pod nazwą danego elementu). Dozwolone jest dołączanie metek także do stereotypów. Najpowszechniej metki stosuje się celem określenia właściwości istotnych w generowaniu kodu oraz zarządzaniu konfiguracjami;

np. $\{wersja = beta\}$
 $\{model = HP\ LaserJet\ 1500L\}$.

Podstawowe pojęcia

Common Warehouse Metamodel	Stereotyp
Computer-Aided Software Engineering	Graficzny
Cykl życia systemu	Tekstowy
Diagram UML	Metodyka
Abstrakcyjny	Obiektowa
Konkretny	OMT
Diagramy dynamiki UML	OOAD
Diagram czynności	OOSE
Diagramy interakcji	Spółeczna
Diagram harmonogramowania	Strukturalna
Diagram komunikacji	Meta Object Facility
Diagram sekwencji	Modelowanie systemów
Diagram sterowania interakcją	Nagłówek diagramu
Diagram maszyny stanowej	Nazwa
Diagram przypadków użycia	Wyróżnik diagramu
Diagramy struktury UML	Parametr
Diagram klas	Notacja
Diagram obiektów	Obiektowość
Diagram pakietów	Dziedziczenie
Diagram struktur połączonych	Hermetyzacja
Diagramy wdrożeniowe	Klasa
Diagram komponentów	Komunikat
Diagram rozlokowania	Obiekt
Klasyfikator	Polimorfizm
Ustrukturyzowany	Object Constraint Language
Mechanizmy rozszerzalności	OMG
Metka	ADTF
Ograniczenie	RTF

Perspektywa	Superstruktura UML
Dynamiczna	System informatyczny
Komponentów	Tworzenie systemów informatycznych (TSI)
Logiczna	UML
Przypadków użycia	Wymienność diagramów UML
Rozłokowania	XML Metadata Interchange
Rama	

Pytania i zadania

1. Scharakteryzuj trendy rozwoju modelowania systemów informatycznych.
2. Jakie było znaczenie i skutki inicjatyw unifikacyjnych w dziedzinie tworzenia systemów informatycznych?
3. Jakie zjawiska sprzyjały wzrostowi znaczenia obiektowości w informatyce?
4. Podaj przykłady obszarów i narzędzi, w których model obiektowy znajduje zastosowanie.
5. Wskaż i scharakteryzuj kluczowe pojęcia obiektowości.
6. Zidentyfikuj nazwy kilku klas odnoszących się do środków transportu. Wskaż przykładowe obiekty tych klas.
7. Zaprezentuj mechanizm dziedziczenia, wykorzystując zidentyfikowane klasy środków transportu.
8. Wymień metodyki źródłowe języka UML.
9. Jakie cele sformułowane zostały w trakcie tworzenia języka UML?
10. Podaj definicję języka UML.
11. Wyjaśnij zależność pomiędzy korporacją Rational oraz organizacją OMG w kontekście rozwoju standardu UML.
12. Przedstaw fazy rozwoju języka UML.
13. Wyjaśnij relacje pomiędzy pojęciami:
 - ♦ superstruktura UML,
 - ♦ infrastruktura UML,
 - ♦ XMI,
 - ♦ wymienność diagramów UML,
 - ♦ MOF.

14. Czy modelowanie w języku UML ogranicza się do systemów informatycznych? Odpowiedź uzasadnij.
15. Przedstaw klasyfikację diagramów UML 2.0. Jakie diagramy nie występowały we wcześniejszych wersjach tego standardu?
16. Jakie postacie może przyjąć dowolny diagram? Wskaż informacje zawarte w nagłówku diagramu.
17. Wyjaśnij zależności pomiędzy następującymi pojęciami:
 - ♦ klasyfikator,
 - ♦ instancja,
 - ♦ klasa,
 - ♦ obiekt.
18. Jakie perspektywy występują w opisie architektury systemu? Omów istotę każdej z nich.
19. Zapisz w jednej kolumnie wszystkie perspektywy architektury systemu, a w drugiej poszczególne diagramy UML. Następnie powiąż odpowiednie perspektywy z odpowiadającymi im diagramami.
20. Jaka jest rola mechanizmów rozszerzalności?
21. Wymień podstawowe mechanizmy rozszerzalności języka UML.
22. Opracuj przykłady:
 - ♦ stereotypów tekstowych,
 - ♦ ograniczeń,
 - ♦ metek.
23. Modyfikacja sieci informatycznej uniwersytetu została udokumentowana na stosownych diagramach. Zaprojektuj stereotypy graficzne odnoszące się do poszczególnych elementów sprzętu sieciowego.
24. Odszukaj w Internecie pod adresem *www.omg.org* aktualne specyfikacje języka UML.

Rozdział 2.

Diagram przypadków użycia

Znaczenie diagramów przypadków użycia

Skuteczność tworzenia systemu informatycznego jest ściśle uzależniona od umożliwienia jednoznacznej specyfikacji i **dokumentowania wymagań** (ang. *requirements*) użytkowników wobec przyszłego systemu. W podejściu strukturalnym opracowano szereg metod wspomagających to zadanie. W przypadku języka UML i metodyki RUP rolę metody definiowania wymagań systemowych pełnią diagramy przypadków użycia (DPU). Określają one funkcjonalność analizowanego i tworzonego systemu oraz sposób komunikowania się użytkowników i klientów systemu z systemem.



Diagram przypadków użycia to graficzne przedstawienie przypadków użycia, aktorów oraz związków między nimi, występujących w danej dziedzinie przedmiotowej.

Poprzez wyrażoną związkami interakcję aktorów z przypadkami użycia zaprezentowaną na diagramach przypadków użycia, z jednej strony aktorzy pełnią określone role wobec systemu, a z drugiej przypadki użycia określają usługi świadczone przez system na rzecz aktorów (użytkowników, klientów). Pomijając podstawowy cel stosowania diagramów przypadków użycia w postaci identyfikacji oraz dokumentacji wymagań, spełniają one następujące **zadania** powiązane z celem głównym:

- ♦ umożliwiają analizę obszaru zastosowań, dziedziny przedmiotowej;
- ♦ pozwalają na opracowanie projektu przyszłego systemu;
- ♦ stanowią przystępną i zrozumiałą platformę komunikacji i współpracy udziałowców (ang. *stakeholders*) systemu — aktorów, twórców systemu, inwestorów i właścicieli;

- ♦ są rodzajem umowy, kontraktu pomiędzy udziałowcami co do zakresu i funkcjonalności przyszłego systemu;
- ♦ stanowią podstawę testowania funkcji systemu na dalszych etapach jego cyklu życia.

Diagramy przypadków użycia przedstawiają usługi systemu świadczone aktorom, bez wskazywania konkretnych rozwiązań technicznych.

Przypadki użycia, które jednoznacznie odnoszą się do procesów biznesowych, nazywamy **biznesowymi** przypadkami użycia (rozdział 15.), natomiast gdy odnoszą się do oprogramowania, nazywa się je **systemowymi** przypadkami użycia.

Podstawowe kategorie pojęciowe oraz notacja graficzna

Diagramy przypadków użycia zawierają następujące **główne kategorie pojęciowe**:

- ♦ przypadki użycia,
- ♦ aktorów,
- ♦ związki.

Kategorie te w każdym diagramie są ze sobą ściśle powiązane.

Przypadek użycia

Użytkownik — klient systemu oczekuje od niego, aby dostarczał zgodnie z wyspecyfikowanymi wcześniej wymaganiami określoną funkcjonalność stanowiącą wartość dla aktora. W ten sposób zostaje określony cel, zadanie, które system ma spełnić wobec aktora. A zatem:



Przypadek użycia to specyfikacja ciągu akcji i ich wariantów, które system (lub inna jednostka) może wykonać poprzez interakcje z aktorami tego systemu.

Przypadek użycia (ang. *use-case*) jest więc kompleksowym działaniem realizowanym w systemie w konsekwencji określonej aktywności **aktora** (ang. *actor*). Zakres danego przedsięwzięcia determinowany jest przez zestaw wszystkich wzajemnie powiązanych przypadków użycia. Pojedynczy przypadek użycia należy traktować jako reprezentanta spójnej jednostki funkcjonalności dostarczanej przez system. Wnoszona funkcjonalność musi mieć istotne znaczenie jako całość z punktu widzenia powiązanego z nią aktora.

Nazwę przypadku użycia stanowi zwięzłe polecenie wykonania funkcji w projektowanym systemie, sformułowane w trybie rozkazującym. Przykładowe przypadki użycia zamieszczono na rysunku 2.1. Przy ich tworzeniu zastosowano obecnie najpowszechniej używaną notację.

Rysunek 2.1.

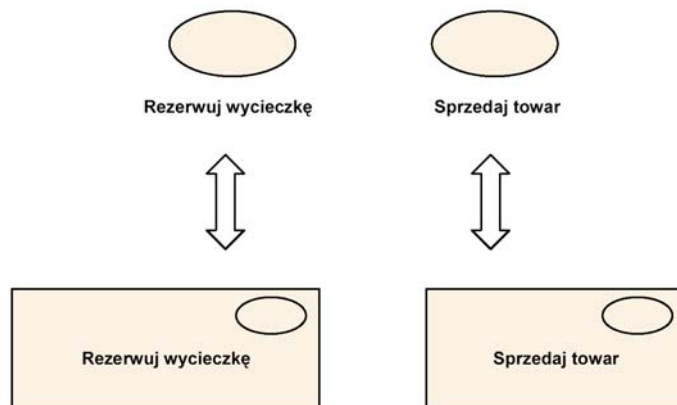
Oznaczenia przypadków użycia



W wersji 2.0 standardu UML zaproponowano również inne, alternatywne notacje przypadków użycia, tożsame w sensie znaczeniowym z notacją podstawową. Przedstawiono je na rysunku 2.2.

Rysunek 2.2.

Alternatywne notacje przypadków użycia



Aktor

Z każdym systemem komunikują się związani z nim aktorzy, w szczególności użytkownicy oraz klienci. A zatem:

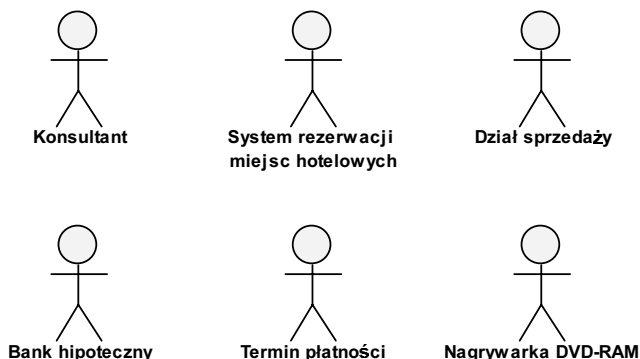


Aktor jest to spójny zbiór ról odgrywanych przez użytkowników przypadku użycia w czasie interakcji z tym przypadkiem użycia.

Aktorzy mogą być **osobowi** lub **niesobowi**. Rolę aktora osobowego może pełnić osoba, zespół, dział, instytucja, organizacja, zrzeszenie organizacji bądź organizacja wirtualna. Nazwy aktorów osobowych często pokrywają się z nazwami stanowisk lub funkcji pełnionych w organizacji, projekcie czy też przedsięwzięciu. Z kolei aktorami niesobowymi są systemy zewnętrzne (podsystemy, bazy danych), urządzenia oraz czas (rysunek 2.3).

Rysunek 2.3.

Podstawowe
rodzaje aktorów



Nazwę aktora wyraża się rzeczownikiem lub określeniem rzeczownikowym w liczbie pojedynczej. Identyfikując aktorów, należy pamiętać, że odzwierciedlają oni nie indywidualne obiekty ze świata rzeczywistego, lecz role pełnione przez te obiekty. Nawiązując do rysunku 2.3, można hipotetycznie stwierdzić, że kierujący firmą konsultingową zatrudni cały sztab *Konsultantów*, a nie tylko jednego. Jednak z punktu widzenia systemu będą oni odgrywać dokładnie tę samą rolę.

Aktor może **użytkować** jeden lub więcej przypadków użycia w danym systemie, natomiast przypadek użycia może być użytkowany przez jednego lub więcej aktorów. Interakcja aktora z przypadkami użycia polega na ich inicjowaniu, dostarczaniu danych, otrzymywaniu danych oraz użytkowaniu realizowanej przez przypadek użycia funkcjonalności.

Związek

Każdy aktor umieszczony na diagramie przypadków użycia powinien być bezpośrednio powiązany z co najmniej jednym przypadkiem użycia. Analogicznie, każdy przypadek użycia użytkowany jest przez co najmniej jednego aktora, chociaż niejednokrotnie są to powiązania pośrednie. W diagramie przypadków użycia nie powinny zatem występować niepowiązane elementy — są one zawsze powiązane **związkami** (ang. *relationships*).



Związek stanowi semantyczne powiązanie pomiędzy elementami modelu.

W diagramach języka UML wyróżnić można cztery **rodzaje związków**:

- ♦ asocjację,
- ♦ uogólnienie,
- ♦ zależność,
- ♦ realizację.

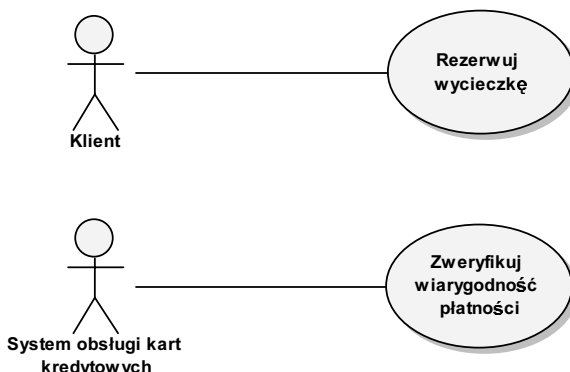
Realizacja w istocie jest szczególnym przypadkiem zależności, niemniej specyfika jej zastosowania uzasadnia potraktowanie jej jako odrębnego rodzaju związku. Najbardziej powszechne zastosowanie znajduje **asocjacja** (ang. *association*).



Asocjacja jest związkiem pomiędzy dwoma lub więcej klasyfikatorami, opisującym powiązania pomiędzy ich instancjami.

W diagramach przypadków użycia asocjacja wskazuje zatem domyślnie na dwukierunkową komunikację pomiędzy aktorem a przypadkiem użycia. Nie jest ona — jak w diagramie przepływu danych, stosowanym w ramach analizy i projektowania strukturalnego — oznaczeniem konkretnego przepływu, np. dokumentów. Może ona dodatkowo występować w formie ze wskazaniem kierunku nawigacji. Ta sytuacja ma miejsce wtedy, gdy z pewnych względów istotne jest jawne przedstawienie inicjatora interakcji (por. rysunek 2.19). Na rysunku 2.4 przedstawiono związki pomiędzy aktorami a przypadkami użycia.

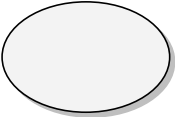
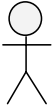
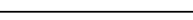
Rysunek 2.4.
*Związki asocjacji
w diagramach
przypadków użycia*



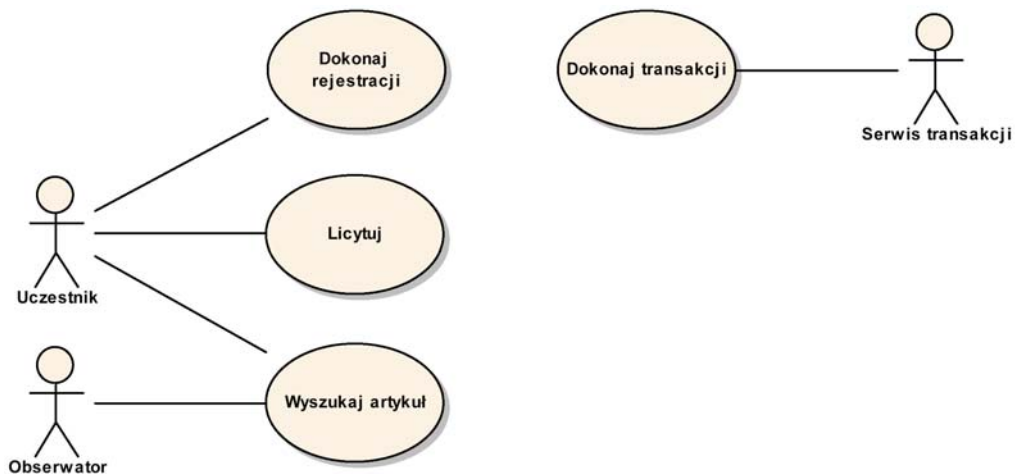
W przeciwieństwie do diagramów klas, gdzie dopuszcza się nazwy asocjacji, w elementarnych związkach typu aktor — przypadek użycia nie formułuje się nazwy związku. Rysunek 2.4 przedstawia podstawowe kategorie pojęciowe diagramu przypadków użycia, wskazujące, że sporządzenie zamówienia dotyczącego wycieczki leży w gestii *Klienta*. Weryfikowanie wiarygodności płatności odbywa się natomiast za pośrednictwem *Systemu obsługi kart kredytowych*, zewnętrznego w stosunku do tworzonego systemu. Na etapie definiowania przypadków użycia i szkicowania ich diagramu nie podaje się żadnych szczegółów dotyczących sposobu, w jaki wspomniane czynności są wykonywane, ani z jakich zasobów systemowych należy skorzystać w celu ich realizacji. Przedstawienie tych informacji to jedno z zadań innych rodzajów diagramów, omawianych w dalszych rozdziałach książki.

Notację graficzną głównych kategorii pojęciowych wykorzystywanych w ramach diagramów przypadków użycia syntetycznie ujęto w tabeli 2.1.

Tabela 2.1. Podstawowe elementy diagramu przypadków użycia

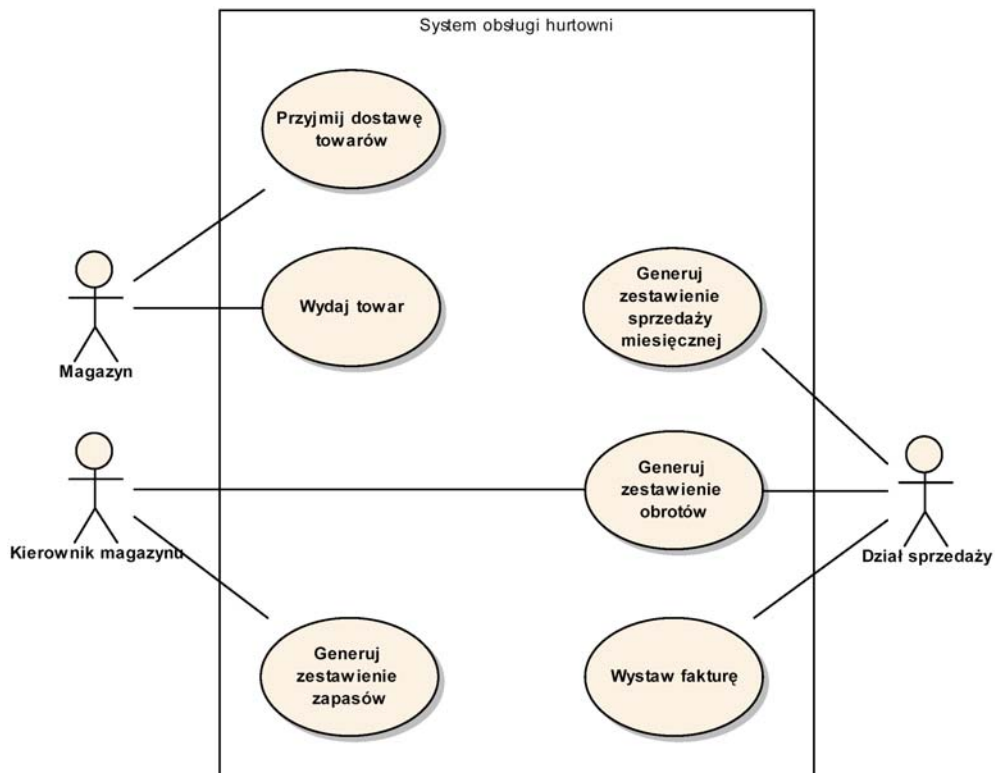
Nazwa pojęcia	Notacja graficzna
Przypadek użycia	
Aktor	
Asocjacja	

Przykładowy diagram przypadków użycia (DPU) zawierający zdefiniowane wyżej elementy zaprezentowano na rysunku 2.5.

**Rysunek 2.5.** Diagram przypadków użycia aukcji internetowej

Rysunek ten przedstawia trzech aktorów: *Uczestnika* aukcji internetowej, *Obserwatora* przebiegu licytacji oraz zewnętrznego w stosunku do systemu *Serwis transakcji*. Tylko zarejestrowany *Uczestnik* może brać bezpośredni udział w licytacji — chociaż zarówno on, jak i anonimowy *Obserwator* mogą swobodnie wyszukiwać dostępne na aukcji artykuły. Z systemem komunikuje się również *Serwis transakcji*, wymieniający z nim informacje niezbędne do poprawnego przeprowadzenia transakcji.

Rysunek 2.6 przedstawia zastosowanie przypadków użycia do opisu funkcjonowania hurtowni. W celu zaznaczenia **granicz obszaru zastosowań** przyszłego systemu, jego dziedziny przedmiotowej, można wprowadzić prostokąt grupujący przypadki użycia. Zawiera on tytuł określający nazwę analizowanej dziedziny przedmiotowej.



Rysunek 2.6. DPU systemu obsługi hurtowni

Granica obszaru zastosowań reprezentuje zakres funkcjonalny przyszłego systemu. Aktyorki komunikujących się z systemem umieszcza się poza granicą obszaru zastosowań. Na przedstawionym na rysunku 2.6 diagramie z systemem obsługi hurtowni komunikuje się *Magazyn*, w którym księguje się dostawy towarów oraz ich wydawanie. Naturalnie na diagramie przypadków użycia nie definiuje się, jakie konkretnie informacje składają się na wspomniane księgowanie. Zadaniem opisywanego diagramu jest tylko wskazanie, że są to wymagania systemu. Bieżące stany magazynowe podlegają nadzorowi *Kierownika magazynu*. Zarówno *Kierownik magazynu*, jak i *Dział sprzedaży* mają uprawnienia do generowania zestawienia obrotów. Ponadto *Dział sprzedaży* ma dostęp do zestawień okresowych i funkcji związanych z fakturowaniem.

Zaawansowane składniki diagramu

Do zaawansowanych koncepcji diagramów przypadków użycia należy zaliczyć:

- ♦ rozbudowę DPU poprzez różnicowanie związków,
- ♦ zależność zawierania,
- ♦ zależność rozszerzania,

- ♦ uogólnienie,
- ♦ rodzaje aktorów,
- ♦ liczebność,
- ♦ nawigację,
- ♦ realizację,
- ♦ przypadki użycia typu CRUD,
- ♦ diagram kontekstowy,
- ♦ dokumentację przypadków użycia.

Rozbudowa DPU poprzez różnicowanie związków

Jak zaznaczono, asocjacja to podstawowy rodzaj związku wykorzystywany w konstruowaniu diagramów przypadków użycia. Poza nią na diagramach przypadków użycia modelować można związki zależności, uogólnienia i realizacji. Związki asocjacji występują wyłącznie pomiędzy aktorami a przypadkami użycia. Pozostałe rodzaje związków pozwalają pokazać, w jakich relacjach znajdują się poszczególne przypadki użycia. Ponadto związki uogólnienia umożliwiają udokumentowanie relacji pomiędzy aktorami. Dzięki różnym rodzajom związków można tworzyć **złożoną strukturę** modelu przypadków użycia. Stanowi on wówczas zestaw logicznie powiązanych diagramów. Porządkującą rolę mogą odegrać tu diagramy pakietów, przedstawione w rozdziale 13. W ten sposób można tworzyć bardzo rozbudowane diagramy przypadków użycia, precyzyjnie odwzorowujące rzeczywistość i uwzględniające logiczne relacje pomiędzy poszczególnymi elementami DPU.

Szczególne możliwości rozbudowywania diagramów przypadków użycia stwarzają zależności (ang. *dependencies*).



Zależność to taki związek pomiędzy dwoma elementami modelowania, w którym zmiana jednego z nich, niezależnego, wpływa na drugi, zależny.

I tak w obrębie pojedynczego diagramu przypadków użycia zależności w DPU występują w postaci stereotypowanej, przede wszystkim jako zależności:

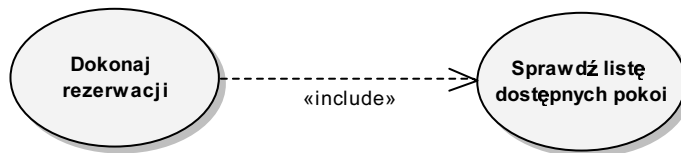
- ♦ zawierania,
- ♦ rozszerzania.

Zależności zawierania

Zależność zawierania «*include*» przedstawia powiązanie pomiędzy **przypadkiem zawierającym**, tj. bazowym przypadkiem użycia, a **przypadkiem zawieranym**. Jest to związek obligatoryjny. Zawierany przypadek użycia nie jest wykonywany samodzielnie, lecz wyłącznie przy odwołaniu się do większego, zawierającego przypadku użycia (rysunek 2.7).

Rysunek 2.7.

Zależność zawierania
«include»



Sens tworzenia zależności zawierania na diagramie przypadków użycia występuje w dwóch sytuacjach — gdy:

- ♦ istnieje możliwość **wydzielenia** w formie zawieranego przypadku użycia pewnej spójnej, wspólnej dla kilku innych przypadków użycia funkcjonalności; jest to przykład praktycznego zastosowania zasady ponownego użycia — nie zachodzi konieczność powielania analogicznej treści w wielu przypadkach użycia; tym samym często występująca funkcjonalność jest reprezentowana jednokrotnie na diagramie i wykorzystywana przez inne przypadki użycia;
- ♦ interakcje aktor-system wyrażone w dokumentacji scenariusza tego przypadku są bardzo liczne.

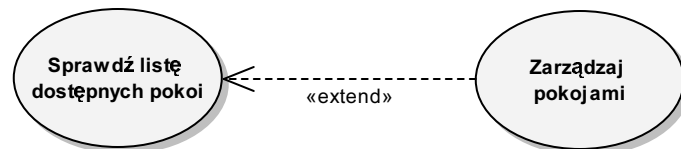
Na diagramie na rysunku 2.7 wykonanie przypadku użycia *Dokonaj rezerwacji* pociąga za sobą konieczność zweryfikowania dostępności pokoi. Uzyskuje się to poprzez zainicjowanie zawieranego przypadku użycia *Sprawdź listę dostępnych pokoi*. Po wykonaniu tego przypadku następuje wykonanie funkcjonalności przypadku zawierającego *Dokonaj rezerwacji*. Ten sam zawierany przypadek użycia *Sprawdź listę dostępnych pokoi* może być wykorzystywany przez inne przypadki zawierające, które go przywołują. Przykładowo może on zostać użyty w podsystemie rozdysponowania obsługi hotelowej.

Zależności rozszerzania

Zależność rozszerzania «*extend*» przedstawia powiązanie pomiędzy **rozszerzanym przypadkiem użycia**, tj. przypadkiem bazowym, a **przypadkiem rozszerzającym**. Związek ten, w odróżnieniu od zależności zawierania, ma charakter opcjonalny. Funkcjonalność reprezentowana przez rozszerzający przypadek użycia może, ale nie musi zostać włączona do rozszerzanego przypadku użycia (rysunek 2.8). Włączenie przypadku rozszerzającego wymaga spełnienia określonego warunku.

Rysunek 2.8.

Zależność rozszerzania
«extend»



Tworzenie zależności rozszerzania znajduje uzasadnienie, o ile funkcjonalność reprezentowana przez rozszerzany przypadek użycia ma zostać **uzupełniona** o kilka dodatkowych kroków. Nie mają one być jednak automatycznie wykonywane przy każdym odwołaniu do tego przypadku użycia, a jedynie w określonych sytuacjach. W przypadku

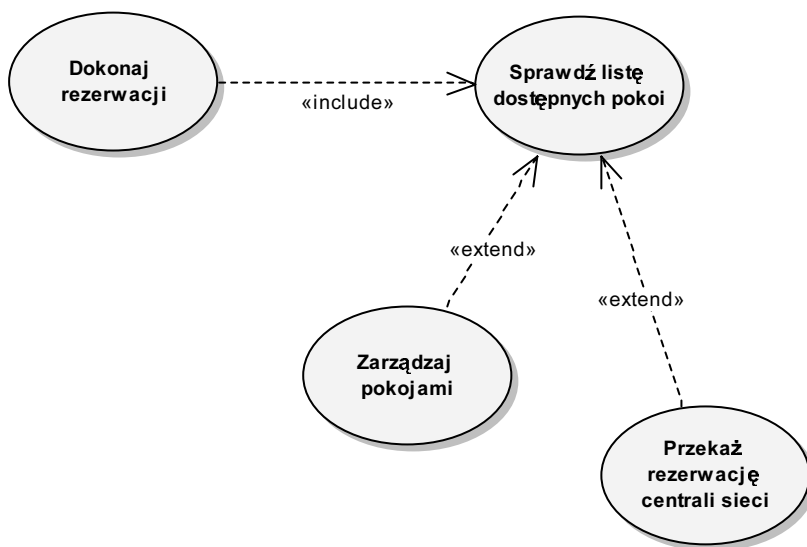
powyższego wycinka DPU *Sprawdzenie listy dostępnych pokoi* nie jest jednoznaczne z *Zarządzaniem pokojami*. System hotelowy, oprócz wyszukiwania pokoi spełniających wyspecyfikowane przez klienta kryteria, musi bowiem uwzględniać takie elementy jak:

- ♦ przypisanie pokojowi innej kategorii,
- ♦ zmiana ceny wynajmu,
- ♦ zmiana statusu pokoju na „dla niepalących”,
- ♦ czasowe wyłączenie pokoju z użytkowania.

Wymienione sytuacje są reprezentowane przez przypadek użycia *Zarządzanie pokojami*.

Zależność zawierania jest **skierowana** od przypadku zawierającego do zawieranego, natomiast zależność rozszerzania — od przypadku rozszerzającego do rozszerzanego. Zależności zawierania i rozszerzania występują na diagramach przypadków użycia niezależnie od siebie (rysunek 2.9). Na rysunku tym przypadkiem zawierającym jest przypadek *Dokonaj rezerwacji*. Zawiera on przypadek *Sprawdź listę dostępnych pokoi*. Przypadek ten musi być wykonywany każdorazowo przy realizacji przypadku *Dokonaj rezerwacji*. Z kolei przypadek *Sprawdź listę dostępnych pokoi* może być rozszerzany o funkcjonalność przypadków *Zarządzaj pokojami* i *Przełącz rezerwację centrali sieci*. Są one zatem opcjonalne.

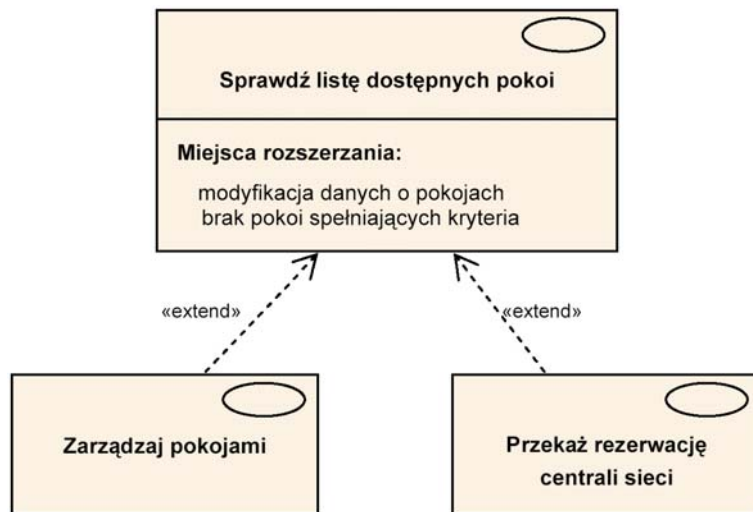
Rysunek 2.9.
Różne rodzaje
zależności
stereotypowanych



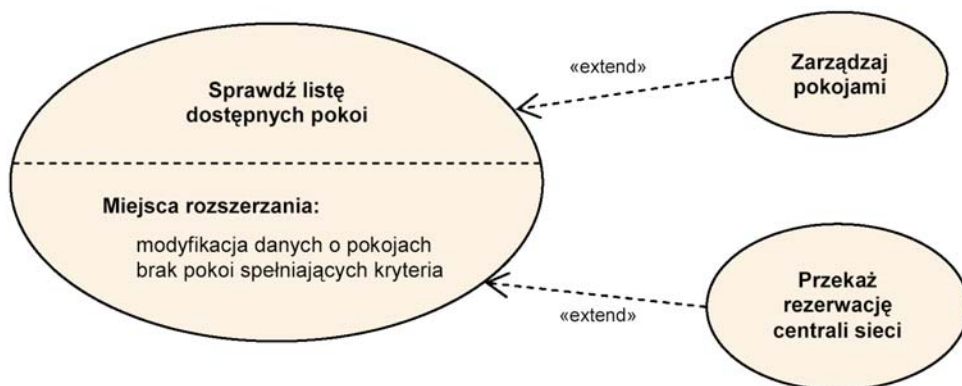
Jak wspomniano, dodatkowa funkcjonalność zawarta w przypadkach rozszerzających nie jest wykonywana automatycznie. Przypadek bazowy może być rozszerzany o przypadki rozszerzające po spełnieniu określonych warunków. Warunki te określa się w przypadku bazowym jako **miejsca rozszerzania** bądź punkty rozszerzania (ang. *extension points*). Po wykonaniu wszelkich działań związanych z rozszerzeniem kontynuowane są działania należące do przypadku bazowego. Zależności rozszerzania zaprezentowane na rysunku 2.9, wzbogacone o miejsca rozszerzania, przedstawia rysunek 2.10.

Rysunek 2.10.

Miejsca rozszerzania
w przypadku użycia



Analogiczne informacje można wnieść do diagramu, stosując alternatywną notację (rysunek 2.11), jak również przy użyciu notatek (rysunek 2.12).

**Rysunek 2.11.** Miejsca rozszerzania — notacja alternatywna

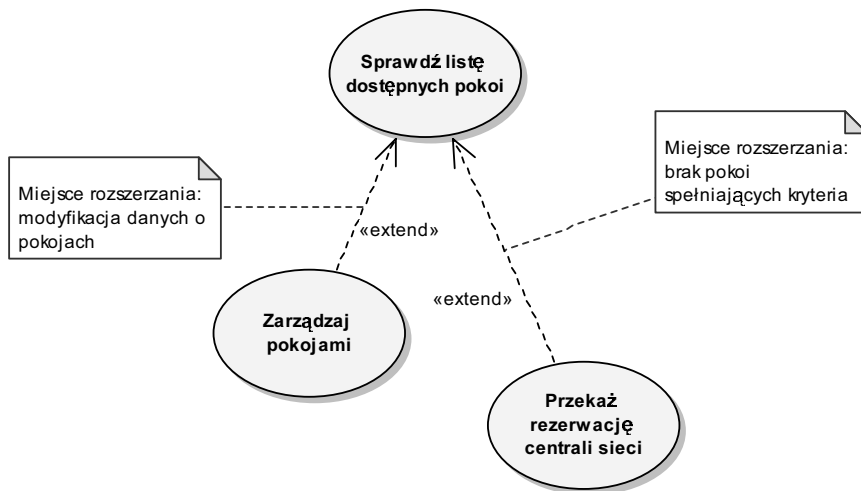
Uogólnienia

Związki **uogólnienia** (ang. *generalizations*) dotyczą w kontekście charakteryzowanego diagramu zarówno przypadków użycia, jak i aktorów.

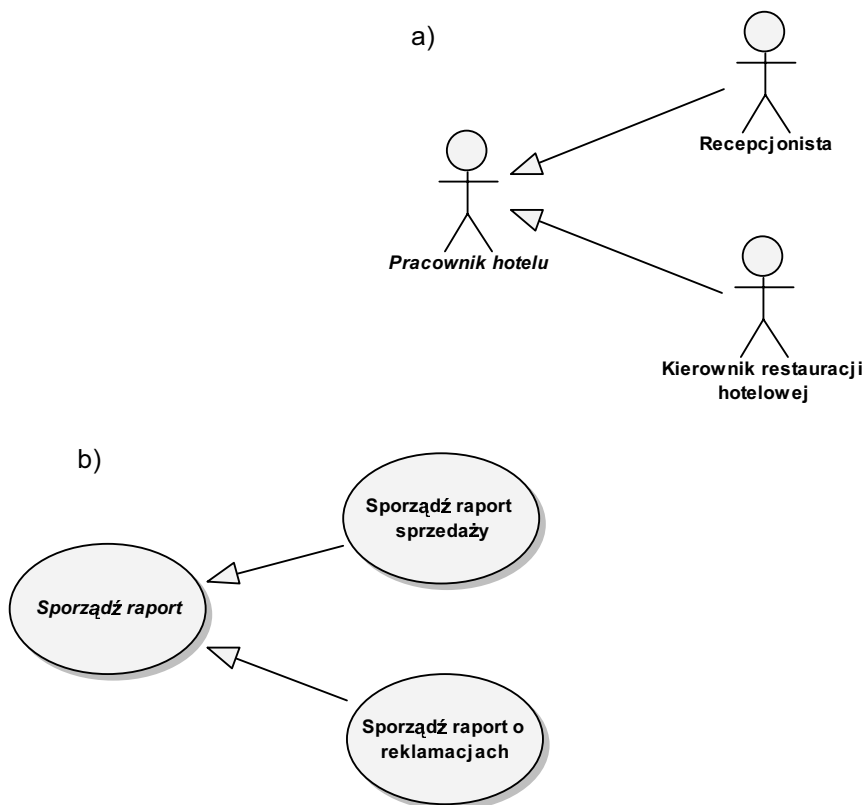


Uogólnienie to związek o charakterze taksonomicznym pomiędzy klasyfikatorem ogólnym a specjalizowanym.

Element specjalizowany z założenia dziedziczy wszelkie cechy elementu ogólnego. Rysunek 2.13a przedstawia przykłady uogólnienia aktorów, podczas gdy rysunek 2.13b — uogólnienia przypadków użycia.



Rysunek 2.12. Notatki w opisie miejsc rozszerzania



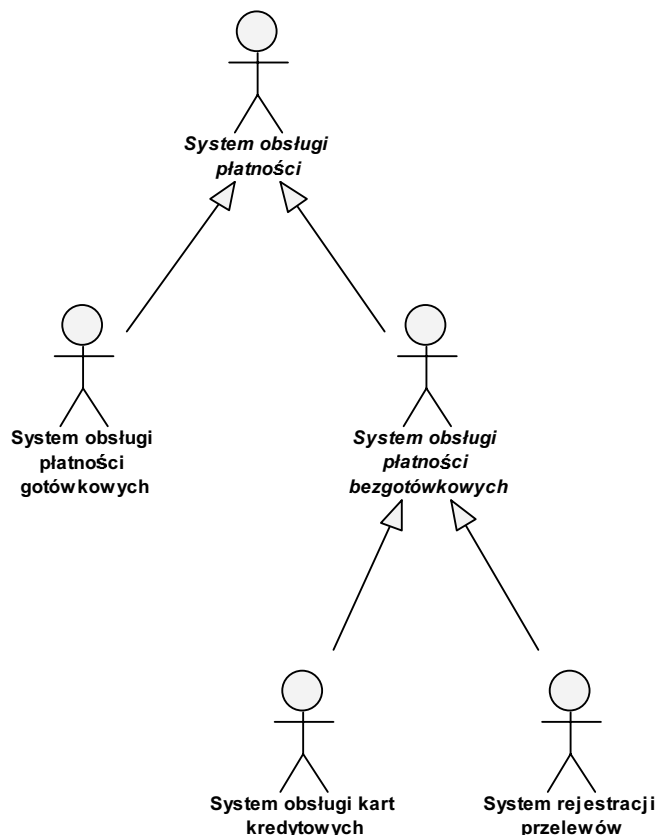
Rysunek 2.13. Przykłady uogólnień

Powyższy rysunek opisuje *Sporządzanie raportu*, przy czym raport sprzedaży charakteryzuje się zarówno cechami podobnymi do raportu o reklamacjach — takimi jak zawartość nagłówka, podobna procedura generowania czy analogiczny układ graficzny — jak i cechami odmiennymi, wynikającymi z innych danych źródłowych oraz innego przeznaczenia.

Za pomocą związku uogólnienia można wyrazić zależności o znacznie wyższym stopniu złożoności poprzez wskazanie dalszych elementów dziedziczących. Prowadzi to do powstania **struktury hierarchicznej** przedstawionej na rysunku 2.14.

Rysunek 2.14.

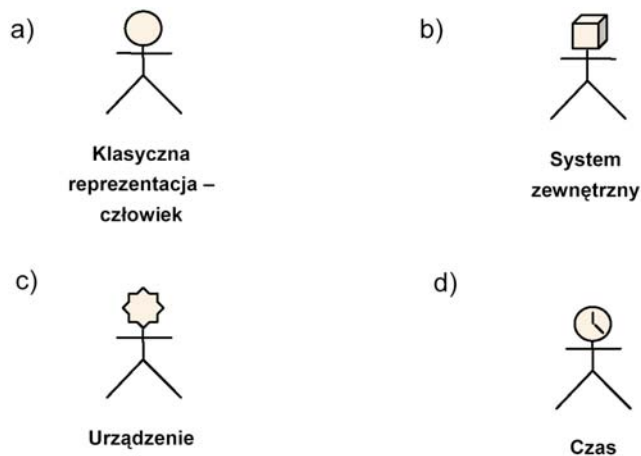
Hierarchia dziedziczenia wyrażona poprzez związki uogólnienia



Rodzaje aktorów

Język UML jest elastyczny, co przejawia się między innymi w umożliwieniu wprowadzania nowych pojęć oraz oznaczeń zwanych **stereotypami**. W odniesieniu do diagramów przypadków użycia ta cecha języka UML pozwala np. wprowadzać różnorodność w oznaczaniu aktorów. Notacja aktorów może w szczególności przybrać postać stereotypów graficznych. I tak uzasadnione jest wyróżnienie czterech rodzajów aktorów (rysunek 2.15):

Rysunek 2.15.
Stereotypy graficzne aktorów



- ♦ ludzi oraz zespołów,
- ♦ systemów zewnętrznych,
- ♦ urządzeń,
- ♦ czasu.

Alternatywą dla stereotypów graficznych, będących odpowiednikami ikon, są opisy tekstowe umieszczone w podwójnych nawiasach trójkątnych — np. aktor, który jest systemem zewnętrznym, może dodatkowo mieć oznaczenie «system». Takie oznaczenie zamieszczono chociażby na rysunku 2.19. Szczególnie istotne znaczenie ma możliwość umieszczania na diagramie przypadków użycia **czasu**, determinującego przede wszystkim rozpoczęcie bądź zakończenie jakiegoś okresu. Uprozczone diagramy przypadków użycia, na których aktorami są odpowiednio: ludzie, czas, urządzenia oraz systemy zewnętrzne, przedstawiono na rysunku 2.16.

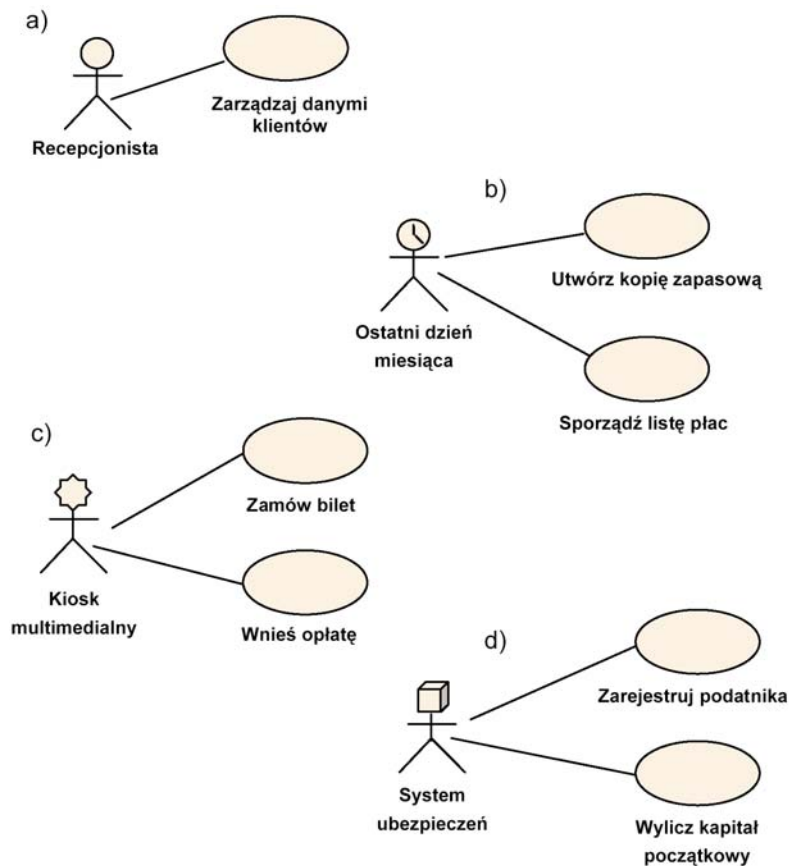
Tekstową odmianę stereotypów zaprezentowano na rysunku 2.17, zachowując analogię do aktorów z rysunku 2.16.

Liczebność

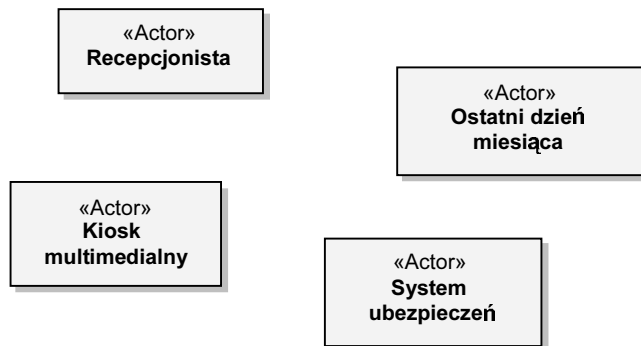
Diagramy przypadków użycia pozwalają na określenie **liczebności** (ang. *multiplicity*) końcówek związku asocjacji pomiędzy aktorami i przypadkami użycia. W nawiązaniu do przykładu aukcji, omawianą sytuację można przedstawić w sposób zaprezentowany na rysunku 2.18.

Diagram ten odwołuje się ponownie do wybranego wycinka mechanizmu funkcjonowania aukcji internetowej, wskazując, że z pojedynczym *Uczestnikiem* związany jest jeden i tylko jeden przypadek użycia *Dokonaj rejestracji*. Funkcjonalność związana z *Wyszukiwaniem artykułu*, *Licytowaniem* czy też *Dokonywaniem transakcji* w odniesieniu do pojedynczego aktora może być inicjowana wielokrotnie. W szczególnym przypadku może nigdy nie zostać wykorzystana.

Rysunek 2.16.
Zastosowanie
stereotypów
graficznych aktorów

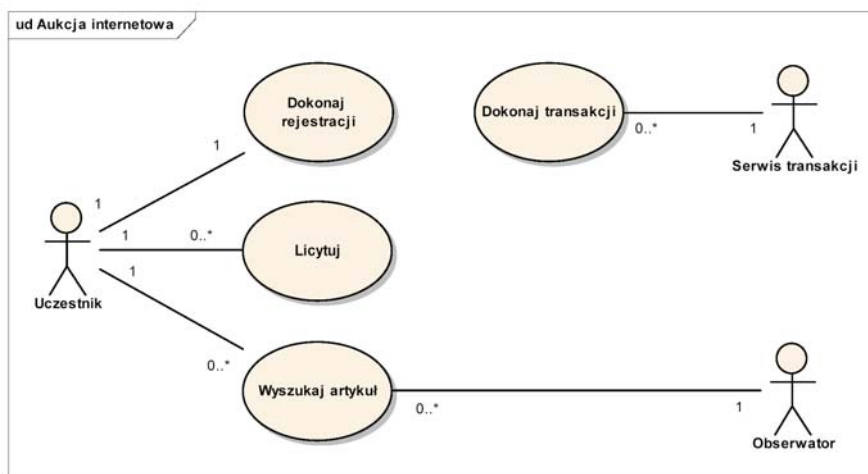


Rysunek 2.17.
Stereotypy tekstowe
aktorów



Nawigacja

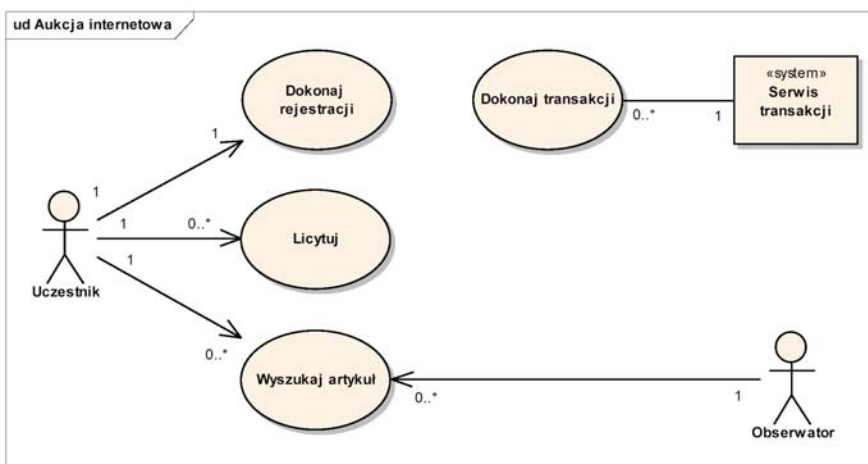
Asocjacje na diagramach przypadków użycia są zawsze **binarne**, łącząc w każdym przypadku dwie instancje klasyfikatorów, tj. aktora oraz przypadek użycia. Jak zasygnalizowano wcześniej, związki pomiędzy aktorami i przypadkami użycia obrazują



Rysunek 2.18. Liczebność na diagramach przypadków użycia

wzajemną interakcję, a nie przepływy danych — stąd z zasady asocjacje nie są oznaczane strzałkami. Jednak dla podkreślenia strony inicjującej dodatkowo asocjację można wzbogacić o wskazanie kierunku nawigacji (ang. *navigability*) w postaci strzałki.

I tak asocjację ze wskazaniem kierunku **nawigacji** na diagramie przypadków użycia przedstawia rysunek 2.19. Aktora *Serwis transakcji* przedstawiono na nim za pomocą stereotypu tekstowego. Stroną inicjującą interakcję z systemem w przypadku *Dokonowywania rejestracji*, *Licytowania* oraz *Wyszukiwania artykułu* jest zawsze *Uczestnik* bądź *Obserwator* aukcji. W odniesieniu do przypadku *Dokonaj transakcji* zastosowanie tej prawidłowości byłoby już nieuzasadnione. Oczywiście oznaczanie nawigacji jest opcjonalne i stosowane tylko wtedy, gdy funkcja inicjatora przypadku użycia lub aktora jest istotna i wymaga udokumentowania na diagramie.



Rysunek 2.19. Nawigacja na diagramach przypadków użycia

Realizacja

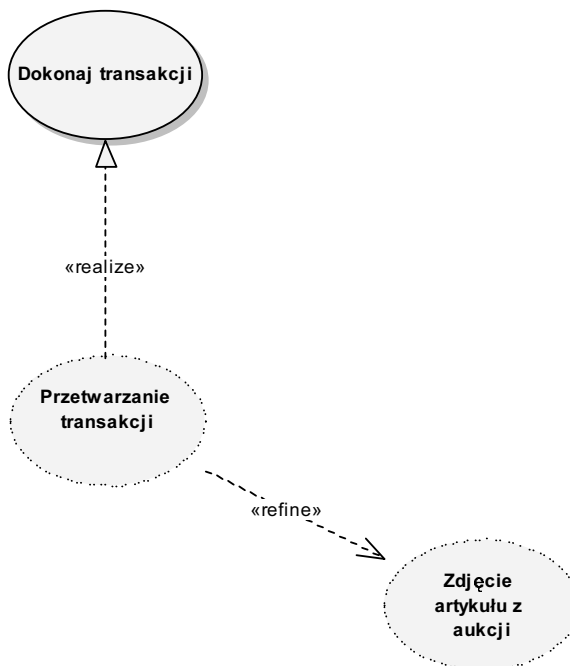
Ostatnim rodzajem związków używanych w języku UML są **realizacje** (ang. *realizations*).



Realizacja to związek znaczeniowy między klasyfikatorami, z których jeden określa kontrakt, a drugi zapewnia wywiązanie się z niego.

Związki realizacji w odniesieniu do przypadków użycia pozwalają modelować relacje występujące pomiędzy ogólnym, funkcjonalnym opisem systemu w postaci diagramów przypadków użycia a jego wdrożeniem. W ten sposób diagram przypadków użycia zostaje jawnie powiązany z innymi diagramami (zarówno statycznymi, jak i dynamicznymi), które zawierają informacje niezbędne do jego poprawnego rozwinięcia i implementacji. Modele opisujące wdrożenie przypadku użycia określane są mianem **współdziałań** (ang. *cooperations*). Współdziałania można łączyć w ciągi coraz bardziej szczegółowych, dalej idących specyfikacji za pomocą zależności stereotypowej *«refine»* (rysunek 2.20), która wskazuje, że element źródłowy jest bardziej szczegółowy niż element docelowy.

Rysunek 2.20.
Związek realizacji



Przypadki użycia typu CRUD

Wiele przypadków użycia wiąże się z przechowywaniem i użytkowaniem danych. Typową funkcjonalnością takich przypadków użycia jest tworzenie, aktualizowanie, usuwanie i wyszukiwanie danych. Jest to znana z baz danych konwencja CRUD:

- ♦ *Create* — tworzenie, wprowadzanie;
- ♦ *Read* — odczytywanie, wyszukiwanie;
- ♦ *Update* — aktualizacja, modyfikowanie;
- ♦ *Delete* — usuwanie, skreślanie.

Osoba odpowiedzialna za identyfikację i specyfikację przypadków użycia, w odniesieniu do przypadków użycia typu CRUD ma do wyboru dwie drogi postępowania. I tak w przypadku zamówienia może ona wybrać jedną z dwóch możliwości:

- ♦ zaprojektować elementarne przypadki użycia *Wprowadź zamówienie*, *Wyszukaj zamówienie*, *Aktualizuj zamówienie*, *Usuń zamówienie*;
- ♦ wyspecyfikować uogólniony przypadek użycia *Zarządzaj zamówieniami* lub *Utrzymuj dane zamówienia*.

Wybór jednego z tych podejść uzależniony jest od liczby interakcji aktora z systemem, niezbędnych do wykonania ww. przypadków użycia.

Stosowanie nazw ścieżkowych

Nazwa przypadku użycia może być prosta lub ścieżkowa. Nazwa prosta jest rutynowym sposobem określania przypadków użycia, natomiast w ramach stosowania nazwy ścieżkowej nazwę przypadku użycia poprzedza nazwa **pakietu**, w którym dany przypadek użycia się znajduje (rysunek 2.21).

Rysunek 2.21.

*Nazwa prosta
a nazwa ścieżkowa*

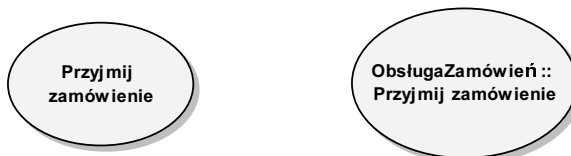
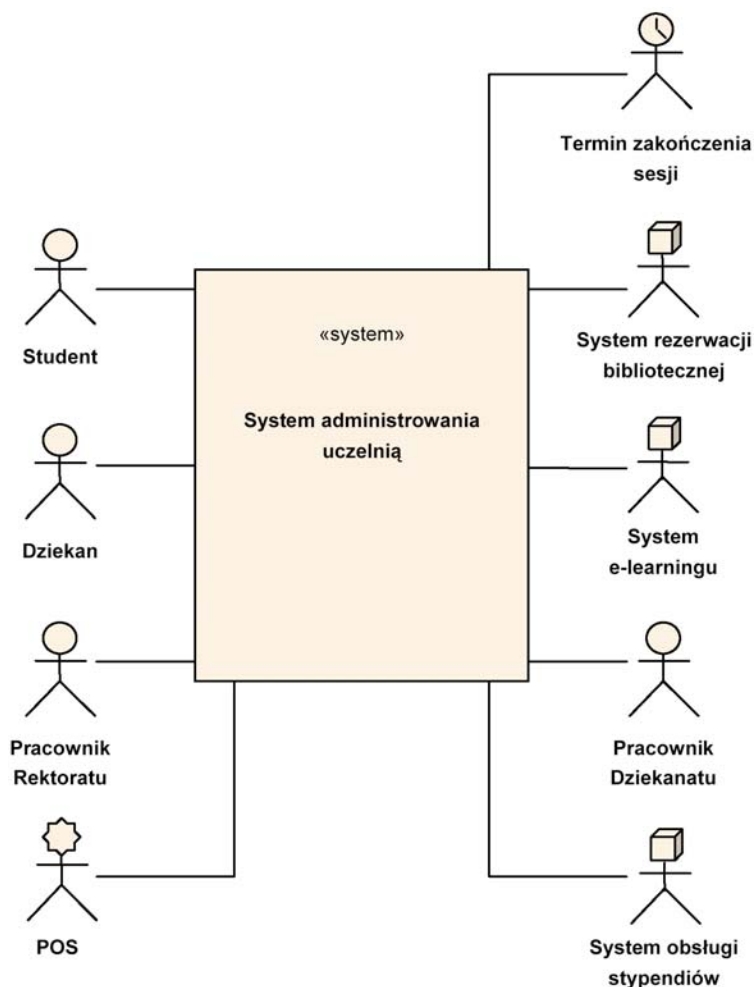


Diagram kontekstowy

W odniesieniu do diagramu przypadków użycia zastosować można znane z podejścia strukturalnego pojęcie **diagramu kontekstowego**. Stanowi on zestawienie aktorów będących w interakcji z danym systemem traktowanym w kategorii pojedynczego procesu. Przykład diagramu kontekstowego przedstawia rysunek 2.22.

Diagramy kontekstowe mogą okazać się pomocne w **identyfikacji** zbiorowości aktorów przed sporządzeniem pełnego DPU. Powyższy rysunek wykorzystuje ponadto urozmaicone stereotypy aktorów celem pełniejszego oddania otoczenia systemu. Do aktorów osobowych zaliczyć należy *Studenta*, *Pracownika rektoratu*, *Pracownika dziekanatu*, jak i samego *Dziekana*. *System administrowania uczelnią* współpracuje z kilkoma systemami zewnętrznymi, do których zaliczyć należy: *System rezerwacji bibliotecznej*, *System e-learningu* oraz *System obsługi stypendiów*. Bardzo istotny w odniesieniu do funkcjonowania uczelni jest aspekt czasu, którego elementami kulminacyjnymi są *Terminy*

Rysunek 2.22.*Diagram kontekstowy*

zakończenia sesji. W tym okresie występuje zwiększona aktywność dziekanatów i innych jednostek organizacyjnych, związana chociażby z rejestracją protokołów egzaminacyjnych i zaliczeniowych, przeprowadzaniem egzaminów dyplomowych i magisterskich wraz z wystawianiem dyplomów ukończenia studiów, czy też rozliczeniami z filiami głównej biblioteki uczelni i bibliotekami wydziałowymi. Natomiast w miarę upowszechniania się nowoczesnych form płatności jednym z najintensywniej wykorzystywanych urządzeń jest terminal POS (*Point of Sale*), który umożliwia regulowanie zobowiązań wobec uczelni z wykorzystaniem kart płatniczych.

Dokumentacja przypadków użycia

Diagramy przypadków użycia przedstawiają bardzo ogólny obraz systemu, stanowiąc punkt startowy jego tworzenia, lecz nie pozwalając na przedstawienie wielu istotnych informacji, które są niezbędne w realizacji dalszych faz cyklu życia systemu. Stąd

każdy przypadek użycia powinien być uzupełniony o stosowną dokumentację, charakteryzującą **scenariusze** tego przypadku użycia (ang. *use-case scenarios*). Ma to krytyczne znaczenie podczas modelowania systemów rozbudowanych.

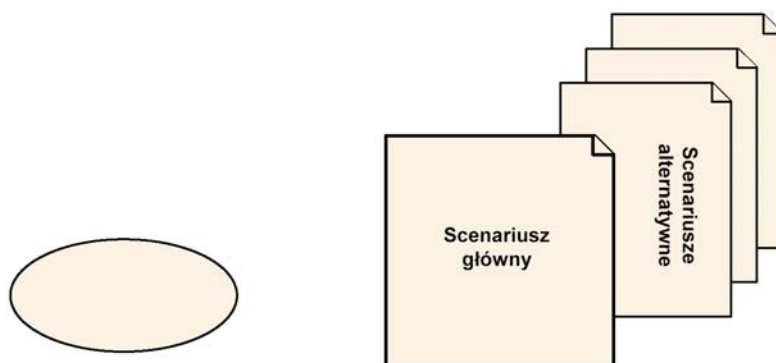


Scenariusz stanowi określony ciąg akcji dokumentujący zachowanie.

Dla danego przypadku użycia zawsze należy wyróżnić scenariusz **główny**. Dokumentacja przypadku użycia może ponadto zawierać scenariusze **alternatywne** (rysunek 2.23). Łącznie scenariusze te precyzyjnie opisują pełną funkcjonalność reprezentowaną przez dany przypadek użycia. W praktycznych zastosowaniach występują sytuacje zdeterminowane, charakteryzujące się niewystępowaniem alternatyw. Naturalnie w takich przypadkach opisuje się wyłącznie scenariusz główny.

Rysunek 2.23.

*Przypadek użycia
i jego scenariusze*



Istnieje szereg sposobów dokumentowania każdego przypadku użycia. Liczba uwzględnianych detali może wahać się w zależności od istotności danego przypadku użycia oraz ryzyka z nim związanego. Opisy te mogą przybrać postać:

- ♦ niesformalizowanego tekstu,
- ♦ formalnego tekstu strukturalnego,
- ♦ pseudokodu,
- ♦ tabeli (tabela 2.2).

Scenariuszowi głównemu odpowiada sekwencja interakcji wymienionych w wierszu „Główny przepływ zdarzeń” szablonu dokumentacji. Z kolei scenariusze alternatywne powstają na zasadzie kombinacji zdarzeń składających się na główny przepływ zdarzeń oraz zdarzeń wymienionych w wierszu „Alternatywne przepływy zdarzeń”.

Na podstawie propozycji szablonu stosowanego do tabelarycznego dokumentowania przypadku użycia, który to szablon przedstawia tabela 2.2, opracowano scenariusze przypadku użycia *Anuluj rezerwację*. Jest to jeden z wielu przypadków użycia systemu hotelowego. Mimo że jest to przypadek względnie nieskomplikowany, przedstawia

funkcjonalność, która musi koniecznie zostać zaimplementowana w systemie. Jak można zaobserwować na diagramie, ze wspomnianym przypadkiem użycia komunikują się aktorzy *Recepcjonista* oraz *Kierownik recepcji*. Dokumentacja szczegółowo opisuje główny scenariusz przypadku użycia oraz wskazuje alternatywne przepływy zdarzeń (tabela 2.3). Do ważnych elementów dokumentacji należą ponadto:

- ♦ warunek wstępny, który określa, że co najmniej jeden pokój lub apartament hotelowy musi być zarezerwowany;
- ♦ warunek końcowy, wskazujący, że po wykonaniu funkcjonalności związanej z tym przypadkiem użycia system odnotuje pokój lub (i) apartament jako dostępny.

Tabela 2.2. Szablon dokumentacji przypadku użycia

Nazwa:	Pełna nazwa przypadku użycia
Numer:	Numer identyfikacyjny przypadku użycia
Twórca:	Dane twórcy przypadku użycia, np. imię, nazwisko, stanowisko
Poziom ważności:	Określenie poziomu ważności przypadku z perspektywy użytkownika, np. niski, średni, wysoki
Typ przypadku użycia:	Określenie typu przypadku użycia z punktu widzenia jego złożoności oraz ważności dla zaspokojenia potrzeb użytkownika, np. ogólny/szczegółowy; niezbędny/istotny/przeciętnie istotny/moło istotny
Aktorzy:	Lista aktorów będących w związku z przypadkiem użycia
Krótki opis:	Krótką, ogólną charakterystyką przypadku użycia
Warunki wstępne:	Charakterystyka koniecznych warunków inicjujących przypadek
Warunki końcowe:	Charakterystyka stanu systemu po realizacji przypadku użycia
Główny przepływ zdarzeń:	Wypunktowana i scharakteryzowana lista przepływów zdarzeń zachodzących podczas realizacji przypadku użycia; scenariusz główny
Alternatywne przepływy zdarzeń:	Wypunktowana i scharakteryzowana lista możliwych, alternatywnych przepływów zdarzeń przypadku użycia
Specjalne wymagania:	Wypunktowana i scharakteryzowana lista dodatkowych zidentyfikowanych wymagań niefunkcjonalnych, które mogą być istotne przykładowo podczas projektowania czy kodowania
Notatki i kwestie:	Lista wszelkich komentarzy dotyczących przypadku użycia i lista pozostałych otwartych kwestii, które powinny zostać rozwiązane wraz z propozycjami osób, które mogłyby je rozwiązać

Źródło: Opracowanie własne na podstawie [RUP-2003]

Konsekwentne dokumentowanie poszczególnych przypadków użycia prowadzi do odnajdywania nieścisłości i przeoczeń powstałych podczas tworzenia wstępnej wersji diagramu przypadków użycia.

Tabela 2.3. Dokumentacja przypadku użycia „Anuluj rezerwację”

Nazwa:	Anuluj rezerwację
Numer:	5
Twórca:	Anna Krotoszyńska — Projektant
Poziom ważności:	Średni
Typ przypadku użycia:	Ogólny
Aktorzy:	Recepcjonista, Kierownik recepcji
Krótki opis:	Anulowanie istniejącej rezerwacji pokoju lub apartamentu
Warunki wstępne:	Co najmniej jeden pokój lub apartament hotelowy musi być zarezerwowany
Warunki końcowe:	System odnotowuje pokój lub (i) apartament jako dostępny
Główny przepływ zdarzeń:	1. Recepcjonista weryfikuje rezerwację, uruchamiając funkcję „Rezerwacje” 2. System wyświetla okno z informacjami o rezerwacjach (pokoje i apartamenty hotelowe) 3. Pracownik recepcji zaznacza rezerwację do anulowania i uruchamia funkcję „Anuluj rezerwację” 4. System wyświetla komunikat „Czy anulować zaznaczone rezerwacje?” 5. Pracownik recepcji potwierdza operację anulowania zaznaczonych rezerwacji 6. System potwierdza wykonanie operacji komunikatem „Anulowano wybrane rezerwacje” i odświeża ekran monitora
Alternatywne przepływy zdarzeń:	2a. System wyświetla komunikat „Brak rezerwacji” 3a. Pracownik recepcji rezygnuje z anulowania rezerwacji 3b. Jeśli podczas rezerwacji podany został adres e-mail, pracownik może wysłać do klienta pocztą elektroniczną informację o anulowaniu rezerwacji
Specjalne wymagania:	1. Wysoka niezawodność systemu 2. Czas przetwarzania operacji anulowania rezerwacji nie może przekroczyć 5 sekund
Notatki i kwestie:	Brak

Proces tworzenia diagramu przypadków użycia

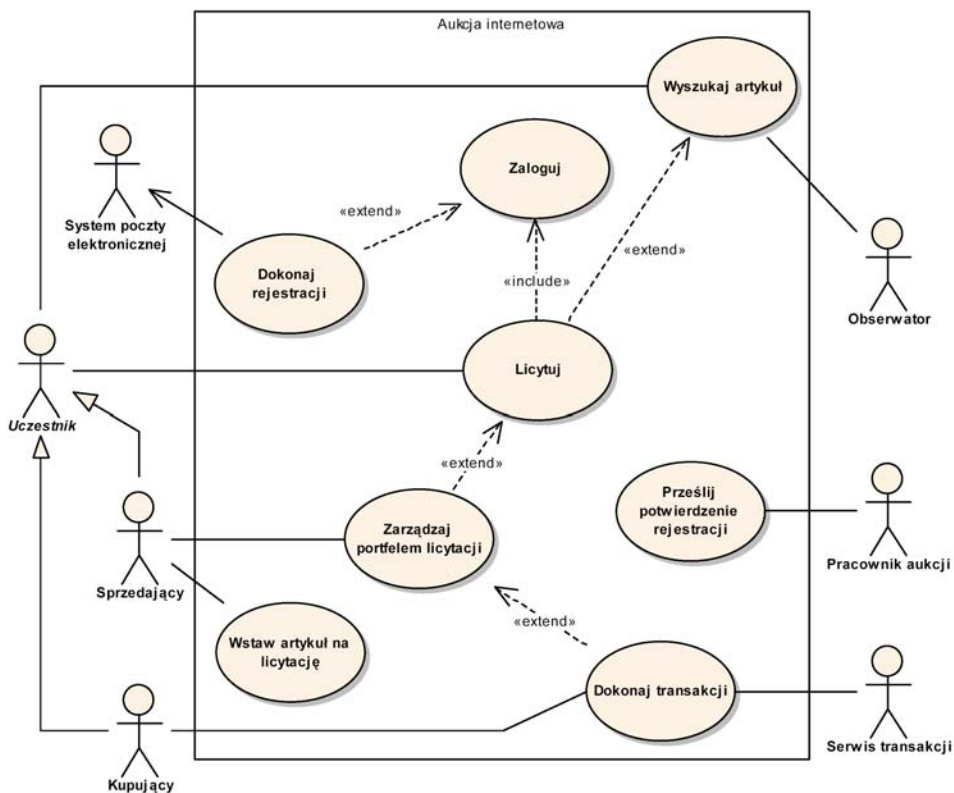
Proces tworzenia diagramu przypadków użycia jest procesem iteracyjnym. Kluczowe znaczenie przypisać należy następującym **etapom**:

1. identyfikacji aktorów,
2. opcjonalnemu opracowaniu diagramu kontekstowego,
3. identyfikacji przypadków użycia,

4. opracowaniu związków — w szczególności asocjacji,
5. wykorzystaniu wszystkich kategorii zaawansowanych do opracowania diagramu przypadków użycia,
6. udokumentowaniu przypadków użycia z wykorzystaniem szablonów.

Studium diagramu przypadków użycia

Omówione wcześniej (w punktach *Podstawowe kategorie pojęciowe oraz notacja graficzna* oraz *Zaawansowane składniki diagramu*) podstawowe oraz zaawansowane elementy DPU zawiera rysunek 2.24, przedstawiający system aukcji internetowej. Umożliwia on sprzedaż różnych artykułów w drodze licytacji. Aby licytacja mogła się odbyć, zarówno *Sprzedający*, jak i artykuł — który później staje się przedmiotem licytacji — muszą być zarejestrowane w systemie. Podczas licytacji system aukcji zatwierdza kwoty proponowane przez *Kupujących*. Jeśli podczas licytacji artykuł znajdzie nabywcę, *Kupujący* ustala z nim szczegóły płatności oraz przesyłki. W przeciwnym razie *Sprzedający* może usunąć artykuł z bazy systemu lub wystawić artykuł na nowej *Licytacji*.



Rysunek 2.24. System aukcji internetowej

Aktorzy *Kupujący* oraz *Sprzedający* są uszczegółowieniami abstrakcyjnego aktora *Uczestnik*. Oprócz wymienionych, na diagramie przypadków użycia systemu aukcji internetowej zaprezentowano czterech innych aktorów. I tak aktor *Serwis transakcji* reprezentuje zewnętrzny system informatyczny obsługujący transakcje. Aktor *Pracownik aukcji* wykorzystuje funkcjonalność związaną z potwierdzaniem rejestracji *Sprzedających*. Polega ona, między innymi, na pobraniu właściwych danych powstałych podczas rejestracji *Sprzedającego*, w tym kodu sprzedaży. Po wydrukowaniu odpowiedniego dokumentu jest on przesyłany tradycyjną pocztą na adres *Sprzedającego*. Kolejny aktor przedstawiony na diagramie, *Obserwator*, może wyłącznie wyszukiwać artykuły w katalogu aukcji. Nieosobowym aktorem jest *System poczty elektronicznej*. Rola ta jest powiązana z przypadkiem *Dokonaj rejestracji*, ponieważ system wysyła potwierdzenie rejestracji przez Internet.

Sprzedający korzysta z funkcjonalności aż siedmiu przypadków użycia. I tak może on:

- ♦ *Dokonać rejestracji* na aukcji;
- ♦ *Wstawić artykuł na licytację*;
- ♦ przeglądać ofertę aukcji poprzez *Wyszukiwanie artykułów* będących przedmiotem licytacji;
- ♦ *Logować się* do systemu, dzięki czemu po identyfikacji może on aktywnie uczestniczyć w licytacjach i proponować kwoty — a zatem *Licytować*;
- ♦ *Zarządzać portfelem licytacji*;
- ♦ *Dokonywać transakcji* przy współudziale nieosobowego aktora *Serwis transakcji*.

Z kolei *Kupujący* może korzystać z funkcjonalności wszystkich przypadków użycia udostępnionych *Sprzedającemu* z wyłączeniem przypadków użycia *Zarządzaj portfelem licytacji* oraz *Wstaw artykuł na aukcję*.

Podstawowe pojęcia

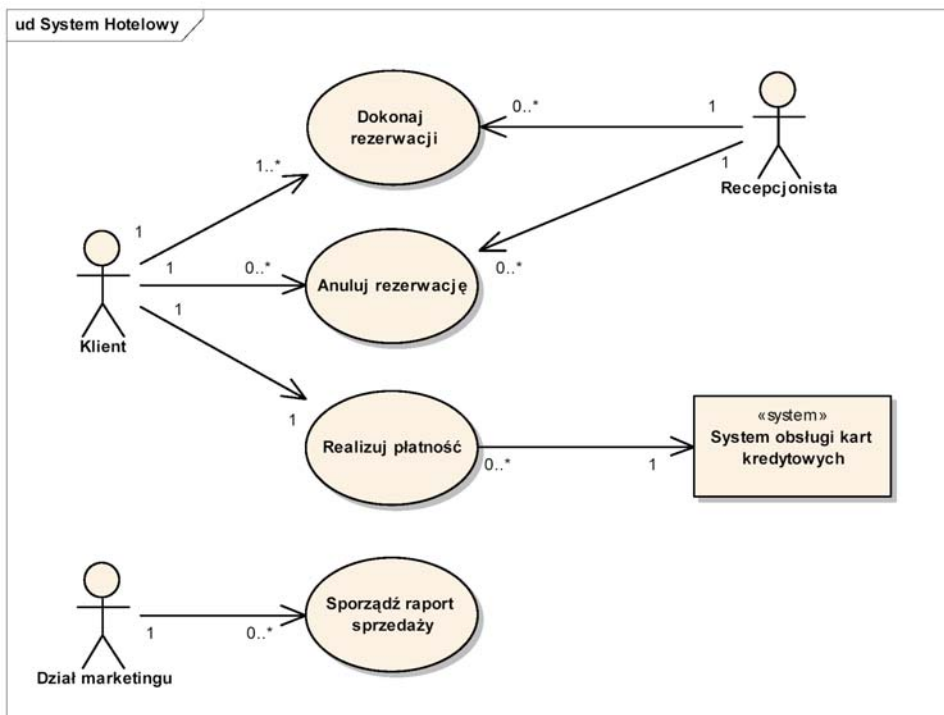
Aktor	Liczebność
Nieosobowy	Miejsce rozszerzania
Osobowy	Nawigacja
Diagram kontekstowy	Nazwa ścieżkowa
Diagram przypadków użycia	Notatka
Definicja	Pakiet
Proces tworzenia	Przeływ zdarzeń
Dziedzina przedmiotowa	Alternatywny
Granica obszaru zastosowań	Główny

Przypadek użycia	Stereotyp
Biznesowy	Udziałowiec systemu
Rozszerzający	Współdziałanie
Rozszerzany	Wymaganie
Systemowy	Związek
Typu CRUD	Asocjacja
Zawierający	Realizacja
Zawierany	Uogólnienie
Scenariusz	Zależność
Scenariusz główny	Zależność rozszerzania
Scenariusze alternatywne	Zależność zawierania

Pytania i zadania

1. Diagramy przypadków użycia pełnią bardzo ważną rolę w procesie tworzenia systemu. Jak jest uzasadnienie ich stosowania?
2. Zdefiniuj i zinterpretuj następujące pojęcia:
 - ♦ diagram przypadków użycia,
 - ♦ przypadek użycia,
 - ♦ aktor,
 - ♦ związek.
3. Wymień podstawowe kategorie aktorów w diagramach przypadków użycia. Podaj przykłady.
4. Wyjaśnij zależności pomiędzy pojęciami:
 - ♦ asocjacja,
 - ♦ związek.
5. Przedstaw podstawowe przypadki użycia związane ze studentem w systemie obsługi dziekanatu.
6. Naszkicuj kompletny diagram przypadków użycia opisujący funkcjonowanie dziekanatu. Szczególną uwagę poświęć identyfikacji innych aktorów.
7. W jaki sposób można określić zakres systemu na diagramie przypadków użycia? Zaproponuj sposób zaprezentowania na tym diagramie podziału systemu na podsystemy.

8. Wyjaśnij istotę zależności rozszerzania, zawierania oraz uogólnień. Czym różni się zależność rozszerzania od zależności zawierania?
9. Wskaż kilka sposobów specyfikowania miejsc rozszerzania na diagramach przypadków użycia. Który z nich uznałbyś za najefektywniejszy i dlaczego?
10. W jakich sytuacjach zasadne jest zastosowanie zależności zawierania?
11. Jakie kategorie pojęciowe diagramów przypadków użycia można opisywać z wykorzystaniem związku uogólnienia? Podaj przykłady.
12. Uzasadnij znaczenie stereotypów w kontekście specyfikowania aktorów.
13. Wyjaśnij zależności pomiędzy pojęciami:
 - ♦ realizacja,
 - ♦ współdziałanie.
14. Wymień wszystkie elementy zaprezentowane na diagramie zamieszczonym na rysunku 2.25:



Rysunek 2.25. Rezerwacje w systemie hotelowym

15. Do jakiej kategorii modelowania diagramów przypadków użycia odnoszą się pojęcia liczebności oraz nawigacji?
16. Jakie dodatkowe informacje wnoszą do diagramu nazwy ścieżkowe?

17. Rozwiń skrót CRUD.
18. Uzasadnij zastosowanie diagramów kontekstowych. Wyszukaj w literaturze nazwę metodyki, w której po raz pierwszy wprowadzono pojęcie diagramu kontekstowego oraz nazwisko jej autora.
19. Jakie są wspólne elementy diagramów kontekstowych i pełnego DPU?
20. Wyjaśnij pojęcie scenariusza. Wymień rodzaje scenariuszy.
21. W jakich postaciach dokumentować można przypadki użycia?
22. Opracuj dokumentację dla dwóch wybranych przypadków użycia z diagramu przedstawiającego system obsługi hurtowni (rysunek 2.6).
23. Który wiersz tabelarycznego szablonu dokumentacji przypadku użycia odpowiada scenariuszowi głównemu?
24. Opracuj dokumentację przypadku użycia typu CRUD niezbędnego do zarządzania kontami klientów w systemie bankowym.
25. Korzystając z dokumentacji opracowanej zgodnie z poleceniem z poprzedniego punktu, zaprojektuj elementarne, specjalizowane przypadki użycia reprezentujące funkcjonalność niezbędną do zarządzania kontami klientów w systemie bankowym.
26. Wyjaśnij i uzasadnij przebieg procesu tworzenia diagramu przypadków użycia.

Rozdział 3.

Diagram klas

Znaczenie diagramów klas

Diagramy klas, obok diagramów przypadków użycia, zaliczyć należy do najczęściej stosowanych i zarazem kluczowych rodzajów diagramów w języku UML. Są one ponadto powszechnie rozpoznawanym elementem najpopularniejszych metodyk i technik obiektowych. Przedstawiają statykę systemu, stanowiąc przede wszystkim podstawę przyszłej **obiektywnej bazy danych**. Główne elementy diagramów klas mają znaczny wpływ na układ i zawartość innych diagramów UML.



Diagram klas to graficzne przedstawienie statycznych, deklaratywnych elementów dziedziny przedmiotowej oraz związków między nimi.

Podstawowe kategorie pojęciowe oraz notacja graficzna

Paradygmat podejścia obiektowego opiera się na zasadniczych pojęciach **obiektu** (ang. *object*) i klasy.



Obiektem jest każdy byt — pojęcie lub rzecz — mający znaczenie w kontekście rozwiązywania problemu w danej dziedzinie przedmiotowej.

Wszystko, co wiadomo o obiekcie, jest reprezentowane przez wartości atrybutów — czyli cech statycznych tego obiektu. Natomiast zachowanie obiektu wyrażone jest w operacjach określających usługi, które oferuje obiekt. Zainicjowanie operacji skutkuje użytkowaniem i (lub) modyfikacją danych reprezentowanych przez wartości atrybutów. Obiekty charakteryzują się unikatową tożsamością, jest więc możliwe ich jednoznaczne

wyodrębnienie ze zbiorowości innych obiektów, nawet gdy wszystkie wartości przechowywane przez te obiekty są identyczne. W istocie obiekt jest wyróżniany poprzez samo istnienie, a nie cechy opisowe.

Jednocześnie można powiedzieć, że dowolny obiekt jest instancją abstrakcyjnego pojęcia, jakim jest **klasa** (ang. *class*) obiektu. Podstawę identyfikacji klasy stanowią grupy obiektów charakteryzujące się:

- ♦ identyczną strukturą danych, tj. takimi samymi **atrybutami**;
- ♦ identycznym zachowaniem, tj. takimi samymi **operacjami**;
- ♦ identycznymi związkami;
- ♦ identycznym znaczeniem w określonym kontekście.



Klasa jest uogólnieniem zbioru obiektów, które mają takie same atrybuty, operacje, związki i znaczenie.

Każda klasa zawiera więc zestaw informacji istotnych z punktu widzenia kontekstu systemu. Zestaw atrybutów, operacji i związków z innymi klasami może być szerszy lub węższy w zależności od wymagań dotyczących przyszłego systemu. W diagramach tych klasę standardowo przedstawia się jako prostokąt złożony z **trzech sekcji**:

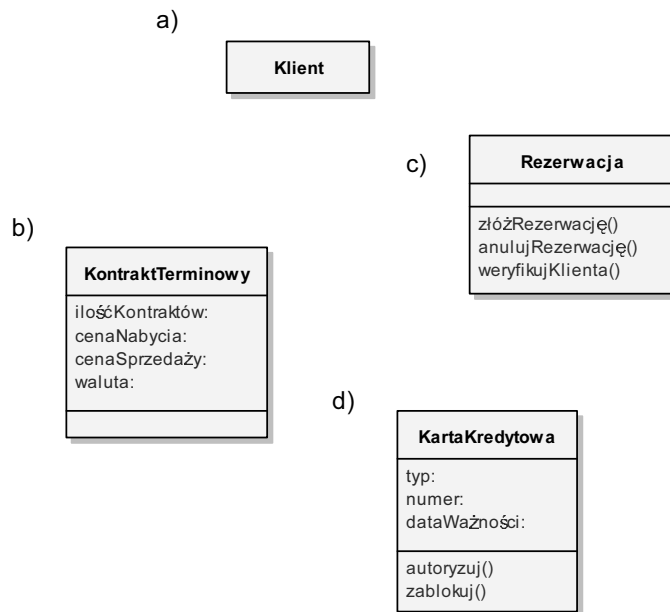
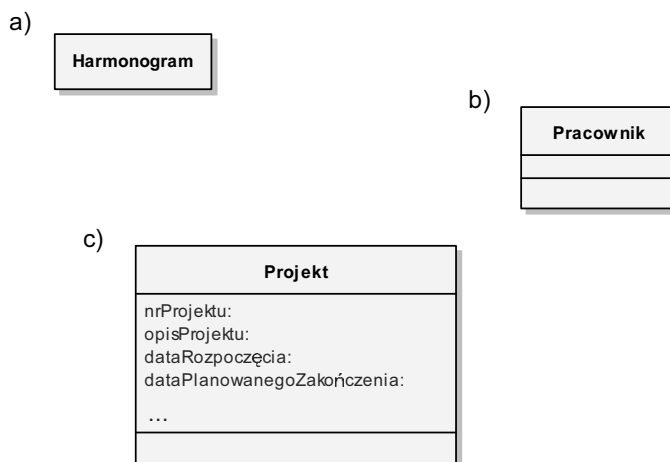
- ♦ nazwy klasy,
- ♦ zestawu atrybutów,
- ♦ zestawu operacji.

Rysunek 3.1 ujmuje możliwe **kombinacje** graficznej prezentacji klas:

- ♦ sama nazwa klasy,
- ♦ nazwa klasy z zestawem atrybutów,
- ♦ nazwa klasy z zestawem operacji,
- ♦ nazwa klasy z zestawem atrybutów i operacji.

W związku z różnorodnością możliwych sposobów specyfikowania klas należy wyróżnić następujące opcje ich prezentacji graficznej:

- a) sama nazwa klasy umieszczona w **jednosekcyjnym** bloku oznacza, że sekcje atrybutów i operacji zostały wyspecyfikowane, lecz nie są w sposób jawny zamieszczone na diagramie klas (rysunek 3.2a);
- b) alternatywnie, klasę przedstawia się jako blok złożony z trzech sekcji z nazwą w pierwszej sekcji i **niewyspecyfikowanymi** atrybutami i operacjami (rysunek 3.2b);
- c) jeśli liczba atrybutów lub operacji jest większa, to ich wyliczanie w odpowiednich sekcjach można przerwać **wielokropkiem**, co należy interpretować jako przypisanie klasie jeszcze innych atrybutów i operacji — niewymienionych bezpośrednio w specyfikacji (rysunek 3.2c).

Rysunek 3.1.*Przykłady opisu klas***Rysunek 3.2.***Opcje specyfikacji klas*

Liczbę sekcji w każdej klasie można zwiększać, dodając np. sekcje zawierające zobowiązania (punkt *Zaawansowane składniki diagramu*) bądź wyjątki. Sekcje te można dodawać opcjonalnie w przypadkach, gdy jest to niezbędne dla podniesienia precyzji diagramu klas, jednak nadmierne stosowanie dodatkowych sekcji może obniżyć przejrzystość diagramu.

Klasy obiektów — podobnie jak inne elementy języka UML — powiązane są różnego rodzaju związkami. W diagramach klas powszechnie stosuje się wszystkie cztery **rodzaje związków** języka UML, czyli:

- ♦ asocjacje,
- ♦ uogólnienia,
- ♦ zależności,
- ♦ realizacje.

Podstawowe cechy związku asocjacji w ujęciu diagramu klas zostały scharakteryzowane poniżej, podczas gdy pozostałe rodzaje związków omówiono w punkcie *Zaawansowane składniki diagramu*.

Asocjacja

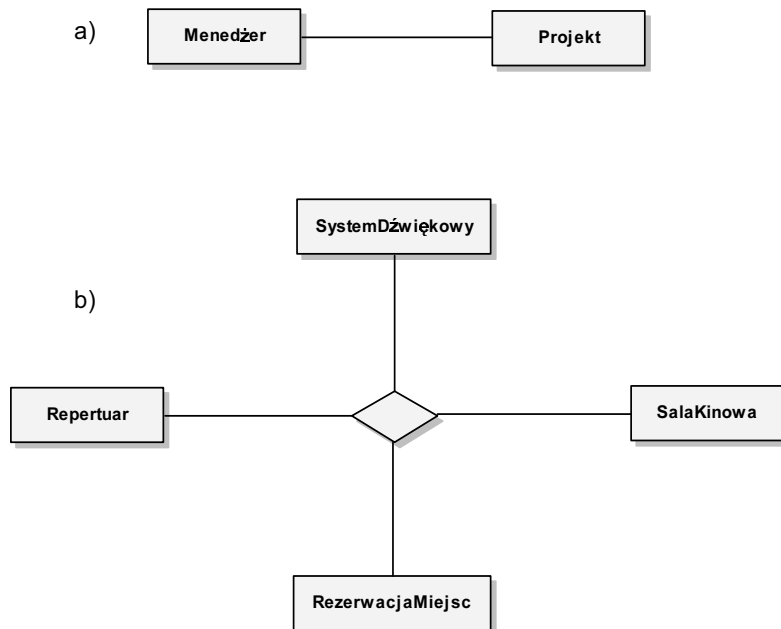
Głównym rodzajem związków między klasami są **asocjacje**. Definicję asocjacji przedstawiono w rozdziale drugim. Asocjacja opisuje zbiór powiązań pomiędzy obiektami, tak jak klasa obiektu — zbiór obiektów stanowiących jej instancje. Wyróżnia się dwa rodzaje asocjacji:

- ♦ binarne,
- ♦ n-arne.

W praktyce dominują **asocjacje binarne** (ang. *binary associations*), dlatego poświęcono im główną część rozważań na temat związków asocjacji. Rysunek 3.3 przedstawia:

- ♦ asocjację binarną harmonogramowania projektu (rysunek 3.3 a).
- ♦ asocjację n-arną systemu kinowego (rysunek 3.3 b).

Rysunek 3.3.
*Asocjacje
pomiędzy klasami*



Asocjacje n-arne można również nazwać asocjacjami n-argumentowymi, a więc trój-argumentową, czteroargumentową itd. Zilustrowane na rysunku 3.3 asocjacje opisują w sposób elementarny związki między klasami. Pełna semantyczna interpretacja związku wymaga wprowadzenia szeregu dodatkowych elementów opisu. Tak więc asocjację można dokładnie sprecyzować poprzez zdefiniowanie jej następujących **cech**:

- ♦ nazwy,
- ♦ ról powiązanych klas,
- ♦ nawigacji,
- ♦ liczebności,
- ♦ agregacji.

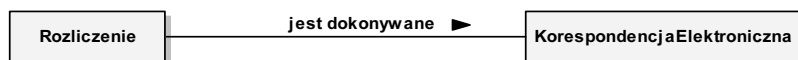
Wymienione cechy mają charakter podstawowy. Możliwe jest rozszerzanie ich listy w zależności od specyfiki projektu i potrzeb zespołu projektowego.

Nazwy asocjacji

Istnieją różne sposoby charakteryzowania asocjacji. Podając nazwę asocjacji, określa się istotę danego związku. W celu uniknięcia nieporozumień przy nazwie asocjacji można podać kierunek jej odczytu. Istnieje szereg możliwości oznaczania **nazw** asocjacji. Mogą one być:

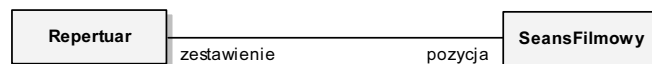
- ♦ nienazwane (rysunek 3.3);
- ♦ nazwane, z opcjonalnym zamieszczeniem znacznika wskazującego kierunek interpretacji asocjacji (rysunek 3.4);

Rysunek 3.4.
Asocjacja nazwana



- ♦ scharakteryzowane poprzez role klas pełnione w asocjacji (rysunek 3.5);

Rysunek 3.5.
Asocjacja z wyspecyfikowanymi rolami



- ♦ nazwane i równocześnie scharakteryzowane przez role (rysunek 3.6).

Rysunek 3.6.
Asocjacja nazwana z wyspecyfikowanymi rolami



Rysunek 3.4 wskazuje, że poszczególne transakcje finansowe reprezentowane przez obiekty klasy *Rozliczenie* powiązane są z odpowiednimi dokumentami elektronicznymi, które są instancjami klasy *KorespondencjaElektroniczna*. Dokumenty te stanowią podstawę dokonania transakcji.

Role

Każdą asocjację można również interpretować dwustronnie poprzez podanie **ról** (ang. *roles*) pełnionych przez powiązane ze sobą klasy. Rola asocjacji w związku binarnym jest powinnością pełnioną przez jedną klasę obiektu wobec drugiej klasy. W asocjacji n-arnej role można przypisać każdej z powiązanych klas.

Nazwy ról umieszcza się po obydwu stronach asocjacji. Rola spełniana przez daną klasę lokowana jest bezpośrednio przy klasie określonej. Przykład asocjacji opisanej wyłącznie za pomocą ról przedstawiono na rysunku 3.5. Kompleksowy opis asocjacji zawierający jej nazwę i role zaprezentowano na rysunku 3.6.

Poszczególne seanse, będące instancjami klasy *SeansFilmowy*, powiązane są z bieżącym *Repertuarem* kinowym. Repertuar w istocie jest sumarycznym zestawieniem seansów — a więc dany seans stanowi jedną z pozycji repertuaru.

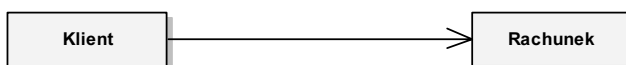
Na powyższym rysunku zaprezentowano wycinek diagramu klas systemu harmonogramowania projektów informatycznych. *Pracownik* działu IT firmy, z którym bezpośrednio powiązano *Projekt*, automatycznie klasyfikowany jest jako jego kierownik. Projekt potraktowano w kategoriach zlecenia zarządzanego przez kierownika.

Nawigacja

Do wykonywania operacji na obiektach poszczególnych klas niezbędne jest przesyłanie komunikatów pomiędzy nimi. Dla każdej asocjacji istniejącej pomiędzy klasami komunikowanie domyślnie odbywa się w obydwu kierunkach asocjacji. Jest to nawigacja dwukierunkowa. W praktyce występują sytuacje, w których wskazanie **kierunku nawigacji** zwiększa efektywność komunikowania się (rysunek 3.7). Mamy wtedy do czynienia z nawigacją jednokierunkową.

Rysunek 3.7.

Asocjacja z kierunkiem nawigacji

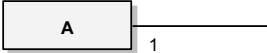
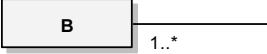
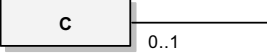
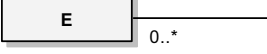
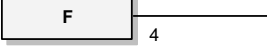
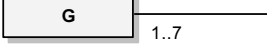
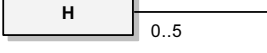
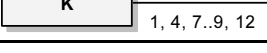


Asocjacja ze wskazanym kierunkiem nawigacji nie oznacza bezwzględnego zablokowania przejścia pomiędzy klasami w dwu kierunkach. Dwukierunkowość może być bowiem osiągnięta w sposób mniej efektywny dzięki wykorzystaniu asocjacji pomiędzy innymi, wzajemnie powiązanymi klasami, zamieszczonymi na diagramie pomiędzy klasą docelową a źródłową.

Liczebność

Klasom odpowiadają określone liczby obiektów. **Liczebność** to specyfikacja zakresu dopuszczalnej liczby obiektów danej klasy biorących udział w danym związku. A zatem jest to określenie liczby obiektów jednej klasy, które wiążą się z jednym obiektem drugiej klasy znajdującym się po przeciwnej stronie tej samej asocjacji. Tabela 3.1 przedstawia notację liczebności asocjacji stosowaną w języku UML.

Tabela 3.1. Oznaczanie liczebności asocjacji

Oznaczenie	Opis	Przykład
1	dokładnie jeden	
1..*	jeden lub wiele	
0..1	zero lub jeden	
*	wiele	
0..*	zero lub wiele	
n	dokładnie n ($n > 1$)	
1..n	od 1 do n	
0..n	od 0 do n	
n..m	od n do m ($n, m > 1$)	
n..*	więcej niż n	
n, m, o..p, q	liczebność złożona	

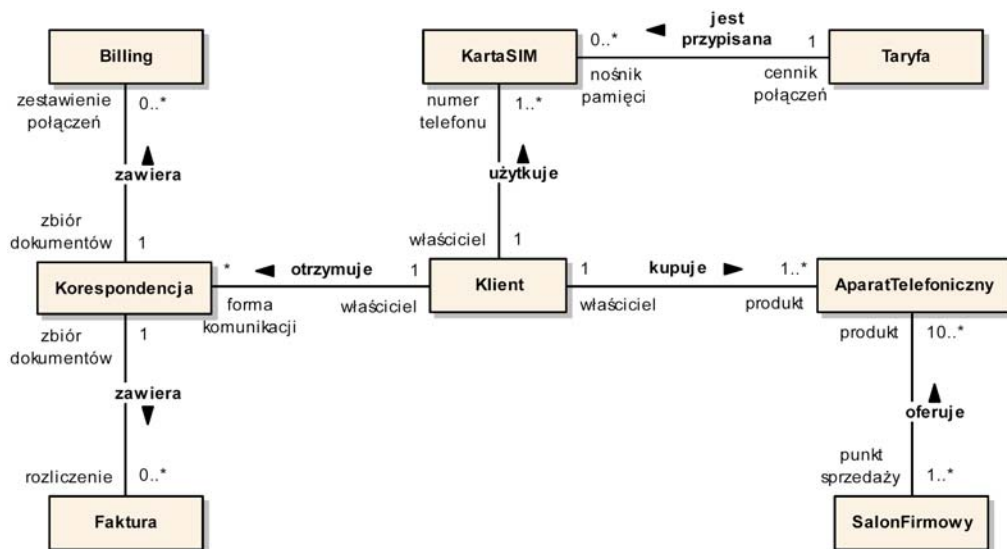
Poprzez n rozumie się określoną, znaną liczbę naturalną. Natomiast w przypadku chęci zasygnalizowania, że w określonej asocjacji bierze udział znaczna, lecz nieznaną liczbą obiektów, używa się symbolu gwiazdki (*).

Rysunek 3.8 przedstawia diagram klas ze wskazaniem liczebności asocjacji.

Interpretacja liczebności asocjacji polega na każdorazowym rozpatrzeniu sytuacji, gdy po jednej jej stronie znajduje się jeden obiekt, natomiast liczba obiektów, które odpowiadają mu w drugiej klasie, jest ustalana. I tak wskazana na rysunku 3.8 asocjacja *kupuje* łącząca klasy *Klient* i *AparatTelefoniczny* oznacza, że jeden *Klient* może kupić jeden lub wiele *AparatówTelefonicznych*, natomiast jeden *AparatTelefoniczny* może być zakupiony tylko przez jednego *Klienta*.

Agregacja

Kolejną cechą asocjacji jest agregacja (ang. *aggregation*), która opisuje związek całość-część pomiędzy klasami. W rzeczywistości istnieje wiele złożonych obiektów składających się z części, które w połączeniu stanowią spójną całość. Wyróżnia się dwa rodzaje agregacji:



Rysunek 3.8. Zastosowanie liczebności asocjacji

- ♦ agregację całkowitą — kompozycję;
- ♦ agregację częściową.

Dla wymienionych pojęć używa się alternatywnych nazw. Kompozycja nazywana jest również agregacją silną lub składową, natomiast agregacja częściowa — agregacją słabą lub współdzieloną. Agregacja całkowita jest oznaczona przez wypełniony romb, natomiast częściowa przez pusty romb.

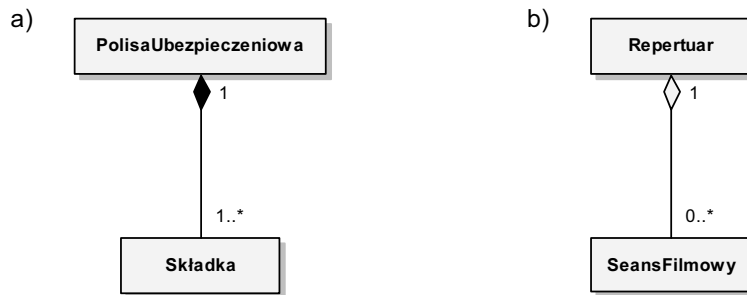
W obydwu rodzajach agregacji występują dwa podstawowe pojęcia:

- ♦ agregat, czyli obiekt stanowiący całość;
- ♦ segment, czyli część.

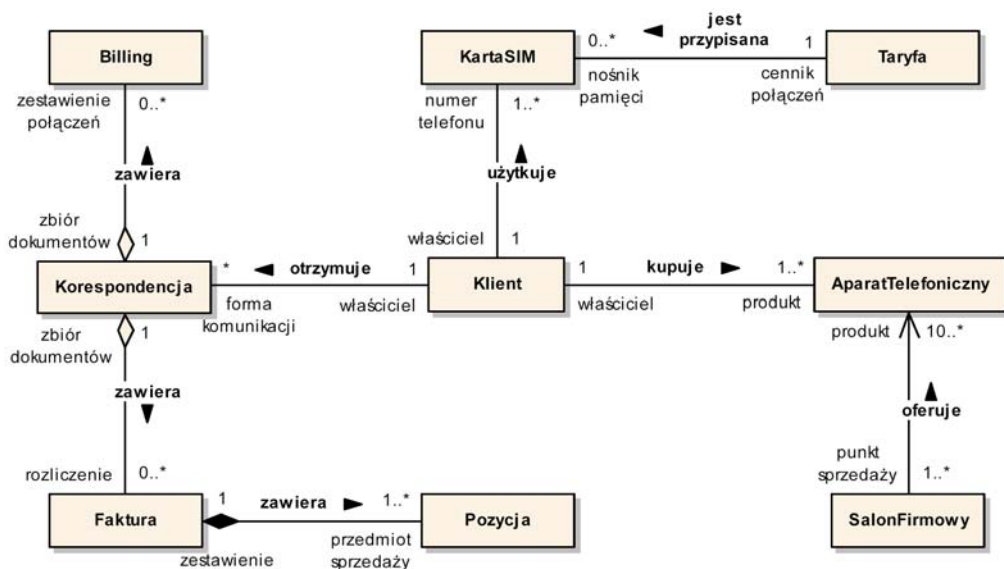
W przypadku agregacji całkowitej obiekty-segmenty będące częściami agregatów nie mogą samodzielnie i niezależnie funkcjonować. Usunięcie agregatu powoduje **automatyczną** likwidację wszystkich segmentów będących jego częściami. Natomiast agregacja częściowa wskazuje na asocjację, w której usunięcie obiektu będącego agregatem nie powoduje likwidacji obiektów będących jego częściami, czyli obiektów-segmentów. Obiekty współdzielone mogą zatem funkcjonować **samodzielnie**, niezależnie od agregatu. Rysunek 3.9 przedstawia agregację całkowitą (rysunek 3.9a) i agregację częściową (rysunek 3.9b).

Rysunek 3.9 prezentuje elementy obiektowych baz danych systemów informatycznych firmy ubezpieczeniowej oraz kompleksu kinowego. Na jego podstawie można stwierdzić, że *Składka* jest niezbędnym elementem *PolisyUbezpieczeniowej* i nie może w żadnym wypadku poprawnie funkcjonować w oderwaniu od niej. Potraktowanie *SeansówFilmowych* w kategoriach agregacji częściowej wskazuje jednak, że są one integralną częścią *Repertuaru*, ale przykładowo w razie jego zastąpienia nie zostaną utracone, lecz mogą zostać zaadaptowane do jego następcy.

Rysunek 3.9.
Agregacja całkowita
a agregacja częściowa



Rysunek 3.10 przedstawia diagram klas zaprezentowany na rysunku 3.8, rozbudowany o dodatkowe kategorie modelowania, w tym agregacje.



Rysunek 3.10. Diagram klas z uwzględnieniem agregacji

Na pierwszy rzut oka wartość informacyjna przekazywana przez uaktualniony diagram nie jest zasadniczo zwiększona. Wciąż centralnym elementem bazy danych pozostaje klient sieci telefonii komórkowej, o którym gromadzone są informacje w postaci obiektów klasy *Klient*. Do każdego zarejestrowanego *Klienta* przyporządkowano przynajmniej jedną *KartęSIM*, którą *Klient* w praktyce użytkuje. *Taryfa* wykorzystywana przez *Klienta* przypisywana jest oddzielnie do każdej *KartySIM*, aby w przypadku posiadania przez danego *Klienta* kilku *AparatówTelefonicznych* w danej sieci umożliwić mu niezależne stosowanie różnych *Taryf* — na przykład taryfy o niskim stałym abonamencie dla telefonu wykorzystywanego prywatnie, natomiast dla telefonów służbowych tej charakteryzującej się wysokim abonamentem i niskimi kosztami zmiennymi. Mogą istnieć *KartySIM* bez przyporządkowanej *Taryfy* — sytuacja ta ma miejsce chociażby w przypadku rozwiązania umowy przez *Klienta*. Zważywszy, że najczęstszym sposobem rozpoczęcia korzystania z usług danej sieci telefonii komórkowej jest transakcja

wiązana — kupno nie tylko *KartySIM*, lecz także zawierającego ją *AparatuTelefonicznego* — sieć dla celów marketingowych i księgowych przechowuje informacje o dostępnych modelach *AparatówTelefonicznych* oraz ich przypisaniu do własnych *Klientów*. Sprzedaż dokonywana jest przez *SalonyFirmowe*. Wskazanie na uaktualnionym diagramie kierunku nawigacji pomiędzy klasami *SalonFirmowy* oraz *AparatTelefoniczny* informuje, że do odpowiedzialności *SalonuFirmowego* należy wskazanie wszystkich sprzedanych przez tę placówkę *AparatówTelefonicznych*, natomiast zależność ta nie funkcjonuje w drugą stronę — na podstawie danych reprezentowanych przez klasę *AparatTelefoniczny* nie można stwierdzić, który salon dokonał sprzedaży. Jednocześnie *SalonFirmowy* musi oferować minimum 10 *AparatówTelefonicznych*.

Rysunek 3.10 uporządkowuje ponadto kwestie związane z *Korespondencją* docierającą do *Klienta*. *Klient* otrzymuje wiele zestawów dokumentów traktowanych jako obiekty klasy *Korespondencja*. W praktyce dokonywane jest to okresowo, najczęściej co miesiąc. Opisywany diagram klas jednoznacznie wskazuje, że elementem takiej *Korespondencji* standardowo jest *Billing*, a zatem zestawienie połączeń, oraz *Faktura*. Liczebności dotyczące opisywanych asocjacji wskazują, że pojedyncza *Korespondencja* może zawierać kilka *Billingów* bądź *Faktur*. Przykładowo ma to miejsce, jeśli *Klient* posiada w danej sieci kilka aktywnych *AparatówTelefonicznych* rozliczanych oddzielnie. W przypadku wprowadzonych do systemu druków okolicznościowych *Faktury* oraz zestawienia połączeń nie są oczywiście dołączane, stąd wskazanie tego związku jako opcjonalnego w postaci liczebności 0..*. Usunięcie z systemu zapisów dotyczących przesłanej *Korespondencji* pociąga za sobą jednocześnie usunięcie wpisów dotyczących *Billingów* i *Faktur*. Ponieważ *Faktura*, poza wykonanymi połączeniami, może opiewać także na inne produkty bądź usługi (cena aparatu telefonicznego, koszt aktywacji, opłata za modyfikację planu taryfowego itd.), jej poszczególne pozycje przechowywane są w obiektach osobnej klasy *Pozycja*. Warto zaznaczyć, że *Pozycja* powiązana jest z *Fakturą* za pomocą silnej odmiany związku agregacji (kompozycji), co wskazuje, iż usunięcie z systemu wpisu dotyczącego *Faktury* zlikwiduje wszystkie jej części reprezentowane przez obiekty klasy *Pozycja*.

Zaawansowane składniki diagramu

Diagram klas charakteryzuje się znaczną liczbą zaawansowanych elementów i koncepcji modelowania. Obejmują one:

- ♦ rodzaje diagramów klas;
- ♦ zobowiązania;
- ♦ widoczność;
- ♦ atrybuty i operacje statyczne;
- ♦ nazwy klas, atrybutów i operacji;
- ♦ notację atrybutów i składnię operacji;
- ♦ klasy asocjacyjne;

- ♦ asocjacje zwrotne i wielokrotne;
- ♦ kwalifikację;
- ♦ uogólnienia, klasy abstrakcyjne oraz konkretne;
- ♦ zależność;
- ♦ realizację.

Rodzaje diagramów klas

Diagramy klas, ze względu na znaczną liczbę elementów, jakie potencjalnie mogą na nich wystąpić, cechują się zróżnicowanym poziomem szczegółowości. W celu uzyskania odpowiedniej efektywności oraz skuteczności procesu tworzenia SI należy rozróżnić ściśle powiązane ze sobą dwa **poziomy tworzenia** diagramów klas:

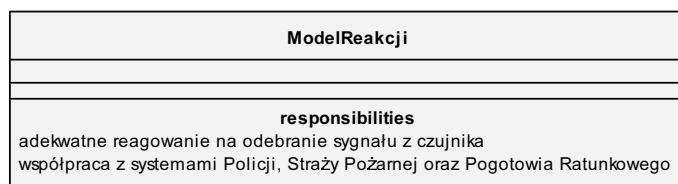
- ♦ poziom konceptualny,
- ♦ poziom implementacyjny.

Pomiędzy wyróżnionymi poziomami możliwe są rozwiązania pośrednie, będące wynikiem kolejnych iteracji tworzenia systemu. **Konceptualny diagram klas** zawiera wyłącznie podstawowe elementy, cechując się przystępnością nazewnictwa klas, atrybutów i operacji. Jest zrozumiały dla użytkownika i wynika bezpośrednio z analizy dziedziny przedmiotowej. Z kolei **implementacyjny diagram klas** jest stopniowo wzbogacany o elementy opisu niezbędne dla prawidłowej specyfikacji modelu, takie jak typy danych, zobowiązania, widoczność, statyczność, klasy asocjacyjne, kwalifikacje, uogólnienia, zależności czy też realizacje. Dobór tych elementów modelu jest uzależniony od specyfiki danego projektu. Klasy na poziomie implementacyjnym można dodatkowo oznaczyć stereotypem «*implementation class*».

Zobowiązania

Elementy techniki diagramów klas mogą posłużyć do określenia **zobowiązań** (ang. *responsibilities*) klas, poprzedzającego opracowanie pełnego projektu. Najczęściej możliwość tę wykorzystuje się w celu wypracowania koncepcji realizacji scenariuszy przypadków użycia, głównie za pomocą diagramów interakcji (rozdziały 6. – 10.). Zobowiązania klasy można nakreślić na podstawie wyników modelowania biznesu czy też definiowania wymagań systemu. A zatem zobowiązanie jest kontraktem lub odpowiedzialnością klasy. Klasę *ModelReakcji* zawierającą sekcję zobowiązań przedstawia rysunek 3.11.

Rysunek 3.11.
*Sekcja zobowiązań
w klasie*



System nadzoru pomieszczeń przechowuje w klasie *ModelReakcji* operacje, które są wykonywane w przypadku odebrania przez jeden z czujników zainstalowanych w monitorowanych pomieszczeniach sygnału wskazującego na zagrożenie. Innym zobowiązaniem zamieszczonej na rysunku klasy jest automatyczne informowanie odpowiednich służb publicznych o zaistniałym zagrożeniu.

Specyfikacja zobowiązań pełni rolę **pomocniczą** w procesie projektowania diagramów klas, gdyż dostarcza wysokopoziomowego opisu zadań poszczególnych klas. W ten sposób, unikając koncentrowania się na wyszczególnianiu obszernej listy atrybutów i operacji danej klasy, twórca systemu może rozważyć, czy nie jest ona obciążona nadmiarową listą. Klasy ze zobowiązaniami są w naturalny sposób przekształcane w diagram klas z pełną specyfikacją atrybutów i operacji. Klasę, której podstawową zawartość stanowią zobowiązania oraz lista współpracujących klas, nazywa się **kartą CRC** (*Class-Responsibility-Collaboration card*).

Widoczność

Istnieją zróżnicowane wymagania względem dostępu do poszczególnych atrybutów i operacji różnych klas w systemie. W systemie musi być zapewniona możliwość pełnej ochrony niektórych danych czy też zapewnienia ich dostępności tylko dla pewnej części obiektów innych klas składających się na statykę danej aplikacji. Uwarunkowania te spełniane są przez cechę **widoczność** (ang. *visibility*), która oznacza poziom dostępności atrybutów i operacji dla innych klas. Tabela 3.2 przedstawia stosowane w języku UML poziomy widoczności.

Tabela 3.2. *Poziomy widoczności*

Poziom	Symbol	Charakterystyka
Publiczny	+	obiekty wszystkich klas w systemie mają dostęp do atrybutu lub operacji
Prywatny	–	tylko obiekty danej klasy mają dostęp do atrybutu lub operacji
Chroniony	#	wyłącznie obiekty klas dziedziczących z danej klasy mają dostęp do atrybutu lub operacji
Pakietowy	~	tylko składowe pakietu, do którego dana klasa należy, mają dostęp do atrybutu lub operacji

Istnieje reguła wskazująca, by atrybuty klas definiować jako prywatne w celu minimalizacji zagrożenia utraty kontroli nad dostępem do atrybutów i operacji poszczególnych klas systemu. Dostęp do nich z zewnątrz umożliwia się za pośrednictwem publicznych operacji.

Sposób oznaczania poziomów widoczności atrybutów i operacji w ramach klas przedstawia rysunek 3.12.

Rysunek 3.12.

Oznaczenie
widoczności w klasie

Licytacja	
+	dataRozpoczęcia:
+	czasTrwaniaLicytacji:
+	cenaWywoławcza:
~	dostawa:
-	idLicytacji:
+	utwórzLicytację()
+	zamknijLicytację()
+	wyświetlOpisArtykułu()
#	przypiszKupującego()
-	modyfikujOpis()

Atrybuty i operacje statyczne

Każdy obiekt klasy przechowuje zazwyczaj własne, odmienne, modyfikowalne wartości atrybutu bądź operacji. Jednocześnie wszystkie instancje danej klasy mogą przechowywać pewne stałe, identyczne wartości — np. stały mnożnik, rodzaj waluty, operację tworzenia lub usuwania. W przypadku zmiany takiej wartości w jednym obiekcie zmiana ta jest automatycznie wprowadzana w innych obiektach tej klasy. Opisaną cechę atrybutów i operacji nazywa się **statycznością**. Tak więc zarówno atrybuty, jak i operacje mogą być:

- ♦ egzemplarzowe — indywidualne dla każdego obiektu danej klasy;
- ♦ statyczne — tożsame we wszystkich obiektach klasy.

Atrybuty i operacje statyczne podkreśla się linią ciągłą.

Nazwy klas, atrybutów i operacji

Wyjaśnienie pojęcia widoczności pozwala na sprecyzowanie **reguł nadawania nazw** klasom, atrybutom i operacjom. Nazwa samej klasy na poziomie koncepcyjnym najczęściej jest wyrażeniem rzeczownikowym. Z kolei na poziomie implementacyjnym przybiera ona postać łańcucha znaków z wyszczególnieniem początku każdego wyrazu składowego za pomocą wielkiej litery. Przykładami poprawnych nazw klas są *Aukcja*, *SesjaGieldowa*, *PochodnyInstrumentFinansowy*. Również nazewnictwo atrybutów i operacji jest zróżnicowane dla poziomu koncepcyjnego i implementacyjnego diagramu klas. Przykładowe nazwy obu kategorii zawierają tabele 3.3 i 3.4.

Tabela 3.3. Konwencja nazewnictwa atrybutów

Atrybut	Nazwa na poziomie koncepcyjnym	Nazwa na poziomie implementacyjnym
pojemność karty SIM	Pojemność Karty	<i>pojemnośćKarty</i>
adres salonu firmowego	Adres Salonu	<i>adresSalonu</i>
numer telefonu komórkowego	Nr Telefonu Komórkowego	<i>nrTelefonuKomórkowego</i>

Tabela 3.4. Konwencja nazewnictwa operacji

Operacja	Nazwa na poziomie konceptualnym	Nazwa na poziomie implementacyjnym
zablokowanie karty SIM	Zablokuj Kartę	<i>zablokujKartę()</i>
zmiana aktualnych stawek wszystkich taryf	Zmień Stawki Taryf	<i>zmieńStawkiTaryf()</i>
wysłanie krótkiej wiadomości tekstowej	Wyślij SMS	<i>wyślijSms()</i>

Notacja atrybutów i składnia operacji

Specyfikacja typów danych określających poszczególne atrybuty jest pozornie nieskomplikowaną czynnością. Istotnie, nie nastręcza trudności — pod warunkiem bycia świadomym tego, że typy danych uzależnione są od **struktury języka programowania**, w którym nastąpi wdrożenie. Dlatego też narzędzia CASE różnych producentów udostępniają niejednolity ich zestaw. Podsumowanie najpowszechniej używanych typów danych przedstawia tabela 3.5.

Tabela 3.5. Standardowe typy danych

Typy danych	Narzędzie Rational Rose	Platforma .NET
Logiczne	Boolean	Boolean
Stałooprzecinkowe	Byte	Byte
	Integer	Short
	Long	Integer
Zmiennoprzecinkowe		Long
	Single	Single
	Double	Double
		Decimal
Znakowe	String	String
		Char
Inne	Date	Date
	Object	Object
	Currency	
	Variant	

Notacja atrybutów na poziomie implementacyjnym charakteryzuje się standardową składnią. Ich prezentacja na diagramie odbywa się wg następującej formuły:

```
<atrybut> ::= [<widoczność>] ["/"] <nazwa-atrybutu> ["/"] <typ>
["/] <liczebność> ["/] ["/] <wartość-początkowa> ["/] { <określenie-
-właściwości> ["/] }
```

Analogicznie, również dla operacji istnieje formuła składniowa:

```
<operacja> ::= [<widoczność>] <nazwa-operacji> [ "(" <lista-parametrów> ")" ]
[":" <określenie-właściwości>]
```

Lista parametrów wskazuje parametry operacji oddzielone przecinkami. Składnia parametru jest analogiczna do składni atrybutu, z tą różnicą, że przed jego nazwą zamiast opcjonalnej widoczności oraz symbolu ukośnika (wskazującego atrybuty pochodne) umieszcza się przedrostek wskazujący kierunek przekazywania parametru. Jest to również element opcjonalny, domyślnie przyjmujący wartość *in*.

Liczba uwzględnianych parametrów może być dowolna. Charakterystyczne kombinacje informacji dotyczące atrybutów przedstawia tabela 3.6, podczas gdy najczęściej spotykane elementy składni operacji zawarto w tabeli 3.7.

Tabela 3.6. Przykłady zastosowania typów danych i domyślnych wartości atrybutów

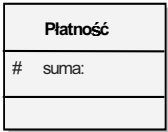
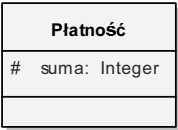
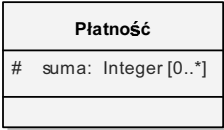
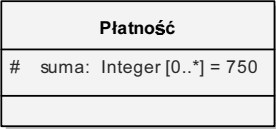
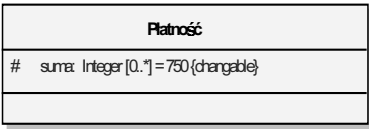
Przykład	Interpretacja
 <pre>classDiagram class Płatność { # suma: }</pre>	Wyłącznie nazwa atrybutu ze wskazaniem widoczności
 <pre>classDiagram class Płatność { # suma: Integer }</pre>	Nazwa atrybutu wraz ze wskazaniem jego typu i widoczności
 <pre>classDiagram class Płatność { # suma: Integer [0..*] }</pre>	Nazwa atrybutu (ze wskazaniem typu i widoczności) oraz jego liczebność
 <pre>classDiagram class Płatność { # suma: Integer [0..*] = 750 }</pre>	Zaprezentowana powyżej sytuacja, poszerzona o wartość początkową atrybutu
 <pre>classDiagram class Płatność { # suma: Integer [0..*] = 750 {changeable} }</pre>	Chroniony atrybut typu <i>Integer</i> o liczebności 0 lub wiele, mający wartość początkową 750 z przypisaną właściwością <i>changeable</i>

Tabela 3.7. Przykłady zastosowania składni operacji

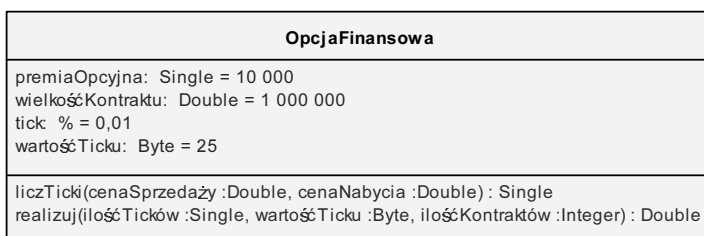
Przykład	Interpretacja
<pre> classDiagram class Płatność { + obliczPłatność() } </pre>	Nazwa operacji ze wskazaniem widoczności
<pre> classDiagram class Płatność { + obliczPłatność() : Long } </pre>	Nazwa operacji wraz ze wskazaniem widoczności i właściwości, która wskazuje typ wyniku
<pre> classDiagram class Płatność { + obliczPłatność(pozycja : Byte, obniżka : Boolean) } </pre>	Nazwa operacji (ze wskazaniem widoczności) poszerzona o listę parametrów ze wskazaniem ich typów
<pre> classDiagram class Płatność { + obliczPłatność(in pozycja : Byte = 50, obniżka : Boolean) : Long } </pre>	

Publiczna operacja zwracająca wynik typu *Long* — zawierająca atrybuty: *pozycja* typu *Byte* (o początkowej wartości 50) — oraz *obniżka* typu *Boolean* — uzupełniona o jawne wskazanie kierunku przekazywania parametru (*in*).

Omówione w niniejszym podrozdziale elementy opisu atrybutów i operacji zawarto na rysunku 3.13.

Rysunek 3.13.

Złożona reprezentacja atrybutów i operacji



Powyższy rysunek przedstawia klasę *OpcjaFinansowa*, która zawiera cztery atrybuty i dwie operacje. Klasa ta powstała na wczesnym etapie tworzenia systemu zaprezentowanego w punkcie *Studium diagramu klas*. Wartość pojedynczych kontraktów terminowych, których przykładem jest opisywana opcja, jest rzędu miliona USD, zatem atrybut *wielkośćKontraktu* domyślnie przyjmuje tę właśnie wartość. Ze względu na możliwość przypisywania do niego znacznych kwot zdecydowano się na typ danych *Double*. Tak samo wyrażony w procentach atrybut *tick* przyjmuje domyślnie wartość 0,01, natomiast kolejny atrybut niezbędny w opisie kontraktu terminowego, *wartośćTicku*, 25 USD. Zważywszy, że *wartościTicków* to niewielkie liczby, w tym przypadku wystarczający jest typ *Byte*. *PremiaOpcyjna*, suma istotnie większa, lecz nigdy nie tak

duża jak *wielkośćKontraktu*, jest atrybutem typu *Single*. W tym przypadku prawdopodobnie wystarczający byłby typ *Integer* — w większości języków programowania ograniczony z góry liczbą 65 536 — lecz przyjęcie pojemniejszego typu danych ma na celu zabezpieczenie się przed ewentualnym wystąpieniem błędu przepełnienia zmiennej. Należy podkreślić, że zamieszczona na opisywanym rysunku klasa jest rezultatem jednej z początkowych iteracji tworzenia dokumentacji statyki systemu, zatem nie uwzględniono na nim wszelkich niezbędnych w tej sytuacji atrybutów i operacji oraz znaczników widoczności, typowych dla tego poziomu szczegółowości. Tak samo atrybutowi *premiaOpcyjna* przypisano początkowo wartość domyślną w wysokości 10 000 USD, z której w kolejnych iteracjach zrezygnowano. Powodem było to, że wartość *premiOpcyjnej* jest różna w niemal każdym przypadku — zależy ona bowiem od ryzyka związanego z transakcją — i zazwyczaj znacznie niższa niż na wskazanym przykładzie. Operacja *liczTicki* zwraca liczbę *ticków* — czyli w istocie podaje zmianę ceny instrumentu bazowego kontraktu terminowego na podstawie cen nabycia oraz cen sprzedaży tego instrumentu. Ponownie w ramach środków ostrożności przyjęto bardziej pojemny typ danych dla zwracanej wartości. Operacja *realizuj*, zawierająca trzy parametry, oblicza zysk albo stratę z tytułu zawarcia kontraktu terminowego. Ze względu na to, że ten rodzaj instrumentów finansowych niesie ze sobą znaczne ryzyko, a ich nabywca rzadko ogranicza się do zawarcia pojedynczych kontraktów, zarówno potencjalne zyski, jak i straty są olbrzymie, więc do ich wyrażenia przewidziano bardzo pojemny typ danych *Double*.

Język UML umożliwia obliczanie wartości **atrybutów pochodnych** (ang. *derived attributes*). Poprzedzone są one znakiem ukośnika „/”. Ich wartość ustala się na zasadzie przypisania określonej formuły obliczeniowej, która korzysta z wartości innych atrybutów na diagramie klas. Formuła zazwyczaj jest zapisywana jako notatka odnosząca się do atrybutu pochodnego. W odniesieniu do scharakteryzowanego przykładu opcji finansowej atrybutem pochodnym może być atrybut */wartośćTransakcji*, który jest iloczynem atrybutu *wielkośćKontraktu* z klasy *OpcjaFinansowa* oraz atrybutu przechowującego liczbę zawartych kontraktów, umieszczonego w tej samej klasie bądź dowolnej klasie powiązanej.

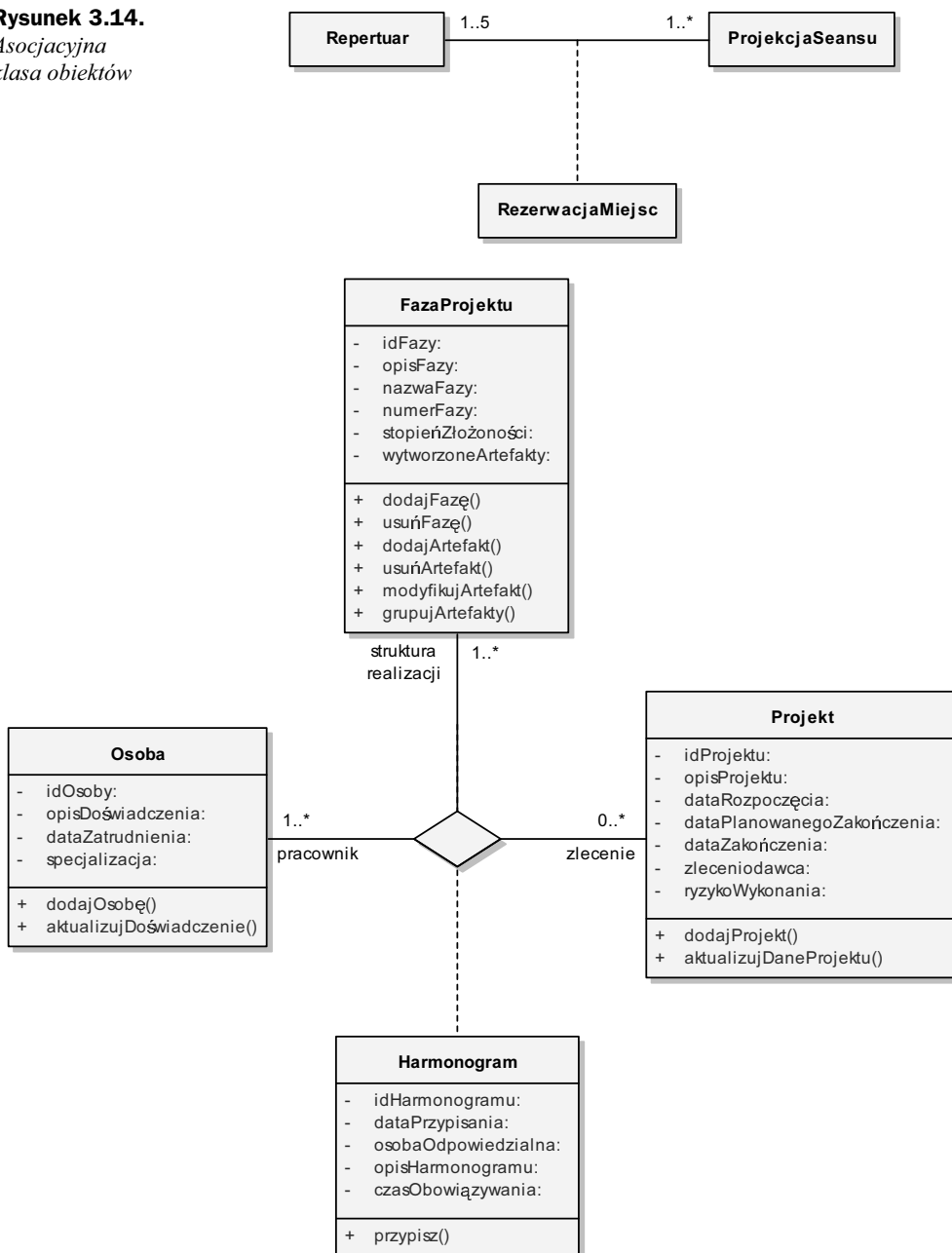
Klasy asocjacyjne

Klasa asocjacyjna (ang. *association class*) umożliwia bardziej precyzyjny opis związku między klasami. Służy do przedstawiania asocjacji w postaci klas. Zawiera zatem, podobnie jak inne klasy, nazwę, atrybuty i operacje. Liczbę sekcji można zwiększać. Każdej asocjacji można przypisać co najwyżej jedną instancję tego rodzaju klasy. Na diagramie klas klasę asocjacyjną wprowadza się poprzez ścieżkę asocjacyjną w postaci linii przerywanej (rysunek 3.14).

I tak *RezerwacjaMiejsc* jest klasą asocjacyjną dokumentującą asocjację pomiędzy *Reper tuarem* a *ProjekcjąSeansu*. Z jednym Repertuarem związanych jest wiele *ProjekcjiSeansu*. Przewidziano możliwość tworzenia kilku *Repertuarów*, np. dziennego, wieczornego, weekendowego. Wstępnie przyjęto ograniczenie liczby różnych repertuarów (5). Przedstawianie asocjacji w postaci klasy dotyczy także asocjacji n-arnych (rysunek 3.15).

Rysunek 3.14.

Asocjacyjna
klasa obiektów

**Rysunek 3.15.** Związek n-arny i klasa asocjacyjna

Opisywany rysunek dotyczy harmonogramowania projektu. I tak klasa asocjacyjna *Harmonogram* specyfikuje asocjację łączącą trzy inne klasy:

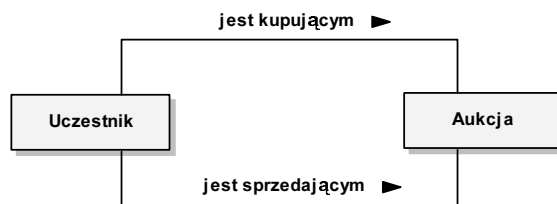
- ♦ *Osobę*,
- ♦ *Projekt*,
- ♦ *FazęProjektu*.

Klasy te są szczegółowo scharakteryzowane poprzez swoje atrybuty, operacje oraz widoczności. Na diagramie wyspecyfikowano ponadto role oraz liczebności.

Asocjacje zwrotne i wielokrotne

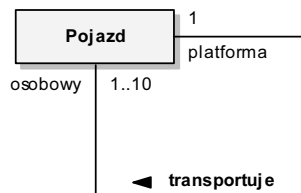
Powiązane klasy mogą pełnić względem siebie kilka odmiennych ról. Tym samym klasy te mogą być połączone kilkoma asocjacjami. Asocjacje takie nazywa się **asocjacjami wielokrotnymi**. Każda z tych asocjacji winna być nazwana lub scharakteryzowana na podstawie ról, jak na rysunku 3.16.

Rysunek 3.16.
Asocjacje wielokrotne między klasami



Istnieje możliwość przedstawiania asocjacji wiążącej daną klasę z samą sobą. Asocjacja taka nazywana jest **asocjacją zwrotną**. Przykład asocjacji zwrotnej zamieszczono na rysunku 3.17.

Rysunek 3.17.
Asocjacja zwrotna

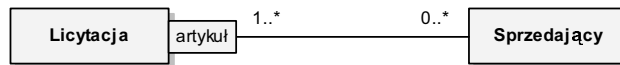


Kwalifikacja

Ostatnią omawianą cechą asocjacji jest kwalifikacja. Umożliwia ona wyszukiwanie obiektów związanych z daną asocjacją. Kwalifikacji dokonuje się za pomocą **kwalifikatora** (ang. *qualifier*) — atrybutu lub listy atrybutów, których wartości porządkują zbiór obiektów tej klasy występujących w danej asocjacji. Przykład kwalifikatora zamieszczono na rysunku 3.18.

Rysunek 3.18.

Asocjacja
kwalifikowana



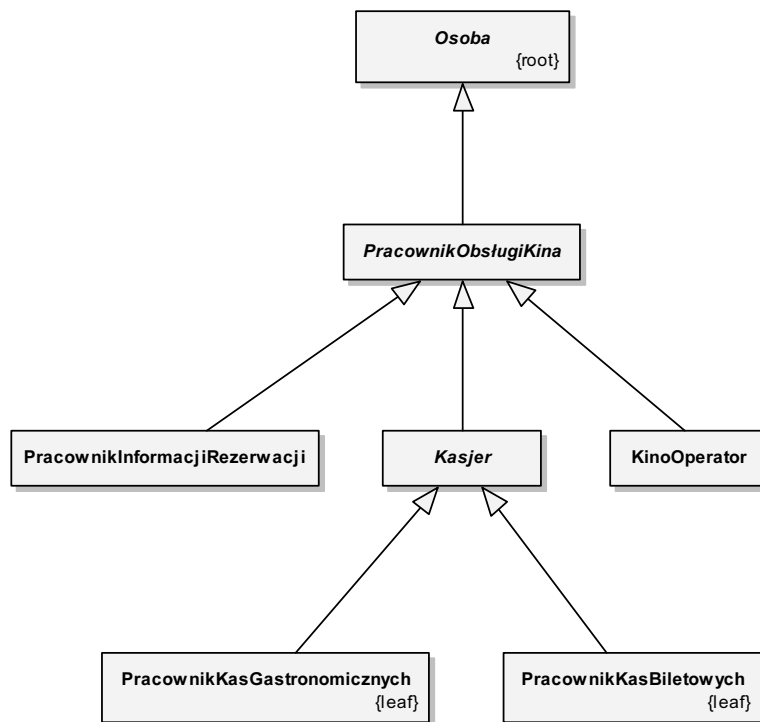
Uogólnienia, klasy abstrakcyjne oraz konkretne

W hierarchiach klas powiązanych związkami uogólnienia występują **klasy abstrakcyjne** oraz konkretne. Klasy abstrakcyjne nie mają konkretnych instancji obiektów, lecz stanowią uogólnienie obiektów konkretnych znajdujących się na niższych poziomach hierarchii. Nazwy klas abstrakcyjnych — podobnie jak innych abstrakcyjnych kategorii pojęciowych — pisane są kursywą. Klasom abstrakcyjnym można przypisywać atrybuty i operacje podlegające **dziedziczeniu** zgodnie z hierarchią klas.

Klasy na najniższym poziomie hierarchii nazywane są „liśćmi” (oznaczenie *{leaf}*), natomiast na najwyższym — klasami podstawowymi albo „korzeniami” (oznaczenie *{root}*). Jawne wskazanie klas będących korzeniami i liśćmi zastosowano na rysunku 3.19.

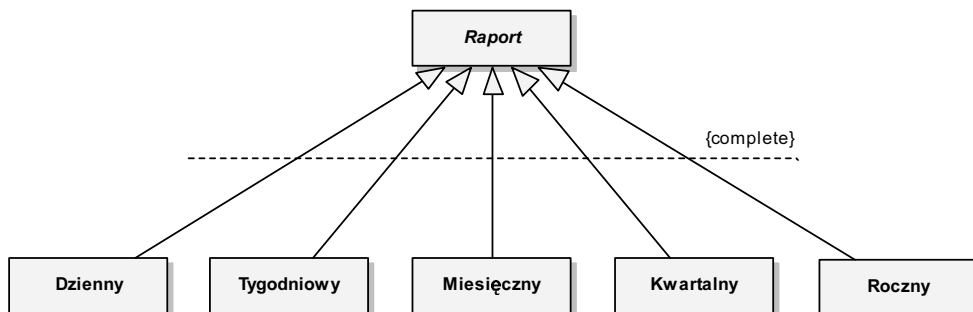
Rysunek 3.19.

Dziedziczenie klas



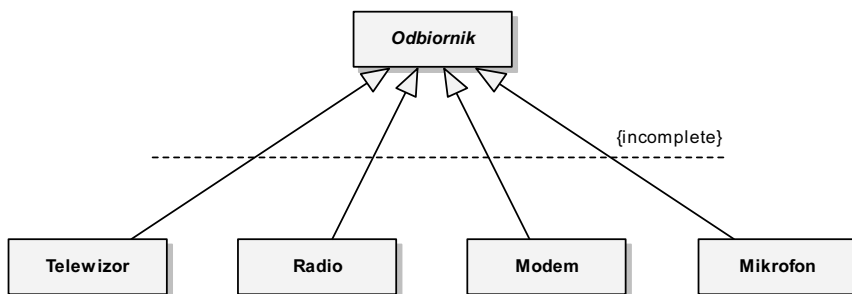
Związkom uogólnienia, występującym pomiędzy klasami obiektów znajdujących się w danej hierarchii klas, przypisać można cztery podstawowe **ograniczenia**. Obejmują one:

- ♦ *{complete}* — zawiera pełen zestaw podklas dla danej klasy (rysunek 3.20);



Rysunek 3.20. Ograniczenie typu *{complete}*

- ♦ *{incomplete}* — oznacza, że zestaw podklas przypisanych danej klasie nie jest pełny (rysunek 3.21); wiadomo, że istnieje więcej klas specjalizowanych nie mających istotnego znaczenia w rozpatrywanym kontekście;



Rysunek 3.21. Ograniczenie typu *{incomplete}*

- ♦ *{disjoint}* — domyślny rodzaj uogólnienia, oznaczający, że każda podklasa w hierarchii klas jest rozdzielna (rysunek 3.22), tzn. posiada tylko jedną klasę bezpośrednio nadrzędną;
- ♦ *{overlapping}* — oznacza, że w hierarchii klas określone klasy mogą mieć wspólne podklasy (rysunek 3.22); sytuacja taka określana jest mianem dziedziczenia wielokrotnego.

Ograniczenia te odnoszą się do związków uogólnienia wyspecyfikowanych pomiędzy klasami.

Ze związkiem uogólnienia wiąże się pojęcie **dyskryminatora** (ang. *generalization set*), który jest określeniem kryterium dokonania klasyfikacji w uogólnieniu. Na ogół jest to kryterium domyślne, lecz można je wyrazić wprost.

Dyskryminator stwarza możliwość systematyzowania związków uogólnień. Dyskryminatory wskazujące kryterium formy płatności (*Gotówka*, *Przelew*, *KartaPłatnicza*) oraz kryterium terminu płatności (*Należność*, *Wpływ*) zawiera rysunek 3.23 przedstawiający

Rysunek 3.22.

Ograniczenia
typu {disjoint}
i {overlapping}

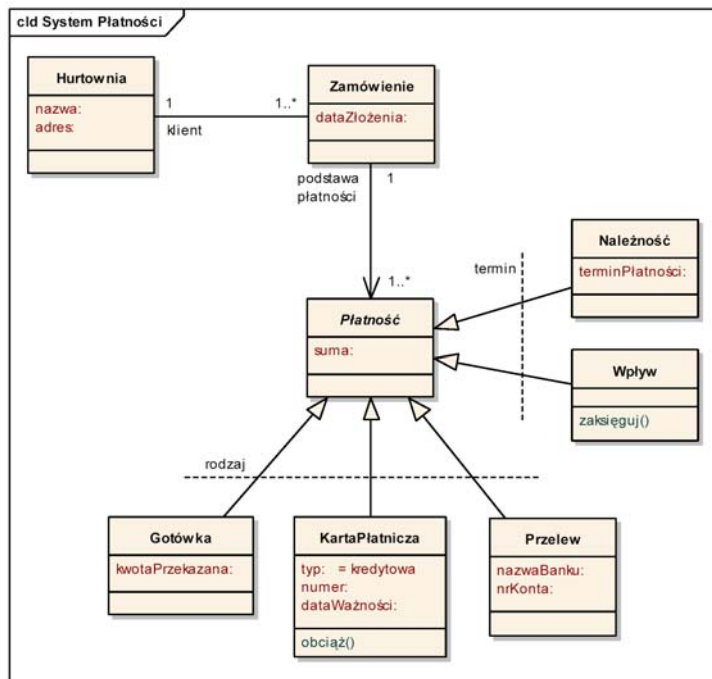
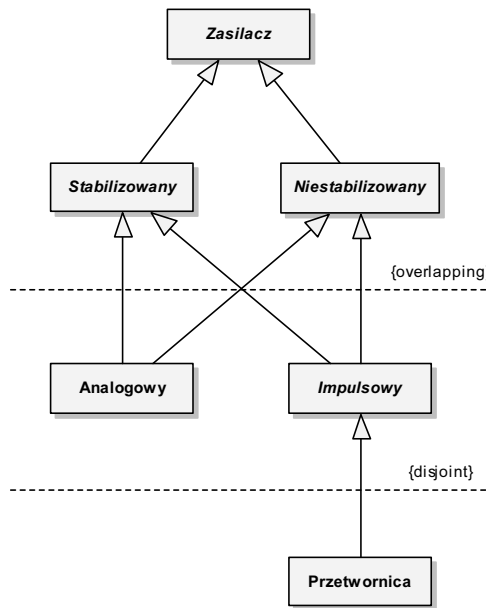
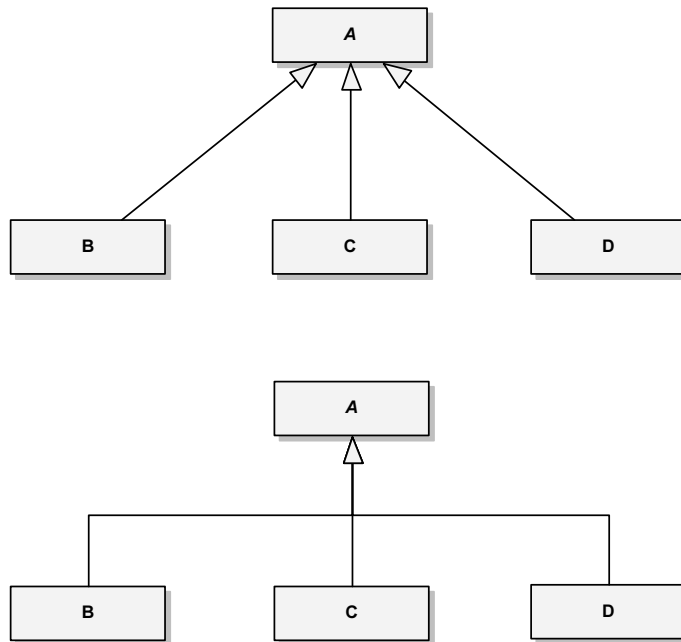
**Rysunek 3.23.** Diagram klas systemu płatności

diagram klas systemu płatności. Pojęcie dyskryminatora można połączyć ze stosownymi ograniczeniami, wprowadzając np. klasyfikacje nie tylko typu {disjoint}, lecz także {overlapping}, {complete} oraz {incomplete}.

Podsumowując związki uogólnienia, należy zaznaczyć, że istnieją dwie konwencje graficznej prezentacji związków uogólnienia (rysunek 3.24). Są one znaczeniowo **tożsame**.

Rysunek 3.24.

Konwencje graficznej prezentacji uogólnień

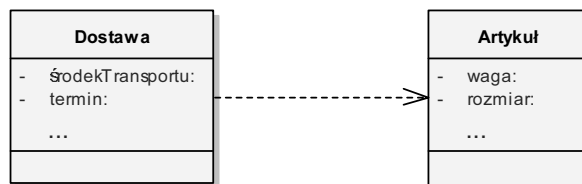


Zależność

Mniejsze znaczenie w diagramach klas odgrywa związek **zależności**, zdefiniowany w poprzednim rozdziale. Związek ten oznacza, że niezależna klasa obiektów wykorzystuje klasę zależną, jak przedstawiono na rysunku 3.25. Istnieje szereg rodzajów zależności; ich opis można uszczegółowić poprzez podanie stereotypów dostępnych w języku UML i znajdujących zastosowanie w odniesieniu do diagramów klas. Klasa niezależna jest także nazywana docelową, natomiast zależna źródłową.

Rysunek 3.25.

Związek zależności



Realizacja

Klasy mogą brać również udział w związkach realizacji. Jak zaznaczono w rozdziale drugim, jedna strona tego związku wskazuje klasyfikator określający kontrakt, natomiast druga — klasyfikator zapewniający wywiązanie się z niego. W opisywanym kontekście

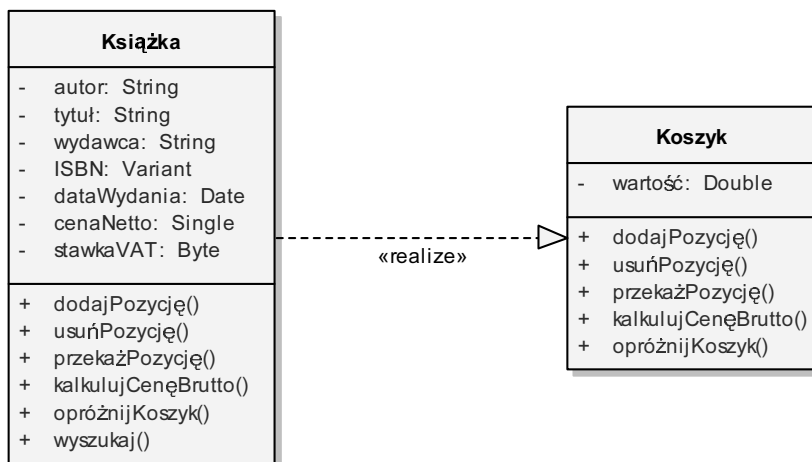
realizacja wskazuje na związek pomiędzy interfejsem a klasą. Identyfikuje **interfejsy** (ang. *interfaces*), które zapewniają realizację usługi klasy (rysunek 3.31).



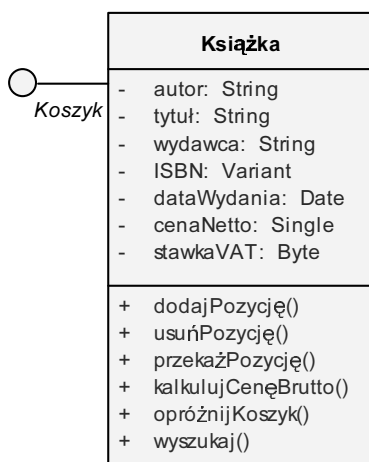
Interfejs to zestaw operacji, które wyznaczają usługi oferowane przez klasę lub komponent.

Istnieje szereg konwencji przedstawiania interfejsów na diagramach klas (rysunek 3.26a, rysunek 3.26b). Interfejsy szerzej omawia rozdział jedenasty.

a)



b)



Rysunek 3.26. Związek realizacji i interfejs

Poszczególne usługi oferowane przez interfejs *Koszyk* w postaci operacji *dodajPozycję*, *usuńPozycję* itd. są realizowane poprzez klasę *Książka*.

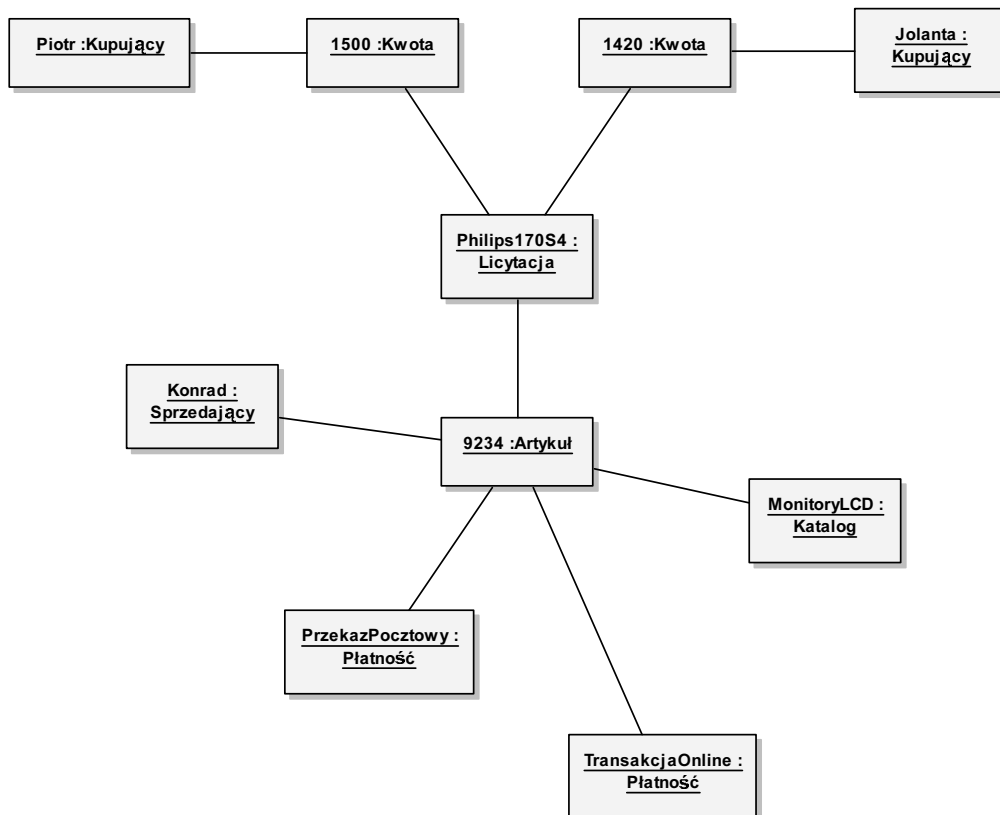
Diagram obiektów

W celu ułatwienia zrozumienia diagramu klas o wysokim stopniu złożoności można posłużyć się **diagramami obiektów**.



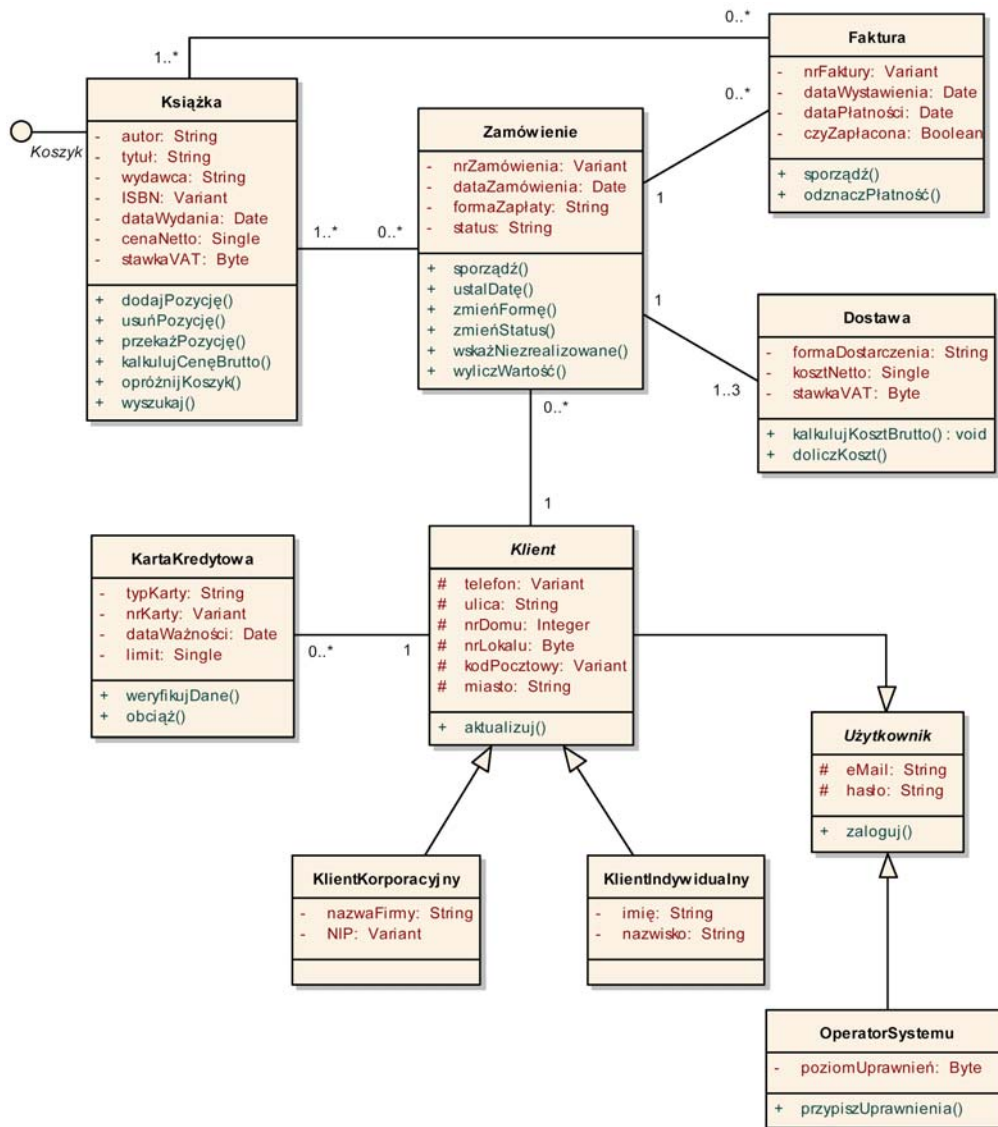
Diagram obiektów to wystąpienie diagramu klas, odwzorowujące strukturę systemu w wybranym momencie jego działania.

Diagramy obiektów z reguły zawierają wyłącznie elementarne informacje. I tak rysunek 3.27 przedstawia odwzorowanie obiektów systemu aukcji internetowej podczas wstawiania artykułu na aukcję przez sprzedającego. Diagram klas tego systemu jest tematem części studialnej niniejszego rozdziału (rysunek 3.31).



Rysunek 3.27. Przykład diagramu obiektów

Z kolei na rysunku 3.28 oraz 3.29 zamieszczono odpowiednio: diagram klas systemu elektronicznej księgarni oraz diagram obiektów opisujący strukturę tego systemu w momencie dokonywania zakupu z użyciem karty kredytowej.

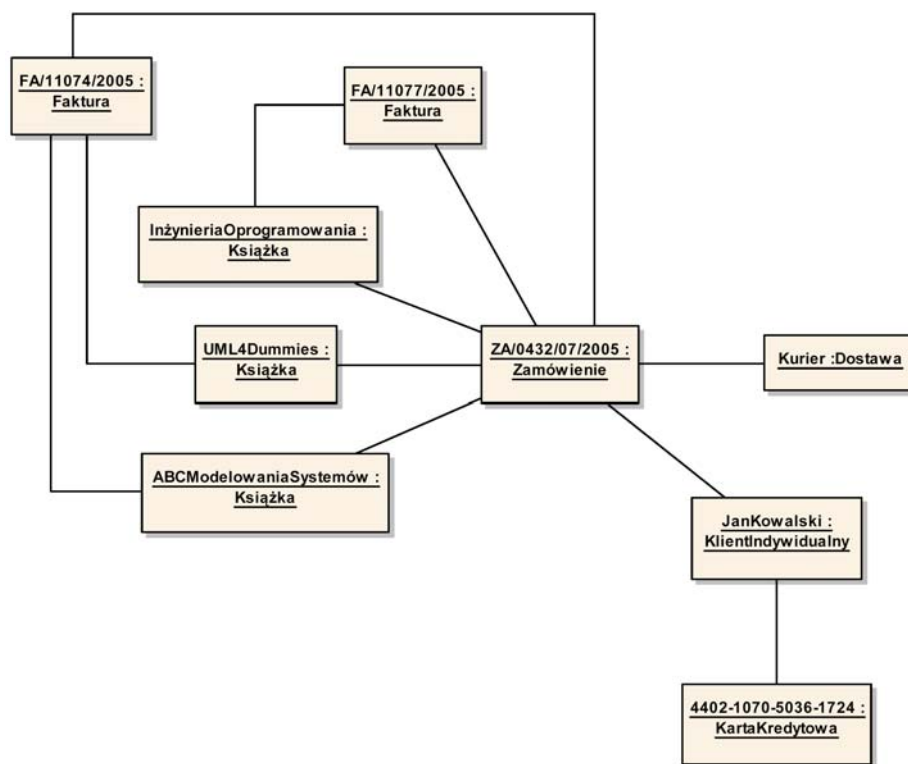


Rysunek 3.28. Przykład klas systemu księgarni elektronicznej

Proces tworzenia diagramu klas

W procesie tworzenia diagramu klas wyróżnić można następujące etapy:

1. zidentyfikowanie i nazwanie klas;
2. opcjonalne określenie zobowiązań klas;



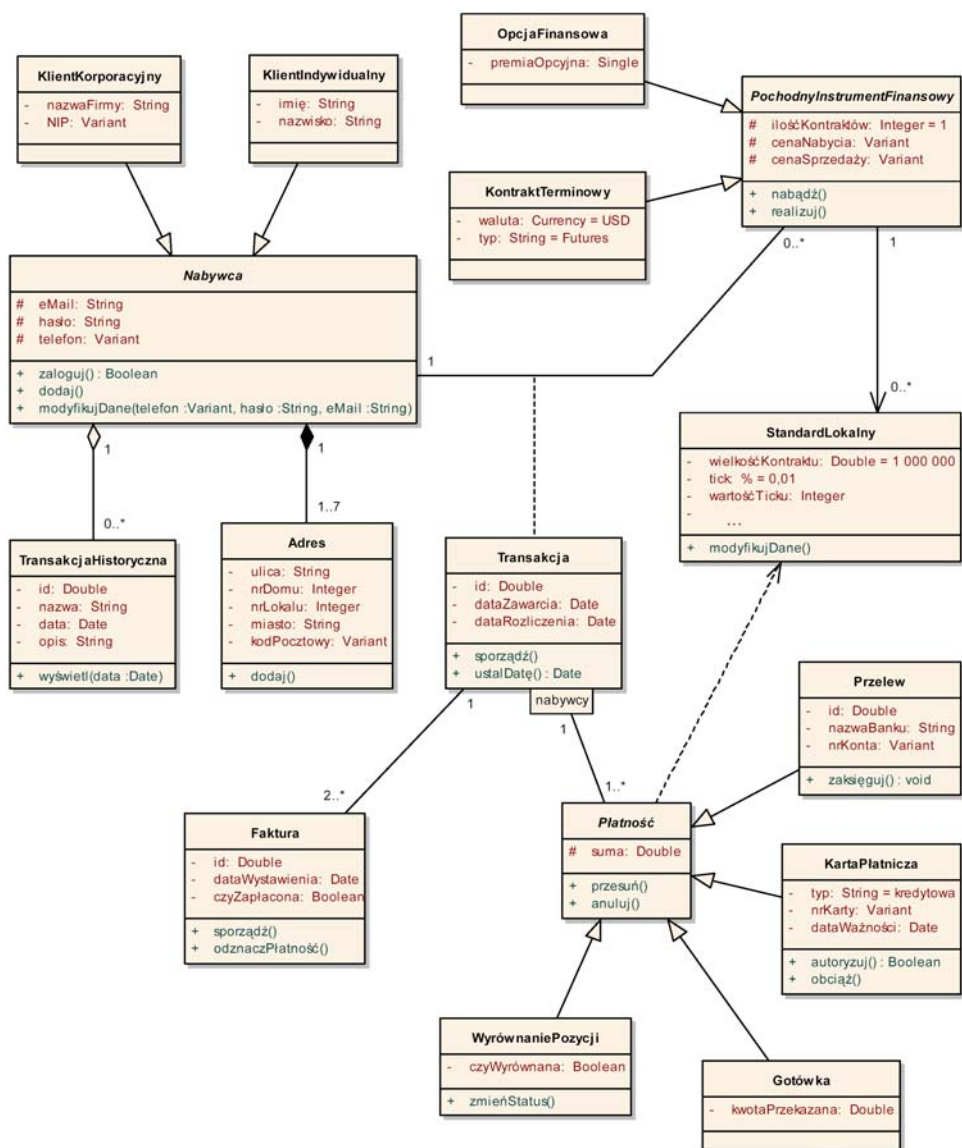
Rysunek 3.29. Diagram obiektów systemu księgarni elektronicznej

3. połączenie poszczególnych klas z wykorzystaniem związków asocjacji;
4. zidentyfikowanie oraz nazwanie atrybutów i operacji;
5. wyspecyfikowanie asocjacji z użyciem wszystkich jej cech (nazwy, ról, nawigacji, liczebności, agregacji, kwalifikacji);
6. opracowanie innych rodzajów związków, tj. uogólnień, zależności i realizacji;
7. pełne, precyzyjne wyspecyfikowanie atrybutów i operacji zgodnie ze składnią opisaną w niniejszym rozdziale;
8. opcjonalne opracowanie diagramów obiektów.

Należy zaznaczyć, że zgodnie z istotą iteracyjno-przyrostowego procesu TSI etapy te są realizowane wielokrotnie, stosownie do przyrostu dokumentacji i zwiększania jej szczegółowości.

Studium diagramu klas

Rysunek 3.30 podsumowuje omówione we wcześniejszych punktach podstawowe oraz zaawansowane elementy diagramu klas.



Rysunek 3.30. Diagram klas finansowego systemu transakcyjnego

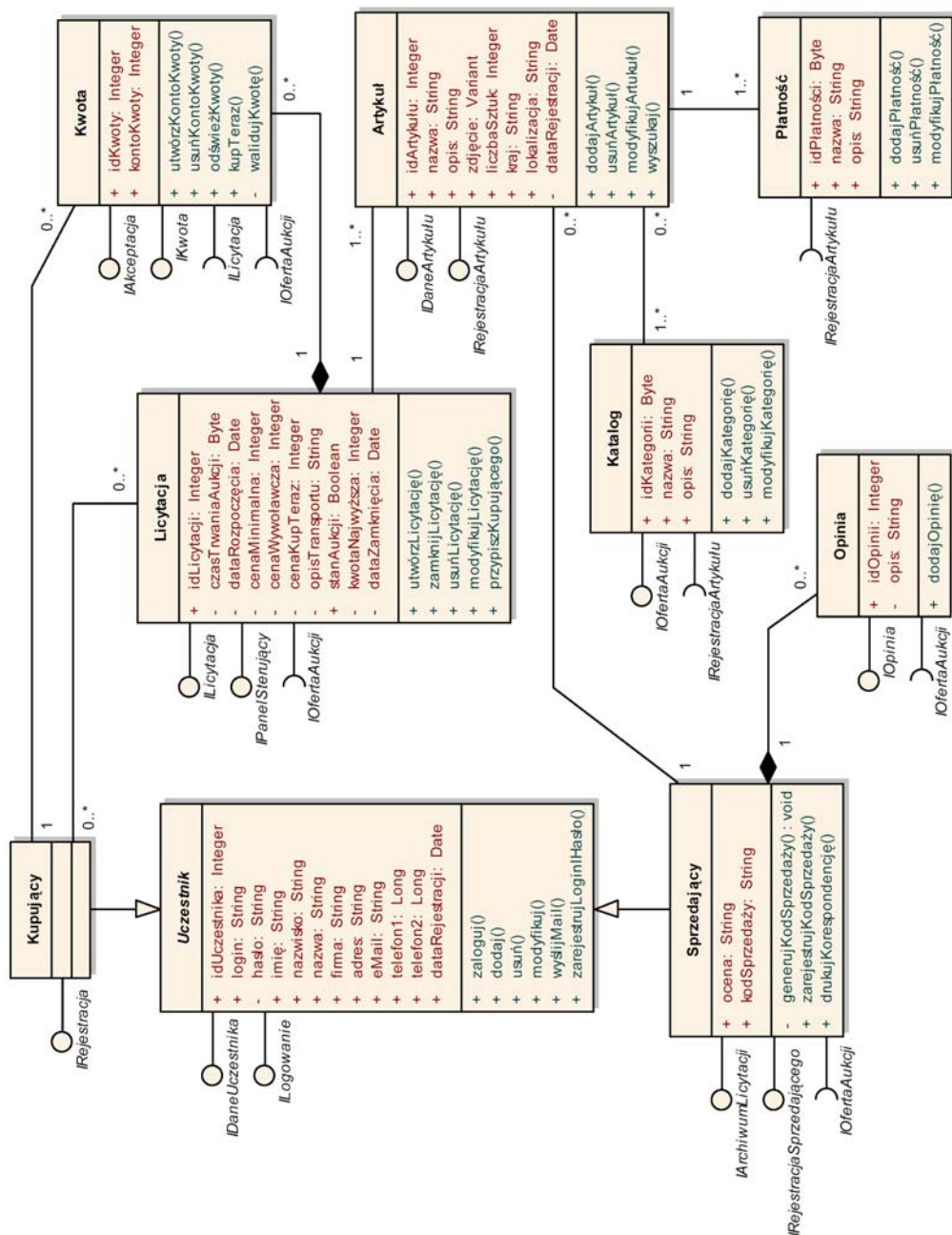
Rolą systemu finansowego pokazanego na zamieszczonym niżej diagramie klas jest obsługa elektronicznego obrotu pochodnymi instrumentami finansowymi. Przedmiotem transakcji są *PochodneInstrumentyFinansowe* (PIF). Są to instrumenty, których cena zależy od innych instrumentów finansowych (np. kursu wybranej waluty czy też kursu akcji). Istotą transakcji jest nabycie albo zbycie instrumentu finansowego za określoną cenę w określonym dniu, a następnie zbycie albo odkupienie go w przyszłości — również po wcześniej ustalonej cenie i w określonym dniu. Zysk albo strata na transakcji jest wynikiem wahań kursu w czasie bądź ceny instrumentu, na którym *PochodneInstrumentFinansowy* bazuje.

W systemie obsługiwane są poszczególne rodzaje PIF: *KontraktyTerminowe* oraz tzw. *OpcjeFinansowe*. Ze względu na konieczność rozróżniania kontraktów terminowych typu forward oraz futures, w klasie *KontraktTerminowy* zaimplementowano atrybut *typ*, przy czym wartość domyślną tego atrybutu stanowi kontrakt typu futures. Cechą szczególną *OpcjiFinansowych* jest możliwość niezrealizowania transakcji w przypadku niekorzystnego układu cen. Dodatkowym kosztem płaconym przez klienta za to „ubezpieczenie” jest tzw. premia opcyjna — opłata manipulacyjna uiszczana dodatkowo w momencie dokonania transakcji. Jej wysokość jest różna (atrybut *premiaOpcyjna* klasy *OpcjaFinansowa*). Transakcje obejmujące PIF są zazwyczaj standaryzowane (klasa *StandardLokalny*), przy czym standardem mogą być np. regulacje danej giełdy prowadzącej obrót PIF. Atrybut *tick* opisuje minimalną zmianę ceny instrumentu bazowego, natomiast atrybut *wartośćTicku* — finansową wartość tej minimalnej zmiany. Wszystkie osoby zarejestrowane w systemie reprezentowane są przez obiekty klasy *Nabywca*, przy czym w systemie wyróżnia się *Nabywców* będących osobami fizycznymi (*KlientIndywidualny*) oraz firmami (*KlientKorporacyjny*). Dowolny klient może posiadać szereg zarejestrowanych *Adresów* korespondencyjnych (maksymalna liczba obsługiwanych adresów wynosi 7). Klasa *Transakcja* reprezentuje operację obrotu PIF. Z każdą *Transakcją* związane są przynajmniej dwie kopie *Faktury* (faktura dla klienta oraz jej kopia, pozostająca w jednostce prowadzącej obrót *PochodnymiInstrumentamiFinansowymi*) oraz *Płatność*. *Płatności* można dokonać jednorazowo bądź w kilku transzach. Fakt całkowitego opłacenia *Faktury* odnotowywany jest w systemie (operacja *odznacznPłatność* klasy *Faktura*). Istnieje możliwość fizycznego dokonania *Płatności* z wykorzystaniem *KartyPłatniczej*, *Przelewu* bądź *Gotówki* — w przypadku tzw. wirtualnych PIF (niewymagających fizycznego dostarczenia towaru oraz środka płatniczego) stosuje się tzw. *WyrównaniePozycji*. W tej sytuacji transfery środków pieniężnych składających się na transakcję w praktyce nie występują, rozliczeniu podlega wyłącznie zysk albo strata na transakcji.

Z kolei diagram zaprezentowany na rysunku 3.31 przedstawia schemat bazy danych systemu aukcji internetowej. Wyszczególnione klasy (np. klasa *Artykuł*) opisują wszystkie obiekty, które mają takie same atrybuty, operacje, związki i znaczenie. Zilustrowany diagram klas — poza samymi klasami i związkami pomiędzy nimi — przedstawia interfejsy, które reprezentują usługi oferowane przez klasę. Przykładami interfejsów, których usługi są realizowane podczas przypadku użycia „Dokonaj rejestracji”, są: *IRejestracjaSprzedającego* oraz *IPanelSterujący*.

Podstawowe pojęcia

Agregacja	Atrybut
Agregat	Chroniony
Całkowita	Pakietowy
Częściowa	Prywatny
Segment	Publiczny
Asocjacja	Statyczny



Rysunek 3.31. Model koncepcyjny bazy danych systemu aukcji internetowej

Diagram klas	Operacja
Definicja	Chroniona
Implementacyjny	Pakietowa
Konceptualny	Prywatna
Proces tworzenia	Publiczna
Diagram obiektów	Statyczna
Dyskryminator	Rola
Dziedziczenie	Widoczność
Interfejs	Zobowiązanie
Karta CRC	Związek
Klasa	Asocjacja
Abstrakcyjna	Binarna
Asocjacyjna	N-arna
Konkretna	Wielokrotna
Kwalifikacja	Zwrotna
Kwalifikator	Realizacja
Liczebność	Uogólnienie
Nawigacja	Zależność
Obiekt	
Ograniczenie	
{complete}	
{disjoint}	
{incomplete}	
{overlapping}	

Pytania i zadania

1. Uzasadnij znaczenie diagramu klas.
2. Jakie kategorie pojęciowe decydują o stanie obiektu, a jakie o jego zachowaniu?
3. Wyjaśnij relacje pomiędzy klasą a obiektem.
4. Jakie elementy może zawierać symboliczny opis danej klasy? Wymień najczęściej stosowane kombinacje tego opisu. Podaj przykłady.

5. Przedstaw atrybuty, operacje i inne elementy klasy obiektu *Licytacja*.
6. W jaki sposób oznacza się fakt pominięcia w opisie klasy mniej istotnych atrybutów i (lub) operacji?
7. Wymień rodzaje asocjacji. Jaka jest podstawowa różnica pomiędzy nimi?
8. Wymień cechy asocjacji. Oceń ich praktyczne znaczenie.
9. W jaki sposób na diagramach klas oznacza się nawigację dwukierunkową, a w jaki jednokierunkową?
10. Jakie oznaczenia liczebności są twoim zdaniem absolutnie niezbędne do opisu diagramu klas na poziomie konceptualnym? Jakie widzisz zagrożenia stosowania oznaczenia * na poziomie implementacyjnym?
11. Wskaż podstawową różnicę pomiędzy agregacją a kompozycją.
12. Powiązana ze *ZintegrowanymSystememInformatycznym BazaDanych* jest niezbędnym elementem takiego systemu i nie może w żadnym wypadku poprawnie funkcjonować w oderwaniu od niego. Przedstaw to powiązanie na diagramie klas.
13. Jak inaczej nazywana jest kompozycja?
14. Zidentyfikuj klasy systemu obsługi rezerwacji hotelowych. Połącz je związkami, tworząc konceptualny diagram klas.
15. W jaki sposób uzasadniłbyś zasadność różnicowanie diagramów klas względem poziomów?
16. Wyjaśnij pojęcie zobowiązania klasy. Jak nazywa się klasę, której podstawową zawartość stanowią zobowiązania oraz lista współpracujących klas?
17. Wymień i scharakteryzuj poziomy widoczności występujące w języku UML. Porównaj je z poziomami widoczności występującymi w obiektowych językach programowania, takich jak C++ oraz JAVA.
18. Wyjaśnij koncepcję statyczności.
19. Wyjaśnij konwencję nazywania klas odpowiednio na:
 - ♦ konceptualnych diagramach klas,
 - ♦ implementacyjnych diagramach klas.
20. Omów składnię atrybutów oraz operacji w języku UML.
21. Wzbogać przykładowe klasy zamieszczone na rysunku 3.1 o typy danych, domyślne wartości atrybutów, listy parametrów operacji oraz oznaczenia widoczności.
22. Wskaż podobieństwa i różnice pomiędzy asocjacjami wielokrotnymi i zwrotnymi. Podaj przykład zastosowania asocjacji zwrotnej w dziedzinie przedmiotowej funkcjonowania księgarni.

23. Przedstaw asocjację pomiędzy klasami *Osoba* oraz *Projekt* w postaci klasy asocjacyjnej *Harmonogram*. Uzupełnij opis o atrybuty, operacje, liczebności oraz nazwy ról.
24. Wyjaśnij relacje pomiędzy pojęciami:
- ♦ dziedziczenie,
 - ♦ klasa konkretna,
 - ♦ obiekt,
 - ♦ dyskryminator,
 - ♦ uogólnienie.
25. Jak graficznie odróżnia się klasę abstrakcyjną od klasy konkretnej? O jakie informacje wzbogacają model oznaczenia *{root}* i *{leaf}*?
26. Zilustruj zastosowanie kategorii pojęciowej dyskryminatora w odniesieniu do dwóch kryteriów dokonania klasyfikacji w uogólnieniu klasy *KontoBankowe*.
27. Opisz konwencje graficznej prezentacji związków uogólnienia.
28. Zaproponuj klasy, od których zależna byłaby klasa *Rachunek* opisująca rachunek telefoniczny.
29. Wyjaśnij pojęcie kwalifikatora.
30. Przekształć konceptualny diagram klas systemu obsługi rezerwacji hotelowych w implementacyjny diagram klas, poprzez zastosowanie zwiększających wartość informacyjną tego diagramu zaawansowanych kategorii pojęciowych.
31. Zaproponuj interfejsy najistotniejszych klas systemu zaprezentowanego na rysunku 3.30.
32. Jaką rolę spełniają diagramy obiektów?
33. Podaj przykładowe obiekty klas systemu obsługi rezerwacji hotelowych.
34. Opracuj diagram obiektów odpowiadający diagramowi klas zamieszczonemu na rysunku 3.10.

Rozdział 4.

Diagram czynności

Znaczenie diagramów czynności

Diagramy czynności są jednym z rodzajów diagramów języka UML opisujących dynamikę systemu.



Diagram czynności to graficzne przedstawienie sekwencyjnych i (lub) współbieżnych przepływów sterowania oraz danych pomiędzy uporządkowanymi ciągami czynności, akcji i obiektów.

Diagramy czynności stosuje się w **modelowaniu**:

- ◆ wysokopoziomowych procesów biznesowych,
- ◆ systemów oraz podsystemów,
- ◆ scenariuszy przypadków użycia,
- ◆ procesów systemowych charakteryzujących się dużą liczbą równoległych czynności i sytuacji decyzyjnych,
- ◆ operacji,
- ◆ algorytmów.

Zakres dokumentowania czynności i akcji w wersji 2.0 standardu UML uległ znacznemu poszerzeniu w stosunku do wersji 1.x.

Podstawowe kategorie pojęciowe oraz notacja graficzna

Diagramy czynności składają się z następujących **podstawowych elementów**:

- ♦ czynności,
- ♦ akcji,
- ♦ przepływów sterowania,
- ♦ początku,
- ♦ końca,
- ♦ zakończenia przepływu.

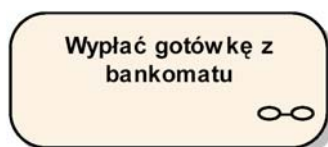
Tabela 4.1 zawiera charakterystykę podstawowych kategorii pojęciowych występujących na diagramie czynności.

Czynności a akcje

Czynności (ang. *activities*) na diagramach mogą cechować się rozbudowaną funkcjonalnością, tj. mogą reprezentować niezwykle złożone procesy biznesowe bądź algorytmy przetwarzania. Dla osiągnięcia precyzyjnego ich opisu niezbędna staje się **dekompozycja** czynności. Czynności mogą być tym samym dekomponowane na zhierarchizowane podczynności. Notacja czynności będącej przedmiotem dekompozycji jest wzbogacana poprzez zamieszczenie stosownego symbolu w jej prawej dolnej części (rysunek 4.1). Podczynność jest dekomponowana w formie dodatkowego diagramu czynności.

Rysunek 4.1.

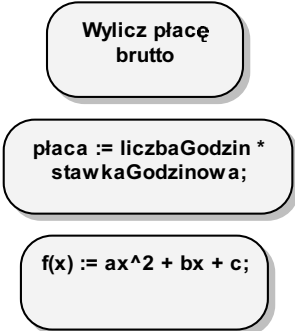
*Pojedyncza czynność
uszczegółowiona
na odrębnym
diagramie*



Proces dekompozycji może być kontynuowany aż do poziomu akcji (ang. *actions*), a więc elementarnych jednostek specyfikacji dynamiki systemu. Proces ten przedstawia rysunek 4.2.

Na najniższym poziomie abstrakcji na diagramach czynności reprezentowane są przepływy sterowania i danych pomiędzy logicznie uporządkowanymi ciągami akcji. Kontekst akcji stanowią czynności, uogólniające ich zagnieżdżoną sekwencję. Podstawowe **różnice** pomiędzy czynnościami a akcjami prezentuje tabela 4.2.

Tabela 4.1. Podstawowe kategorie pojęciowe diagramów czynności

Nazwa	Notacja graficzna	Definicja/interpretacja
Czynność		 Czynność to określone zachowanie złożone z logicznie uporządkowanych ciągów podczynności, akcji oraz obiektów w celu wykonania pewnego procesu.
Akcja		 Akcja to elementarna jednostka specyfikacji zachowania, która reprezentuje transformację lub przetwarzanie w modelowanym systemie.
Przeływ sterowania		 Przeływ sterowania to relacja między dwoma czynnościami bądź akcjami, wskazująca, że po wykonaniu źródłowej czynności albo akcji sterowanie zostanie przekazane do docelowej czynności albo akcji.
Początek		Początek to punkt rozpoczęcia przepływu sterowania i danych inicjujący funkcjonowanie diagramu czynności. Standardowo w diagramach czynności występuje jeden początek. W złożonych systemach, szczególnie tych funkcjonujących w czasie rzeczywistym, może wystąpić więcej niż jeden początek.
Koniec		Koniec to punkt zatrzymania wszystkich przepływów sterowania i danych na diagramie czynności. Na jednym diagramie czynności może wystąpić więcej niż jeden koniec.
Zakończenie przepływu		Zakończenie przepływu to punkt zatrzymania wybranego przepływu sterowania. Na jednym diagramie czynności może wystąpić więcej niż jedno zakończenie przepływu.

Rysunek 4.2.
Dekompozycja czynności

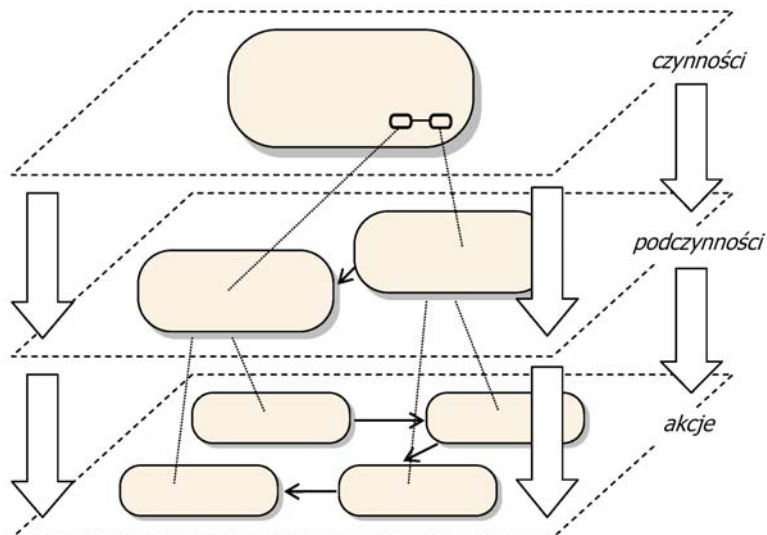
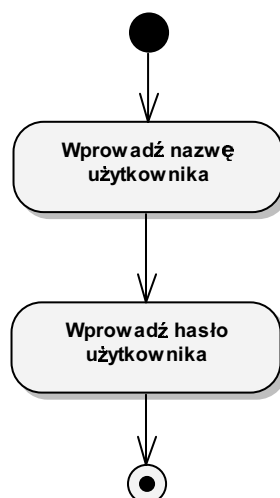


Tabela 4.2. *Różnice pomiędzy czynnościami a akcjami*

Kryterium	Czynność	Akcja
Istota	podzielna	niepodzielna; z natury nie ulega przerwaniu
Poziom ogólności	ogólna	szczegółowy przypadek; może być konsekwencją czynności
Dekompozycja	dozwolona	niedozwolona
Czas realizacji	znaczący	nieznaczny

Przykład diagramu czynności zawierającego wejście, wyjście, dwie czynności oraz przepływy sterowania pomiędzy wymienionymi elementami zamieszczono na rysunku 4.3, który dotyczy logowania użytkownika.

Rysunek 4.3.
Logowanie użytkownika do systemu operacyjnego



Zaawansowane składniki diagramu

Diagramy czynności w początkowych wersjach dotyczyły możliwości dokumentowania przepływu sterowania w systemie. W wersji 2.0 języka UML wiele uwagi poświęcono kwestii precyzyjnego dokumentowania również przepływów danych. W związku z tym kolejno omówione zostaną następujące **kategorie zaawansowane**:

- ♦ przepływy sterowania,
- ♦ akcje,
- ♦ przepływy danych,
- ♦ partycje diagramów czynności,
- ♦ obszar rozszerzenia,
- ♦ obszar przerwania,
- ♦ manipulator wyjątków.

Przepływy sterowania

Do udokumentowania **przepływów sterowania** (ang. *control flows*) w diagramie czynności niezbędna jest umiejętność rozumienia i posługiwania się następującymi kategoriami pojęciowymi:

- ♦ znacznikiem sterowania;
- ♦ przepływami decyzyjnymi: decyzjami, łącznikami, złączeniami;
- ♦ przepływami współbieżnymi: rozwidleniami, scaleniami.

Przepływ sterowania zachodzi pomiędzy kolejnymi, logicznie uporządkowanymi czynnościami lub akcjami. W szczególnych przypadkach można przypisać mu nazwę. Przepływ sterowania charakteryzuje się zdolnością przesyłania **znaczników sterowania** (ang. *tokens*). Znaczniki sterowania to abstrakcyjne kategorie pojęciowe, użyteczne w monitorowaniu i realizacji procesu sterowania na diagramie czynności. Nie mają osobnego oznaczenia. Wystąpienie znacznika sterowania na rysunku 4.3 w odniesieniu do czynności *Wprowadź nazwę użytkownika* należy interpretować jako aktualne wykonywanie tej czynności. Po jej wykonaniu znacznik sterowania zostaje przekazany za pośrednictwem przepływu sterowania do kolejnej czynności — *Wprowadź hasło użytkownika*. Z kolei przekazanie znacznika do wyjścia diagramu na rysunku 4.3 powoduje zniszczenie tego znacznika.

Znacznik sterowania

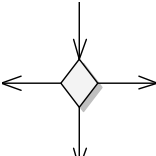
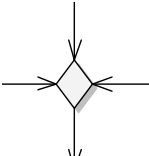
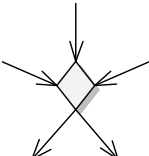
Zdolnością **generowania** znaczników sterowania odznacza się początek diagramu czynności. Z kolei przekazanie dowolnego znacznika sterowania do końca diagramu powoduje **zniszczenie** wszystkich aktualnie występujących na diagramie znaczników

sterowania. W odróżnieniu od powyższej sytuacji, przekazanie dowolnego znacznika sterowania do zakończenia przepływu niszczy tylko ten znacznik sterowania. Znaczniki sterowania mogą być kopiowane bądź niszczone także przez inne elementy diagramów czynności. Przepływ znacznika sterowania w procesach decyzyjnych i współbieżności wyjaśniono w kolejnych podpunktach niniejszego rozdziału.

Przepływy decyzyjne

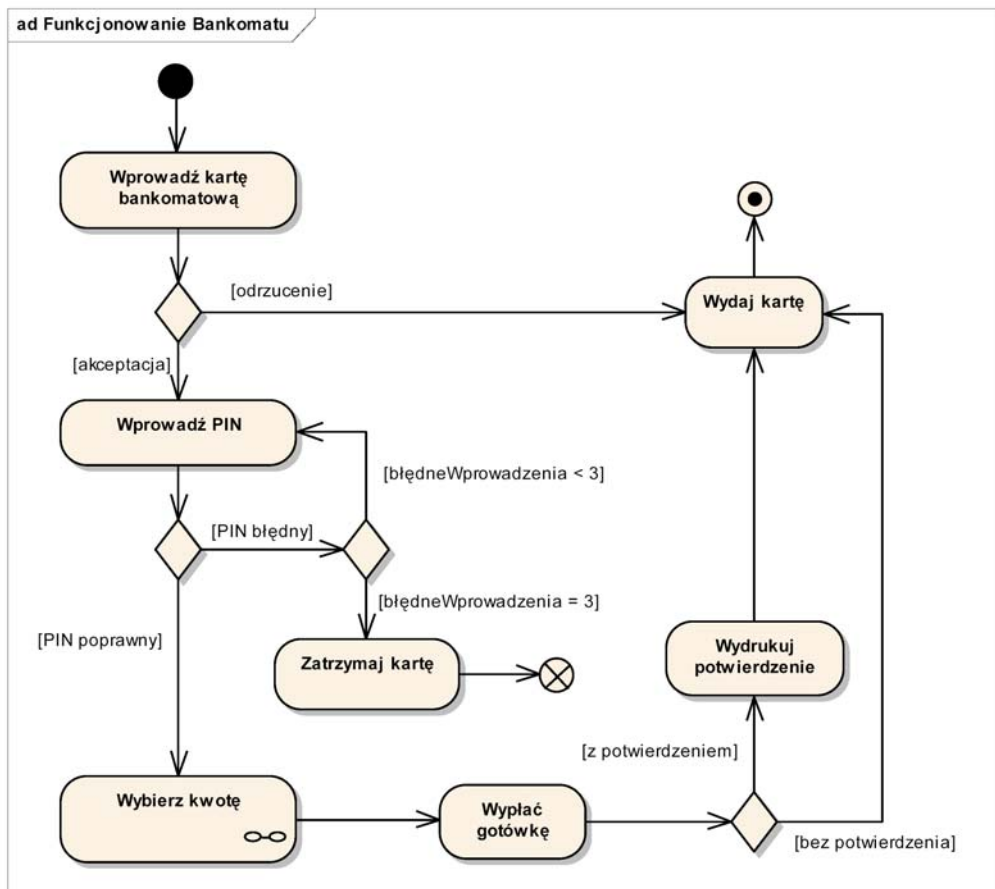
We współczesnym biznesie ma miejsce bardzo wiele złożonych sytuacji decyzyjnych, które znajdują odzwierciedlenie w systemie. Stąd diagramy czynności, w których czynności bądź akcje są uporządkowane w sposób sekwencyjny, należą do rzadkości. Specyfikacja procesu za pomocą tego diagramu oznacza konieczność rozważenia wielu przepływów alternatywnych, uzależnionych od spełnienia warunków czy wykonania iteracji. Sytuacje te można definiować dzięki **blokom decyzyjnym**, które mają charakter decyzji lub złączenia. Bloki te oznaczane są graficznie za pomocą symbolu rombu (tabela 4.3).

Tabela 4.3. Notacja bloków decyzyjnych

Kategoria	Notacja
Decyzja	
Złączenie	
Integracja funkcji decyzji i złączenia	

Decyzja

Wyjście **decyzji** (ang. *decision node*) stanowią dwa lub więcej przepływów sterowania, z których tylko jeden może zostać zrealizowany. Decyzja z zasady charakteryzuje się tym, że ma jeden przepływ wejściowy i przynajmniej parę przepływów wyjściowych. Każdy znacznik sterowania wprowadzany przepływem sterowania do decyzji zostaje odpowiednio przekierowany w zależności od spełnienia warunku. Zasady modelowania przepływów decyzyjnych na diagramach czynności ilustruje rysunek 4.4.



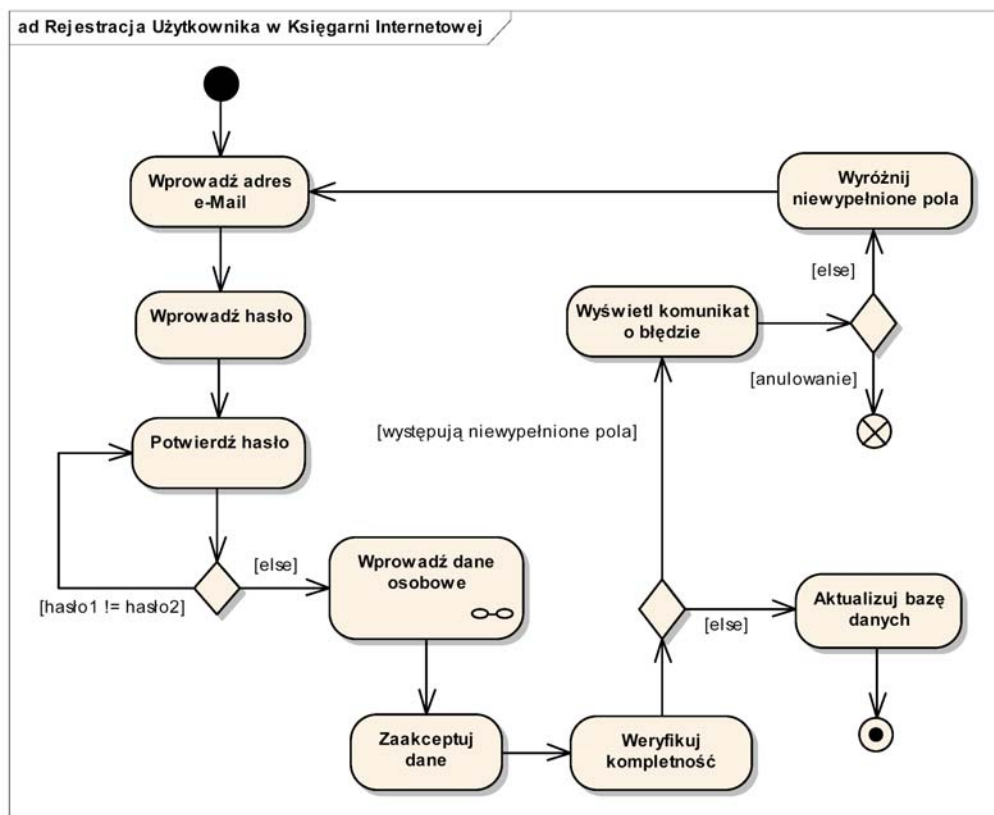
Rysunek 4.4. Funkcjonowanie bankomatu

Wybór jednego z przepływów alternatywnych determinowany jest przez wynik ściśle określonego wyrażenia logicznego określanego mianem **warunku** (ang. *guard condition*). Warunki umieszczane są w nawiasach kwadratowych. Warunki te muszą się wzajemnie wykluczać w odniesieniu do danej decyzji, jak zaprezentowano na rysunku 4.4. Ilustruje on diagram czynności charakteryzujący funkcjonowanie bankomatu. Pierwszą realizowaną czynnością jest *Wprowadzenie karty bankomatowej* do czytnika kart. Czytnik weryfikuje fizyczne cechy karty. Jeżeli odpowiadają one wzorcowi, karta jest akceptowana i klient proszony jest o *Wprowadzenie PIN-u*. W przeciwnym razie następuje *Wydanie karty*. Jeżeli wprowadzony przez klienta numer PIN jest poprawny, oprogramowanie bankomatu przechodzi do realizacji czynności *Wybierz kwotę*. Na zaprezentowanym diagramie została ona potraktowana w sposób ogólny, a następnie uszczegółowiona na rysunku 4.7. Błędne *Wprowadzenie PIN-u* skutkuje natomiast:

- ♦ powrotem do czynności *Wprowadzania PIN-u*, o ile liczba pomyłek jest mniejsza od trzech;
- ♦ *Zatrzymaniem karty* ze względów bezpieczeństwa, jeśli klient pomyli się po raz trzeci.

Po dokonaniu *Wyboru kwoty* następuje *Wyplacenie gotówki*. O ile klient sobie tego życzy, kolejną czynnością jest *Wydrukowanie potwierdzenia*. Ostatecznie następuje *Wydanie karty* klientowi.

Jeden z przepływów wyjściowych danej decyzji można oznaczyć słowem kluczowym *else*. Przepływ ten zostanie zrealizowany tylko i wyłącznie w przypadku niespełnienia warunków zdefiniowanych dla wszystkich innych przepływów wynikowych dotyczących danej decyzji. Zastosowanie słowa kluczowego *else* przedstawiono na rysunku 4.5.

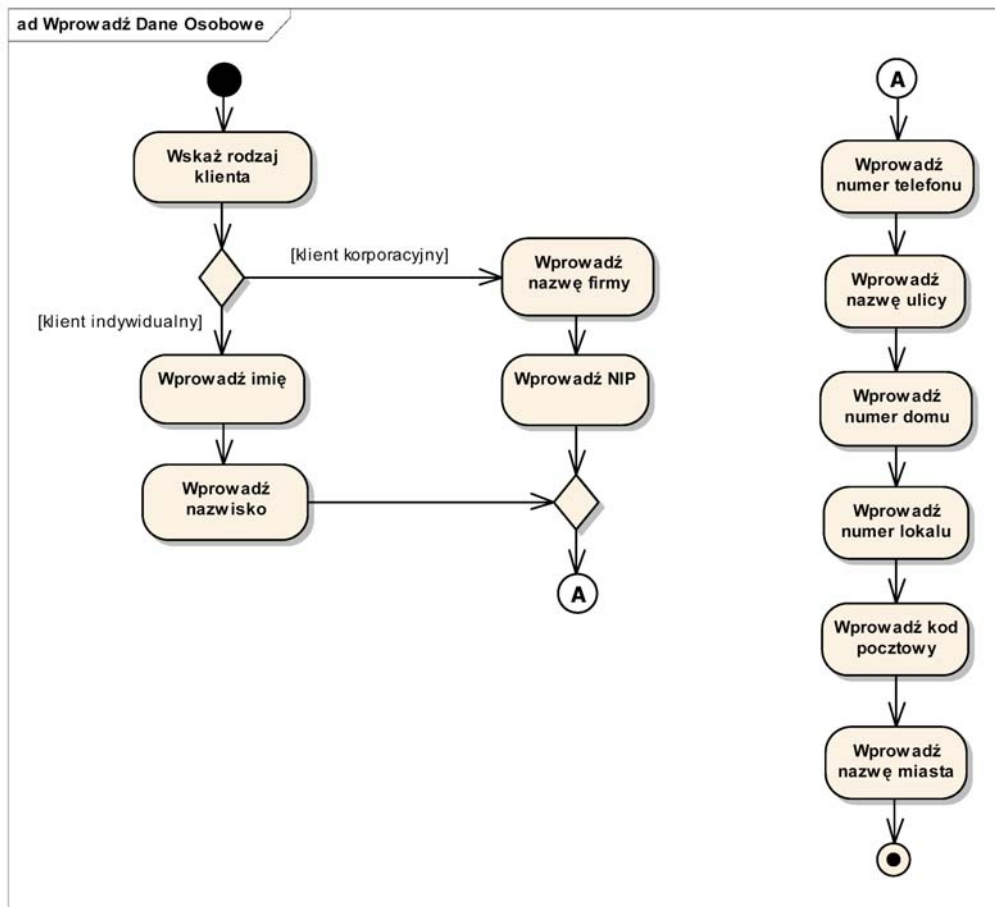


Rysunek 4.5. Rejestracja użytkownika w księgarni internetowej

Rysunek 4.5 przedstawia diagram czynności opisujący proces rejestracji w systemie księgarni internetowej. Pierwszą wykonywaną przez klienta czynnością jest *Wprowadzenie adresu e-mail*, na podstawie którego jest on później identyfikowany w systemie. Przy każdym logowaniu się do systemu klient posługuje się wybranym przez siebie hasłem, toteż na etapie rejestracji muszą zostać zrealizowane czynności *Wprowadź hasło* oraz *Potwierdź hasło*. Przejście do kolejnego etapu rejestracji jest możliwe tylko i wyłącznie w przypadku wprowadzenia na formatce rejestracji tego samego ciągu znaków do dwóch niezależnych pól oznaczających hasło. Procedura ta jest zabezpieczeniem przed popełnieniem technicznego błędu podczas definiowania hasła, a w konsekwencji niemożnością późniejszego zalogowania się do systemu. Zakres danych osobowych niezbędny do poprawnej rejestracji w systemie księgarni internetowej jest

znaczny, zatem *Wprowadzanie danych* opisano na towarzyszącym mu diagramie czynności (rysunek 4.6). Po wypełnieniu formatki rejestracji klient akceptuje wszystkie zadeklarowane dane poprzez naciśnięcie przycisku „zarejestruj” (czynność *Zaakceptuj dane*). Konsekwencją wykonania wskazanej czynności jest *Weryfikacja kompletności* formularza. Tym samym:

- ♦ Jeżeli rejestrujący się użytkownik wypełnił wszystkie pola formatki, realizowana jest czynność *Aktualizuj bazę danych*. Po jej wykonaniu rejestracja ulega zakończeniu.
- ♦ Jeżeli chociaż jedno z wymaganych pól pozostało niewypełnione, system *Wyświetla komunikat o błędzie*, i o ile potencjalny użytkownik wyraża wolę kontynuowania procedury rejestracji, są *Wyróżniane niewypełnione pola*. Formatka rejestracji wyświetlana jest ponownie, co umożliwia uzupełnienie bądź modyfikację wprowadzonych danych. Klient może także zrezygnować z kontynuowania rejestracji, w związku z czym nie będzie mógł w przyszłości korzystać z części transakcyjnej systemu.



Rysunek 4.6. Specyfikacja czynności „Wprowadź dane osobowe”

Łącznik

Zaznaczona na rysunku 4.5 czynność *Wprowadź dane osobowe* została uszczegółowiona na rysunku 4.6. Zachowanie spójności dokumentacji wymusza tożsamość nazwy diagramu uszczegóławiającego i nazwy czynności nadrzędnej. Opisywany diagram uszczegóławiający prowadzi pojęcie **łącznika** (ang. *activity edge connector*). Umożliwia on przerwanie przepływu sterowania i wznowienie go w innym miejscu diagramu czynności. Łączniki są szczególnie użyteczne w efektywniejszym organizowaniu i zwiększaniu przejrzystości złożonego diagramu czynności. Dokumentacja języka UML wskazuje także na możliwość wznowienia przepływu sterowania na odrębnym diagramie czynności.

Jak zaprezentowano na rysunku 4.6, potencjalny użytkownik systemu księgarni internetowej w ramach udostępniania danych osobowych na samym początku *Wskazuje rodzaj klienta*. Następnie — w zależności od tego, czy jest klientem indywidualnym, czy korporacyjnym — wprowadza odpowiednio:

- ♦ imię i nazwisko,
- ♦ nazwę firmy i numer NIP.

Kolejne czynności, realizowane w ustalonej sekwencji, to:

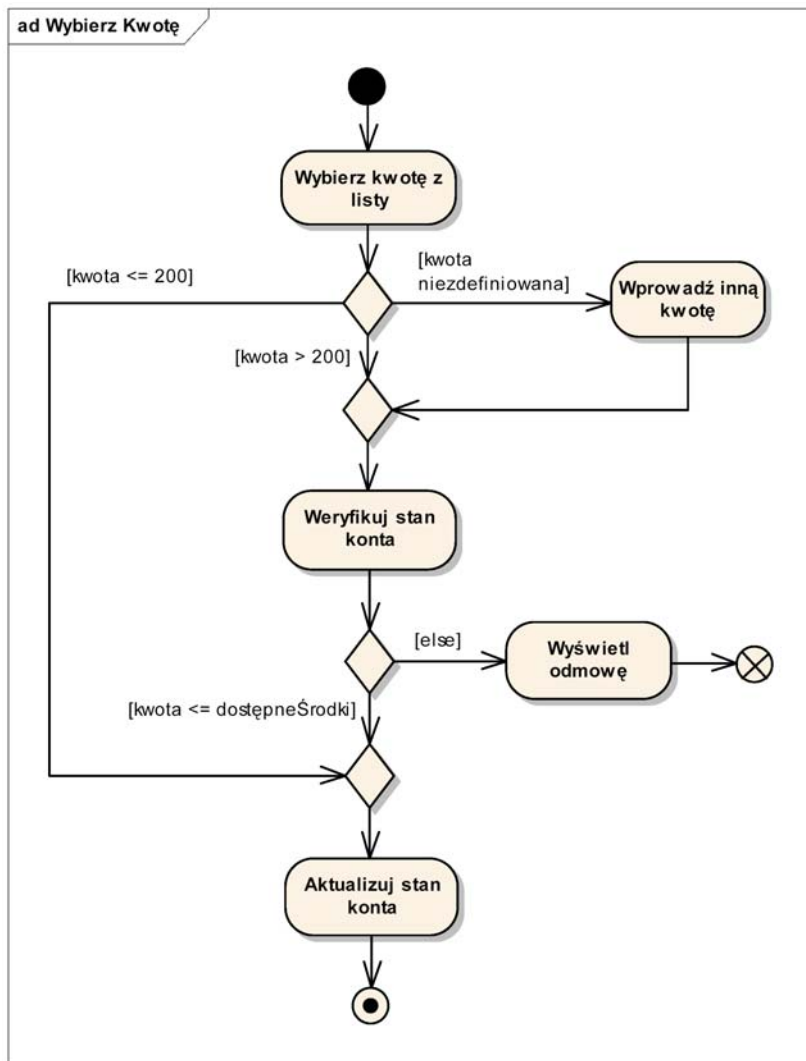
- ♦ *Wprowadź numer telefonu,*
- ♦ *Wprowadź nazwę ulicy,*
- ♦ *Wprowadź numer domu,*
- ♦ *Wprowadź numer lokalu,*
- ♦ *Wprowadź kod pocztowy,*
- ♦ *Wprowadź nazwę miasta.*

Złączenie

Drugą formą bloku decyzyjnego jest **złączenie** (ang. *merge node*). Zawiera ono szereg przepływów wejściowych oraz jeden przepływ wyjściowy. Złączenie nie ma funkcji o charakterze synchronizacyjnym — każdy zrealizowany przepływ sterowania będący wejściem do złączenia pociąga za sobą automatyczne wykonanie wyjściowego przepływu sterowania. Oznacza to, że znacznik sterowania z dowolnego wejściowego przepływu sterowania jest przekazywany do wyjściowego przepływu sterowania. Ideę złączenia przedstawiono na rysunku 4.7.

W celu uwzględnienia złożoności przepływów decyzyjnych na diagramie dozwolona jest specyfikacja decyzji oraz złączenia alternatywnych przepływów w ramach jednego, **zintegrowanego** bloku decyzyjnego. Diagram czynności opracowany na podstawie rysunku 4.7, ilustrujący opisaną sytuację, zaprezentowano na rysunku 4.8.

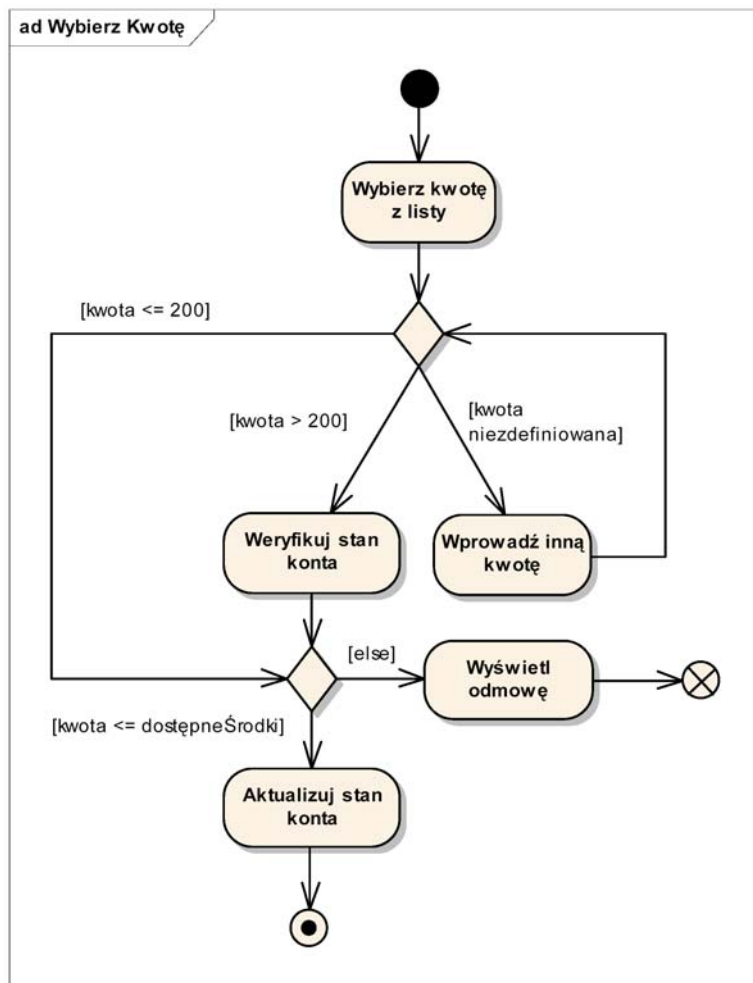
Rysunek 4.7.
Wykorzystanie
złączy
na przykładzie
wyboru kwoty



Rysunki 4.7 i 4.8 dokumentują wybór kwoty w ramach bardziej ogólnego procesu komunikowania się z interfejsem bankomatu w celu wypłacenia gotówki. Pierwszą realizowaną czynnością jest *Wybranie kwoty z listy wyświetlanej na ekranie*. W przypadku wskazania konkretnej sumy wykonywane są następujące czynności:

- ♦ Natychmiastowa *Aktualizacja stanu konta*, o ile wskazana kwota nie przekracza 200 złotych.
- ♦ *Weryfikacja stanu konta*, o ile wskazana kwota jest większa niż 200 złotych. W wyniku pozytywnej weryfikacji następuje *Aktualizacja stanu konta*. W przeciwnym razie interfejs *Wyświetla odmowę*, w związku z czym wybieranie kwoty zostaje przerwane.

Rysunek 4.8.
Zintegrowanie
decyzji i złączenia



Jeżeli klient nie decyduje się wskazać konkretnej pozycji na liście dostępnych kwot, może ręcznie *Wprowadzić inną kwotę*. Dalsze przepływy pomiędzy poszczególnymi czynnościami wykonywane są tak jak pokazano powyżej.

Przepływy współbieżne

Obok przepływów decyzyjnych, w diagramach czynności można dokumentować współbieżne przepływy sterowania. Przybierają one postać rozwidlenia lub scalenia (tabela 4.4).

Rozwidlenie (ang. *fork node*) cechuje się występowaniem jednego wejściowego przepływu sterowania oraz co najmniej dwoma przepływami wynikowymi. Następuje więc rozdzielenie jednego przepływu wejściowego na wiele wynikowych. Przepływ wejściowy inicjuje skopiowanie znacznika sterowania i przekazanie poszczególnych kopii

Tabela 4.4. Notacja wykorzystywana w dokumentowaniu procesów współbieżnych

Kategoria	Notacja
Rozwidlenie	
Scalenie	
Zintegrowanie funkcji rozwidlenia i scalenia	

do wszystkich współbieżnych przepływów. Odwrotną funkcję do rozwidlenia spełnia **scalenie** (ang. *join node*), które oznacza przekazanie sterowania z wielu współbieżnych, wejściowych przepływów sterowania do jednego wynikowego. Rozwidlenia i scalenia są ze sobą merytorycznie związane, lecz liczba wszystkich przepływów wynikowych rozwidlenia nie musi być zgodna z liczbą współbieżnych przepływów wejściowych w scaleniu. W punkcie scalenia równoległe procesy ulegają synchronizacji, natomiast wszystkie znaczniki sterowania z wyjątkiem jednego — zniszczeniu (rysunek 4.9).

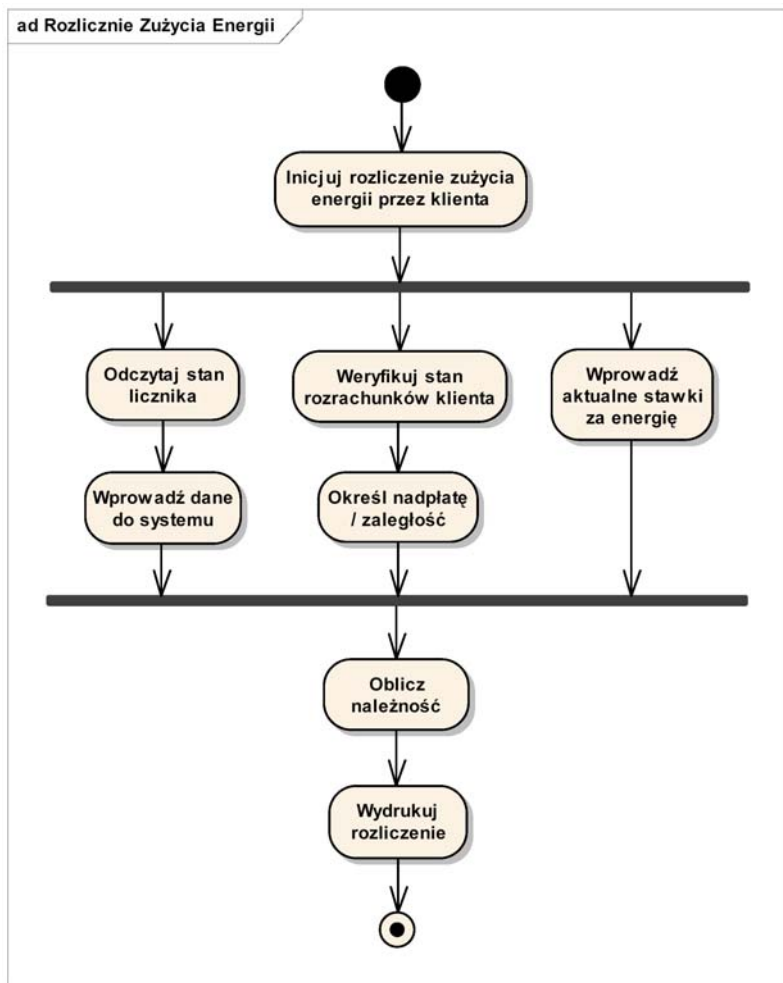
W przypadku scalenia można zdefiniować **specyfikację scalenia** (ang. *join specification*), którą jako ograniczenie zapisuje się na diagramie czynności na wysokości scalenia. Jest to wyrażenie przyjmujące tylko dwie wartości — prawda lub fałsz. W momencie przekazania przez jakikolwiek z przepływów wejściowych znacznika sterowania do scalenia następuje obliczenie wartości wyspecyfikowanego wyrażenia. Znacznik sterowania jest przekierowywany do przepływu wynikowego tylko i wyłącznie w przypadku prawdziwości wyrażenia. Jeżeli natomiast wartością jest fałsz, znacznik sterowania jest niszczone. Przykładowy sposób wskazania na diagramie czynności specyfikacji scalenia zaprezentowano na rysunku 4.10.

Podobnie jak miało to miejsce w przypadku przepływów decyzyjnych, istnieje możliwość zintegrowania funkcji rozwidlenia i scalenia. Opisywaną sytuację przedstawia rysunek 4.11.

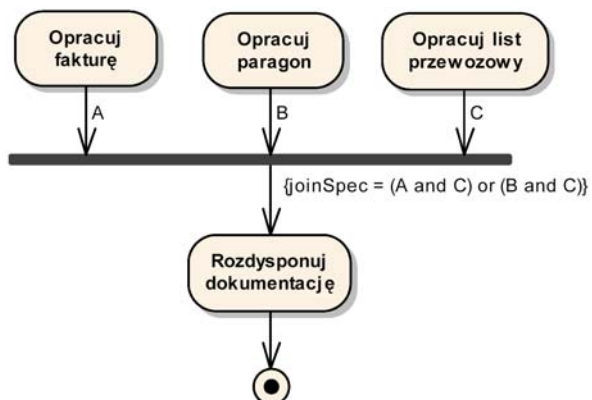
Akcje

Każda z czynności może zostać przedstawiona w formie **szczególnego przypadku** diagramu czynności, w którym w miejsce czynności wprowadza się akcje — elementarne, niepodzielne jednostki zachowania, dynamiki systemu. Diagramy takie służą

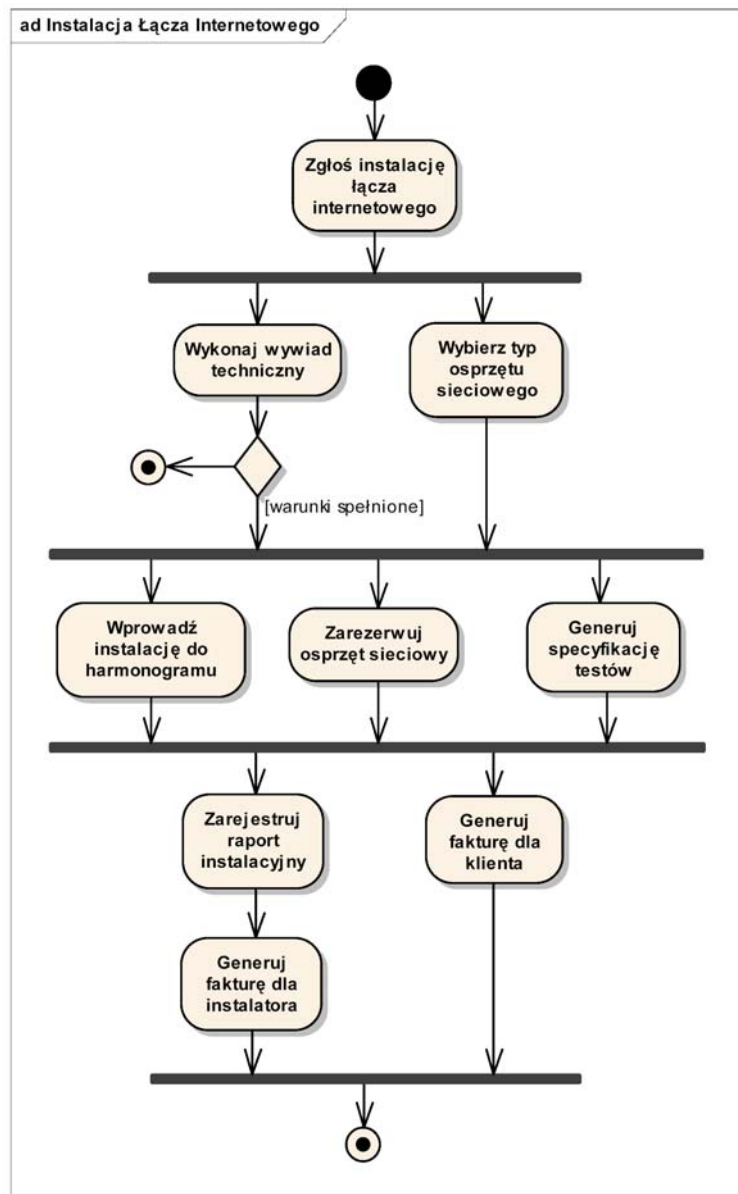
Rysunek 4.9.
Rozliczenie konta odbiorcy energii cieplnej



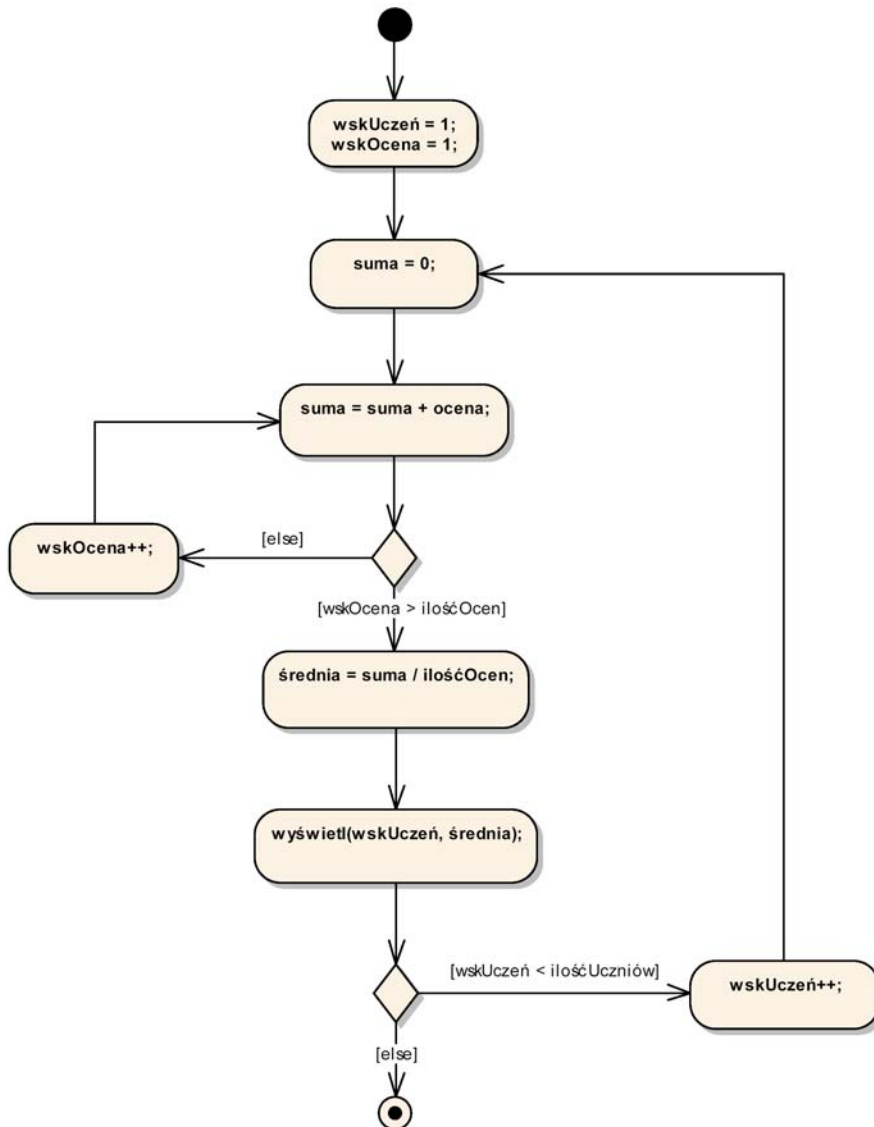
Rysunek 4.10.
Specyfikacja scalenia



Rysunek 4.11.
Zintegrowanie
funkcji rozwidlenia
i scalenia

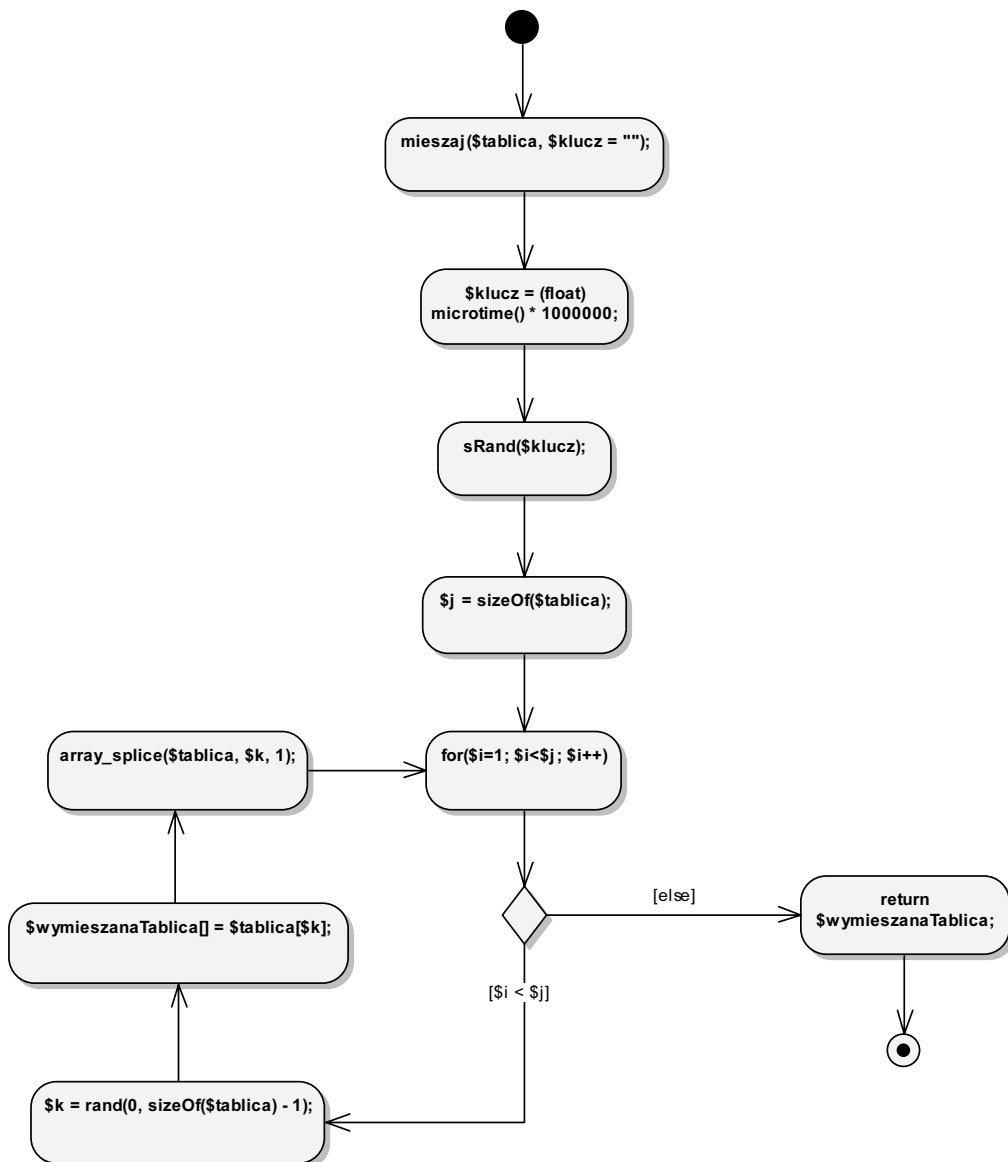


najczęściej do bezpośredniego opracowania programistycznych specyfikacji. Przykładem diagramu czynności zorientowanego na pojęcie akcji jest obliczanie średniej (rysunek 4.12) czy też losowanie pytań (rysunek 4.13). Obydwa przykłady algorytmów dotyczą akcji systemu elektronicznego egzaminowania. Fakt specyfikowania algorytmów przez opisywane diagramy — a zatem bezpośrednie nawiązanie do kodu źródłowego — spowodował, że w wersji UML 2.0 wyróżniono czterdzieści sześć różnych akcji. Umożliwiają one bezpośrednią konwersję akcji na specyfikację kodu języka



Rysunek 4.12. Specyfikacja algorytmu — obliczanie średniej

programowania z wykorzystaniem generatora kodu. Szczegółowa klasyfikacja i charakterystyka poszczególnych akcji dostępna jest w dokumentacji standardu UML [OMG-2004, strony 229 – 314]. Ze względu na liczebność i zakres merytoryczny poszczególnych akcji nie omawia się ich w niniejszym opracowaniu.



Rysunek 4.13. Specyfikacja algorytmu — losowanie pytań

Przepływy danych

Zgodnie z istotą obiektowości poszczególne czynności lub (i) akcje zobrazone na diagramie czynności są wykonywane z udziałem obiektów. Istnieje możliwość bezpośredniego zobrażenia opisanej sytuacji poprzez wprowadzenie **przepływu danych** (ang. *data flow*) pomiędzy danym obiektem a czynnościami lub akcjami mającymi

wpływ na ten obiekt. Przepływy danych umożliwiają bardziej rozbudowaną specyfikację diagramów czynności. Wystąpienie obiektu na diagramie czynności znajduje uzasadnienie w 3 przypadkach:

- ♦ wskazywana jest odpowiedzialność obiektu,
- ♦ obrazowany jest przepływ obiektu,
- ♦ zmieniany jest stan obiektu.

Do **udokumentowania** przepływów danych niezbędna jest umiejętność rozumienia i posługiwania się następującymi kategoriami pojęciowymi:

- ♦ przekaznikiem danych,
- ♦ zestawem przekazników,
- ♦ parametrem czynności,
- ♦ wagami,
- ♦ sygnałami,
- ♦ buforem centralnym,
- ♦ składnicą danych.

Graficznie przepływ danych oznaczony jest analogicznie jak przepływ sterowania. Przepływ ten modeluje przekazanie danych z i do czynności (akcji). Obsługa obiektów w momencie realizacji procesu wypożyczania książki została zaprezentowana na diagramie czynności zamieszczonym na rysunku 4.14.

Przekazniki danych

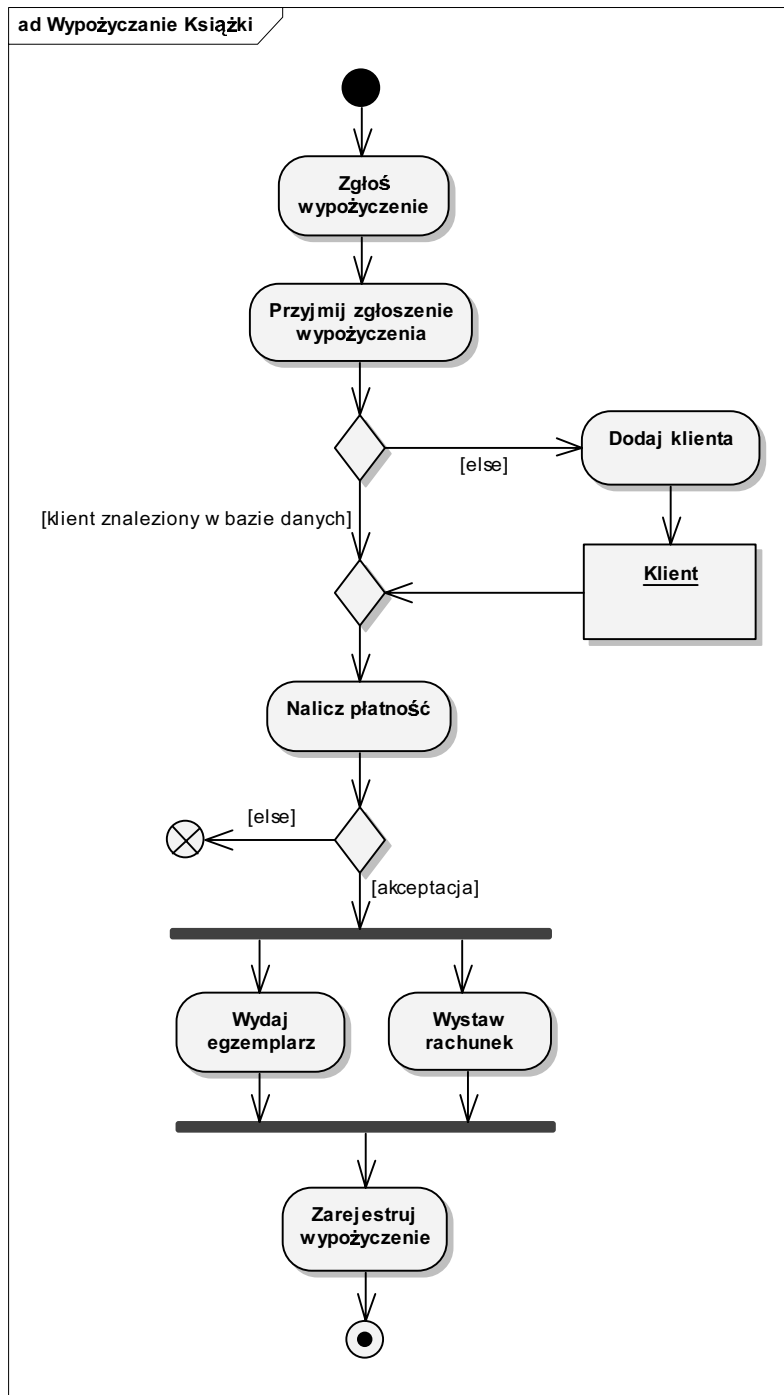
Przepływ obiektów na diagramach czynności pozwala na modelowanie przepływu danych do i z obiektów. Każdy obiekt musi być powiązany z przynajmniej jedną czynnością albo akcją. Najprostszym sposobem zaznaczenia przepływu obiektów na diagramie czynności jest umieszczenie obiektu pomiędzy dwoma czynnościami wraz z zaznaczonym wejściowym oraz wyjściowym przepływem danych. Sposób ten zilustrowano na rysunku 4.15.

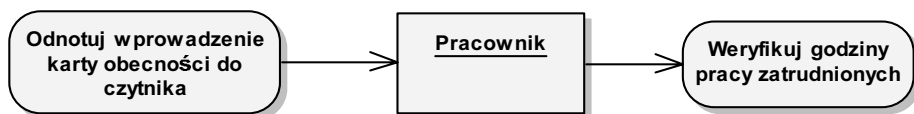
Drugą z możliwości, zmniejszającą graficzne przeładowanie diagramu, jest oznaczenie obiektu poprzez umieszczenie stosownych **przekazników danych** (ang. *pins*) w postaci małych kwadratów na wyjściu i wejściu czynności realizujących przepływ danych do i z obiektu. Sposób ten zaprezentowano na rysunku 4.16.

W obydwu przypadkach w sposób jawny określa się nazwę obiektu bądź przekazników. Przekaznik na diagramie można ponadto przedstawić w sposób anonimowy, sygnalizujący fakt wystąpienia obiektu oraz przepływu danych pomiędzy czynnościami lub (i) akcjami. Tak więc przekazniki mają postać jawną, opisaną na rysunku 4.16, oraz anonimową, jak zaprezentowano na rysunku 4.17.

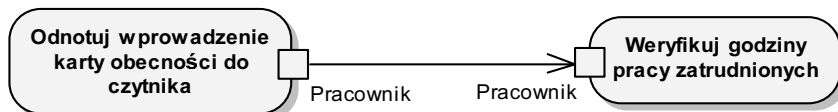
Rysunek 4.14.

*Proces
wypożyczania
książki*





Rysunek 4.15. Przepływ obiektu



Rysunek 4.16. Przepływ obiektu z wykorzystaniem przekazników danych



Rysunek 4.17. Przepływ obiektu — anonimowa notacja przekazników

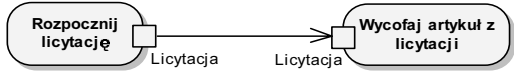
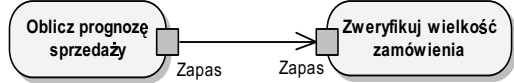
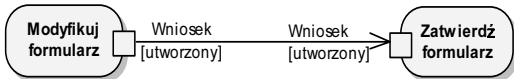
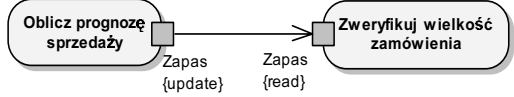
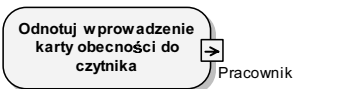
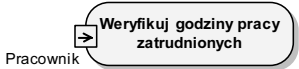
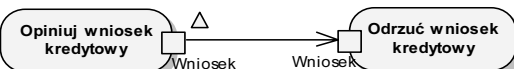
Istnieje szereg możliwości bardziej precyzyjnego przedstawienia przekazników danych. Tabela 4.5 przedstawia wzbogacanie ich notacji stosownie do specyfikowania na diagramie czynności ich szczególnego charakteru.

Przedstawione w tabeli 4.5 obiekty i przepływy danych dotyczą atrybutów pojedynczego obiektu i związanych z nim przekazników danych. Często jednak pomiędzy czynnościami występuje większa liczba przepływów danych przedstawianych za pomocą przekazników danych. Przekazniki te, a więc związane z nimi przepływy danych, mogą być łączone w **zestawy**, które określają uwarunkowania wykonania czynności. Zazwyczaj poszczególne przekazniki danych rozumiane są jako wzajemnie niesprzeczne. W konsekwencji kilka przekazników danych może być wykonanych łącznie — na zasadzie **koniunkcji**. Może jednak wystąpić sytuacja, w której wykonanie jednej grupy zgodnych przepływów danych oznaczonych przekaznikami prowadzi do wykluczenia wykonania innej grupy przepływów. W tym przypadku grupy te są wykonywane na zasadzie **alternatywy**. Przekazniki przetwarzane zgodnie z zasadą koniunkcji łączy się w zestawy, graficznie wyróżnione za pomocą obramowania. Może wystąpić kilka tak wyróżnionych zestawów przekazników danych wejściowych i wyjściowych względem określonej czynności. Obramowane zestawy przekazników wykluczają się i są wykonywane alternatywnie. Każdy przekaznik może należeć do kilku alternatywnych zestawów przekazników danych. Przepływy danych z i do zestawów przekazników mogą być przesyłane do jednej lub wielu czynności (rysunek 4.18).

Jak zaprezentowano na rysunku 4.18, w celu *Zmodyfikowania branżowej segmentacji klienta* należy dysponować danymi dotyczącymi konkretnego klienta oraz wartości przeprowadzonych przez niego transakcji. Pierwsze z nich przechowywane są w obiekcie klasy *Klient*, natomiast drugie wynikają:

- ♦ bezpośrednio z wartości poszczególnych transakcji zapisanych w atrybutach obiektów klasy *Transakcja* powiązanych z danym klientem,
- ♦ pośrednio z aktualnego przyporządkowania klienta do konkretnego segmentu.

Tabela 4.5. Przekazniki danych

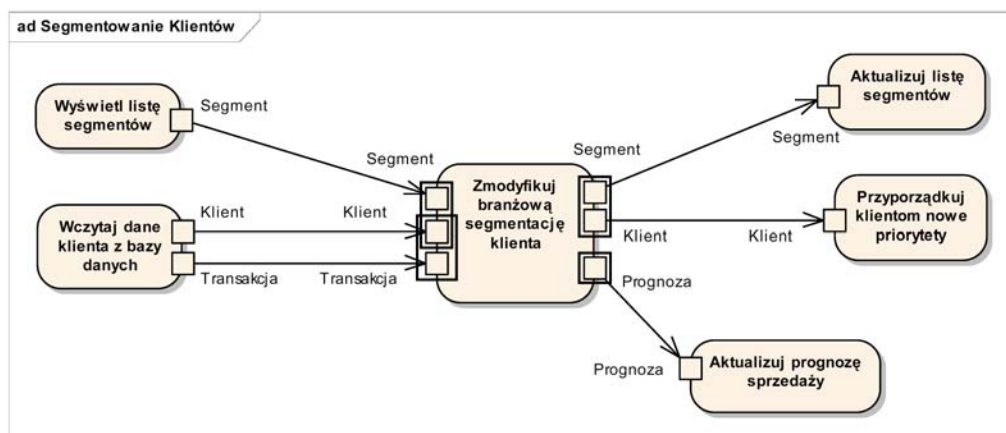
Notacja	Interpretacja
	Przekaznik danych oznaczający przepływ danych z czynności do obiektu.
	Przekaznik danych związany z przepływem danych o charakterze strumieniowym, tj. takim, w którym dane przesyłane pomiędzy czynnościami źródłową a docelową nie są modyfikowane. W sposób jawny akcentuje, że nie są tu realizowane przepływy danych oparte na iteracjach czy współbieżności.
	Przekaznik danych uzupełniony o stan obiektu.
	Przekaznik danych uzupełniony o ograniczenia. Najbardziej typowe ograniczenia dotyczą operacji tworzenia, wyszukiwania, aktualizacji, usuwania danych (CRUD). Poza ograniczeniami typu CRUD można zapisać w nawiasach klamrowych inne ograniczenia. Na rysunku obok zapas jest aktualizowany czynnością <i>Oblicz prognozę sprzedaży</i> , a następnie odczytany przez czynność <i>Zweryfikuj wielkość zamówienia</i> .
	Wejściowy oraz wyjściowy przekaznik danych. Oznaczenie stosowane w przypadku pominięcia bezpośredniego powiązania przekazników danych odpowiednich obiektów. Odpowiednik diagramu zamieszczonego na rysunku 4.16. Pomijanie oznaczenia przepływu danych należy stosować w szczególnych sytuacjach.
	
	Przekaznik danych o charakterze wyjątku. Wyjątek oznacza zainicjowanie czynności nie będących czynnościami rutynowymi w kontekście danego procesu. Na rysunku obok sytuacją wyjątkową w kontekście przyznawania kredytu jest <i>Odrzucenie wniosku kredytowego</i> .

Tak więc realizacja czynności *Zmodyfikuj branżową segmentację klienta* wymaga wykonania alternatywnych zestawów przepływów danych pobierających atrybuty obiektów:

- ♦ klasy *Klient* oraz klasy *Transakcja*,
- ♦ klasy *Klient* oraz klasy *Segment*.

W wyniku *Zmodyfikowania branżowej segmentacji klienta* można alternatywnie:

- ♦ dokonać *Aktualizacji prognozy sprzedaży* na podstawie danych zawartych w obiekcie klasy *Prognoza*;
- ♦ przeprowadzić *Aktualizację listy segmentów* oraz *Przyporządkować klientom nowe priorytety* na podstawie danych przechowywanych odpowiednio w obiektach klas *Segment* i obiekcie klasy *Klient*.

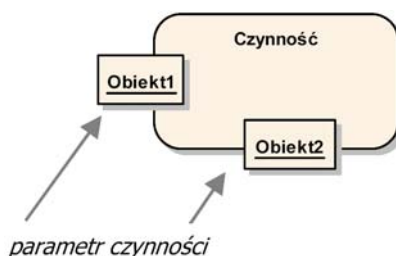


Rysunek 4.18. Zestawy przekaźników sterowania w segmentowaniu klientów

Parametr czynności

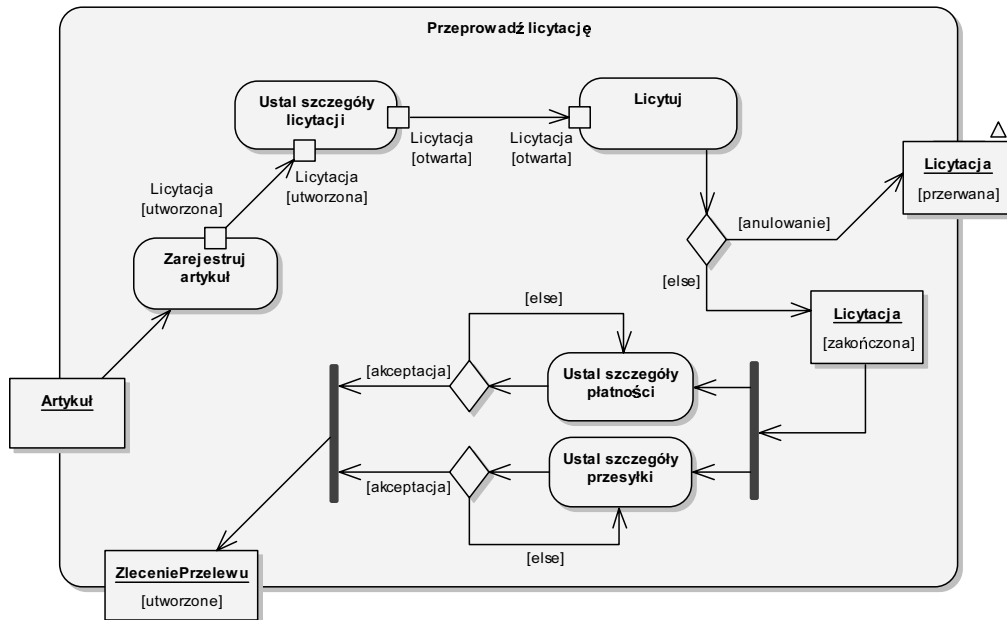
Parametry czynności (ang. *activity parameter nodes*) pełnią rolę uzupełniającą w opisie przepływów danych w dekomponowanym, hierarchicznie uporządkowanym diagramie czynności. Notacja parametru czynności przedstawia się jak na rysunku 4.19. Oznacza ona, że na obwodzie czynności umieszcza się symbol obiektu lub obiektów spełniających rolę parametrów.

Rysunek 4.19.
Notacja parametrów
czynności



Jednym z opcjonalnych elementów parametru jest stan obiektu. Wewnątrz parametryzowanej czynności można na diagramie czynności zilustrować zaawansowane kategorie pojęciowe, które są reprezentowane zgodnie z logiką ich stosowania. Praktyczne zastosowanie parametrów czynności ilustruje rysunek 4.20.

Jak pokazano na rysunku 4.20, *Przeprowadzanie licytacji* jest złożonym zachowaniem systemu. Obejmuje wiele podczynności, przepływów sterowania, przepływów danych, parametrów oraz innych podstawowych i zaawansowanych kategorii pojęciowych. Opisywany diagram przedstawia specyfikowanie czynności *Przeprowadź licytację* z zastosowaniem parametrów. Parametrami są w tym przypadku wyszczególnione i zapisane na granicach czynności obiekty, tj. *Artykuł*, *Licytacja* oraz *ZleceniePrzelewu*. Oznacza to, że wymienione obiekty są przetwarzane przez podczynności będące składowymi *Przeprowadzania licytacji*. Jak pokazano, parametry w postaci obiektów mogą



Rysunek 4.20. Przeprowadzanie licytacji

być przekazywane do czynności (obiekt *Artykuł*), jak również pobierane z czynności (obiekty *Licytacja* oraz *ZleceniePrzelewu*). Logika licytacji w systemie obejmuje podczynności:

- ♦ *Zarejestruj artykuł*,
- ♦ *Ustal szczegóły licytacji*,
- ♦ *Licytuj*,
- ♦ *Ustal szczegóły płatności*,
- ♦ *Ustal szczegóły przesyłki*.

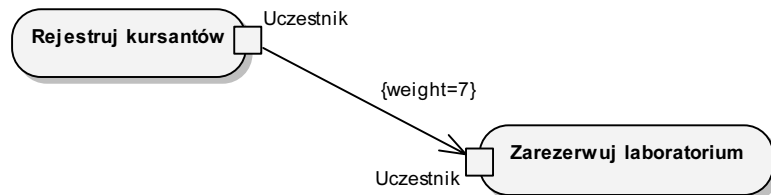
Dwie ostatnie z wymienionych czynności realizowane są współbieżnie. Mogą być powtarzane dowolną ilość razy — aż do osiągnięcia jednomyślności wszystkich uczestników danej licytacji.

Wagi

Podczas modelowania obiektów można posłużyć się kategorią pojęciową **wagi** (ang. *weight*). Waga jest ograniczeniem, więc zaznacza się ją w nawiasach klamrowych nad przepływem. Wskazuje minimalną liczbę znaczników sterowania, które muszą być przekazane ze źródła do czynności lub akcji docelowej. Źródłem może być wyłącznie obiekt. Jeżeli waga wynosi 0, oznacza to, że wszystkie znaczniki sterowania są przekazywane do czynności docelowej, niezależnie od ich liczby. Jeśli natomiast liczba znaczników sterowania zaznaczonych w wadze nie spełni określonego minimum, wówczas żaden ze

znaczników nie zostanie przekazany do czynności docelowej. Interpretację rozumienia wagi przedstawiono na rysunku 4.21. Waga przybierająca postać ograniczenia {waga=7} oznacza, że należy *Zarezerwować laboratorium* tylko w przypadku, gdy na kurs zgłosi się minimum siedmiu uczestników. Inny przykład, dotyczący minimalnej liczby dwustu faksów do zainicjowania *Archiwizacji plików*, zawiera rysunek 4.25.

Rysunek 4.21.
Zastosowanie wagi
na diagramie



Sygnały

Na diagramach czynności mogą występować sygnały. Przesyłanie sygnałów ma charakter asynchroniczny. A zatem **sygnał** (ang. *signal*) można zdefiniować jako specyfikację asynchronicznego bodźca inicjującego czynność lub akcję. Wyróżnia się sygnały nadawcze oraz odbiorcze, wysyłane i odbierane przez czynności i akcje. Oznaczenia sygnałów przedstawia tabela 4.6.

Tabela 4.6. Oznaczenia sygnałów

Nazwa	Notacja
Sygnał nadawczy	
Sygnał odbiorczy	
Czas	

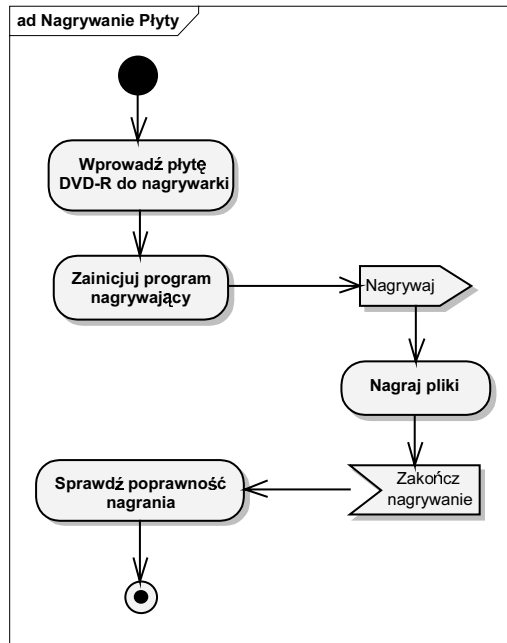
Role sygnałów nadawczych i odbiorczych w procesie nagrywania płyt CD przedstawia rysunek 4.22, natomiast sygnały czasowe na diagramach czynności w odniesieniu do systemu harmonogramowania zadań przedstawiono na rysunku 4.23.

Bufor centralny

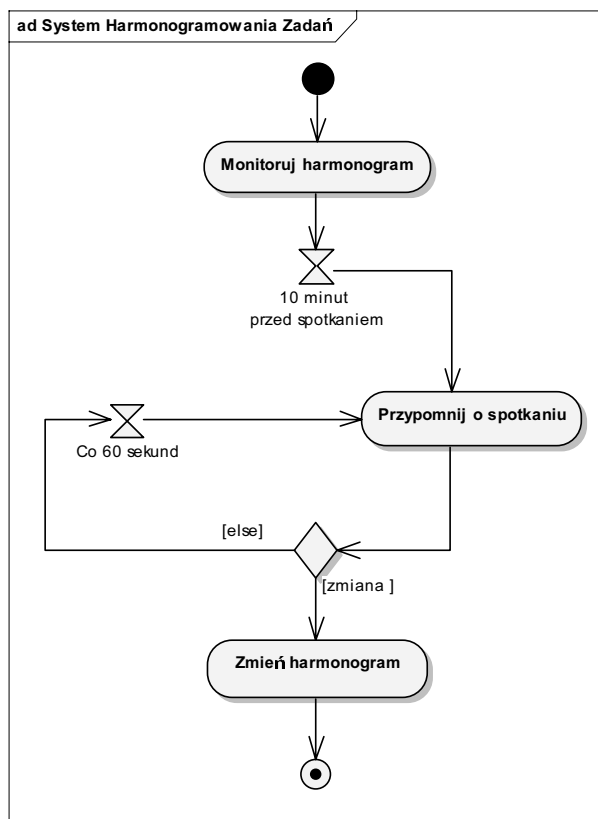
We współczesnych systemach informatycznych, procesach biznesowych czy zaawansowanych procesach przetwarzania danych istnieje potrzeba wdrożenia procedur kolejkowania. Rozwiązaniem tego problemu na poziomie projektowania systemu w języku UML 2.0 jest kategoria **buforu centralnego** (ang. *central buffer node*). Jest on specyficznym obiektem służącym do zarządzania przepływami danych z różnych obiektów źródłowych i do różnych docelowych. Bufor nie jest powiązany bezpośrednio

Rysunek 4.22.

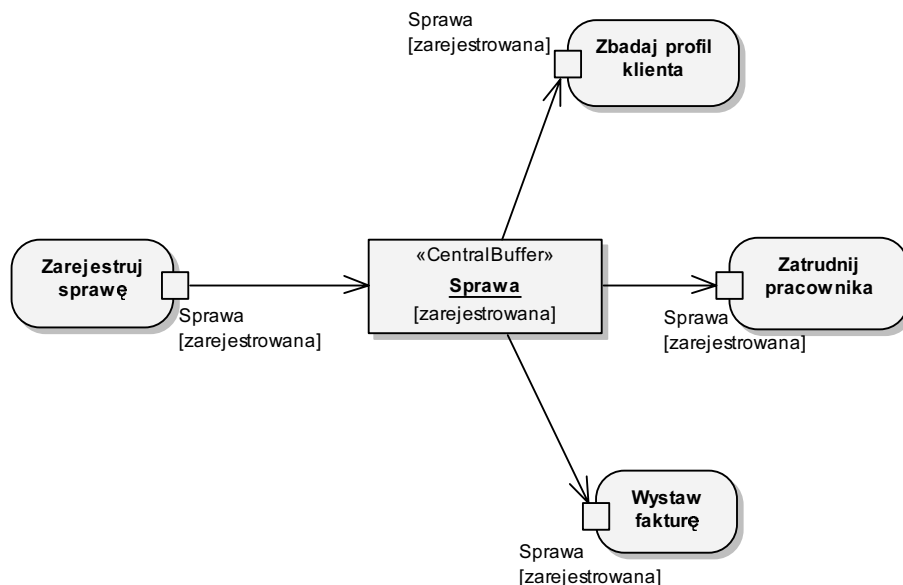
*Zastosowanie
sygnałów w procesie
nagrywania płyt DVD*

**Rysunek 4.23.**

*System
harmonogramowania
zadań*



z czynnościami. Obiekty źródłowe (wejściowe) są przekazywane wraz ze znacznikami sterowania do bufora centralnego. Bufor centralny przyjmuje obiekty i przekazuje je do obiektów wynikowych. Przechowanie obiektów i ich znaczników w buforze umożliwia ich dalsze przesyłanie w celu wykonania określonych czynności. Bufor centralny oznacza się stereotypem «*CentralBuffer*» i nadaje nazwę przyjmowanych, przechowywanych i przekazywanych obiektów o określonym stanie. Bufor centralny różni się od innych obiektów tym, że nie jest powiązany z czynnościami czy też akcjami na zasadzie przekaźnika danych. Rysunek 4.24 przedstawia przepływ spraw w systemie elektronicznego obiegu dokumentów WFM.



Rysunek 4.24. Przepływ spraw w systemie elektronicznego obiegu dokumentów WFM

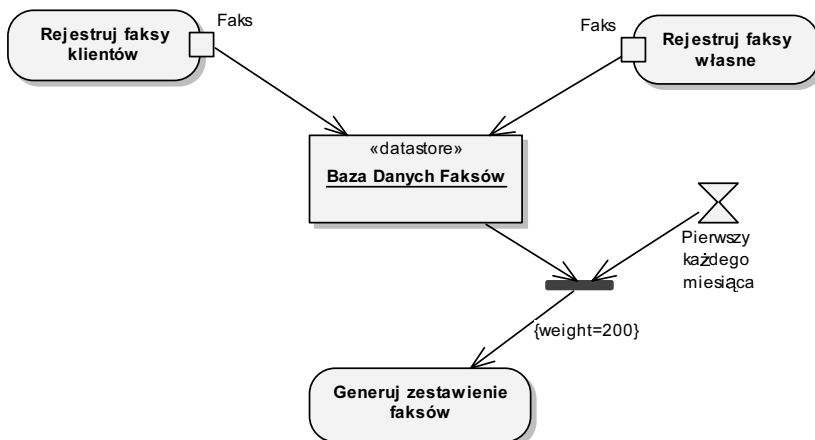
Podczas wykonywania czynności *Zarejestruj sprawę* system generuje obiekt *Sprawa*. Jest on przekazywany w stanie *zarejestrowana* do bufora centralnego. Przyjmuje on obiekt sprawy wraz ze znacznikiem sterowania i automatycznie ustawia go w odpowiednim miejscu kolejki. W momencie nadejścia obsługi odpowiedniego zarejestrowanego obiektu sprawy, bufor centralny — w zależności od rodzaju sprawy — przekazuje obiekt i jego znacznik sterowania do inicjowanej czynności. I tak w przypadku sprawy dotyczącej *Zbadania profilu klienta* znacznik sterowania trafia do działu marketingu. Jeśli sprawa dotyczy *Zatrudnienia pracownika*, znacznik wraz z obiektem przesyłany jest do działu kadr, natomiast jeśli dotyczy *Wystawienia faktury*, sprawa przekazywana jest do księgowości.

Składnica danych

Składnica danych (ang. *data store node*) jest rodzajem bufora centralnego dla danych stałych przechowywanych w dłuższym okresie. Na diagramie czynności oznacza się ją stereotypem «*datastore*». Przykładem są tu dane kadrowe czy dotyczące technologii

wyrobów. W przypadku wysyłania danych ze składnicy danych dotyczących konkretnego obiektu jest on kopiowany. W składnicy pozostają dane stałe, które jednocześnie mogą być pobrane dla potrzeb realizacji określonej czynności przez czynność będącą na zewnątrz składnicy danych. Znacznik sterowania tego obiektu ulega skopiowaniu. Stąd też składnica danych jest powiązana bezpośrednio z czynnościami i innymi elementami diagramu, podczas gdy bufor łączył się jedynie z obiektami. Funkcja buforu jest jednak bardziej zaawansowana — dotyczy kolejkovania obiektów, podczas gdy składnica jedynie pasywnie przechowuje dane obiektów (rysunek 4.25).

Rysunek 4.25.
Składnica danych systemu obsługi faksów



W firmie realizującej zdalnie usługi zaufania (np. usługi notarialne oraz certyfikacyjne), oprócz poczty elektronicznej standardowo wykorzystywany jest faks. System informatyczny FaksPro został zaprojektowany do obsługi za pomocą modemu i komputera klasy PC faksów przychodzących i wychodzących. System przechowuje w składnicy danych szczegółowe informacje o każdym fakse przychodzącym lub wychodzącym. Składnica danych przechowuje następujące informacje o każdym fakse: data wysłania, data odebrania, numer telefonu, treść. Faksy wysyłane i przychodzące są archiwizowane w *BazieDanychFaksów* w wyniku *Rejestrowania faksów klienta* oraz *Rejestrowania faksów własnych*. W momencie wykonywania czynności, która operuje na danych przechowywanych w składnicy, system kopiuje znacznik sterowania obiektu, przekazując go do odpowiedniej czynności lub akcji. Funkcjonalność aplikacji obejmuje również *Generowanie zestawień faksów*. Pierwszego dnia każdego miesiąca system podejmuje próbę wygenerowania zestawienia. Jeżeli liczba faksów wynosi 200 lub więcej, system wykonuje wspomniane zadanie. W innym przypadku lista faksów nie jest drukowana. System podejmuje kolejną próbę wygenerowania listy faksów w następnym miesiącu.

Partycje diagramów czynności

Diagramy czynności w wersji podstawowej pozwalają na określenie przepływu sterowania i danych pomiędzy czynnościami a obiektami. Opis ten może być wzbogacony, we wskazanym powyżej zakresie, o dodatkowe kategorie zaawansowane, takie jak oznaczenia przepływów decyzyjnych czy współbieżnych. Taki diagram czynności może

całkowicie spełniać zadania, które zostały określone w punkcie *Znaczenie diagramów czynności*. Jednak istnieje możliwość uwzględnienia na tych diagramach kolejnych elementów opisu, takich jak miejsce realizacji czynności czy wskazanie instancji klasyfikatora odpowiedzialnej za jej funkcjonowanie. Zaprezentowanie tych informacji wymaga zmiany układu graficznego diagramu czynności. Efekt ten osiąga się poprzez zastosowanie w starszych wersjach UML (1.x) **torów**, a w wersji 2.0 bardziej złożonej kategorii pojęciowej — **partycji** (ang. *activity partitions*).

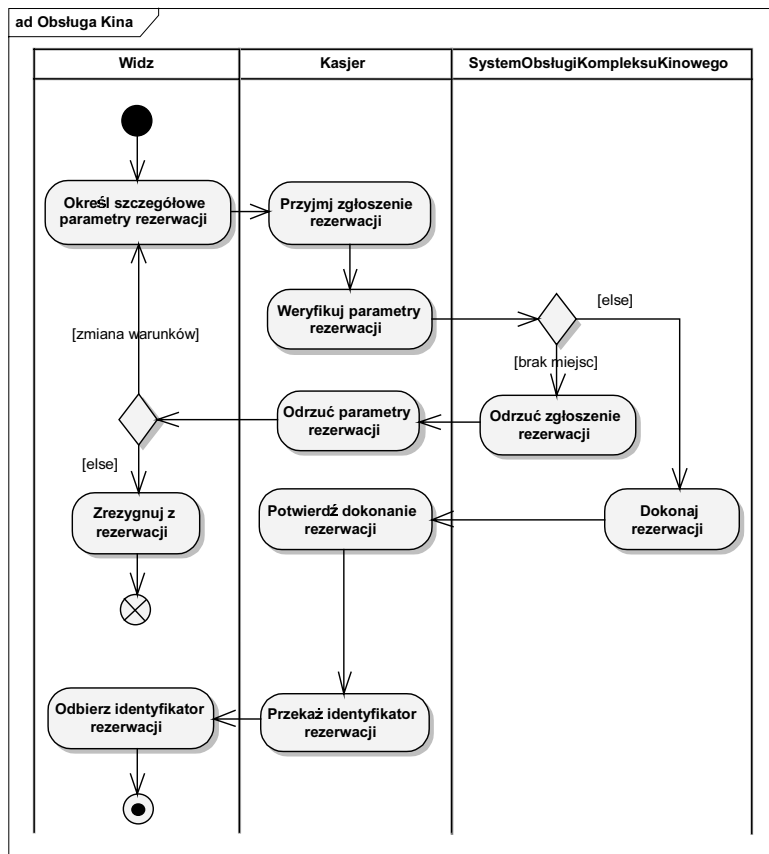


Partycja jest mechanizmem grupowania elementów diagramu czynności powiązanych przepływami sterowania i przepływami danych, pełniących określoną, wspólną rolę na diagramie.

Partycja jest wyznaczana na diagramie w postaci pionowego lub poziomego toru, w którym pomiędzy dwoma równoległymi liniami występują poszczególne elementy diagramu czynności. Każda partycja jest jednoznacznie identyfikowana poprzez swoją nazwę. Na rysunku 4.26, który dokumentuje proces obsługi kina, uwzględniono trzy instancje klasyfikatorów — *Widza*, *Kasjera* oraz *SystemObsługiKompleksuKinowego*. Odpowiadają im partycje o tych samych nazwach.

Rysunek 4.26.

Wykorzystanie partycji w opisie obsługi kina



Partycja obejmuje wyłącznie funkcjonalność reprezentowaną przez zawarte w niej elementy diagramu czynności. Partycje mogą być hierarchicznie podzielone na **subpartycje** (ang. *activity subpartitions*). Partycja lub subpartycja tworzona jest na zasadzie wyodrębnienia jednorodnego procesu, instancji klasyfikatora, zjawiska czy też zmiennej występującej w biznesie. Taki jednorodny proces, instancję klasyfikatora, zjawisko czy zmienną — reprezentowaną w partycji lub subpartycjach — określa się mianem **kryterium** (ang. *dimension*). Przykładami kryteriów są:

- ♦ organizacja firmy;
- ♦ miejsca powstawania kosztów;
- ♦ położenie geograficzne;
- ♦ aktorzy i inne instancje klasyfikatorów;
- ♦ wartości i atrybuty.

Tak więc dla partycji *JednostkaOrganizacyjna* kryterium jest *OrganizacjaFirmy*, natomiast subpartycjami *DziałKadr*, *DziałSprzedaży* czy też *DziałMarketingu*.

Ten sposób deklarowania kryteriów partycji można zrealizować na diagramie czynności w układzie pionowym lub poziomym. Stąd najczęściej wyróżniane są dwa odrębne kryteria. W określonych, bardziej złożonych procesach biznesowych można osiągnąć większy stopień szczegółowości dzięki przecięciu kryterium poziomego z pionowym. Każdy cząstkowy obszar przecięcia odpowiadający obydwu kryteriom partycji (subpartycji) tworzy **komórkę** (ang. *cell*). Zawiera ona funkcjonalność wspólną dla obu partycji (rysunek 4.27). Na diagramie czynności można wyodrębnić większą liczbę kryteriów i odpowiadających im partycji w układzie pionowym bądź poziomym. Powoduje to jednak wzrost złożoności tworzonego diagramu. Sytuacja to dotyczy rozbudowanych, wielokryterialnych przypadków użycia.

Rysunek 4.27 przedstawia przetwarzanie wymagania systemowego. Współczesne przedsiębiorstwa często zdobywają przewagę nad konkurentami dzięki wdrożeniu nowoczesnego zaawansowanego systemu informatycznego i jego efektywnej eksploatacji. Eksploatacja systemu jest częścią jego cyklu życia i wymaga sprawnego zarządzania wymaganiami użytkowników i zmianami systemu. Nowoczesne oprogramowanie służące do zarządzania wymaganiami systemowymi opiera się na technologii WFM. Dostęp do niego mają użytkownicy z uprawnieniami do zgłaszania wymagań systemowych oraz menedżerowie, którzy mogą akceptować i modyfikować status oraz kolejność realizowania zmian systemu. Przedstawienie zależności pomiędzy pracownikami dysponującymi różnymi prawami dostępu do różnych aplikacji i ich modułów może okazać się trudnym zadaniem nawet dla doświadczonego analityka. Wynika to między innymi z konieczności uwzględnienia logiki przepływów danych i czynności zarówno systemów oprogramowania, jak i biznesowych. Aby wiernie przedstawić te zależności, wykorzystano macierzowy opis partycji, składający się z:

- ♦ kryterium poziomego, tj. partycji *ZarządzanieWymaganiamiWFM*,
- ♦ kryterium pionowego, tj. partycji *Pracownik*.

że użytkownik oczekuje jego szybkiego zaimplementowania — wymaganie jest przekazywane do *KierownikaDziałuIT*. Jeśli wymaganie ma status inny niż krytyczny, trafia do konkretnej kolejki wymagań oczekujących na realizację. Następnie *UżytkownikSystemuBIS* analizuje planowane terminy realizacji zgłoszonego przez siebie wymagania. I tak:

- ♦ jeśli planowany termin realizacji jest satysfakcjonujący, przepływ czynności zaprezentowany na diagramie ulega zakończeniu;
- ♦ Jeśli termin realizacji wymagania jest dla *UżytkownikaSystemuBIS* zbyt odległy, zgłasza on nowe wymaganie, dotyczące zmiany terminu realizacji wymagania pierwotnego.

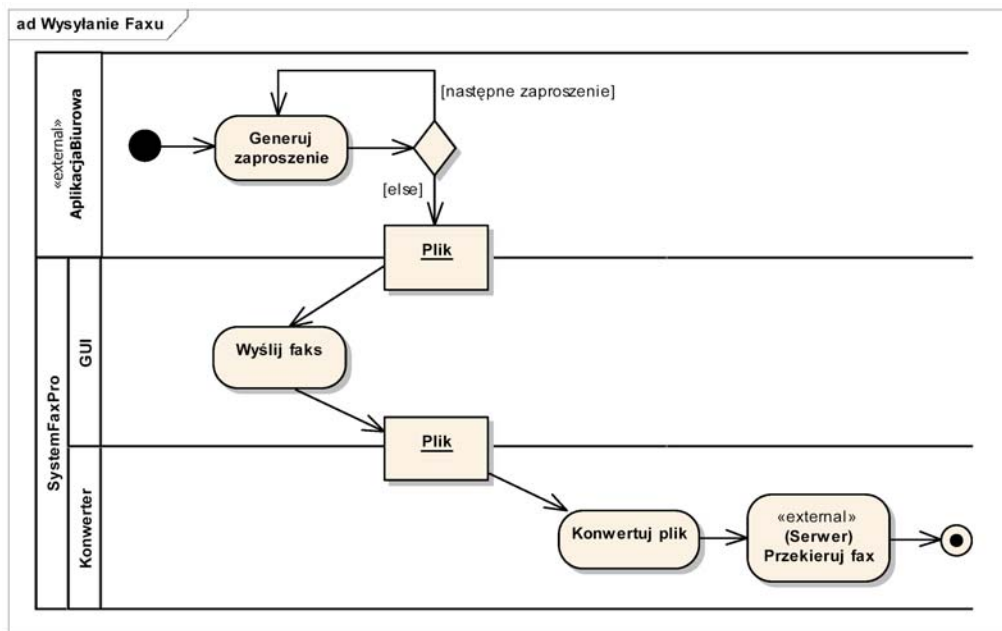
W odniesieniu do wymagań przekazanych do *KierownikaDziałuIT* następuje *Analizowanie użyteczności* wymagania. Czynność jest wykonywana w ramach subpartycji *ObsługaWymagań*. W przypadku potwierdzenia wysokiej użyteczności wymagania jest ono akceptowane, a *KierownikDziałuIT* równolegle *Aktualizuje harmonogram* oraz *Przydziela specjalistów*. W przypadku uzyskania innych wyników *Analizy użyteczności* następuje *Usunięcie wymagania z kolejki*. W obu przypadkach *UżytkownikSystemuBIS* może *Sprawdzić status wymagania*. Jeśli zostanie ono zaakceptowane, użytkownik *Analizuje terminy realizacji* wymagania. Wykorzystuje w tym celu *HarmonogramRealizacjiWymagań*. Po zapoznaniu się z planowanymi terminami realizacji *UżytkownikSystemuBIS* podejmuje decyzję. Przepływ sterowania jest w tym momencie realizowany analogicznie jak w przypadku wymagań o statusie innym niż krytyczny.

Tak więc specyfikację partycji na danym diagramie czynności należy rozpocząć od wyodrębnienia kryteriów, wg których czynności będą grupowane. Czynności te korespondują z partycjami. Występują tu następujące **możliwości specyfikacji**:

- ♦ wyodrębnianie partycji w układzie pionowym lub poziomym;
- ♦ wyodrębnianie subpartycji w ramach hierarchicznie uporządkowanych poziomych lub pionowych partycji;
- ♦ wyodrębnianie partycji oraz związanych z nimi subpartycji w układzie macierzowym w celu określenia komórek.

Obok partycji wewnętrznych, dotyczących dokumentowanego przypadku użycia, na diagramie czynności można uwzględnić **partycje zewnętrzne**, które dotyczą obiektów i czynności będących poza zakresem funkcjonalnym dokumentowanego przypadku. Czynności lub obiekty użytkowane przez sąsiadujące partycje umieszcza się na **granicznych liniach** partycji (rysunek 4.28).

System informatyczny FaksPro został zaprojektowany do obsługi faksów przychodzących i wychodzących przy użyciu modemu i komputera klasy PC. Funkcjonalność aplikacji obejmuje również generowanie raportów o faksach (plikach) przychodzących i wychodzących. Raport taki zawiera datę wysłania lub odebrania faksu, liczbę faksów w danym okresie, numery odbiorców, treść itd. Często stosowaną funkcją aplikacji jest wysłanie treści pliku utworzonego w zewnętrznej *AplikacjiBiurowej* z uwzględnieniem konwersji najpopularniejszych formatów plików. Aby wykonać funkcję konwersji, system wykorzystuje wbudowany *Konwerter*. Zmienia on źródłowy format pliku



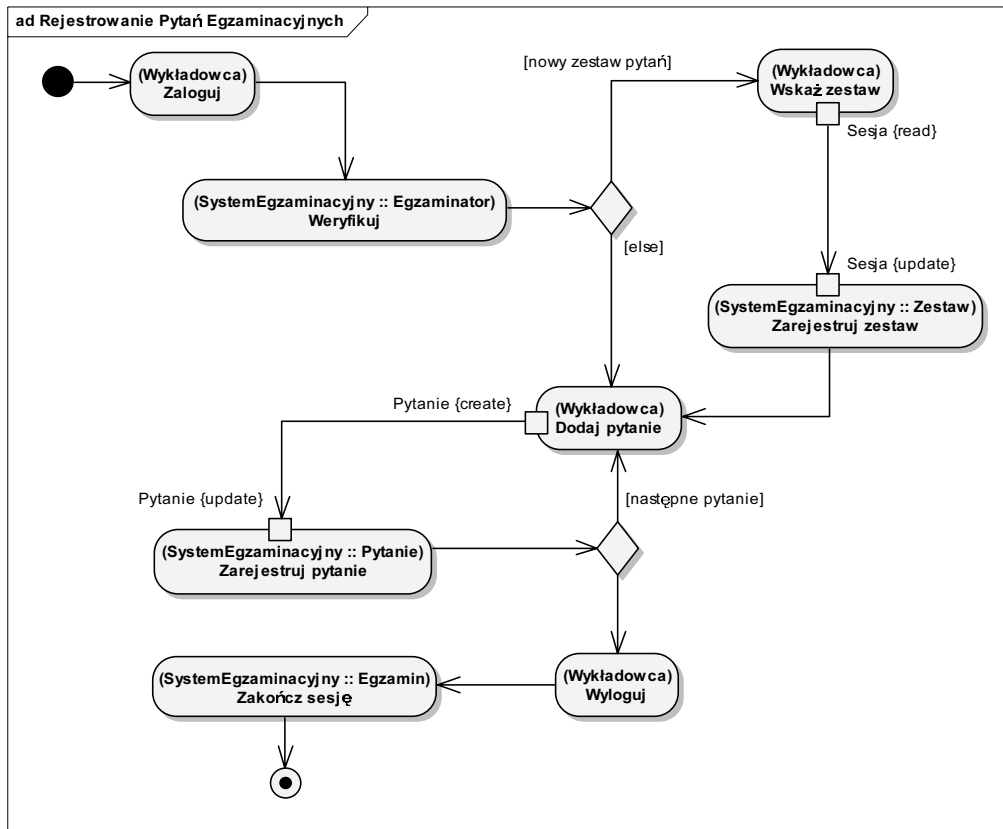
Rysunek 4.28. Partycje i subpartycje wewnętrzne oraz zewnętrzne

na odpowiedni, zgodny z modulem wysłania wiadomości. W aplikacji biurowej generowane są zaproszenia w postaci *Pliku*. Plik przekazywany jest do partycji *SystemFaxPro*, złożonej z dwóch subpartycji. W pierwszej z nich na podstawie pliku realizowana jest czynność *Wyślij faks*. W wyniku tej czynności plik jest przekazywany do *Konwertera*, gdzie wykonywane są kolejno czynności *Konwertuj plik* i *Przekieruj faks*.

Partycje wizualizują przepływy sterowania i danych pomiędzy poszczególnymi czynnościami (akcjami) i obiektami w odniesieniu do wybranych kryteriów. Przy bardzo rozbudowanych diagramach czynności można posłużyć się **alternatywną notacją**, która polega na eliminowaniu linii granicznych poszczególnych partycji i subpartycji. W tym przypadku nazwy partycji umieszcza się w nawiasach okrągłych w obrębie czynności (akcji), ponad jej nazwą. W przypadku, gdy elementy diagramów czynności znajdują się na granicy dwóch partycji, nazwy tych partycji rozdziela się przecinkiem. I wreszcie — jednoznaczne wskazanie subpartycji umożliwiające jest poprzez wyspecyfikowanie ciągu składającego się odpowiednio z:

- ♦ nazwy partycji,
- ♦ podwójnego dwukropka,
- ♦ nazwy subpartycji;

w nawiasach okrągłych w obrębie każdego elementu przypisanego do tej subpartycji. Ten sposób specyfikowania partycji i subpartycji ilustruje rysunek 4.29.



Rysunek 4.29. Alternatywna notacja partycji i subpartyji

Obszar rozszerzenia

Specyficzną rolę w diagramach czynności pełnią **obszary rozszerzenia** (ang. *expansion regions*).



Obszar rozszerzenia jest ściśle zdefiniowanym fragmentem diagramu czynności z jednoznacznie wyspecyfikowanymi wejściami i wyjściami, wykonywanym wielokrotnie, stosownie do liczby elementów na wejściu.

Wejścia i wyjścia obszaru rozszerzenia nazywane są **przełącznikami rozszerzenia** (ang. *expansion nodes*). Zawierają one zbiory danych o ustalonej liczbie elementów tego samego typu. Rolą obszarów rozszerzenia jest umożliwienie sekwencyjnego, współbieżnego bądź iteracyjnego wykonywania jego funkcjonalności kolejno dla każdej z elementarnych danych ze zbioru symbolizowanego przez przełącznik rozszerzenia. Jedną z tych trzech opcji zapisuje się w formie słowa kluczowego w obrębie obszaru rozszerzenia. Tak więc dane wejściowe są kolejno pobierane z wejściowego przełącznika. Stosownie do treści słowa kluczowego wykonywany jest obszar rozszerzenia. W rezultacie

jego funkcjonowania powstaje kolejno zbiór danych wyjściowych. Dopiero po skompletowaniu pełnego zbioru wyjściowego jest on przekazywany na zewnątrz obszaru rozszerzenia.

Obszar rozszerzenia przyjmuje na diagramie graficzną postać czynności z krawędziami kreślonymi linią przerywaną. Na linii tej umieszczane są poszczególne przekaźniki rozszerzenia. Przekaźniki rozszerzenia przyjmują postać nazwanych prostokątów podzielonych na komórki zawierające elementarne dane tego samego typu. Każdy przekaźnik wejściowy i wyjściowy jest więc jednorodnym zbiorem danych. Może istnieć dowolna liczba przekaźników, przy czym:

- ♦ liczba przekaźników wejściowych nie musi być równa liczbie przekaźników wyjściowych,
- ♦ typy danych mogą się różnić w przypadku różnych przekaźników.

Przez poszczególne przekaźniki rozszerzenia przechodzą oznaczone strzałkami, kolejno wykonywane przepływy danych. W obszarze rozszerzenia można zapisać typ realizowanego rozszerzenia. Dostępne są następujące typy rozszerzenia:

- ♦ *stream* — sekwencyjne;
- ♦ *parallel* — współbieżne;
- ♦ *iterative* — iteracyjne.

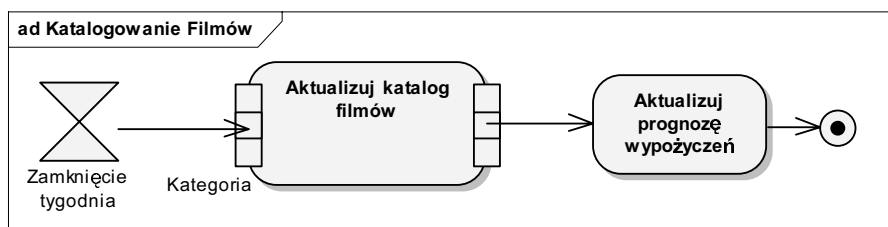
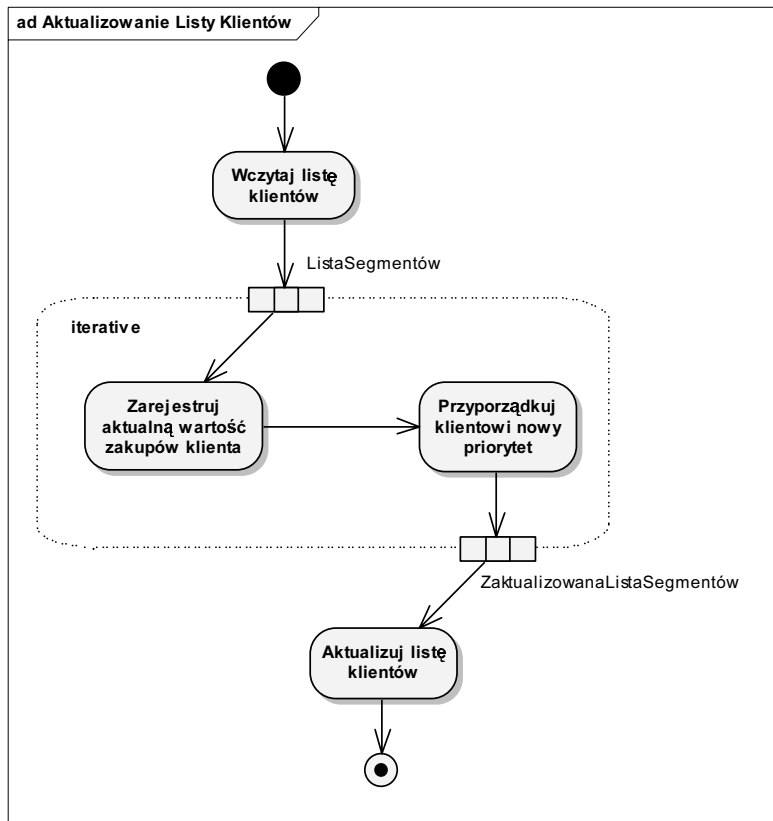
Zasady funkcjonowania obszaru rozszerzenia ilustruje rysunek 4.30.

Rysunek 4.30 przedstawia aktualizowanie listy klientów względem priorytetów w systemie CRM. Cechą każdego systemu klasy CRM jest możliwość tworzenia programów lojalnościowych. Właściwe zastosowanie tej koncepcji pozwala lepiej zaspokajać potrzeby klientów, co powoduje zwiększenie sprzedaży produktów i usług oferowanych przez firmę. Ten mechanizm jest ściśle związany z segmentacją klientów, czyli kwalifikowaniem każdego klienta do określonego segmentu klientów o określonej charakterystyce jakościowej i ilościowej. Taka klasyfikacja pozwala na nadawanie klientom priorytetów. Menedżerowie firm stosujących systemy CRM co pewien czas analizują segmenty i ich charakterystykę. Gdy zaistnieje taka potrzeba, dodają nowe segmenty lub modyfikują charakterystyki istniejących. Aby dodać nowy segment(y), operator systemu CRM *Wczytuje listę klientów*. Wczytana lista zawiera podział klientów na konkretną liczbę segmentów. Następnie system wykonuje dwie następujące po sobie czynności: *Rejestruje aktualną wartość zakupów klienta* oraz *Przyporządkowuje klientowi nowy priorytet*. Te dwie czynności są wykonywane w obszarze rozszerzenia oznaczonym słowem kluczowym *iterative*. Takie zakwalifikowanie obszaru rozszerzenia oznacza, że iteracyjnie dla każdego segmentu wszyscy klienci są weryfikowani pod kątem przyporządkowania właściwego priorytetu. Po wykonaniu czynności system *Aktualizuje listę klientów*. Zaktualizowana lista może być podzielona na inną liczbę segmentów, niż miało to miejsce pierwotnie.

W obszarze rozszerzenia występować może szereg czynności lub akcji. Możliwe jest również ograniczenie obszaru rozszerzenia do jednej czynności. W takim przypadku można użyć uproszczonej notacji, jak zaprezentowano na rysunku 4.31.

Rysunek 4.30.

Obszar
rozszerzenia
i przekazniki
rozszerzenia

**Rysunek 4.31.** Uproszczona notacja przekazników rozszerzenia

Przedstawiono na nim katalogowanie filmów w systemie informatycznym wypożyczalni. Systemy informatyczne wykorzystywane przez firmy wypożyczające kasety VHS i płyty DVD muszą mieć funkcję katalogowania kaset i płyt w kategorii wg określonych kryteriów. Wyszczególnione kategorie przekładają się bezpośrednio na cenę wypożyczenia filmu. Najważniejszym kryterium stosowanym przy katalogowaniu jest liczba dni, które upłynęły od premiery filmu, oraz liczba wypożyczeń. Przykładowo wypożyczenie płyty DVD zakwalifikowanej do kategorii MegaHIT kosztuje najwięcej, ponieważ film niedawno miał premierę oraz cieszy się dużym zainteresowaniem klientów wypożyczalni. Zatem system wypożyczalni przechowuje w bazie danych cennik powiązany z kategoriami filmów wraz z opisującymi je szczegółowymi kryteriami.

Dzięki odpowiedniej, zaimplementowanej procedurze system automatycznie raz w tygodniu uruchamia skrypt, który uaktualnia przynależność każdego z filmów do odpowiedniej kategorii. Automatycznie zmienia się cena wypożyczenia filmu. Po *Aktualizacji katalogu filmów* zmienia się liczba filmów danej kategorii. Może to mieć istotny wpływ na ogólną liczbę wypożyczeń filmów danej kategorii, dlatego następną czynnością jest *Aktualizacja prognozy wypożyczeń*.

Obszar przerwania

Przeptyw sterowania na diagramach czynności może skutkować przerwaniem funkcjonowania pewnych wyodrębnionych obszarów zwanych **obszarami przerwania** (ang. *interruptible activity regions*).

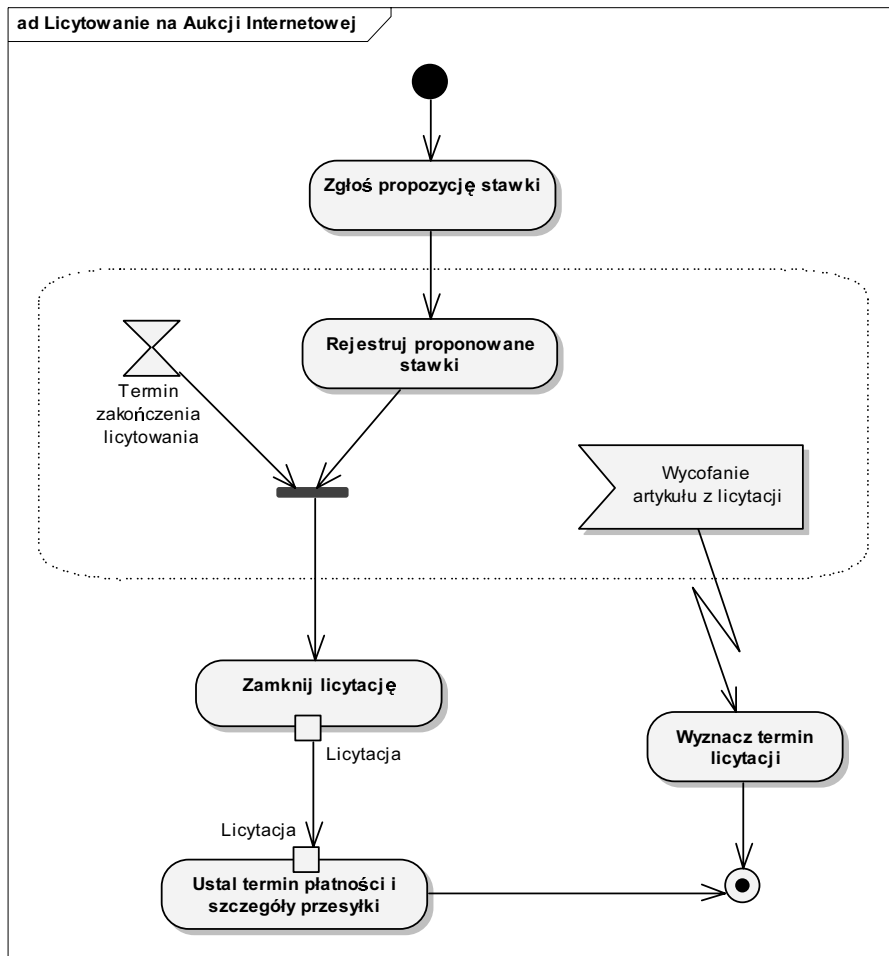


Obszar przerwania stanowi grupa czynności, w obrębie której w wyniku działania przepływu przerwania realizacja wszystkich czynności jest bezzwłocznie przerywana.

Początek **przepływu przerwania** (ang. *interrupting edge*) zaznacza się w obrębie obszaru przerwania, natomiast jego koniec poza nim. W momencie przekroczenia granicy obszaru przerwania przez jakikolwiek ze znaczników sterowania następuje zniszczenie wszystkich pozostałych znaczników w tym obszarze. Oznacza to, że tylko jeden znacznik zostaje przekazany do elementu docelowego. W związku z tak radykalnym skutkiem działania przepływu przerwania należy dokładnie przeanalizować potencjalny obszar przerwania przed nadaniem mu takiego statusu.

Charakterystyka obszaru przerwania ma zastosowanie w odniesieniu do pojęcia czynności, nie zaś akcji, która jako elementarna jednostka zachowania nie ulega przerwaniu. Opisywany obszar na diagramie przyjmuje graficzną postać czynności z krawędziami kreślonymi linią przerywaną. Przepływ przerwania reprezentowany jest w formie „błyskawicy” (rysunek 4.32). Alternatywnym oznaczeniem przepływu przerwania jest umieszczenie stosownego stereotypu, ikony błyskawicy, nad linią przepływu sterowania wychodzącego z obszaru przerwania. Ponieważ przepływ sterowania inicjowany jest często w sposób asynchroniczny, w takich przypadkach zasadne jest jawne zilurowanie sygnału przekazującego znacznik sterowania do przepływu przerwania.

Na aukcjach internetowych sprzedający rejestrują artykuły, które później stają się przedmiotem licytacji. Podczas rejestracji sprzedający określa czas trwania licytacji oraz, opcjonalnie, cenę minimalną i cenę „kup teraz”. W momencie otwarcia licytacji jest ona uwzględniana w katalogu dostępnych licytacji. Od tej chwili kupujący z odpowiednimi uprawnieniami *Zgłaszają propozycje stawek*, a system *Rejestruje proponowane stawki*. System rejestruje stawki do momentu upływu czasu trwania licytacji. Wtedy wysyłany jest sygnał, który uruchamia czynność *Zamknięcia licytacji*. Następnie sprzedający *Ustala terminy płatności i szczegóły przesyłki* z kupującym. Jeśli jednak jeszcze podczas trwania licytacji sprzedający nie jest zadowolony z proponowanych stawek, decyduje się na *Wycofanie artykułu z licytacji*. Następnie wyznacza nowy termin licytacji. Na diagramie rejestrowanie stawek zostało umieszczone w obszarze przerwania, ponieważ w momencie wystąpienia przepływu przerwania wszystkie znaczniki sterowania w tym obszarze stają się nieistotne i ulegną zniszczeniu.



Rysunek 4.32. Obszar przerwania w systemie aukcji internetowej

Manipulator wyjątków

W diagramach czynności realizacja określonych czynności czy też akcji może wymagać dodatkowych środków bezpieczeństwa dla stabilnego funkcjonowania systemu. Wskazują one na sytuacje wyjątkowe, które nie są rutynowe w kontekście danego procesu. Sytuacje te należy każdorazowo identyfikować i na bieżąco obsługiwać w systemie. Obsługę wyjątków można osiągnąć poprzez wyspecyfikowanie odpowiadającym im **manipulatorów wyjątków** (ang. *exception handlers*).



Manipulator wyjątków określa czynności, które należy wykonać, jeśli określony wyjątek wystąpi w trakcie wykonania czynności chronionej.

Tak więc wyjątki występujące w czynnościach chronionych są powiązane odpowiednimi manipulatorami wyjątków z czynnościami zapewniającymi obsługę sytuacji wyjątkowej. W przypadku zaistnienia wyjątku uruchamiany jest manipulator, który inicjuje funkcjonowanie czynności związanej z tym wyjątkiem poprzez przekaznik wyjątków. Manipulatory wyjątków należy nazywać. Czynności zapewniające obsługę wyjątków mogą nie mieć innych wejść i wyjść poza manipulatorami. Nieuwzględnienie wyjątku i związanego z nim manipulatora może mieć nieprzewidywalne skutki dla wykonania czynności chronionej.

Funkcjonowanie manipulatora wyjątków przedstawia rysunek 4.33 oraz rysunek 4.34. Pierwszy z nich przedstawia zachowanie systemu elektronicznego egzaminowania. System podczas *Losowania numerów pytań* z tablicy odwołał się do jej nieistniejącego indeksu. Sytuacja stała się krytyczna dla poprawnego przeprowadzenia egzaminu. System natychmiastowo wstrzymał akcję losowania numerów pytań i *Przeindeksował tablicę*.

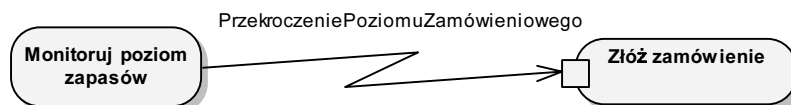
Rysunek 4.33.

Indeks poza zakresem



Rysunek 4.34.

Przekroczenie poziomu zamówieniowego



W systemie sterowania zapasami (rysunek 4.34) stan zapasów wszystkich artykułów jest na bieżąco monitorowany. Dla każdego z artykułów wyznaczony jest poziom zapasu, przy którym należy złożyć zamówienie na partię zapasu odpowiednią do utrzymania ciągłości sprzedaży pełnego asortymentu. W przypadku przekroczenia tego poziomu — a zatem zaistnienia sytuacji krytycznej — manipulator wyjątków uruchamia czynność, która zapewnia obsługę sytuacji wyjątkowej. Rolę manipulatora wyjątków spełnia poziom zapasu zamówieniowego.

Proces tworzenia diagramu czynności

Proces tworzenia diagramów czynności opiera się na następujących etapach:

1. zidentyfikowaniu podstawowych czynności i sygnałów na podstawie scenariusza przypadku użycia;
2. połączeniu czynności i sygnałów za pomocą przepływów sterowania;
3. opcjonalnym przeprowadzeniu dekompozycji czynności do poziomu akcji;
4. identyfikacji decyzyjnych i współbieżnych przepływów sterowania;
5. wprowadzeniu przepływów danych z wykorzystaniem kategorii modelowania, takich jak:

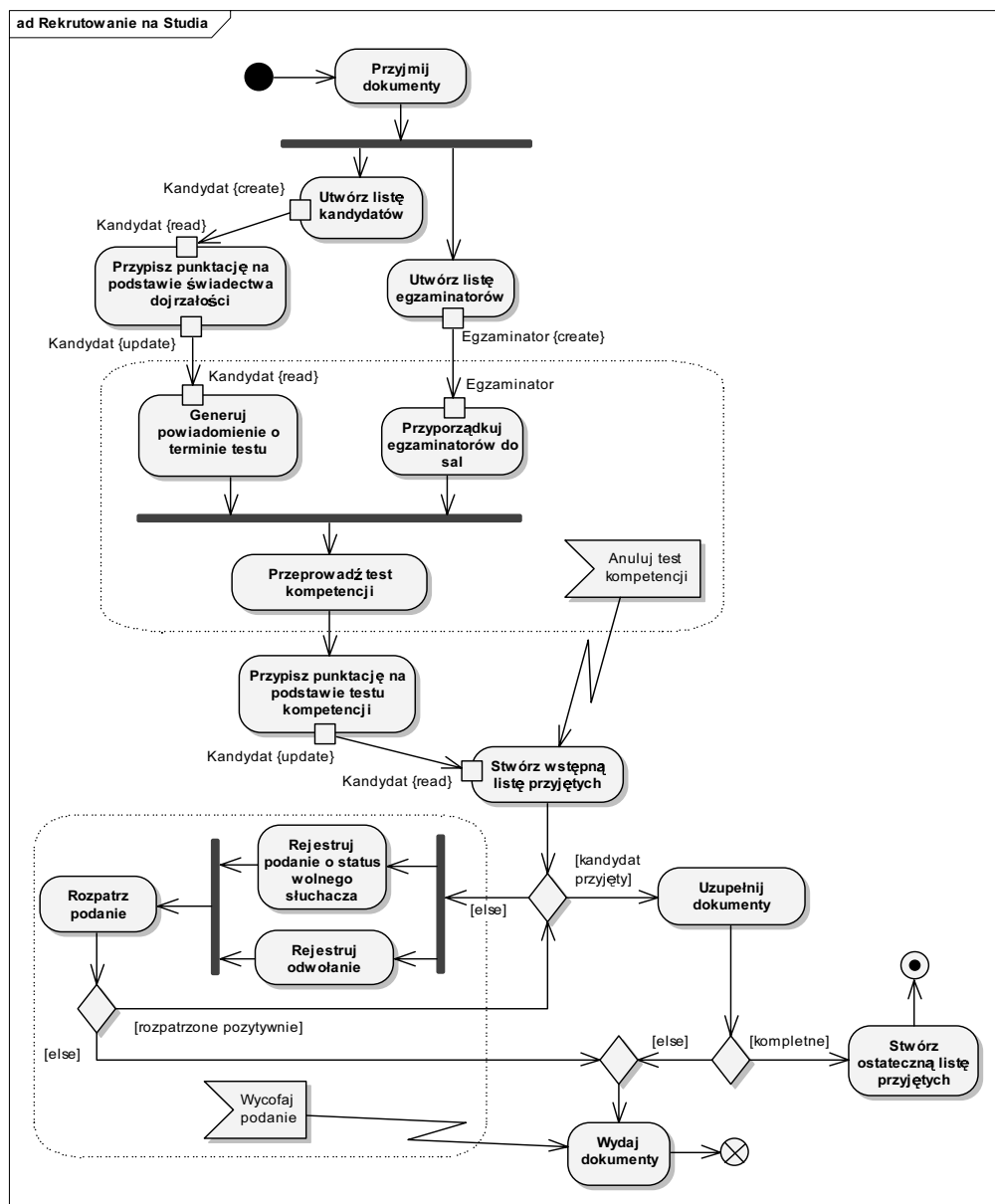
- ♦ przekaźniki danych,
 - ♦ parametry czynności,
 - ♦ bufor centralne,
 - ♦ składnice danych;
6. wyznaczeniu kryteriów, partycji i subpartycji diagramu;
 7. wprowadzeniu obszarów specjalizowanych:
 - ♦ obszaru rozszerzenia,
 - ♦ obszaru przerwania;
 8. wprowadzeniu manipulatorów wyjątków;
 9. identyfikacji akcji i opracowaniu adekwatnych diagramów akcji.

Proces tworzenia diagramów czynności powinien być realizowany zgodnie z koncepcją iteracyjno-przyrostowego cyklu życia systemu informatycznego. Oznacza to, że diagram czynności jest uszczegóławiany w kolejnych iteracjach danej fazy, co powoduje przyrost jego dokumentacji aż do osiągnięcia pełnego jej zakresu.

Studium diagramu czynności

Diagram czynności zawierający szereg elementów podstawowych oraz zaawansowanych — w tym procesy decyzyjne i współbieżne, sygnały, przekaźniki danych z ograniczeniami oraz obszary przerwania — przedstawia rysunek 4.35. Dziedzinę przedmiotową stanowi w tym przypadku proces rekrutowania na studia.

Jak zaprezentowano na rysunku, proces rekrutowania na studia jest inicjowany poprzez przyjmowanie i odnotowywanie w systemie dokumentów poszczególnych kandydatów. Dane poszczególnych kandydatów przechowywane są w systemie jako obiekty klasy *Kandydat*. Stosownie do ocen uzyskanych na egzaminie dojrzałości każdemu kandydatowi jest przypisywana punktacja. Wprawdzie akty prawne normujące rekrutację na studia wyższe nie uwzględniają możliwości egzaminowania kandydatów z przedmiotów, z których otrzymali oni ocenę wyszczególnioną na świadectwie dojrzałości, uczelnie mają daleko posuniętą autonomię we wprowadzaniu tzw. testów kompetencji. Stosuje się je najczęściej w przypadku bardzo dużej liczby kandydatów, kiedy niemożliwe jest jednoznaczne i obiektywne wskazanie liczby kandydatów co najwyżej nieznacznie odbiegającej od wyznaczonego limitu. Przykładowo założenia rekrutacyjne mogą uwzględniać przyjęcie 80 osób, natomiast kandydaci zajmujący miejsca od 70 do 120 na liście uporządkowanej malejąco według liczby uzyskanych punktów uzyskują taki sam wynik punktowy. Z tego powodu, oprócz listy ubiegających się o przyjęcie, w systemie równolegle tworzona jest lista *Egzaminatorów*. Dane poszczególnych egzaminatorów przechowywane są w systemie jako obiekty klasy *Egzaminator*. System *Generuje powiadomienie o terminie testu* adresowane do każdego z kandydatów, jednocześnie *Przyporządkowując egzaminatorów do sal*. Dopiero po zakończeniu obu tych równoległych czynności możliwe jest *Przeprowadzenie testu kompetencji*.



Rysunek 4.35. Przeprowadzanie rekrutacji na studia

Projektanci systemu przewidzieli możliwość rezygnacji z przeprowadzania egzaminu przez władze wydziałowe. Niezależnie od konkretnego terminu *Anulowania testu kompetencji*, wszystkie prowadzące do niego czynności przygotowawcze są natychmiast przerywane. W szczególnym przypadku — jeśli test anulowano w trakcie jego przeprowadzania — nie występuje uzasadnienie kontynuowania testu. O ile test kompetencji odbył się, jego wyniki są doliczane do punktacji powstałej na podstawie analizy

świadectwa maturalnego. Pozwala to na *Stworzenie wstępnej listy przyjętych* na I rok studiów. Jeżeli test kompetencji został anulowany, podstawą tworzenia wstępnej listy jest punktacja przypisana poszczególnym kandydatom na podstawie ich świadectw dojrzałości. Każdy zakwalifikowany kandydat jest zobligowany do *Uzupełnienia dokumentów*. Kandydaci, którzy dopełnili tego obowiązku, są uwzględniani podczas realizacji czynności *Stwórz ostateczną listę przyjętych* — tym samym zostają studentami. W przeciwnym razie są im *Wydawane dokumenty*. Kandydaci zweryfikowanie negatywnie mogą:

- ♦ *Zarejestrować podanie o status wolnego słuchacza,*
- ♦ *Zarejestrować odwołanie.*

Podanie jest rozpatrywane przez dziekana wydziału. W przypadku pozytywnego rozpatrzenia podania, po uzupełnieniu dokumentów student trafia na ostateczną listę przyjętych. W przeciwnym razie dokumenty są mu zwracane, a rekrutacja uznana za zamkniętą. Analogiczny rezultat ma *Wycofanie podania* w dowolnym momencie.

Podstawowe pojęcia

Algorytm	Partycja
CRUD	Komórka
Czas	Partycja zewnętrzna
Dekompozycja czynności	Subpartycja
Akcja	Tor
Czynność	Początek
Podczynność	Podsystem
Diagram czynności	Proces systemowy
Definicja	Przełącznik
Proces tworzenia	Przełącznik danych
Koniec	Anonimowy
Manipulator wyjątków	Wejściowy
Obiekt	Wyjściowy
Obszar	Przełącznik rozszerzenia
Przerwania	Przełącznik wyjątków
Rozszerzenia	
Ograniczenie	

Przepływ danych	Przepływ współbieżny
Bufor centralny	Rozwidlenie
Parametr czynności	Scalenie
Składnica danych	Specyfikacja scalenia
Strumieniowy	Znacznik sterowania
Sygnał	Scenariusz przypadku użycia
Nadawczy	Stan
Odbiorczy	Zakończenie przepływu
Waga	
Zestaw przekaźników	
Przepływ sterowania	
Przepływ decyzyjny	
Decyzja	
Łącznik	
Warunek	
Złączenie	

Pytania i zadania

1. W jakim celu używa się diagramów czynności?
2. Wymień podstawowe elementy diagramu czynności.
3. Zilustruj graficznie podstawowe kategorie diagramów czynności.
4. Wyjaśnij różnicę między kategoriami:
 - ♦ koniec,
 - ♦ zakończenie przepływu.
5. Wymień różnice pomiędzy czynnościami a akcjami. Czy na pojedynczym diagramie czynności mogą jednocześnie występować czynności i akcje?
6. Czy diagramy czynności mogą być dekomponowane? Jeśli tak, wskaż okoliczności, w jakich przeprowadzenie dekompozycji jest uzasadnione, i kategorie pojęciowe biorące udział w tym procesie. Jeśli nie, uzasadnij dlaczego.
7. Wybierz serwis internetowy mający na celu wspomaganie działalności turystycznej w danym regionie. Użytkując podstawowe kategorie, opracuj diagram czynności tego serwisu.

8. Wymień zaawansowane kategorie diagramu czynności.
9. Co rozumie się pod pojęciem przepływu sterowania?
10. Z jakimi pojęciami należy wiązać przepływy sterowania?
11. Wyjaśnij znaczenie znaczników sterowania na diagramach czynności. Jakie skutki dla znaczników powoduje kategoria pojęciowa *początek*, a jakie *koniec*?
12. W jakich sytuacjach znajdują zastosowanie przepływy decyzyjne?
13. Jaka jest różnica między decyzją a złączeniem? Podaj ich oznaczenia graficzne.
14. W jaki sposób oznacza się warunki w przepływach? Przedstaw w postaci bloku decyzyjnego z zapisem warunków grupowanie klientów w trzech segmentach — A, B i C — w zależności od wielkości zakupów klienta.
15. Jaka jest rola łącznika? Zinterpretuj tę rolę, przekształcając odpowiednio rysunek 4.5.
16. Zilustruj złączenie na diagramie czynności. Wyjaśnij przepływ znacznika sterowania w złączeniu.
17. Kiedy możliwa jest integracja decyzji i złączenia w jednym bloku decyzyjnym? Opracuj w tym aspekcie przepływy decyzyjne w systemie stypendialnym. Opisz warunki.
18. Czym różnią się przepływy współbieżne od decyzyjnych? Wyjaśnij przepływ znaczników sterowania w obydwu rodzajach przepływów.
19. Wyjaśnij znaczenie rozwidlenia i scalenia w przepływach współbieżnych. Podaj interpretację graficzną.
20. Czy współbieżne przepływy są w jakimkolwiek momencie synchronizowane? Jeśli tak, to kiedy to następuje?
21. Zapisz na stosownym diagramie czynności regułę sprzedaży jednostek funduszu inwestycyjnego w przypadku osiągnięcia określonej ceny.
22. Zaproponuj współbieżne biznesowe procesy w systemie bankowym.
23. Makler podejmuje decyzję o sprzedaży jednostek funduszy inwestycyjnych — akcyjnego, stabilnego wzrostu i zrównoważonego — po osiągnięciu pewnych wyznaczonych przez klienta wartości jednostek uczestnictwa. Za pomocą jakich kategorii pojęciowych można zilustrować diagram czynności przedstawiający tę sytuację? Opracuj adekwatny diagram z uwzględnieniem specyfikacji scalenia (ograniczenie *joinSpec*).
24. Operatorzy sieci komórkowych w swej ofercie określają różne rodzaje abonamentów. W ofertach kierowanych do klientów biznesowych równolegle dostarczają promocyjne oferty zakupu aparatów telefonicznych. Wykorzystując specyfikację scalenia, opracuj diagram czynności przedstawiający tę sytuację.
25. Czym różnią się przepływy danych od przepływów sterowania na diagramach czynności?

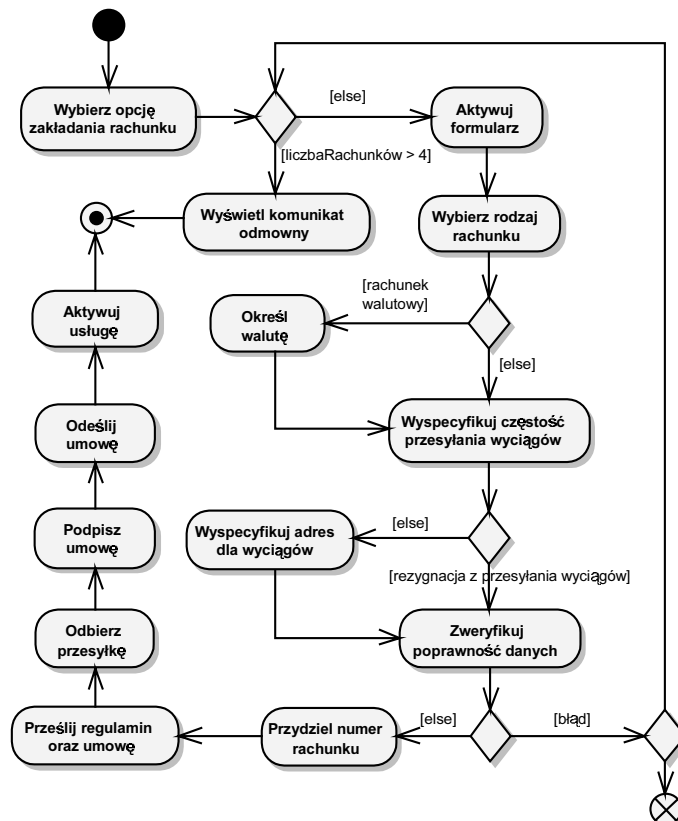
- 26.** Opracuj diagramy czynności przedstawiające akcje składające się na:
- ♦ wyszukiwanie największej liczby ze zbioru liczb,
 - ♦ sortowanie tablicy jednowymiarowej,
 - ♦ sortowanie tablicy dwuwymiarowej,
 - ♦ obliczanie wybranych formuł statystycznych,
 - ♦ wyliczanie kolejnych elementów ciągu Fibonacciego.
- 27.** Wymień podstawowe kategorie pojęciowe przepływów danych.
- 28.** W jakich sytuacjach uzasadnione jest umieszczenie obiektu (obiektów) na diagramie czynności?
- 29.** Co to jest przekaźnik danych? Jakie są opcje jego graficznej prezentacji na diagramach czynności?
- 30.** Kiedy przekaźniki danych łączone są w zestawy?
- 31.** Wskaż zasady użytkowania zestawów przekaźników.
- 32.** Dane są następujące czynności:
- ♦ *Przyjmij kandydata na studia,*
 - ♦ *Wypożycz czasopismo.*
- Wzbogać te czynności o stosowne parametry.
- 33.** W przedsiębiorstwie turystycznym przyjęto zasadę, że wycieczkę realizuje się przy minimum dwudziestu zgłoszeniach. Zapisz tę informację na diagramie czynności w postaci wagi.
- 34.** Wymień rodzaje sygnałów stosowanych na diagramach czynności.
- 35.** Przedstaw role sygnałów następujących urządzeń:
- ♦ skaner,
 - ♦ modem,
 - ♦ zmywarka.
- 36.** Jaka jest różnica między buforem centralnym a składnicą danych?
- 37.** W firmie przechowywane są wyroby i informacje o wyrobach. Jak należy rozumieć i interpretować pojęcie bufora centralnego i składnicę danych w odniesieniu do ruchu wyrobów oraz przechowywania informacji o nich? Interpretację należy oprzeć na rozumieniu znacznika sterowania.
- 38.** Uzasadnij wprowadzenie pojęcia partycji na diagramach czynności.
- 39.** Wyjaśnij zależności pomiędzy pojęciami:
- ♦ kryterium,
 - ♦ subpartycja,
 - ♦ partycja.
- Podaj przykład związany z funkcjonowaniem sklepu internetowego.

40. Przedstaw graficznie zależności pomiędzy pojęciami wymienionymi w poprzednim pytaniu w układzie:
- ♦ poziomym,
 - ♦ pionowym,
 - ♦ macierzowym.
41. W odniesieniu do układu macierzowego partycji zinterpretuj znaczenie komórki.
42. Jaka jest rola partycji zewnętrznych na diagramie czynności?
43. Kiedy należy umieszczać czynności lub obiekty na granicznych liniach partycji?
44. W odniesieniu do przedstawionego na rysunku 4.36 diagramu czynności występują następujące instancje klasyfikatorów:
- ♦ *SystemBankowy*,
 - ♦ *Klient*,
 - ♦ *SystemPocztyElektronicznej*.

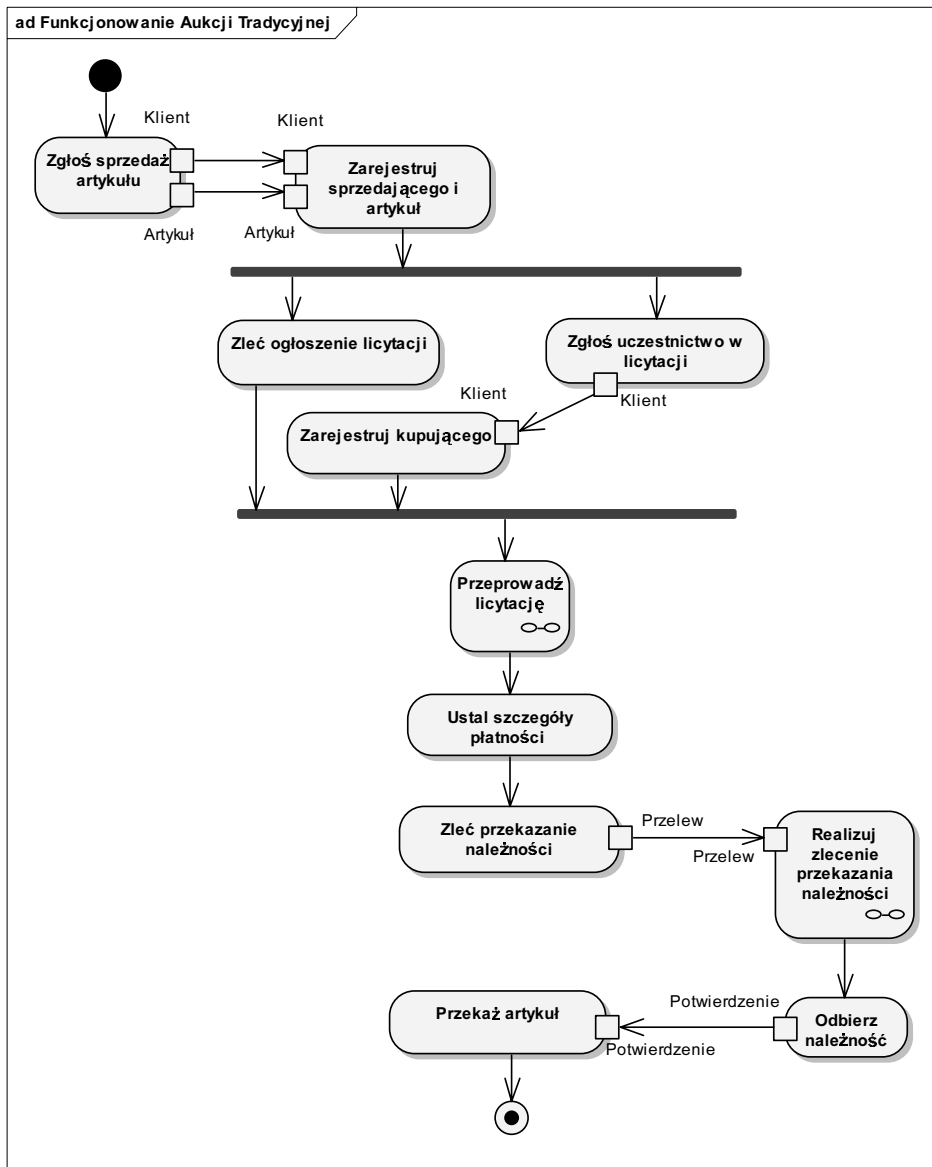
Zidentyfikuj obiekty przekazywane pomiędzy poszczególnymi czynnościami w systemie oraz wzbogać diagram o adekwatne partycje.

Rysunek 4.36.

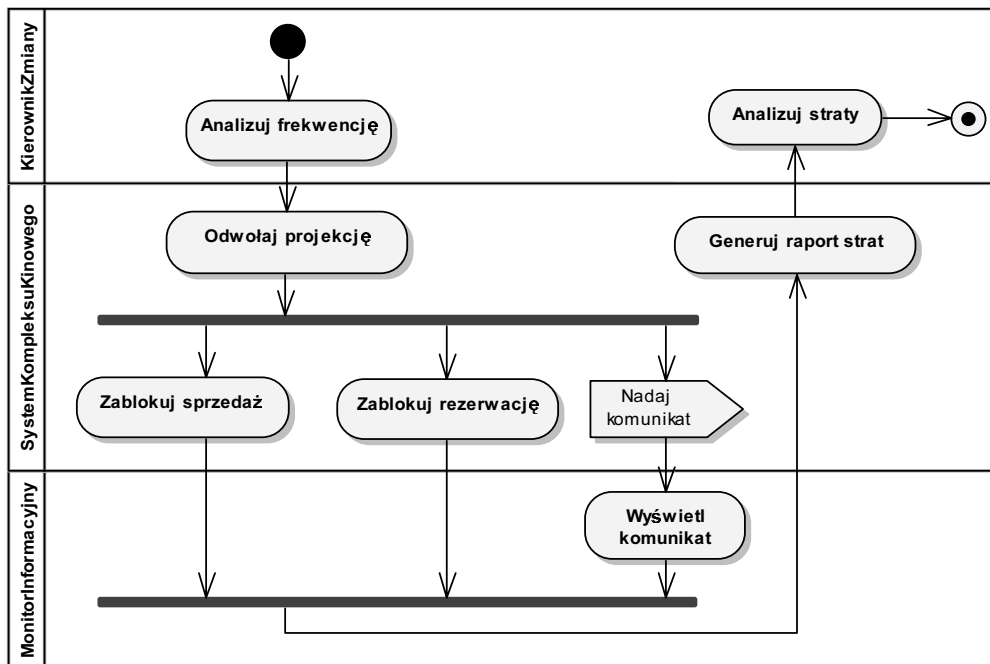
Zamawianie wycieczki



45. Opracuj diagram czynności na podstawie rysunku 4.36, wykorzystując alternatywną notację partycji poprzez umieszczenie ich nazw w czynnościach.
46. W odniesieniu do przedstawionego na rysunku 4.37 diagramu czynności występują następujące instancje klasyfikatorów:
- ♦ *Aukcjoner*,
 - ♦ *Bank*,
 - ♦ *Kupujący*,
 - ♦ *AsystentAukcji*,
 - ♦ *Sprzedający*.
- Opisz sytuację zaprezentowaną na diagramie oraz wzbogać diagram o adekwatne partycje.
47. Opracuj diagram czynności na podstawie rysunku 4.37, wykorzystując alternatywną notację partycji — umieszczanie nazw partycji w czynnościach.
48. W jakich przypadkach zastosowanie partycji staje się szczególnie użyteczne?
49. Z jakimi uwarunkowaniami wiąże się zastosowanie obszaru rozszerzenia?
50. Wyjaśnij szczegółowo role i rodzaje przekaźnika rozszerzenia.
51. Sprawdź, jakiego rodzaju karty uprzywilejowane emituje CitiBank i zinterpretuj naliczanie odsetek jako obszar rozszerzenia dla złotej karty.
52. W jakich sytuacjach przy projektowaniu diagramów czynności należy uwzględnić obszary przerwania?
53. Wyjaśnij znaczenie obszaru przerwania.
54. Zaznacz na rysunku 4.38 obszar(y) przerwania oraz obiekty przekazywane pomiędzy czynnościami. Rozważ zastosowanie takich obiektów jak:
- ♦ *Repertuar*,
 - ♦ *Rezerwacja*,
 - ♦ *Bilet*,
 - ♦ *RaportStrat*.
55. Podaj przykłady procedur programistycznych, w których ma zastosowanie kategoria obszaru przerwania.
56. Jaka rolę pełni manipulator wyjątków i przekaźnik wyjątków?
57. Przedstaw manipulator wyjątków w systemie obsługi poczty elektronicznej w celu identyfikacji zainfekowanych wirusem wiadomości oraz wiadomości z tematami zastrzeżonymi.
58. Opisz i uzasadnij logikę procesu tworzenia diagramów czynności.



Rysunek 4.37. Funkcjonowanie aukcji tradycyjnej



Rysunek 4.38. *Anulowanie rezerwacji biletu kinowego*

Rozdział 5.

Diagram maszyny stanowej

Znaczenie diagramów maszyny stanowej

Obiekty użytkowane w systemie przechodzą w trakcie swojego cyklu życia przez szereg kolejnych stanów. Zjawisko to odzwierciedlane jest przez **maszynę stanową** (ang. *state-machine*).



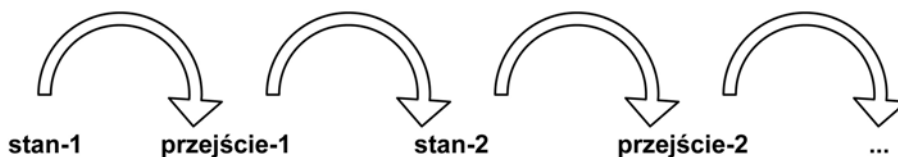
Maszyna stanowa to zachowanie określające sekwencje stanów, przez które przechodzi obiekt bądź interakcja w odpowiedzi na zdarzenia zachodzące w czasie swojego cyklu życia wraz z ich reakcjami na te zdarzenia.

Maszyna stanowa obiektu lub interakcji jest w języku UML dokumentowana poprzez **diagram maszyny stanowej**. Umożliwia on prześledzenie zachowania się obiektu zarówno w zakresie pojedynczego, jak i kilku przypadków użycia.



Diagram maszyny stanowej to graficzne odzwierciedlenie dyskretnego, skokowego zachowania skończonych systemów stan-przejście.

Zachowanie **dyskretne** maszyny stanowej należy interpretować jako stany i przejścia między nimi zorganizowane w sekwencję jak poniżej:



Przeciwieństwem zachowania skokowego jest zachowanie ciągłe, bez wyodrębnienia kolejnych stanów zjawiska. Diagram maszyn stanowych może być wykorzystany jako dokumentacja umożliwiająca bezpośrednie generowanie kodu. Diagramy te są szczególnie użyteczne w modelowaniu systemów czasu rzeczywistego i multimedialnych, chociaż są powszechnie używane.

Podstawowe kategorie pojęciowe oraz notacja graficzna

Podstawowymi pojęciami diagramów maszyny stanowej są:

- ♦ stan,
- ♦ przejście,
- ♦ stan początkowy,
- ♦ stan końcowy.

Graficznie stany przedstawiane są z wykorzystaniem dwóch alternatywnych notacji:

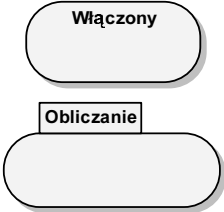
- ♦ klasycznej,
- ♦ zakładkowej.

Notacja klasyczna opiera się na konwencji zapisu stosowanej w przypadku czynności oraz akcji dokumentowanych na diagramach czynności. Alternatywny wybór to **notacja zakładkowa**. Wymaga ona umieszczenia zakładki zawierającej nazwę bezpośrednio nad graficznym oznaczeniem stanu. Podstawowe kategorie modelowania diagramu maszyny stanowej oraz ich definicje albo interpretacje przedstawia tabela 5.1.

Diagram maszyny stanowej zawierający scharakteryzowane podstawowe elementy przedstawiono na rysunku 5.1.

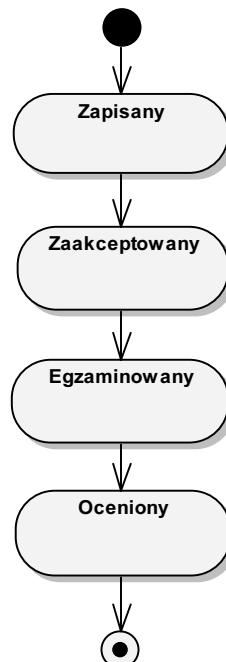
Diagram ten zawiera sekwencję stanów obiektu klasy *Student* w systemie elektronicznego egzaminowania. Pierwszym stanem, w jakim może znaleźć się ten obiekt, jest stan *Zapisany*. Oznacza on, że dany *Student* został umieszczony na liście osób uprawnionych do przystąpienia do egzaminu. W chwili zaznaczenia na liście obecności egzaminowanego, któremu odpowiada dany obiekt klasy *Student*, przechodzi on w stan *Zaakceptowany*. Stan ten — będący konsekwencją polityki bezpieczeństwa wydziału — oznacza, że egzaminowany ma prawo zalogować się do systemu i rozpocząć zdawanie egzaminu. Zalogowanie się egzaminowanego do systemu jest równoznaczne z przystąpieniem do egzaminu. Towarzyszy temu przejście w stan *Egzaminowany*. *Student* znajduje się w tym stanie dopóty, dopóki nie udzieli odpowiedzi na wszystkie pytania składające się na egzamin lub nie upłynie ustalony czas egzaminowania. Obliczana jest wówczas punktacja, a *Student* przechodzi w stan *Oceniony*.

Tabela 5.1. Podstawowe elementy diagramów maszyny stanowej

Nazwa	Notacja	Definicja/interpretacja
Stan		 Stan to okoliczność lub sytuacja, w jakiej się obiekt znajduje w cyklu swojego życia, kiedy spełnia warunek, wykonuje czynność lub czeka na zdarzenie.
Przejście		 Przejście to relacja między dwoma stanami, wskazująca, że obiekt znajdujący się w pierwszym stanie wykona pewne akcje i przejdzie do drugiego stanu, ilekroć zajdzie określone zdarzenie i będą spełnione określone warunki.
Stan początkowy		Stan początkowy to zainicjowanie maszyny lub podmaszyny stanowej.
Stan końcowy		Stan końcowy to zakończenie maszyny lub podmaszyny stanowej.

Rysunek 5.1.

System elektronicznego
egzaminowania
— obiekt klasy Student



Zaawansowane składniki diagramu

Kontekst pojęcia stanu oraz przejścia jest na tyle szeroki pod względem znaczeniowym, że wymaga dalszego uszczegółowienia. Do **kategorii zaawansowanych** maszyny stanowej zaliczają się:

- ♦ sekcje symbolu graficznego stanu,
- ♦ klasyfikacja stanów,
- ♦ obszary współbieżne,
- ♦ pseudostany,
- ♦ rodzaje przejść,
- ♦ protokołowe maszyny stanowe,
- ♦ maszyny stanowe zachowania,
- ♦ zdarzenia.

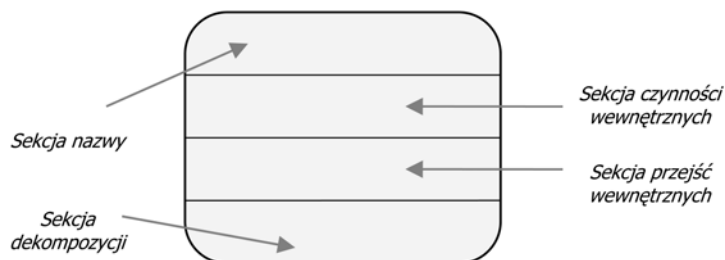
Sekcje symbolu graficznego stanu

W dowolnym stanie zaprezentowanym na diagramie maszyny stanowej można wyróżnić następujące **sekcje** (ang. *compartments*):

- ♦ sekcję nazwy,
- ♦ sekcję czynności wewnętrznych,
- ♦ sekcję przejść wewnętrznych,
- ♦ sekcję dekompozycji — w przypadku stanów złożonych.

Zaznaczanie sekcji w obrębie stanów jest opcjonalne. Wskazane jest oddzielanie ich separatorami w postaci linii ciągłych, jak zaprezentowano na rysunku 5.2.

Rysunek 5.2.
Sekcje stanu



Sekcja nazwy (ang. *name compartment*) zawiera nazwę stanu. Musi ona w sposób jednoznaczny identyfikować stan obiektu. Nazwy stanów różnią się zauważalnie od nazw czynności. Przykładowymi nazwami stanów są:

- ♦ *Czuwanie*,
- ♦ *Obliczanie*,
- ♦ *Zamówiony*,
- ♦ *Oczekująca*.

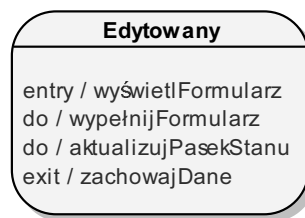
Sekcja czynności wewnętrznych (ang. *internal activities compartment*) zawiera listę czynności wewnętrznych. Są one wykonywane w trakcie przyjmowania określonego stanu przez obiekt. Rodzaj czynności wewnętrznej identyfikowany jest poprzez słowo kluczowe zamieszczone w wydzielonej części stanu, którego dotyczy. I tak w standardzie UML 2.0 predefiniowane są następujące słowa kluczowe:

- ♦ *entry*, które identyfikuje czynność automatycznie wykonywaną na obiekcie w momencie jego przejścia w dany stan. Dla danego stanu może wystąpić jedna taka czynność;
- ♦ *exit*, które identyfikuje czynność wykonywaną w momencie wyjścia obiektu z danego stanu. Dla danego stanu może wystąpić jedna taka czynność;
- ♦ *do*, które identyfikuje czynność wykonywaną w sposób ciągły na obiekcie znajdującym się w danym stanie; jest ona wykonywana zawsze po wykonaniu czynności wewnętrznej oznaczonej słowem kluczowym *entry*; dla danego stanu można zdefiniować wiele takich czynności, które są realizowane niezależnie od siebie.

Poza predefiniowanymi słowami kluczowymi można w ramach rozszerzeń języka UML zdefiniować inne słowa kluczowe. Przykłady czynności wewnętrznych ilustruje rysunek 5.3.

Rysunek 5.3.

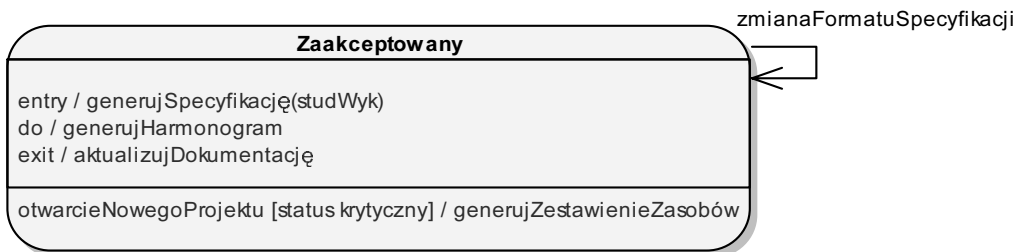
*Czynności
wewnętrzne stanu*



Przedstawia on jeden ze stanów obiektu będącego formularzem zgłoszeniowym na konferencję naukową. Wcześniej utworzony formularz może być *Edytowany*. W momencie uaktywnienia tego stanu następuje *wyświetlenieFormularza* na ekranie. W ramach edycji realizowana jest czynność *wypełnianiaFormularza*. W momencie zmiany stanu formularza zgłoszeniowego z *Edytowany* na inny następuje *zachowanieDanych* w bazie danych.

Sekcja przejść wewnętrznych (ang. *internal transitions compartment*) wskazuje szczególne przypadki przejść, których wykonanie nie prowadzi do zmiany danego stanu. Przejść wewnętrznych nie należy utożsamiać z przejściami zwrotnymi, których stan źródłowy i docelowy są tym samym stanem, mimo że znaczenie jest bardzo zbliżone. W obu przypadkach wykonanie czynności związanej z przejściem nie prowadzi do

zmiany stanu. O ile jednak w przypadku przejścia zwrotnego są wykonywane czynności wejściowe (*entry*) oraz wyjściowe (*exit*) danego stanu, w przypadku przejść wewnętrznych są one pomijane. Przejścia wewnętrzne nie mają odrębnego oznaczenia graficznego, lecz są specyfikowane w obszarze stanu. I tak zastosowanie przejść wewnętrznych w odniesieniu do systemu zarządzania projektami przedstawia rysunek 5.4.



Rysunek 5.4. Przejścia wewnętrzne i zwrotne dla stanu obiektu *Projekt*

Zilustrowano na nim stan klasy obiektów *Projekt*. W firmie programistycznej wykorzystuje się oprogramowanie do monitorowania prowadzonych projektów. *Projekt* może znajdować się w stanie *Zaakceptowany*, jeśli jego zakres zatwierdzą wszyscy udziałowcy. W momencie wejścia obiektu *Projekt* w ten stan następuje *generowanieSpecyfikacji*. Jest ona tworzona na podstawie danych zebranych podczas przeprowadzania studium wykonalności (*studWyk*). Gdy obiekt znajduje się w stanie *Zaakceptowany*, wykonywana jest czynność *generowaniaHarmonogramu*. Przedstawiony stan ma ponadto dwa przejścia:

- ♦ przejście zwrotne inicjowanym zdarzeniem *zmianaFormatuSpecyfikacji*,
- ♦ przejście wewnętrzne inicjowane zdarzeniem *otwarcieNowegoProjektu*.

Pierwsze z wymienionych przejść jest wykonywane, jeśli w firmie nastąpi zmiana formatu specyfikacji. Zainicjowanie tego przejścia skutkuje *aktualizowaniemDokumentacji* projektowej podczas wyjścia ze stanu, a następnie *generowaniemSpecyfikacji* wg aktualnego standardu na wejściu. Drugie z przejść jest przejściem wewnętrznym, co oznacza, że w momencie *otwarciaNowegoProjektu* o statusie krytycznym następuje *generowanieZestawieniaZasobów* firmy. Zgodnie z regułami biznesowymi firmy programistycznej projekty o statusie krytycznym mają wysoki priorytet realizacji. Zatem utworzona lista zasobów posłuży jako wstępna dokumentacja przy ustalaniu terminów realizacji nowego projektu. Status projektu jest w tym przypadku warunkiem, którego spełnienie determinuje tworzenie aktualnego zestawienia zasobów. W przypadku *otwarciaNowegoProjektu* ani dokumentacja, ani specyfikacje nie podlegają aktualizacji.

Klasyfikacja stanów

Można wyróżnić trzy kategorie stanów:

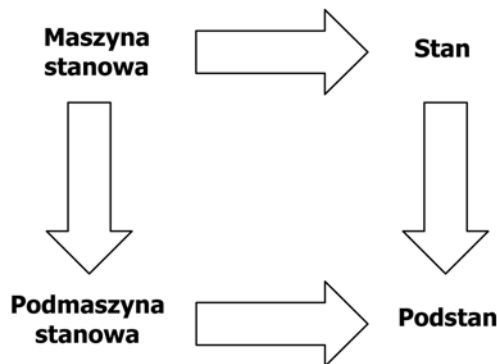
- ♦ proste,
- ♦ złożone,
- ♦ podstany.

Stan prosty (ang. *simple state*) jest stanem, który nie zawiera ani podstanów, ani obszarów współbieżnych. Stany zilustrowane na rysunkach 5.1, 5.3 oraz 5.4 należy zakwalifikować jako stany proste. W odróżnieniu od nich na diagramach maszyny stanowej przedstawiać można stany złożone. **Stan złożony** (ang. *composite state*) jest stanem, który spełnia jeden z wymienionych poniżej wymogów:

- ♦ ma dokładnie jeden obszar współbieżny, zawierający podmaszynę stanową;
- ♦ jest podzielony na dwa lub więcej obszarów współbieżnych.

Każdy stan złożony ma więc **podstany** (ang. *substates*). Mogą to być albo podstany bezpośrednie, czyli znajdujące się dokładnie o jeden poziom niżej w hierarchii, albo podstany pośrednie, czyli wielokrotnie zagnieżdżone. Zakończenie czynności we wszystkich obszarach współbieżnych jest równoznaczne z zakończeniem danego stanu. Relacje pomiędzy maszynami stanowymi a podmaszynami stanowymi przedstawia rysunek 5.5.

Rysunek 5.5.
Maszyny i podmaszyny stanowe



Konwencja specyfikacji stanów złożonych może być:

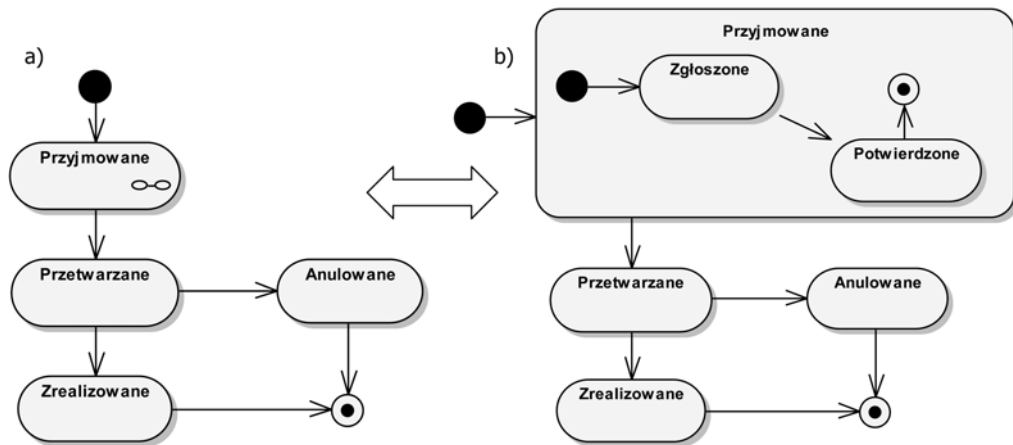
- ♦ ukryta,
- ♦ jawna.

Przykładowo podmaszynę stanową stanu złożonego *Przyjmowanie* można ukryć — co jest wskazywane na diagramie w postaci stereotypu w oznaczeniu stanu (rysunek 5.6a) — lub przedstawić w sposób jawny (rysunek 5.6 b). Notacje te są równoważne, a zatem mogą być stosowane zamiennie. Pierwsza z wymienionych konwencji jest zazwyczaj wprowadzana w przypadku złożonych podmaszyn stanowych. Mniej obszerne podmaszyny stanowe są umieszczane na diagramie w sposób jawny.

Rysunek 5.6 przedstawia maszynę stanową obiektu *Zamówienie*. Jest ono *Zgłaszane* do systemu, a następnie *Potwierdzone*. Oba wymienione stany wskazują na wstępną obsługę zamówienia, więc potraktowano je jako podstany stanu złożonego *Przyjmowane*.

W przypadku, gdy ukrywana podmaszyna stanowa jest specyfikowana na osobnym diagramie, jej nazwę na diagramie nadrzędnym można wyspecyfikować, używając następującej **składni**:

<nazwa-stanu> ”:” <nazwa-podmaszyny-stanowej>

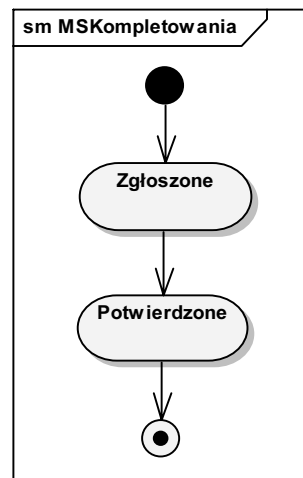


Rysunek 5.6. Podmaszyny stanowe ukryte i zaprezentowane w sposób jawny

Opisywaną sytuację ilustruje rysunek 5.7.

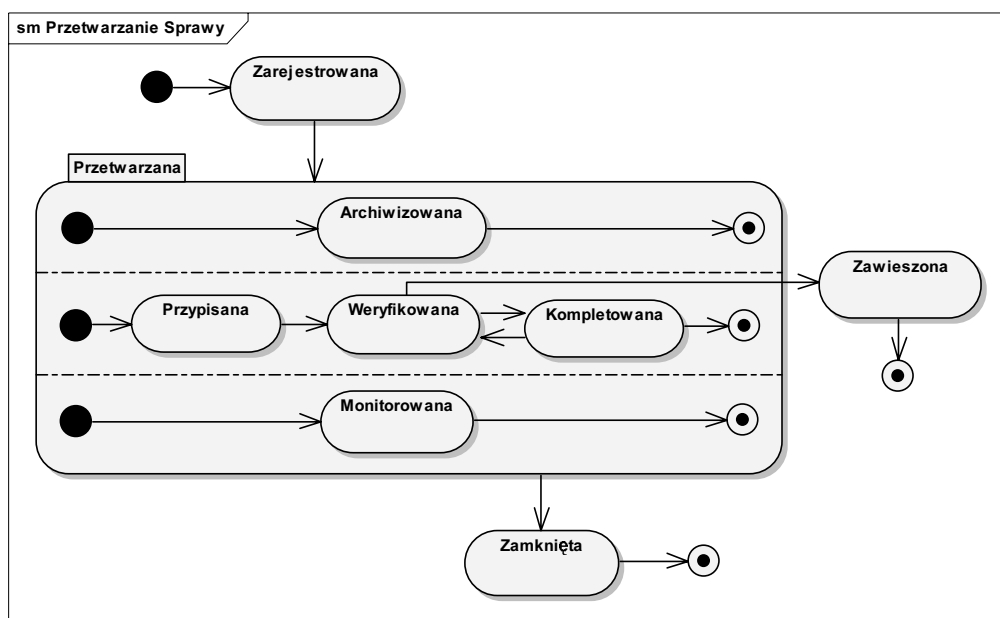
Rysunek 5.7.

Nazewnictwo
podmaszyn stanowych



Obszary współbieżne

Niektóre ze stanów w stanach złożonych i maszynach stanowych są aktywowane współbieżnie. Zapis współbieżności jest możliwy przy wykorzystaniu **obszarów współbieżnych** (ang. *orthogonal regions*). Są to poziome części stanu złożonego zawierające podmaszyny stanowe. W obszarze tym znajdują się więc podstawy i przejścia między nimi. Przejścia mogą przecinać granice stanów złożonych. Obszary współbieżne oddzielone są w symbolu stanu separatorami w postaci linii przerywanych. Wszystkie obszary współbieżne muszą być wykonane, aby stan mógł być zakończony. Zastosowanie obszarów współbieżnych w systemie CRM opartym na technologii zarządzania przepływem pracy (WFM) prezentuje rysunek 5.8.



Rysunek 5.8. Obszary współbieżne w maszynie stanowej przetwarzania sprawy

Określone dokumenty reprezentowane w systemie przez formularze określa się jako sprawy lub zlecenia. Generowane sprawy podlegają *Rejestracji* w systemie. Tylko zaewidencjonowane zbiory dokumentów mogą być następnie *Przetwarzane*. Przetwarzanie sprawy jest procesem, który może być realizowany przez kilka działów firmy. Na każdym etapie przetwarzania sprawa podlega dokumentowaniu — wtedy aktywowany jest stan *Archiwizowana*. Utworzona kopia zapasowa zabezpiecza firmę przed utratą danych w przypadku awarii systemu czy też błędu pracownika. Podczas złożonego stanu *Przetwarzania* podstany *Przypisana*, *Weryfikowana* oraz *Kompletowana* są aktywowane sekwencyjnie. I tak:

- ♦ w pierwszym z nich sprawa zostaje przypisana do konkretnego pracownika firmy, który będzie odpowiedzialny za jej opracowanie;
- ♦ w ramach stanu *Weryfikowanie* następuje sprawdzenie wszelkich wątpliwych informacji zawartych w dokumentacji na podstawie innych przedstawionych dokumentów i źródeł wewnętrznych;
- ♦ przejście wyjściowe ze stanu *Weryfikowania* powoduje aktywację stanu *Kompletowania* dokumentacji sprawy — w ramach tego stanu dostępne informacje są rozbudowywane o źródła zewnętrzne, o ile dotychczasowa dokumentacja i zasoby własne nie są wystarczające do sporządzenia decyzji.

W sytuacji, gdy sprawa wymaga ponownej weryfikacji, reaktywowany jest stan *Weryfikowana*. Negatywne zweryfikowanie informacji zawartych w dokumentacji bądź niemożność ich potwierdzenia skutkuje przejściem sprawy do stanu *Zawieszona*. Zgodnie z logiką zilustrowanej maszyny stanowej sprawa może ze stanu *Przetwarzana* przejść do stanu *Zamknięta*. Aby jednak przejście to zostało zainicjowane, musi zostać wykonane przejście wyjściowe z wykonywanego współbieżnie podstanu *Monitorowana*.

Pseudostany

Na diagramach maszyny stanowej mogą wystąpić rozbudowane ścieżki przejść. Diagramy te uwzględniają zatem wiele możliwości szczegółowego opisu przejść, relacji między przejściami, jak również wpływu przejść na powiązane z nimi stany. Przykładowo przejścia mogą być łączone i rozwidlane. Pełne oddanie merytorycznej zawartości tych złożonych przejść dokonywane jest z wykorzystaniem **pseudostanów** (ang. *pseudostates*).



Pseudostan jest abstrakcyjną kategorią modelowania diagramu maszyny stanowej, umożliwiającą organizowanie złożonych ścieżek przejść.

Opisywana kategoria pojęciowa zdecydowanie wykracza poza podstawowe rozumienie przejścia, a jednocześnie nie spełnia roli stanu w rozumieniu maszyn stanowych. Pseudostany w maszynach stanowych ujmuje tabela 5.2.

Interpretację i przykłady praktycznego zastosowania poszczególnych pseudostanów wymienionych w tabeli 5.2 zilustrowano na rysunkach 5.9 – 5.12. I tak przystępnym przykładem zastosowania pseudostanu w alternatywnych, równoważnych konwencjach graficznych jest decyzja. Rysunek 5.9 przedstawia model decyzji podejmowanej podczas ustalania stanów rachunku bankowego.

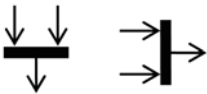
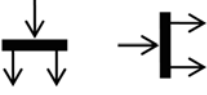
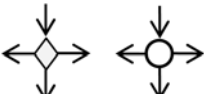
Pseudostany płytkiego wznowienia, rozwidlenia, scalenia, decyzji oraz stanu początkowego i punktu zniszczenia zilustrowano na rysunku 5.10. Przedstawia on maszynę stanową obiektu klasy *Egzamin* w systemie elektronicznego egzaminowania.

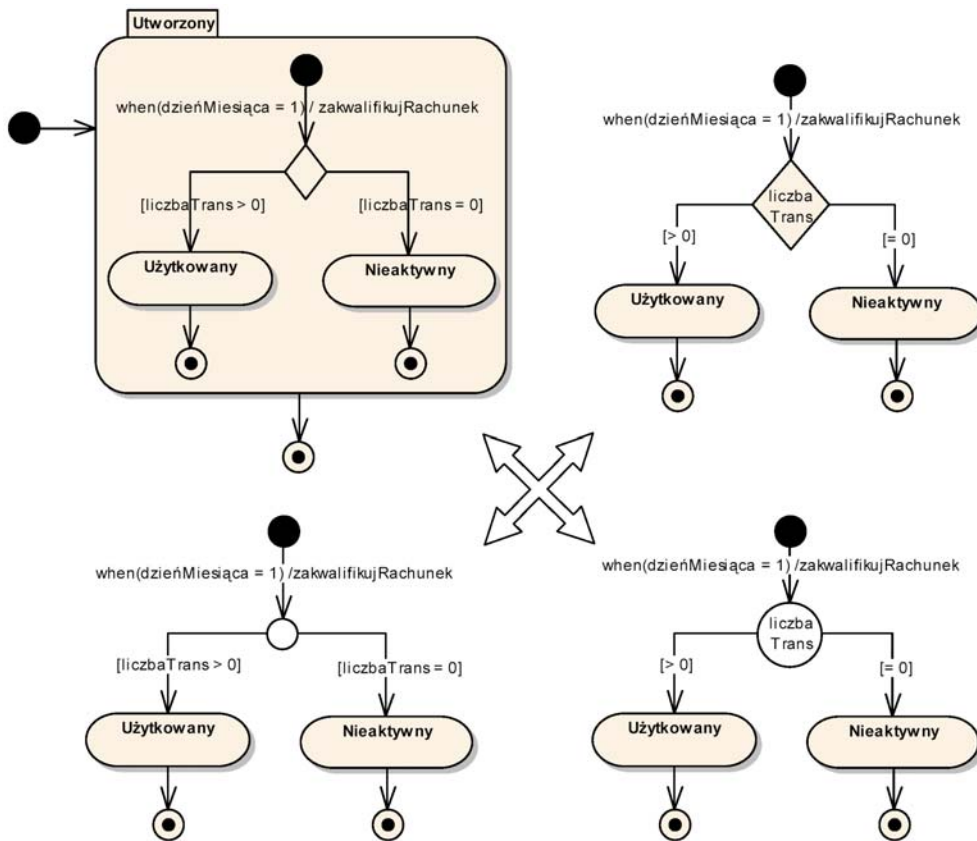
Diagram zawiera dwa stany złożone:

- ♦ *Tworzony*,
- ♦ *Aktywowany*.

W ramach pierwszego z nich wyróżnić można podstany *Planowany* oraz *Edytowany*. W momencie pierwszego aktywowania stanu złożonego *Tworzony* automatycznie aktywowany jest podstan *Planowany*. Po opuszczeniu stanu złożonego — co wiąże się z aktywowaniem stanu *Zatwierdzony* — możliwy jest późniejszy powrót do *Tworzenia*. W tej sytuacji pseudostan płytkiego wznowienia aktywuje albo podstan *Planowany*, albo *Edytowany* — w zależności od tego, który z nich był aktywny w momencie opuszczenia stanu *Tworzony*. Jak zaprezentowano na rysunku, podstan *Edytowany* sam w sobie jest stanem złożonym, a zatem zawiera własną sekwencję podstanów — czyli podmaszynę stanową. Ponieważ na diagramie wykorzystano stan płytkiego wznowienia, nie przechowuje się informacji o ostatnim aktywnym podstanie z wyżej wymienionych. Tak więc nawet jeśli w momencie powrotu do stanu złożonego *Tworzony* aktywowany zostanie podstan *Edytowany*, automatycznie przyjęty zostanie pierwszy z podstanów tego drugiego. Naturalnie przejścia pomiędzy stanami *Tworzony* oraz *Zatwierdzony* mogą być realizowane wielokrotnie. Pseudostan płytkiego wznowienia z każdym opuszczeniem pierwszego z wymienionych stanów jest aktualizowany, co zapewnia powrót do właściwego podstanu z każdym uaktywnieniem stanu złożonego *Tworzony*.

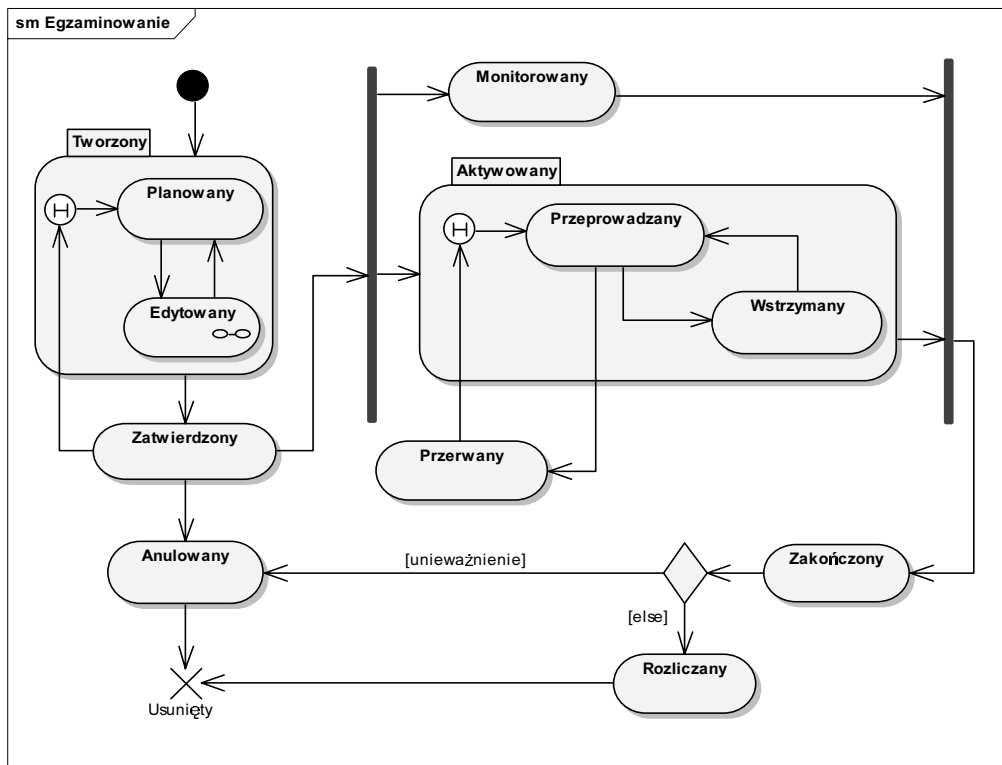
Tabela 5.2. Pseudostany w maszynach stanowych

Nazwa	Notacja	Interpretacja
Początkowy (ang. <i>initial</i>)		Zainicjowanie maszyny lub podmaszyny stanowej. Może zawierać akcje.
Płytkie wznowienie (ang. <i>shallow history</i>)		Wskazuje na ostatni aktywny stan obiektu przed wznowieniem. Nie przechowuje informacji o podstanach tego stanu. W przypadku pierwszej aktywacji pseudostanu wznowienia domyślnie aktywowany jest stan bezpośrednio powiązany z nim za pośrednictwem przejścia.
Głębokie wznowienie (ang. <i>deep history</i>)		Odpowiada ostatniej konfiguracji aktywnej stanu złożonego. Wskazuje na ostatni aktywny stan obiektu przed wznowieniem, przechowując zarazem informacje o podstanach tego stanu.
Scalenie (ang. <i>join node</i>)		Integruje kilka przejść wychodzących z różnych źródłowych obszarów współbieżnych, stanów lub maszyn stanowych.
Rozwidlenie (ang. <i>fork node</i>)		Pozwala na rozdzielenie wejściowego przejścia na dwa lub więcej przejść wyjściowych do różnych obszarów współbieżnych, stanów lub maszyn stanowych.
Punkt węzłowy (ang. <i>junction</i>)		Pozwala na złączenie wielu przejść wejściowych oraz ich rozdzielenie na szereg przejść wyjściowych. Jest on tzw. statycznym rozgałęzieniem warunkowym, co oznacza, że realizacja przejść wejściowych nie determinuje natychmiastowej realizacji przejść wyjściowych. Szczególnymi przypadkami punktu węzłowego są decyzja oraz złączenie.
Decyzja (ang. <i>decision node</i>)		Pozwala na podział przejścia wejściowego na kilka przejść wyjściowych określonych warunkami. Realizacja przejścia wejściowego determinuje natychmiastowy wybór i realizację przejść wyjściowych. W związku z tym decyzja jest dynamicznym rozgałęzieniem warunkowym.
Punkt wejścia (ang. <i>entry point</i>)		Jest łącznikiem specyfikującym przejście ze stanu do podmaszyny stanowej. Umieszcza się go na granicy stanu złożonego; opcjonalnie może być nazwany i (lub) zaznaczony w obszarze podmaszyny stanowej.
Punkt wyjścia (ang. <i>exit point</i>)		Jest łącznikiem specyfikującym przejście z podmaszyny stanowej do stanu. Umieszcza się go na granicy stanu złożonego; opcjonalnie może być nazwany i (lub) zaznaczony w obszarze podmaszyny stanowej.
Punkt zniszczenia (ang. <i>terminate</i>)		Zakończenie maszyny stanowej wskutek zniszczenia adekwatnego obiektu.



Rysunek 5.9. Alternatywne konwencje graficzne pseudostanu decyzji w kwalifikowaniu rachunku bankowego

Analogiczna interpretacja związana jest ze stanem złożonym *Aktywowany*. Przy jego opuszczaniu, czyli dezaktywacji, zapamiętywana jest informacja o aktywnym podstanie. I tak przejście do stanu *Przerwany* — wynikające z przyczyn technicznych, takich jak awaria komputera czy też utrata połączenia sieciowego — nie powoduje konieczności zdawania egzaminu od początku. W momencie powrotu do stanu złożonego *Aktywowany* wznowiany jest odpowiednio podstan *Przeprowadzany* lub *Wstrzymywany*. Jeśli egzamin znajduje się w stanie *Wstrzymywany*, oznacza to, że wykładowca zdecydował się zarządzić chwilową przerwę w egzaminowaniu. Tym samym możliwość interakcji studenta z systemem jest blokowana, a upływ czasu egzaminu zatrzymywany. Współbieżnie z *przeprowadzaniem* albo *wstrzymywaniem* egzamin jest *Monitorowany*. Po jego przejściu w stan *Zakończony* wykładowca podejmuje decyzję, czy rozliczyć egzamin, czy też unieważnić go. Warianty te wzajemnie się wykluczają. W pierwszym przypadku obiekt *Egzamin* przechodzi w stan *Rozliczany*, natomiast w drugim — w stan *Anulowany*. W obu przypadkach realizacja przejścia wyjściowego skutkuje wykasowaniem stosownego obiektu z bazy danych.

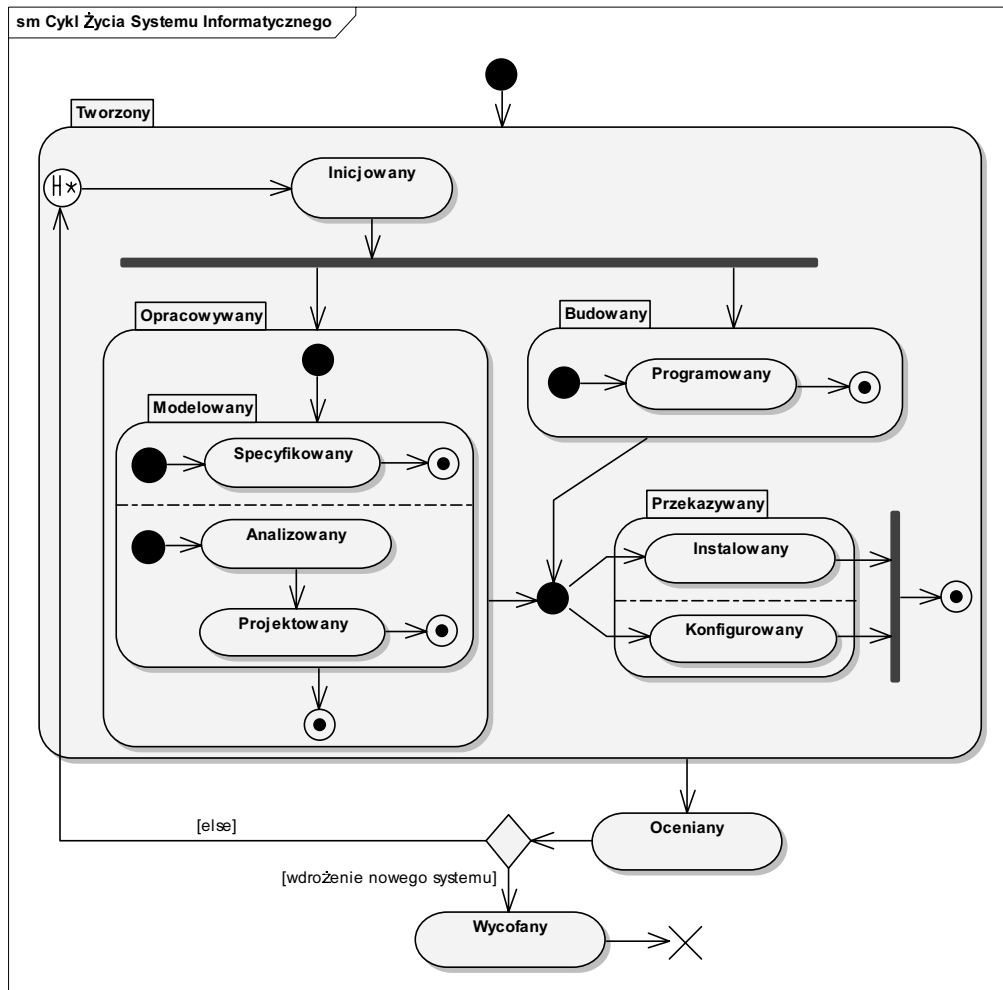


Rysunek 5.10. Wybrane pseudostany w maszynie stanowej systemu elektronicznego egzaminowania

Z kolei rysunek 5.11 zawiera interpretację pseudostanów głębokiego wznowienia, rozwidlenia, scalenia, decyzji, punktu węzłowego oraz stanu początkowego i punktu zniszczenia w połączeniu z obszarami współbieżnymi.

Przedstawia on maszynę stanową dla cyklu życia systemu informatycznego. Pojęcie cyklu życia systemu ściśle wiąże się z dziedziną inżynierii oprogramowania. W modelu obiektowym oznacza on podział procesu tworzenia systemu informatycznego na logicznie powiązane fazy realizacji oraz dyscypliny przepływów pracy. Układ ten ma charakter macierzowy. Zaprezentowany rysunek specyfikuje stany proste i złożone tworzące czteropoziomową hierarchię bezpośrednich i pośrednich zależności. System informatyczny jest w stanie *Tworzony*, gdyż obejmuje zdarzenia skutkujące wytwarzaniem dokumentacji oraz poszczególnych modułów oprogramowania. Stan *Tworzony* jest stanem złożonym i składa się z czterech podstanów:

- ♦ *Inicjowany*,
- ♦ *Opracowywany*,
- ♦ *Budowany*,
- ♦ *Przekazywany*.



Rysunek 5.11. Diagram maszyn stanowej cyklu życia systemu informatycznego

Podstanem aktywowanym w pierwszej kolejności w ramach stanu *Tworzony* jest prosty stan *Inicjowany*. W praktyce następuje rozpoczęcie projektu i wypracowana zostaje wizja systemu, która potwierdza powołanie przedsięwzięcia przez wszystkich jego udziałowców. Opuszczenie stanu *Inicjowany* prowadzi do równoczesnego aktywowania dwóch stanów:

- ♦ *Opracowywany*,
- ♦ *Budowany*.

Pierwszy z nich zawiera podstan *Modelowany*, który jest stanem złożonym podzielonym na dwa obszary współbieżne. Obszar pierwszy obejmuje podstan *Specyfikowany*, w którym analitycy biznesowi definiują wymagania, stosując przypadki użycia i szablony je opisujące. Drugi obszar obejmuje dwa sekwencyjnie aktywowane stany *Analizowany* i *Projektowany*. Wówczas analitycy systemowi oraz architekci analizują

specyfikację wymagań i przekształcają ją na architekturę systemu. Współbieżny ze stanem *Opracowywany* stan złożony *Budowany* zawiera jeden podstan o nazwie *Programowany*. W jego ramach programiści przekształcają projekt w definicję struktury kodu źródłowego, implementują poszczególne klasy itd. Przejścia wyjściowe obu aktywnych, współbieżnych podstanów stanu *Tworzenie* są kierowane do punktu węzłowego. Następnie aktywowane są podstany stanu *Przekazywany*. Stan ten jest podzielony na dwa obszary współbieżne. W ramach stanów *Instalowany* oraz *Konfigurowany* informatycy dostarczają wytworzony system informatyczny, tj. oprogramowanie oraz sprzęt. Przyjmując perspektywę stanu *Tworzony*, stany te są podstanami pośrednimi. Są one natomiast bezpośrednimi podstanami stanu *Przekazywany*.

Przejście wyjściowe ze stanu złożonego *Tworzony* prowadzi do stanu prostego *Oceniany*. W praktyce w ramach tego drugiego analitycy jakości oceniają użyteczność funkcjonującego systemu informatycznego. Stan ten prowadzi do decyzji o dalszym kierunku rozwoju systemu. Pseudostan decyzji rozgałęzia przejście, które w zależności od wyników oceny funkcjonalności powoduje aktywację stanu *Wycofany* lub stanu *Tworzony*. System przyjmuje pierwszy z nich, kiedy zostaje uznany za przestarzały i powinien być zastąpiony nowym systemem informatycznym. Przejście z tego stanu prowadzi wyłącznie do punktu zniszczenia. W praktyce oznacza to wykasowanie kodu źródłowego systemu informatycznego. W przeciwnym razie realizowane jest przejście oznaczone warunkiem przyjmującym postać słowa kluczowego *else*. Wówczas reaktywowany jest stan *Tworzony*. Ponieważ punktem docelowym przejścia jest pseudostan głębokiego wznowienia, wznawiany jest ostatni aktywny stan. Wznowienie to uwzględnia hierarchię podstanów i podmaszyn stanowych.

Rodzaje przejść

Przejścia (ang. *transitions*) występujące w ramach rozbudowanych diagramów maszyny stanowej są znaczeniowo zróżnicowane. Tak więc można wyróżnić następujące rodzaje przejść:

- ♦ **Proste**, zachodzące pomiędzy stanem źródłowym a docelowym.
- ♦ **Zwrotne**, których stanem źródłowym i docelowym jest ten sam stan. Realizacja przejścia zwrotnego nie prowadzi do zmiany stanu.
- ♦ **Wewnętrzne**, będące jednym z elementów danego stanu. Nie wychodzą one ze stanu źródłowego i nie są do niego ponownie wprowadzane. Ich realizacja również nie prowadzi do zmiany stanu.
- ♦ **Lokalne**, oznaczające przejścia pomiędzy dowolnymi podstanami danego stanu złożonego. Realizacja przejść lokalnych nie prowadzi do opuszczenia stanu złożonego.
- ♦ **Zewnętrzne**, opuszczające stan złożony i skierowane do innego stanu lub pseudostanu danej maszyny stanowej.
- ♦ **Wysokiego poziomu** albo **grupowe**, wychodzące ze stanów złożonych i oznaczające konieczność opuszczenia przed przejściem wszystkich podstanów tego stanu złożonego, począwszy od podstanu najbardziej zagnieżdżonego.

- ♦ **Złożone**, będące łańcuchem przejść powiązanych lub rozdzielanych scaleniem, rozwidleniem, punktem węzłowym bądź decyzją.
- ♦ **Automatyczne**, z którym nie jest związane żadne zdarzenie ani warunek.

W modelowaniu rozbudowanych przejść wysokiego poziomu w **hierarchii stanów złożonych** szczególne znaczenie mają następujące pseudostany:

- ♦ punkt wejścia,
- ♦ punkt wyjścia.

I tak zasady stosowania punktów wejścia i wyjścia w odniesieniu do stanów złożonych prezentuje rysunek 5.12. Poza wymienionymi w tabeli 5.2 graficznymi oznaczeniami punktu wejścia i punktu wyjścia zamiennie można stosować zapis tekstowy ze słowem kluczowym *via*. Tak więc składnia nazwy punktu wejścia lub wyjścia na odpowiednim przejściu jest następująca:

via <nazwa>

np. *via Skompletowany*

Protokołowe maszyny stanowe

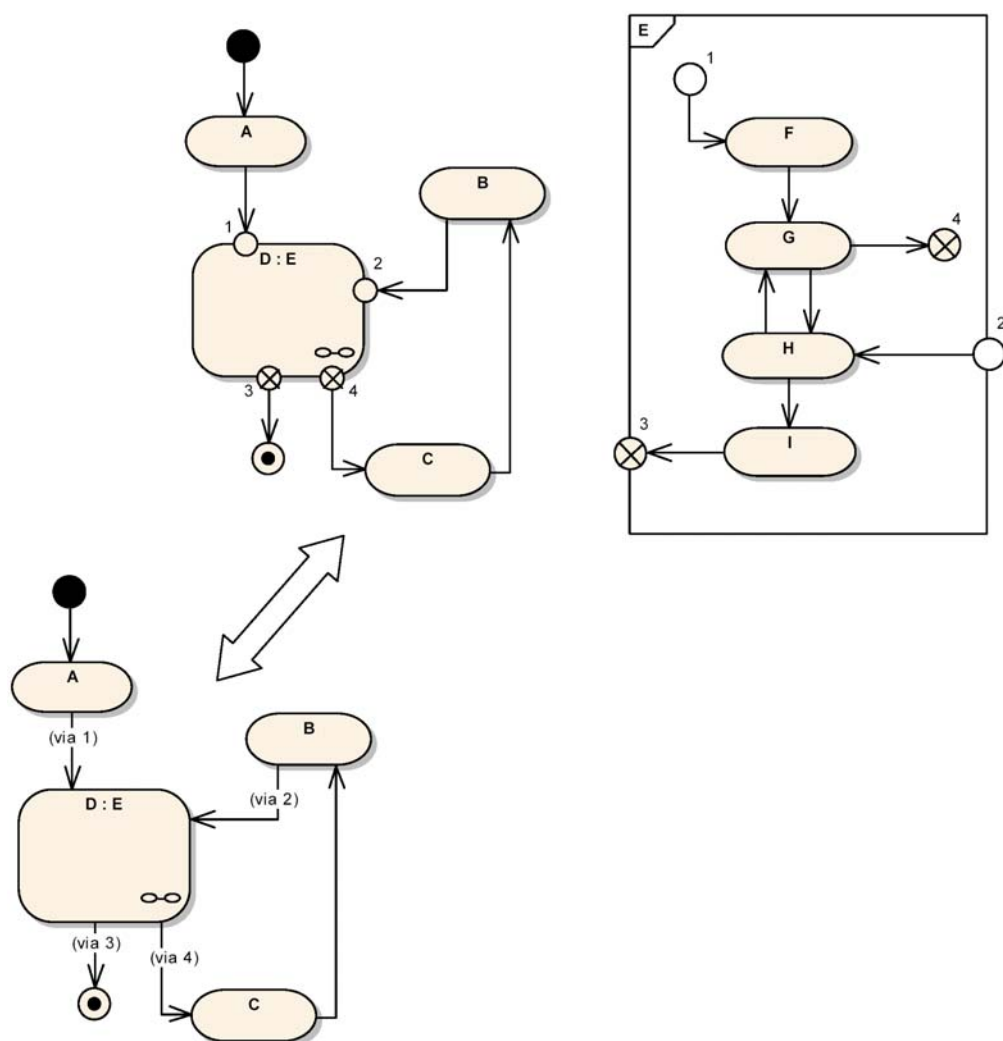
Maszyny stanowe można podzielić na:

- ♦ maszyny stanowe zachowania,
- ♦ protokołowe maszyny stanowe.

Pierwsze z wymienionych przedstawiają przejścia między stanami wielu obiektów w szerszym kontekście zachowania systemu, podsystemu czy też przypadku użycia. Natomiast **protokołowe maszyny stanowe** (ang. *protocol state-machines*) koncentrują się na konkretnym obiekcie. Przedstawiają dozwolone przejścia pomiędzy stanami tego obiektu. W przypadku opracowania dla danej dziedziny przedmiotowej obydwu wymienionych rodzajów maszyn stanowych muszą one być wzajemnie niesprzeczne. Stąd protokołowa maszyna stanowa reprezentuje wszystkie sekwencje operacji na obiekcie, które powodują uaktywnienie poszczególnych stanów tego obiektu. Operacje, które nie generują przejścia obiektu w inny stan, nie są reprezentowane w protokołowej maszynie stanowej. Ten rodzaj diagramów maszyny stanowej oznacza, że nie wskazuje się wewnętrznych czynności stanów, natomiast zaznacza operacje dotyczące przejść między poszczególnymi stanami.

Opis przejścia w protokołowej maszynie stanowej charakteryzuje się następującą **składnią**:

”[” <warunek-wstępny> ”]” <nazwa-operacji> ”/” ”[” <warunek-końcowy> ”]”



Rysunek 5.12. Pseudostany punktów wejścia i wyjścia — alternatywne notacje

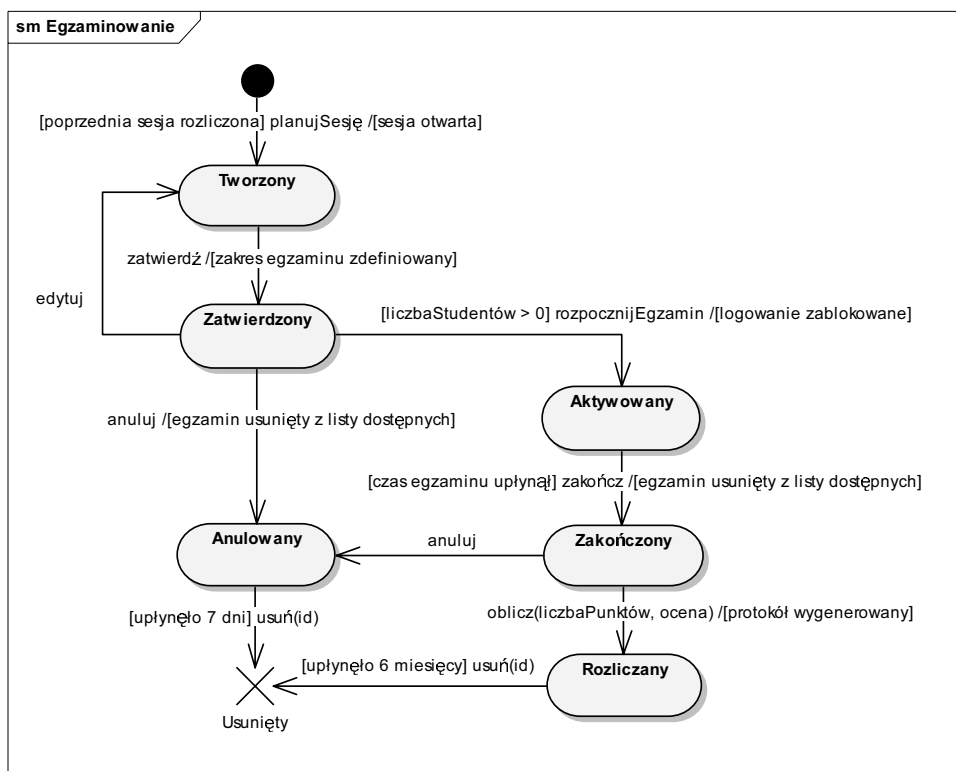
przy czym:

- ♦ *warunek-wstępny* to warunek, który musi być spełniony, aby mogła być wywołana operacja;
- ♦ *nazwa-operacji* wskazuje na jedną z operacji tego obiektu;
- ♦ *warunek-końcowy* to warunek, który jest generowany w konsekwencji wykonania operacji.

Protokołowe maszyny stanowe mogą wystąpić w następujących postaciach:

- ♦ **deklaratywnej**, która w sposób ogólny wskazuje ciąg dopuszczalnych stanów i przejść dotyczących konkretnego obiektu — jest to standardowa postać protokołowej maszyny stanowej;
- ♦ **wykonywalnej**, zawierającej ścisłą i precyzyjną specyfikację przejść oraz związanych z nimi operacji, umożliwiającej bezpośrednie generowanie kodu źródłowego.

Graficznie protokołowe maszyny stanowe zawierają większość kategorii i pojęć maszyn stanowych zachowania. Przedstawia się je w postaci obramowanego diagramu, w nagłówku którego umieszcza się wyróżnik typu diagramu, uzupełniony opcjonalnie o ograniczenie *{protocol}*. Diagramy ilustrujące protokołowe maszyny stanowe mogą stanowić wsparcie w definiowaniu interfejsów i portów na diagramach komponentów. Diagram taki przedstawia rysunek 5.13.



Rysunek 5.13. Diagram protokołowej maszyny stanowej dla obiektu klasy *Egzamin*

Systemy nauczania na odległość umożliwiają w ramach studiów uczestnictwo w wykładach, konferencjach czy też dyskusjach w wirtualnych klasach. Ich funkcjonalność umożliwia również elektroniczne sprawdzanie wiedzy studenta, w tym na odległość. Elektroniczny egzamin jest coraz częściej stosowaną formą, a jego implementacja odpowiada regułom tradycyjnego egzaminowania. Stany obiektu klasy *Egzamin* przedstawia

protokołowa maszyna stanowa zaprezentowana na rysunku 5.13. Pierwszym stanem tego obiektu jest stan *Tworzony*. Warunkiem wejścia w ten stan jest rozliczenie poprzedniej sesji. System nie pozwoli na stworzenie obiektu za pośrednictwem operacji *planujSesję*, o ile warunek ten nie zostanie spełniony. Warunkiem końcowym operacji jest otwarcie sesji. Przejście z opisywanego stanu do stanu *Zatwierdzony* następuje w wyniku wykonania operacji *zatwierdź*, przy czym warunkiem końcowym jest zdefiniowany zakres egzaminu. Praktyka udowodniła, że zakres ten może ulec zmianie. Tak więc *Egzamin* może powrócić do stanu *Tworzony*, jeśli wykładowca postanowi go edytować. Nie jest tym samym wymagane kasowanie uprzednio utworzonego egzaminu i definiowanie nowego. Obiekt powraca do stanu *Tworzony* po wywołaniu operacji *edytuj*.

Zainicjowanie stanu *Aktywowany* umożliwia przejście specyfikowane operacją *rozpocznijEgzamin*. Warunkiem jej wykonania jest dodatnia liczba zalogowanych do egzaminu studentów. Natomiast warunek końcowy blokuje logowanie studentów podczas egzaminu. Następnym stanem, w jakim może znaleźć się obiekt *Egzamin*, jest stan *Zakończony*. Jego aktywacja jest możliwa po wykonaniu operacji *zakończ* i dwóch warunków. Pierwszy z nich, będący warunkiem wstępnym, to upływanie czasu egzaminu. Drugi to warunek usunięcia *Egzaminu* z listy dostępnych egzaminów. Oba te warunki muszą zostać spełnione. *Zakończony* egzamin może zostać rozliczony. Stan *Rozliczany* jest aktywowany w momencie wykonania operacji *oblicz* z jednoczesnym spełnieniem warunku końcowego, tj. wygenerowaniem protokołów. W tym przejściu dodatkowo — w ramach składni operacji — zapisano parametry operacji: *liczbęPunktów* i *ocenę*. Są one atrybutami obiektu *Egzamin*, wykorzystywanymi podczas wykonywania operacji związanej z przejściem.

Po upływie sześciu miesięcy od daty rozliczenia egzaminu operacja *usuń* z parametrem *id* automatycznie kasuje obiekt *Egzamin*. Wartość parametru *id* wskazuje właściwy obiekt. Twórcy systemu nauczania na odległość, realizując wymagania wykładowców, przewidzieli możliwość anulowania zarówno zatwierdzonych, jak i zakończonych egzaminów. *Egzamin* przyjmuje stan *Anulowany* po wykonaniu operacji *anuluj*. Warunkiem końcowym jest tu usunięcie egzaminu z listy dostępnych egzaminów. Anulowany egzamin jest automatycznie usuwany z bazy danych po sześciu dniach. Wykonuje to operacja *usuń* z parametrem *id*. W przedstawionej protokołowej maszynie stanowej operacja *usuń*, podobnie jak operacja *anuluj*, jest wspólna dla dwóch przejść.

Maszyny stanowe zachowania

Maszyny stanowe zachowania (ang. *behavioral state-machines*) przedstawiają przejścia między stanami obiektów w szerszym niż w protokołowych maszynach stanowych kontekście zachowania systemu, podsystemu, przypadku użycia i innych obiektów. Opcjonalny opis przejścia w maszynie stanowej zachowania charakteryzuje się **składnią** zawierającą pojęcia zdarzenia, warunku i czynności, jak następuje:

```
[<sygnatura-zdarzenia>] ["/" <warunek> "/"] ["/" <wyrażenie-czynności>]
```

Poszczególne elementy przedstawionej składni mogą być dalej uszczegóławiane. I tak:

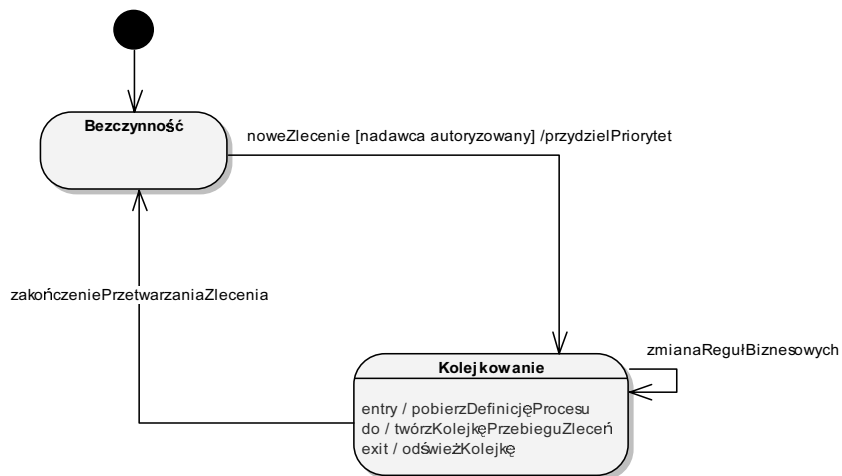
$\langle \text{sygnatura-zdarzenia} \rangle ::= \langle \text{nazwa-zdarzenia} \rangle [\text{"("} \langle \text{lista-parametrów} \rangle \text{"})"]$

$\langle \text{wyrażenie-czynności} \rangle ::= \langle \text{nazwa-akcji} \rangle | \langle \text{składnia-operacji} \rangle [\langle \text{składnia-atrybutu} \rangle]$

W przypadku występowania kilku parametrów są one oddzielane przecinkami. Praktyczne zastosowanie poszczególnych elementów składni przejścia w maszynie stanowej zachowania ilustruje rysunek 5.14.

Rysunek 5.14.

Kolejkowanie projektów jako przykład specyfikacji przejść



Systemy oparte na technologii WFM umożliwiają definiowanie procesu oraz interpretację tej definicji dla pojawiających się zleceń. Stan, w którym system interpretuje definicję zlecenia oraz nadaje jej odpowiedni numer w kolejce obsługi, nazywa się *Kolejkowaniem*. Aktywacja tego stanu jest konsekwencją przejścia ze stanu *Bezczynności*. Pojawienie się nowego zlecenia od autoryzowanego nadawcy powoduje *przypiszeniePriorytetu* obsługi zlecenia. W momencie wejścia w stan *Kolejkowania* wykonywana jest czynność *pobraniaDefinicjiProcesu*. Pozwala to systemowi obsługiwać zlecenia zgodnie z regułami biznesowymi firmy. Gdy system jest w stanie *Kolejkowania*, wykonywana jest czynność *tworzeniaKolejkiPrzebieguZleceń*. Jest ona wykonywana tak długo, jak stan pozostaje aktywny. W momencie wyjścia ze stanu *Kolejkowania* wykonywana jest czynność *odświeżeniaKolejki*. Taka procedura zapewnia utrzymywanie właściwej kolejki zleceń.

Poza wbudowanym modułem poczty elektronicznej systemy WFM oferują możliwość definiowania procesu obsługi zleceń. Odpowiednie narzędzia monitorowania i administrowania pozwalają na ich implementację w trakcie funkcjonowania systemu. Maszyna stanowa przewiduje wystąpienie zdarzenia zmiany reguł biznesowych. W takiej sytuacji wykonywane jest przejście zwrotne, które powoduje wykonanie dwóch czynności oznaczonych słowami kluczowymi *entry* i *exit*. Tak wyspecyfikowane przejście powoduje *odświeżenieKolejki* z uwzględnieniem nowych reguł kolejkowania. System WFM przechodzi w stan *Bezczynności*, gdy obsługa kolejkowania zostanie zakończona.

Zdarzenia

Przejścia z jednego stanu do drugiego inicjowane są przez **zdarzenia** (ang. *events*), stanowiące bodźce wysyłane przez inne obiekty. Wyróżnia się pięć rodzajów zdarzeń (tabela 5.3), przy czym zdarzenie odroczone przyjmuje w praktyce postać kolejgowanego w odmienny sposób zdarzenia o charakterze sygnału, zdarzenia wywołania, zdarzenia czasowego albo zdarzenia zmiany stanu.

Tabela 5.3. Rodzaje zdarzeń

Nazwa zdarzenia	Charakterystyka	Przykład
Sygnał (ang. <i>signal</i>)	Asynchroniczny bodziec przesyłany pomiędzy dwoma obiektami.	♦ PrzeciążenieSieci ♦ NadejścieWiadoomości ♦ OdebranieSygnałuZCzujnika
Zdarzenie wywołania (ang. <i>call event</i>)	Wywołanie przez obiekt operacji innego obiektu, z którym skojarzona jest maszyna stanowa. Zazwyczaj ma ono charakter synchroniczny.	♦ zaloguj(egzaminator) ♦ generujFakturę ♦ zarejestruj(kwota, data)
Zdarzenie czasowe (ang. <i>time event</i>)	Zdarzenie, które występuje po upływie określonego czasu. Opisywane słowem kluczowym <i>after</i> . Czasem początkowym przy obliczaniu wyrażenia jest moment aktywowania stanu obiektu, o ile nie wskazano inaczej.	♦ after(3 godz) ♦ after(tydzień) ♦ after(2 sek since exiting Rozliczany)
Zdarzenie zmiany stanu (ang. <i>change event</i>)	Zdarzenie, które występuje w określonym momencie. Opisywane słowem kluczowym <i>when</i> .	♦ when(dzień = 1 październik) ♦ when(dataSystemowa = 03.00) ♦ when(cena < cenaMinimalna)
Zdarzenie odroczone (ang. <i>deferred event</i>)	Zdarzenie nieobsługiwane w ramach danego stanu i dopisane do kolejki zdarzeń odroczonech. Poszczególne zdarzenia na tej liście są przetwarzane po uaktywnieniu innego stanu, który ich nie odracza. Opisywane słowem kluczowym <i>defer</i> umieszczanym po nazwie zdarzenia i symbolu ukośnika.	♦ zmianaZasadPunktacji / defer ♦ zmianaAlgorytmuLosowania-Pytań / defer ♦ podsumowanieNależności / defer

Dane przejście może zostać zrealizowane wtedy i tylko wtedy, gdy opcjonalny **warunek** (ang. *guard condition*) z nim związany jest spełniony. Po wykonaniu czynności związanych z przejściem następuje aktywacja stanu docelowego.

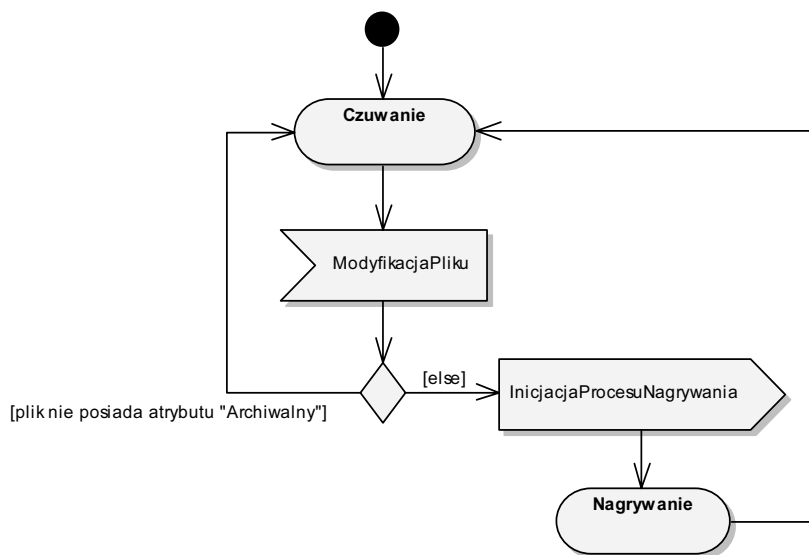
Wystąpienia zdarzeń pomiędzy dwoma stanami i ich obsługę przedstawia się w postaci diagramu z uwzględnieniem następujących oznaczeń:

- ♦ stanów,
- ♦ zdarzeń,
- ♦ akcji,
- ♦ pseudostanów.

Przykład ilustrujący sporządzanie kopii zapasowej, w którym zdarzeniem aktywującym nagrywanie jest modyfikacja pliku, przedstawiono na rysunku 5.15.

Rysunek 5.15.

*Tworzenie kopii
zapasowej pliku*



Proces tworzenia diagramu maszyny stanowej

Proces tworzenia diagramów maszyny stanowej przebiega w następujących **etapach**:

1. określenie rodzaju tworzonej maszyny stanowej i identyfikacja obiektów;
2. zidentyfikowanie poszczególnych stanów maszyny stanowej;
3. określenie hierarchii stanów, podstanów oraz obszarów współbieżnych;
4. powiązanie stanów oraz ich podstanów przejściami;
5. zastosowanie adekwatnych pseudostanów;
6. opracowanie specyfikacji sekcji stanów i przejść zgodnie z przyjętymi składnikami.

Proces tworzenia diagramów maszyny stanowej powinien być realizowany zgodnie z koncepcją iteracyjno-przyrostowego cyklu życia systemu informatycznego. Oznacza to, że diagram maszyny stanowej jest uszczegóławiany w kolejnych iteracjach danej fazy, co powoduje przyrost jego dokumentacji aż do osiągnięcia pełnego jej zakresu.

Studium diagramu maszyny stanowej

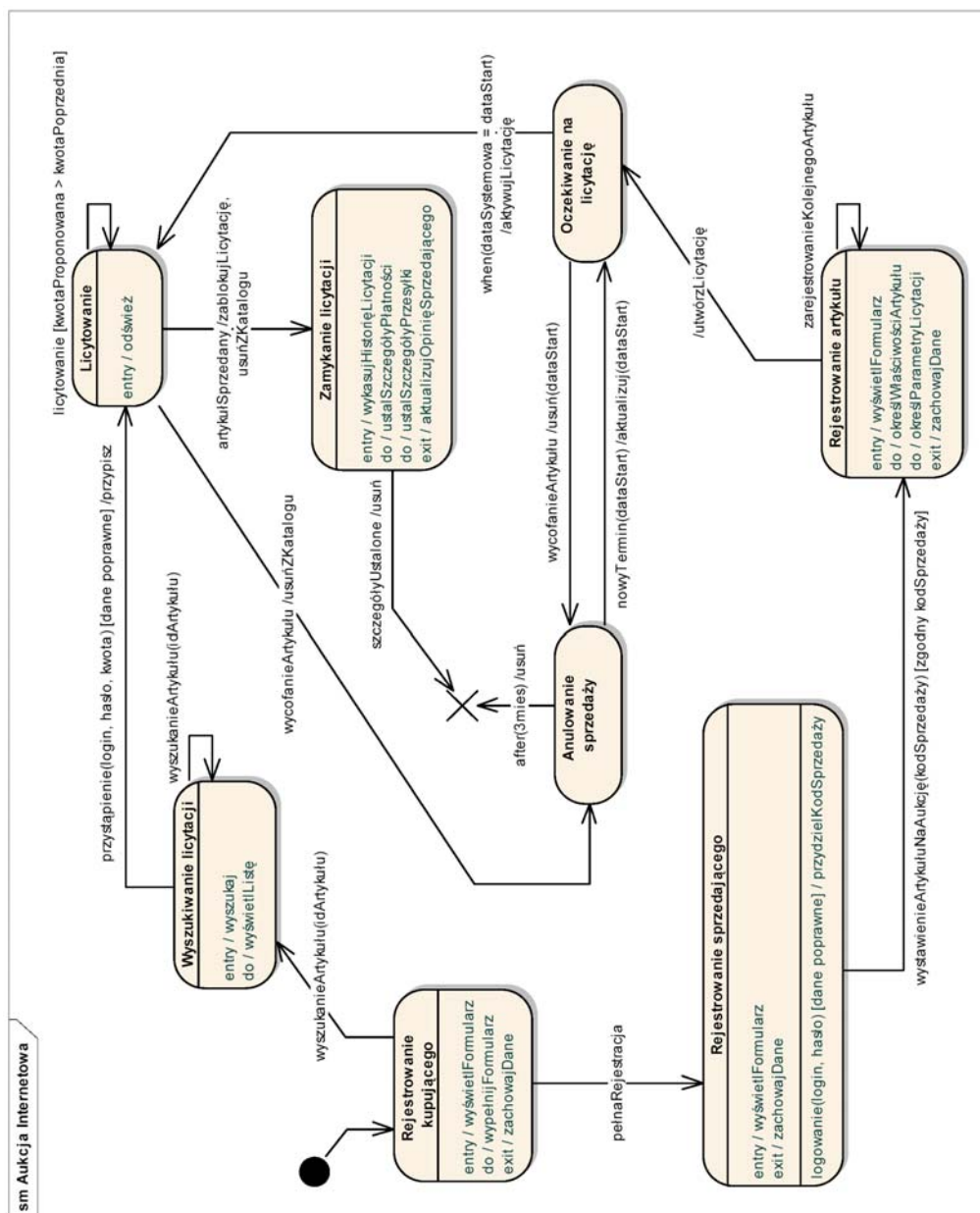
Diagram maszyny stanowej zachowania, zawierający szereg elementów podstawowych oraz zaawansowanych, przedstawia rysunek 5.16. Dziedzinę przedmiotową stanowi w tym przypadku funkcjonowanie systemu aukcji internetowej.

Istnieje szereg modeli biznesu elektronicznego, spośród których znaczącą rolę odgrywają aukcje internetowe. Standardowo umożliwiają one rejestrację uczestników i sprzedawanych artykułów oraz ich licytowanie. Obsługa transakcji jest realizowana opcjonalnie. Pierwszym stanem jest stan *Rejestrowanie kupującego*. W momencie wejścia w ten stan system wyświetla *Formularz*, który wypełnia przyszły uczestnik licytacji. Przejście z tego stanu powoduje *zachowanieDanych*. Może być ono przejściem w stan *Rejestrowanie sprzedającego* lub *Wyszukiwanie licytacji*. W pierwszym przypadku musi wystąpić zdarzenie *pełnejRejestracji*. W praktyce oznacza to, że uczestnik może wystąpić w roli kupującego, jak również sprzedawać artykuły — czyli mieć prawo do tworzenia licytacji w systemie. Zatem podczas stanu *Rejestrowania sprzedającego* kupujący musi się *zalogować*. Następnie system identyfikuje login i hasło. W przypadku potwierdzenia tożsamości kupującego system *przydzielaKodSprzedaży*. Gdy stan zostaje dezaktywowany, *kodSprzedaży* oraz informacje związane ze zdarzeniem wewnętrznym stanu zostają *zachowane* w bazie.

Stan *Rejestrowania sprzedającego* dezaktywowany jest w wyniku wystąpienia zdarzenia *wstawienieArtykułuNaAukcję*. Sprawdzany jest wówczas warunek zgodności wprowadzonego *koduSprzedaży*, co umożliwia systemowi identyfikację uczestnika jako uprawnionego do rejestracji artykułu. Artykuł ten stanie się później przedmiotem licytacji. W chwili wejścia w stan *Rejestrowanie artykułu* system wyświetla *Formularz* rejestracyjny. Sprzedający wykonuje dwie czynności:

- ♦ *określaWłaściwościArtykułu*,
- ♦ *określaParametryLicytacji* — w tym daty otwarcia i zamknięcia licytacji oraz tzw. cenę minimalną i cenę „kup teraz”.

Podanie cen jest opcjonalne. Obie wymienione czynności są poprzedzone słowem kluczowym *do*. W momencie wyjścia ze stanu wykonuje się czynność oznaczona słowem kluczowym *exit*, powodując *zachowanieDanych* wprowadzonych przez sprzedającego. W praktyce występują zdarzenia, w których sprzedający sekwencyjnie rejestrują większą liczbę artykułów. Dlatego też stan *Rejestracji artykułu* ma przejście zwrotne, nazwane adekwatnie do wspomnianego zdarzenia. Po zarejestrowaniu artykułu i zachowaniu danych następuje wykonanie akcji *utworzeniaLicytacji*. Aktywowany jest wówczas stan *Oczekiwanie na licytację*. Stan ten nie jest jednoznaczny z otwarciem



Rysunek 5.16. Maszyna stanowa systemu aukcji internetowej

licytacji, czyli możliwością przypisywania kupujących i rejestrowania kwot proponowanych za artykuł. Jest to stan, w którym zarejestrowany artykuł staje się licytacją. W momencie, gdy data systemowa jest równa dacie otwarcia licytacji, licytacja staje się dostępną w katalogu aukcji i aktywowany jest stan *Licytowania*.

Wracając do stanu *Rejestrowania kupującego*, można zauważyć, że jest możliwe przejście do stanu *Wyszukiwania licytacji*. Następuje to w momencie, gdy zarejestrowany uczestnik wyszukuje w katalogu aukcji interesujące go artykuły. Wykorzystuje on w tym celu formatkę wyszukiwania. Wprowadza nazwę artykułu, potwierdza wybór i oczekuje na odpowiedź. W odpowiedzi system *wyświetla* Listę wszystkich licytacji, na których przedmiotem licytacji jest poszukiwany artykuł. Kupujący może oczekiwać od systemu, że ten *wyszuka* licytację innego artykułu. Przejście zwrotne opisane zdarzeniem *wyszukanieArtykułu*, z parametrem *idArtykułu*, umożliwia realizację opisywanej funkcji. W momencie wykonania przejścia następuje ponowne przeszukanie katalogu aukcji z aktywnymi licytacjami i wyświetlenie na interfejsie zaktualizowanej ich listy. W chwili, w której kupujący decyduje się przystąpić do licytacji, wprowadza on do odpowiednich pól interfejsu login, hasło oraz kwotę. Sytuacja ta zapisana jest jako zdarzenie z parametrami. Warunkiem akceptacji i zarejestrowania proponowanej kwoty jest poprawność loginu i hasła warunkującego identyfikację uczestnika. Wystąpienie zdarzenia spełniającego warunek powoduje wykonanie akcji *przypisania* kupującego do wybranej licytacji. Aktywowany jest wówczas stan *Licytowanie*. Akcja *odświeżenia* powoduje umieszczenie nazwy kupującego wraz z proponowaną kwotą na liście wszystkich uczestników danej licytacji. Podczas stanu *Licytowania* kupujący może zaproponować wyłącznie kwotę większą od poprzednio proponowanej. Przejście zwrotne związane z tym stanem, opisane warunkiem $kwotaProponowana > kwotaPoprzednia$, jest odwzorowaniem tej zasady. Zdarzenie *artykulSprzedany* powoduje zmianę aktywnego stanu na stan *Zamykanie licytacji*.

Zgodnie z regulaminem aukcji artykuł zostaje uznany za sprzedany, jeśli:

- ♦ data systemowa jest równa określonej przez sprzedającego *dacieStop* — co oznacza, że minął czas trwania licytacji — a do licytacji przypisany został co najmniej jeden kupujący;
- ♦ jeden z kupujących zaproponował cenę „kup teraz” lub większą od niej.

Cena „kup teraz” jest opcjonalnie podawana przez sprzedającego podczas rejestrowania artykułu. Wskazuje wysokość proponowanej przez kupujących kwoty, po osiągnięciu której licytacja jest zamykana. W obu tych przypadkach artykuł uważa się za sprzedany. Zatem wystąpienie zdarzenia *artykulSprzedany* powoduje wykonanie czynności *zablokowaniaLicytacji* oraz *usunięciaLicytacjiZKatalogu* prowadzonych licytacji. Moment wykonania przejścia jest jednoznaczny z aktywacją stanu *Zamykania licytacji*. Jego aktywacja powoduje wykonanie czynności oznaczonej słowem kluczowym *entry*, czyli *wykasowanieHistoriiLicytacji*. Odpowiednia operacja usuwa wówczas z bazy danych aukcji internetowej proponowane kwoty innych kupujących.

Sprzedający i kupujący ustalają warunki płatności oraz przesyłki. Po ich ustaleniu system *aktualizujeOpinięSprzedającego* oraz automatycznie usuwa dane dotyczące zarejestrowanego artykułu. Wracając do stanu *Oczekiwania na licytację*, można stwierdzić, że dostępne jest przejście do stanu *Anulowania sprzedaży*. Stan ten jest uwarunkowany wystąpieniem zdarzenia *wycofanieArtykułu* i wykonaniem czynności *usunięcia daty-Start* anulowanej licytacji. Również podczas trwania stanu *Licytowanie* istnieje możliwość *wycofaniaArtykułu* ze sprzedaży. Jednak w takiej sytuacji następuje całkowite usunięcie licytacji z katalogu prowadzonych licytacji. W obu przypadkach aktywowany jest stan *Anulowanie sprzedaży*. Jeśli w okresie następnych trzech miesięcy

sprzedający nie wyznaczy *nowego Terminu* otwarcia anulowanej licytacji, system automatycznie wykona przejście do pseudostanu powiązanego ze stanem *Anulowanie sprzedaży*. Skutkiem tego przejścia jest *usunięcie* z bazy danych wszystkich informacji związanych z artykułem.

Podstawowe pojęcia

Akcja	Wewnętrzne
Czynność	Wysokiego poziomu (grupowe)
Wewnętrzna do	Zewnętrzne
Wewnętrzna entry	Złożone
Wewnętrzna exit	Zwrotne
Diagram maszyny stanowej	Przypadek użycia
Definicja	Pseudostan
Proces tworzenia	Decyzja
Interakcja	Początkowy
Maszyna stanowa	Punkt wejścia
Podmaszyna stanowa	Punkt węzłowy
Protokołowa	Punkt wyjścia
Deklaratywna	Punkt zniszczenia
Wykonywalna	Rozwidlenie
Zachowania	Scalenie
Notacja zakładkowa	Wznowienie
Obszar współbieżny	Głębokie
Separator	Płytkie
Przejście	Sekcja
Automatyczne	Czynności wewnętrznych
Lokalne	Dekompozycji
Proste	Nazwy
Składnia	Przejęć wewnętrznych
Sygnatura zdarzenia	Stan
Warunek	Aktywny
Wyrażenie czynności	Końcowy

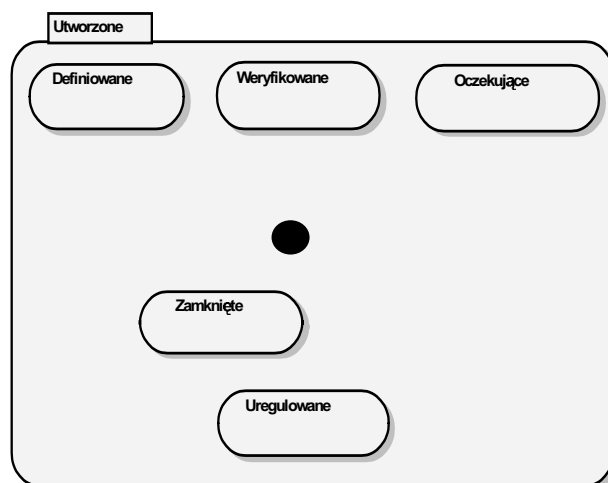
Podstan	Zdarzenie
Bezpośredni	Czasowe
Pośredni	Odroczone
Prosty	Sygnał
Złożony	Wywołania
Składnia nazwy	Zmiany stanu

Pytania i zadania

1. Wskaż rodzaje systemów informatycznych, przy specyfikowaniu których diagramy maszyny stanowej są szczególnie użyteczne. Uzasadnij swój wybór.
2. Jak należy rozumieć pojęcie maszyny stanowej? Wyjaśnij, jak rozumiesz jej dyskretne zachowanie.
3. Wymień podstawowe kategorie pojęciowe diagramu maszyny stanowej.
4. Co rozumie się przez stan obiektu? Czym charakteryzuje się stan początkowy oraz stan końcowy?
5. Jaką rolę pełni przejście w diagramach maszyny stanowej?
6. Zaprezentuj maszynę stanową następujących obiektów:
 - ♦ obiektu *Klient*,
 - ♦ konta bankowego.
7. W jaki sposób diagramy maszyny stanowej mogą być uszczegóławiane poprzez wprowadzenie zaawansowanych kategorii pojęciowych?
8. Jaka jest rola sekcji stanów? Ile wyróżnia się różnych sekcji i co zawierają?
9. Jakimi słowami kluczowymi charakteryzują się czynności wejściowe oraz wyjściowe? Kiedy są uruchamiane?
10. Rozbuduj stany zaprezentowane na rysunku 5.1 o sekcję czynności wewnętrznych.
11. Wyspecyfikuj wybrany stan związany z obiektem *Wypożyczenie* systemu bibliotecznego. Obowiązkowo uwzględnij wszystkie sekcje.
12. Wskaż różnice pomiędzy przejściem zwrotnym a wewnętrznym.
13. Wymień zależności pomiędzy kategoriami pojęciowymi:
 - ♦ stan złożony,
 - ♦ podstan,
 - ♦ podmaszyna stanowa,
 - ♦ obszar współbieżny.

14. Zilustruj konwencję ukrytą i jawną specyfikacji stanów na przykładzie stanu złożonego *Obsługiwane*, który dotyczyłby zlecenia zakupu jednostek uczestnictwa funduszu inwestycyjnego.
15. Zinterpretuj pojęcie obszaru współbieżnego.
16. Rozważ stan złożony obsługi nowego klienta w systemie operatora komórkowego pod kątem obszarów współbieżnych.
17. Czy kategorię pojęciową stanu początkowego należy zakwalifikować jako stan czy pseudostan?
18. W odniesieniu do maszyny stanowej przedstawionej na rysunku 5.8 zaproponuj odpowiedni pseudostan, który umożliwi wielokrotne przetwarzanie danej sprawy. Musi on uwzględniać wznowianie przerwanych przetwarzania sprawy, począwszy od stanu, w którym zostało ono przerwane.
19. Na rysunku 5.11 zastosowano stan głębokiego wznowienia. Sporządź analogiczny diagram maszyny stanowej z wykorzystaniem płytkich wznowień. Rozmieść je w taki sposób, aby logika funkcjonowania maszyny stanowej pozostała niezmienną.
20. Rysunek 5.17 przedstawia stan złożony z punktem węzłowym obiektu klasy *Wypożyczenie* systemu obsługi wypożyczeń płyt DVD. Uzupełnij diagram o stosowne elementy i dokonaj jego opisu.

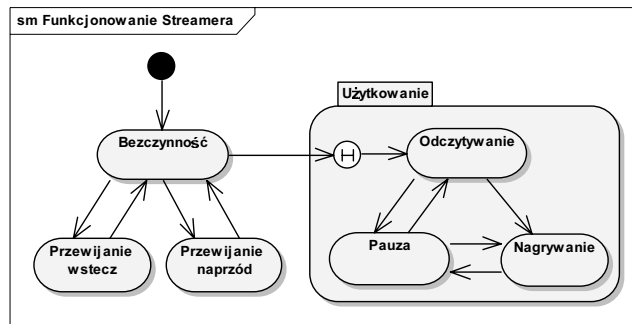
Rysunek 5.17.
Stan złożony obiektu
klasy *Wypożyczenie*



21. Zapoznaj się z funkcjonowaniem komputera wieloprocesorowego. Przedstaw obsługę zadań w tym komputerze w kategoriach pseudostanów scalenia, rozwidlenia, punktu węzłowego i innych. Rozważ opisywane zadanie w kontekście funkcjonowania klastra komputerowego obsługującego metropolitarną sieć komputerową TASK (www.task.gda.pl).
22. Stosując zaawansowane kategorie pojęciowe diagramu maszyny stanowej, opracuj maszynę stanową sieci lokalnej wykorzystującej koncepcję przekazywania znacznika.

23. Wyjaśnij różnicę pomiędzy punktem wyjścia a punktem zniszczenia. Podaj przykład.
24. W ramach rysunku 5.11:
- ♦ zastosuj dla stanu *Opracowywany* ukrytą konwencję zapisu oraz wyodrębnij jego podmaszynę stanową na osobnym diagramie;
 - ♦ zastosuj punkty wejścia i wyjścia na obu diagramach, odpowiednio je nazywając.
25. Wyjaśnij różnice pomiędzy przejściami:
- ♦ wewnętrznym, lokalnym a zewnętrznym;
 - ♦ wysokiego poziomu a złożonym.
26. Wyjaśnij różnice zastosowania maszyn stanowych zachowania oraz protokołowych maszyn stanowych. Która postać protokołowych maszyn stanowych umożliwia bezpośrednie generowanie kodu?
27. Za pomocą jakiego ograniczenia jednoznacznie identyfikuje się diagram protokołowej maszyny stanowej?
28. Wyjaśnij, który człon składni przejścia zwrotnego związanego ze stanem *Kolejkowanie*, zamieszczonym na rysunku 5.14, stanowi zdarzenie, a który czynność? Jaki jest związek między tymi dwoma elementami?
29. Na przykładzie rysunku 5.13 wytłumacz, w jaki sposób na diagramie maszyny stanowej oznacza się warunek. W jakich diagramach języka UML występuje identyczne oznaczenie warunku?
30. Na podstawie protokołowej maszyny stanowej zaprezentowanej na rysunku 5.13 opracuj maszynę stanową zachowania. Uwzględnij zasadę wzajemnego dostosowania i zgodności.
31. Podaj przykład stanu zachowania oraz stanu protokołowego w kontekście systemu rozliczania rozmów telefonicznych.
32. Opierając się na diagramie maszyny stanowej zachowania przedstawionej na rysunku 5.17, opracuj diagramy protokołowych maszyn stanowych dla poszczególnych obiektów:
- ♦ *Artykułu*,
 - ♦ *Kupującego*,
 - ♦ *Sprzedającego*,
 - ♦ *Licytacji*.
33. Uzupełnij przejścia w diagramie zaprezentowanym na rysunku 5.18 o stosowne zdarzenia i warunki.
34. Jakie czynności wewnętrzne ma zaprezentowany na rysunku 5.18 stan złożony *Użytkowanie*? Uzupełnij diagram o inne stany i przejścia związane z funkcjonowaniem streamera.

Rysunek 5.18.
Funkcjonowanie streamera



35. Podaj przykłady następujących rodzajów zdarzeń:

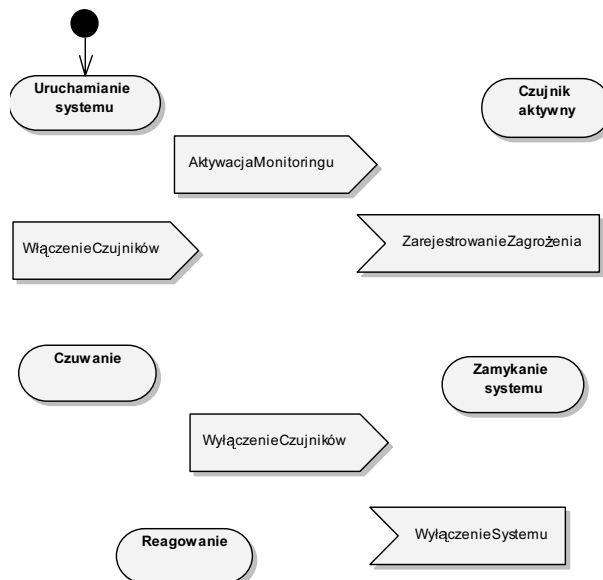
- ♦ sygnału,
- ♦ zdarzenia wywołania,
- ♦ zdarzenia czasowego,
- ♦ zdarzenia zmiany stanu;

dla następujących dziedzin przedmiotowych:

- ♦ funkcjonowania portalu internetowego,
- ♦ tworzenia systemów informatycznych,
- ♦ użytkowania wybranych urządzeń elektronicznych.

36. Analityk zidentyfikował podstawowe stany oraz sygnały systemu monitorowania pomieszczeń (rysunek 5.19). Wykorzystując podstawowe i zaawansowane kategorie pojęciowe, opracuj właściwy diagram maszyny stanowej zachowania tego systemu.

Rysunek 5.19.
Stany i sygnały systemu monitorowania pomieszczeń



Rozdział 6.

Diagramy interakcji

Interakcje

Kluczowym zadaniem diagramów przypadków użycia jest przedstawienie ogólnej wizji funkcjonalności systemu oraz jego wymagań. Wizja ta w kolejnych iteracjach cyklu przyrostowego jest stopniowo uszczegóławiana w postaci dokumentacji analitycznej, projektowej, wdrożeniowej i eksploatacyjnej. Poszczególne przypadki użycia są dokumentowane już w fazie definiowania wymagań. Do podstawowych składników dokumentacji przypadków użycia należy określenie przepływu głównego oraz przepływów alternatywnych. Przepływy te wskazują na **współdziałanie**, komunikację instancji klasyfikatorów w celu realizacji dynamicznych aspektów systemu. Precyzyjne zdefiniowanie komunikowania się poszczególnych instancji klasyfikatorów, decydujące o praktycznej użyteczności tworzonego systemu, realizowane jest z wykorzystaniem różnych rodzajów diagramów interakcji.

W każdym systemie informatycznym występuje szereg elementów wzajemnie oddziałujących na siebie, komunikujących się ze sobą. Te interakcje, szczególnie w systemach czasu rzeczywistego oraz wbudowanych, są niezwykle złożone. Ich analiza, a zwłaszcza opracowanie spójnego projektu systemu, wymaga precyzyjnego opisu wzajemnych interakcji instancji klasyfikatorów. Tak więc **interakcję** (ang. *interaction*) można zdefiniować w następująco:



Interakcja to wymiana bodźców, impulsów i komunikatów pomiędzy instancjami klasyfikatorów w systemie.

Diagramy interakcji

Świadectwem złożoności interakcji w systemie jest zaproponowanie i opracowanie w standardzie UML 2 czterech **rodzajów diagramów interakcji**:

- ♦ sekwencji,
- ♦ komunikacji,
- ♦ harmonogramowania,
- ♦ sterowania interakcją.

Należy zaznaczyć, że pojęcie diagramu interakcji jest pojęciem **abstrakcyjnym** — w praktyce przybiera on postać jednego z czterech wyżej wymienionych rodzajów diagramów. Poszczególne rodzaje diagramów interakcji zostały szczegółowo scharakteryzowane w rozdziałach 6. – 10. Ich syntetyczną charakterystykę przedstawia tabela 6.1.

Tabela 6.1. Rodzaje i charakterystyka diagramów interakcji

Rodzaj diagramu	Charakterystyka
Diagram sekwencji (ang. <i>sequence diagram</i>)	opisuje interakcje pomiędzy instancjami klasyfikatorów systemu w postaci sekwencji komunikatów wymienianych między nimi
Diagram komunikacji (ang. <i>communication diagram</i>)	specyfikuje strukturalne związki pomiędzy biorącymi udział w interakcji instancjami klasyfikatorów oraz wymianę komunikatów pomiędzy tymi instancjami
Diagram harmonogramowania (ang. <i>timing diagram</i>)	reprezentuje na osi czasu zmiany dopuszczalnych stanów, jakie może przyjmować instancja klasyfikatora uczestnicząca w interakcji
Diagram sterowania interakcją (ang. <i>interaction overview diagram</i>)	dokumentuje przepływ sterowania pomiędzy logicznie powiązanymi diagramami i fragmentami interakcji z wykorzystaniem kategorii modelowania diagramów czynności

Diagramy sekwencji i diagramy komunikacji to podstawowe rodzaje diagramów interakcji. Mają charakter izomorficzny. Jedynie te dwa rodzaje diagramów występowały w wersjach 1.x języka UML. Diagramy interakcji zostały w znaczący sposób rozbudowane w wersji 2.0. Zmiana ta dotyczy nie tylko wprowadzenia dwóch nowych rodzajów diagramów interakcji: diagramów harmonogramowania oraz diagramów sterowania interakcją, lecz także istotnego rozbudowania diagramów sekwencji oraz komunikacji o nowe kategorie modelowania.

Składnia komunikatu

Kategorią pojęciową języka UML o podstawowym znaczeniu dla interakcji, zwłaszcza na diagramach sekwencji i komunikacji, jest **komunikat** (ang. *message*).

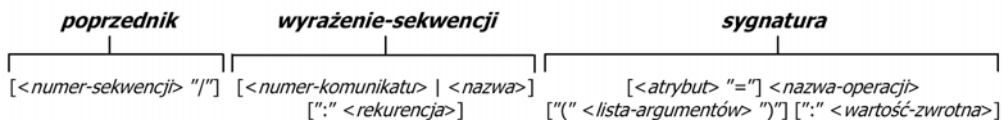


Komunikat to specyfikacja wymiany informacji między obiektami, zawierająca zlecenia wykonania określonej operacji.

Składnia komunikatu w jego pełnej formie przyjmuje następującą postać:

$$\langle \text{komunikat} \rangle ::= [\langle \text{poprzednik} \rangle] [\langle \text{wyrażenie-sekwencji} \rangle] \langle \text{sygnatura} \rangle$$

Każdy z elementów składni jest elementem złożonym, podlegającym dalszemu specyfikowaniu i uszczegóławianiu. Opisywaną sytuację można przedstawić jak poniżej:



I tak opcjonalny fragment formuły składniowej wyrażony jako poprzednik (ang. *predecessor*) przedstawia się następująco:

$$\langle \text{poprzednik} \rangle ::= [\langle \text{numer-sekwencji} \rangle \text{ "/}]$$

Numer sekwencji odpowiada numerowi innego komunikatu i wskazuje, że wysłanie komunikatów „poprzedników” warunkuje wysłanie komunikatu, w którego specyfikacji wskazano poprzedników. W przypadku występowania wielu komunikatów obligatoryjnie realizowanych przed tym komunikatem, są one oddzielane przecinkami;

np. 2 /

1.1.3 /

2.3b, 1.1.3 /

Wyrażenie sekwencji (ang. *sequence expression*) jest również opcjonalnym fragmentem formuły składniowej komunikatu.

$$\langle \text{wyrażenie-sekwencji} \rangle ::= [\langle \text{numer-komunikatu} \rangle \mid \langle \text{nazwa} \rangle] [\text{":"} \text{ " " } \langle \text{rekurencja} \rangle]$$

Numerowanie komunikatów może odbywać się z wykorzystaniem jednego ze sposobów opisanych w rozdziale 8. W szczególnych przypadkach ściśle zdefiniowany numer komunikatu można zastąpić alfanumeryczną nazwą. Zastosowany sposób nazewnictwa musi jednak jednoznacznie wskazywać na kolejność wykonywania komunikatów na diagramie. **Rekurencja** (ang. *recurrence*) może zostać wyrażona jako:

$$\langle \text{rekurencja} \rangle ::= [\langle \text{warunek} \rangle \mid \langle \text{iteracja} \rangle]$$

Warunek przyjmuje przy tym postać:

$$\langle \text{warunek} \rangle ::= \text{"["} \text{ " " } \langle \text{specyfikacja-warunku} \rangle \text{ "]"}$$

np. [wartośćTransakcji > 10000]

[25 < ilość <= 75]

[wzrost > 150, waga >= 60]

[brak autoryzacji]

Iteracja jest natomiast definiowana następująco:

`<iteracja> ::= "*" | "*" [ "<specyfikacja-iteracji>" ]`

np. *

* [i := i..n]

W komunikacie mianem **sygnatury** (ang. *message signature*) określa się wyrażenie:

`<sygnatura> ::= [ "<atrybut>" ] "<nazwa-operacji>" [ "(" "<lista-argumentów>" ")" ] [ ":" "<wartość-zwrotna>" ]`

Opcjonalny **atrybut** jest atrybutem konkretnej klasy, do której wysła się komunikat.

np. nazwa =

nrApartamentu =

Nazwa operacji wskazuje na konkretną operację, która zostanie wykonana przez odbiorcę komunikatu, będącego właścicielem operacji. W sygnaturze nazwa jest elementem obligatoryjnym. Jest zarazem jedynym elementem, którego nie pomija się w składni komunikatu niezależnie od poziomu szczegółowości, na jakim tworzony jest dany diagram interakcji;

np. zwolnijRezerwację

obliczWartość

Lista argumentów specyfikuje opcjonalne parametry dla operacji. W przypadku zapisania wielu argumentów oddziela się je przecinkami. Dodatkowo przed pełną specyfikacją parametru można umieścić **przedrostek** wskazujący kierunek przekazywania parametru. Przedrostek informuje, czy odbiorca komunikatu otrzymuje parametr (przedrostek *in*), czy też parametr jest przekazywany przez odbiorcę na zewnątrz (przedrostek *out*). Niewyspecyfikowanie przedrostka jest równoznaczne z określeniem go jako *in*. Parametry szczegółowo scharakteryzowano w rozdziale 3.

np. (milimetry)

(netto, brutto)

(out_wzrost = 188)

(płeć = mężczyzna, waga = 90, wzrost = 188)

Opcjonalna **wartość zwrotna** wyrażona jest jako lista wartości, które w konsekwencji przesłania komunikatu przez instancję klasyfikatora, będącą nadawcą komunikatu, są jej przekazywane po wykonaniu przez odbiorcę operacji. Stosuje się ją tylko i wyłącznie w przypadku komunikatów zwrotnych. Wartość zwrotna może być wyspecyfikowana tylko w powiązaniu z atrybutem.

np. : 3,14159

: bilans.txt

: zarejestrowany

Podstawowe pojęcia

Argument	Przedrostek
Atrybut	Przypadek użycia
Diagram interakcji	Rekurencja
Diagram harmonogramowania	Iteracja
Diagram komunikacji	Warunek
Diagram sekwencji	Składnia komunikatu
Diagram sterowania interakcją	Poprzednik
Interakcja	Wyrażenie sekwencji
Izomorfizm	Sygnatura
Klasyfikator	Wartość zwrotna
Komunikat	Współdziałanie
Operacja	

Pytania i zadania

1. Wyjaśnij związek między diagramami przypadków użycia a diagramami interakcji.
2. Jak należy rozumieć interakcję? Zinterpretuj ją na przykładzie systemu obsługi poczty elektronicznej.
3. Wymień rodzaje diagramów interakcji.
4. Scharakteryzuj specyfikę poszczególnych diagramów interakcji.
5. Które z wymienionych diagramów interakcji nie znajdowały się w specyfikacji poprzednich wersji języka UML?
6. Jak należy interpretować pojęcie komunikatu w aspekcie interakcji?
7. Jaka jest składnia komunikatu w UML 2.0?
8. Zdefiniuj pojęcie poprzednika. Podaj kilka przykładów praktycznych.
9. Która część specyfikacji komunikatu nazywa się jego sygnaturą? Jakie elementy zawiera?

10. Zinterpretuj następujące komunikaty, wskazując odpowiednie nazwy elementów składni komunikatu:

- ♦ `z = wyliczPoziomZamowienia(artykuł, prognoza)`
- ♦ `wyświetl("Dostępny")`
- ♦ `wyświetl(a = 142)`
- ♦ `[t > x] zapisz`
- ♦ `zapisz`
- ♦ `pobierz(out_nazwa)`

Rozdział 7.

Diagram sekwencji

Znaczenie diagramów sekwencji

Diagramy sekwencji są najczęściej stosowanymi w praktyce spośród wszystkich diagramów interakcji, co można przypisać ich precyzji w opisywaniu i porządkowaniu złożonych **interakcji**. Diagramy te są ściśle powiązane ze scenariuszami przypadków użycia, dokumentując ich funkcjonalność.



Diagram sekwencji jest rodzajem diagramu interakcji, opisującym interakcje pomiędzy instancjami klasyfikatorów systemu w postaci sekwencji komunikatów wymienianych między nimi.

Właściwości diagramów sekwencji doskonale przedstawiają koncepcję obiektowego modelowania systemów informatycznych i pozwalają na bezpośrednie przejście do generowania kodu źródłowego w językach obiektowych, takich jak JAVA, VB .NET czy też C++. Interakcja na diagramie sekwencji opisywana jest w dwóch wymiarach:

- ♦ poziomym — jako oś, na której umieszczono instancje klasyfikatorów biorące udział w interakcji;
- ♦ pionowym — jako oś czasu przedstawiająca ułożone chronologicznie komunikaty.

Wymiar poziomy ma charakter statyczny, natomiast pionowy — dynamiczny. Diagramy sekwencji stwarzają zatem możliwość dokumentowania interakcji w postaci komunikatów wysyłanych oraz odbieranych przez instancje klasyfikatorów systemu. Obrazują one przepływ sterowania w systemie. Oznacza to wyznaczenie kolejności i miejsca wykonywania poszczególnych operacji.

Poza podstawowymi kategoriami pojęciowymi — klasyfikatorem, linią życia, komunikatem oraz ośrodkiem sterowania — diagramy te wzbogacone zostały o cały szereg innych zaawansowanych, omawianych w niniejszym rozdziale pojęć i związanych z nimi graficznych oznaczeń. Do pojęć tych należą różne rodzaje komunikatów, warunki,

tworzenie i niszczenie obiektów, samowywoływanie, rozgałęzienie, iteracja, fragment wyodrębniony, brama. Stwarzają one projektantowi systemu możliwość opracowania bardzo precyzyjnych specyfikacji dynamiki systemu. Szczególną rolę w tym zakresie pełnią operatory interakcji wprowadzone w najnowszej wersji języka UML — 2.0.

Podstawowe kategorie pojęciowe oraz notacja graficzna

Rodzaje diagramów sekwencji

Zakres oczekiwań twórców i użytkowników systemu w zakresie szczegółowości opisu interakcji na diagramie sekwencji jest zróżnicowany. Zależy on również od konkretnej fazy cyklu systemu oraz jej zaawansowania, co determinuje potrzeby w zakresie szczegółowości modelu dynamiki systemu. I tak można wyróżnić trzy **rodzaje diagramów sekwencji**, w zależności od przyjętego stopnia abstrakcji. Są to diagramy:

- ♦ konceptualny,
- ♦ implementacyjny,
- ♦ wystąpieniowy.

W celu opracowania **konceptualnego diagramu sekwencji** (ang. *conceptual sequence diagram*) twórca systemu korzysta z podstawowych kategorii pojęciowych i graficznych. Ten typ diagramu stwarza menedżerom i użytkownikom możliwość szybkiego, ogólnego naszkicowania zakresu i zawartości interakcji dla danej aplikacji. W ten sposób powstaje diagram o ogólnym zakresie i łatwych do zidentyfikowania interakcjach. Konceptualny diagram sekwencji nie został w sposób ścisły zdefiniowany w dokumentacji języka UML 2.0.

Konceptualny diagram sekwencji, choć jest dobrym punktem wyjścia i platformą porozumienia z użytkownikami, nie stanowi wystarczającej podstawy opracowania specyfikacji programistycznej. Rolę tę pełni **implementacyjny diagram sekwencji** (ang. *generic sequence diagram*), którego tworzenie inicjowane jest poprzez wykorzystanie diagramu na poziomie konceptualnym. Diagram ten prezentuje o wiele wyższy poziom precyzji oraz większy semantyczny zakres interakcji, choć dotyczy tej samej dziedziny przedmiotowej. Dokonywane jest to dzięki wykorzystaniu wszystkich kategorii pojęciowych dostępnych w diagramach sekwencji. Implementacyjny diagram sekwencji, oprócz głównego przepływu scenariusza przypadku użycia z odpowiedniego diagramu przypadków użycia, obejmuje także wszystkie przepływy alternatywne tego scenariusza. Diagramy na poziomie implementacyjnym są rutynowymi diagramami sekwencji. Przekazuje się je programistom jako dokumentację projektową lub na ich podstawie automatycznie generuje kod z wykorzystaniem narzędzi CASE.

Diagramy konceptualne i implementacyjne nie wyczerpują wszystkich możliwości prezentowania sekwencji w systemie. Na pełnym, implementacyjnym diagramie sekwencji występuje wiele rozwiązań o charakterze opcjonalnym bądź alternatywnym, których wykonanie jest uzależnione od zdefiniowanych na diagramie warunków. W zależności od ich spełnienia wykonywane jest określone wystąpienie diagramu spośród wszystkich kombinacji głównych i alternatywnych przepływów scenariusza danego przypadku użycia. Wystąpienie diagramu sekwencji opracowane w odniesieniu do ustalonego scenariusza określa się mianem **wystąpieniowego diagramu sekwencji** (ang. *instance sequence diagram*). Danemu implementacyjnemu diagramowi sekwencji zazwyczaj odpowiada wiele wystąpieniowych diagramów sekwencji.

Zależności pomiędzy implementacyjnym a wystąpieniowym diagramem sekwencji można prześledzić, analizując opis przypadku użycia „Anuluj rezerwację” (tabela 2.3). I tak implementacyjny diagram sekwencji opisuje zarówno główny, jak i alternatywne przepływy zdarzeń. Z diagramu implementacyjnego opracowanego na podstawie wspomnianej wyżej tabeli wynikają cztery wystąpieniowe diagramy sekwencji (a, b, c, d). Charakteryzują się one następującą kolejnością przepływów:

- a) 1, 2, ..., 6;
- b) 1, 2a;
- c) 1, 2, 3a;
- d) 1, 2, 3, 3b, 4, 5, 6.

Charakterystykę opisanych rodzajów diagramów sekwencji przedstawia tabela 7.1.

Tabela 7.1. Kategorie modelowania a poziomy diagramów sekwencji

Kryterium	Rama diagramu	Kategorie pojęciowe	Przykładowe rysunki
Konceptualny diagram sekwencji	opcjonalna	podstawowe	7.3, 7.5, 7.6
Implementacyjny diagram sekwencji	wymagana	wszystkie podstawowe i zaawansowane	7.21 – 7.33, 7.38
Wystąpieniowy diagram sekwencji	wymagana	wybrane	7.39

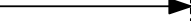
Klasyfikator, komunikat i linia życia

Podstawowymi elementami diagramu sekwencji są:

- ♦ klasyfikator,
- ♦ komunikat,
- ♦ linia życia,
- ♦ ośrodek sterowania.

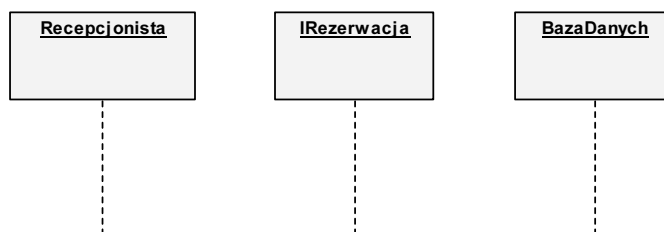
Występują one niezależnie od poziomu abstrakcji, na jakim diagram sekwencji jest tworzony — na poziomie konceptualnym, implementacyjnym czy też wystąpieniowym. Co więcej — w przyjaznej menedżerom, najbardziej ogólnej, strategicznej wersji, konceptualne diagramy sekwencji mogą opierać się wyłącznie na trzech pierwszych z wymienionych pojęć. Bliższą charakterystykę tych kategorii modelowania przedstawia tabela 7.2. Podane w niej definicje są autorskimi modyfikacjami definicji proponowanych przez organizację OMG.

Tabela 7.2. Podstawowe kategorie modelowania diagramów sekwencji

Nazwa pojęcia	Notacja graficzna	Definicja
Obiekt (jako instancja klasyfikatora)		 Klasyfikator to abstrakcyjna kategoria modelowania systemu w języku UML, która uogólnia kolekcję instancji o tych samych cechach.
Linia życia		 Linia życia to powiązana z konkretną instancją klasyfikatora linia na diagramie sekwencji, wskazująca okres istnienia tej instancji.
Komunikat		 Komunikat to specyfikacja wymiany informacji między obiektami, zawierająca zlecenia wykonania określonej operacji.

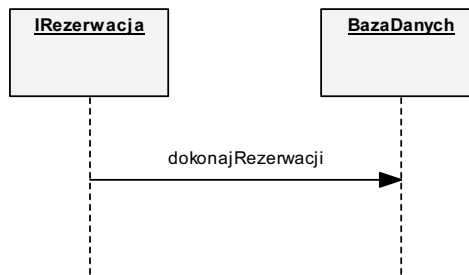
Najczęściej w ramach diagramów sekwencji ilustrowane są instancje **klasyfikatorów** w postaci **obiektów**. Instancje klasyfikatorów biorące udział w interakcji należy rozmieszczać na linii poziomej w kolejności ich występowania. Z każdej instancji klasyfikatora wyprowadza się przerywaną linię nazywaną **linią życia** (ang. *lifeline*), reprezentującą czas życia instancji. Wystąpienie na diagramie sekwencji instancji klasyfikatora jest równoznaczne z wystąpieniem jej linii życia. Obiekty oraz związane z nimi linie życia przedstawia rysunek 7.1.

Rysunek 7.1.
Instancje
klasyfikatorów
z liniami życia

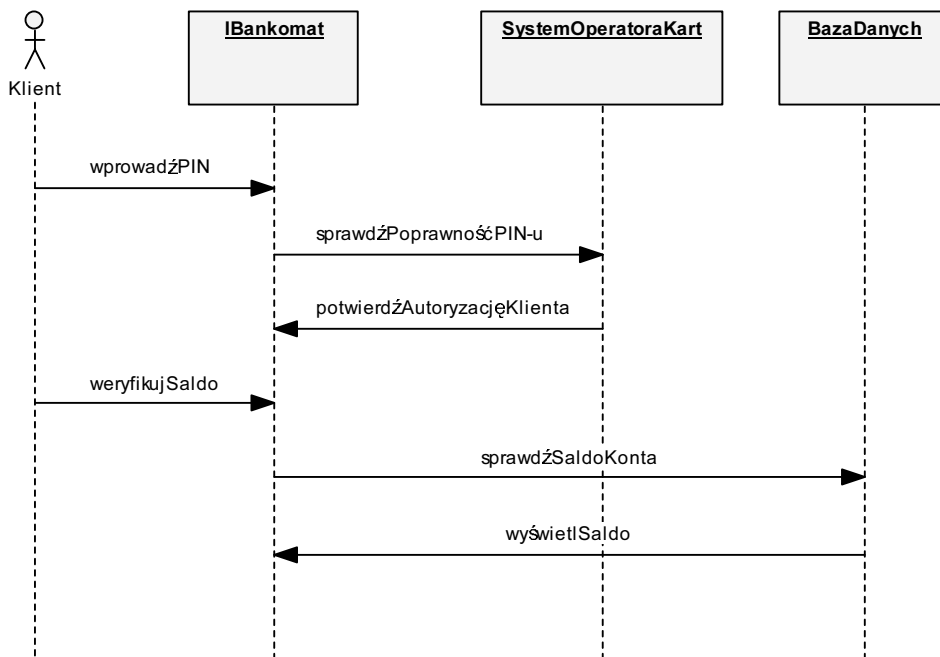


Kolejną podstawową kategorię modelowania, **komunikaty**, rozmieszcza się na osi pionowej diagramu, pomiędzy liniami życia instancji klasyfikatorów (rysunek 7.2).

Rysunek 7.2.
Przesłanie komunikatu



Komunikaty porządkowane są według kolejności ich występowania. Im później następuje wysłanie komunikatu, tym niżej umieszcza się go na diagramie (rysunek 7.3). Istnieje opcja ich numerowania, choć na diagramach sekwencji nie jest to wymagane.



Rysunek 7.3. *Sprawdzanie stanu konta za pośrednictwem bankomatu*

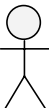
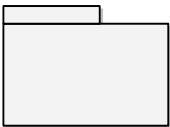
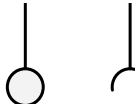
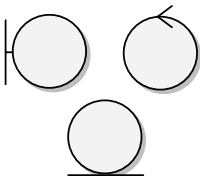
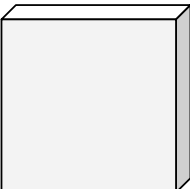
Nie występuje konieczność wysyłania komunikatów wyłącznie do instancji klasyfikatorów sąsiadujących na diagramie z nadawcą. Z linii życia danej instancji klasyfikatora można przesłać komunikaty do wszystkich innych instancji klasyfikatorów występujących na danym diagramie sekwencji.

Rodzaje klasyfikatorów

Jak zaznaczono, diagramy sekwencji zazwyczaj przedstawiają sposób wzajemnej współpracy obiektów. Diagramy te mogą jednak — z wyjątkiem wystąpieniowych diagramów sekwencji — opisywać interakcję pomiędzy instancjami różnych rodzajów

klasyfikatorów stosowanych we wszystkich innych rodzajach diagramów języka UML. Podstawowe rodzaje klasyfikatorów przedstawianych w ramach diagramów sekwencji zawiera tabela 7.3. Twórcy systemu mogą poszerzać spektrum klasyfikatorów poprzez stereotypy — propozycje własnych oznaczeń graficznych i pojęciowych.

Tabela 7.3. Wybrane rodzaje klasyfikatorów w języku UML

Rodzaj	Notacja graficzna	Rodzaj	Notacja graficzna
Aktor		Pakiet	
Przypadek użycia		Współdziałanie	
Klasa		Interfejs	
Klasa analityczna		Węzeł	
Artefakt		Komponent	
Sygnał			
			

Ośrodek sterowania

Linia życia reprezentuje okres życia instancji klasyfikatora. W niektórych okresach swego życia instancja ta jest w stanie czuwania, podczas gdy w innych jest aktywowana przez komunikaty, przejmując sterowanie interakcją i podejmując zlecone działania. Po wykonaniu operacji zleconych komunikatem i wysłaniu odpowiednich komunikatów wynikowych instancja klasyfikatora może ponownie przechodzić w stan czuwania. Sytuacje te powodują, że w odniesieniu do linii życia zaproponowano nową kategorię pojęciową, tj. **ośrodek sterowania** (ang. *execution specification*).

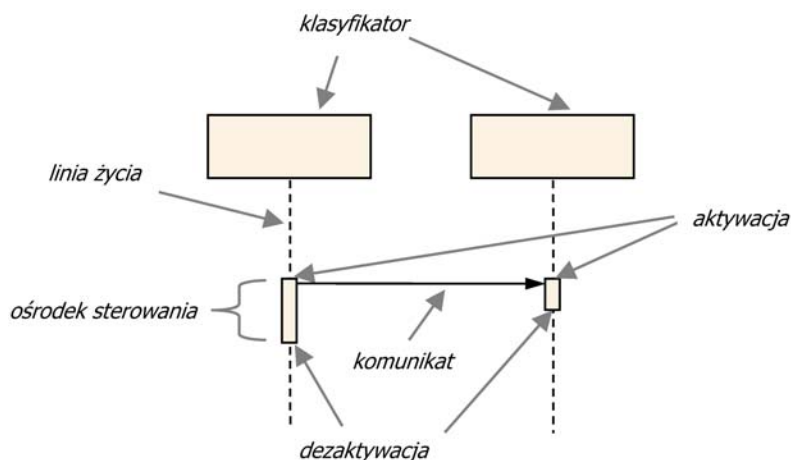


Ośrodek sterowania jest specyfikacją wykonywania czynności, operacji lub innej jednostki zachowania w ramach interakcji.

W praktyce realizacja funkcjonalności instancji klasyfikatora — wskazania na linii życia z wykorzystaniem ośrodka sterowania — oznacza wykonywanie operacji przetwarzania, obliczania, komunikowania się z innymi instancjami klasyfikatorów czy też wykonywania algorytmów złożonych z elementarnych akcji. Na diagramie ośrodek sterowania przyjmuje postać prostokąta umieszczonego na linii życia instancji klasyfikatora (rysunek 7.4).

Rysunek 7.4.

Kategorie pojęciowe diagramów sekwencji



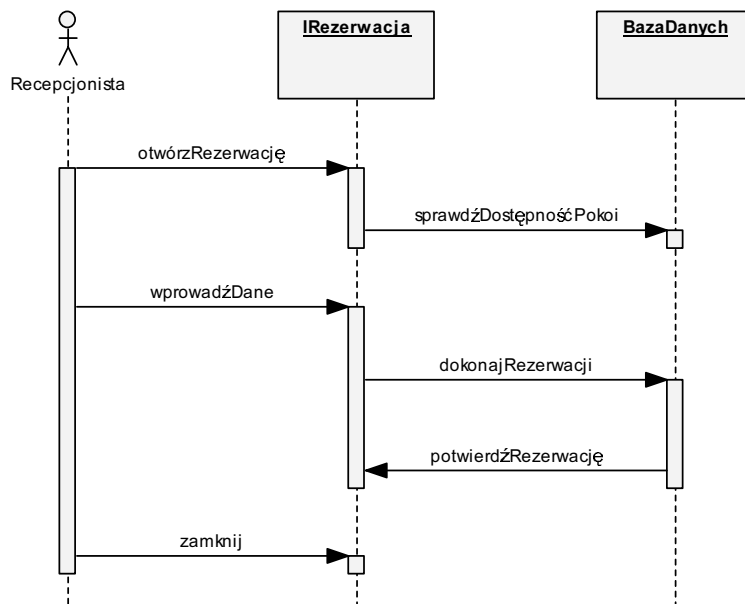
Ośrodek sterowania jest inicjowany **aktywacją**, a przejście w stan czuwania **dezaktywacją**. Zastosowanie ośrodków sterowania podczas modelowania procesu komunikowania się recepcjonisty z systemem rezerwacji hotelowej przedstawia rysunek 7.5.

Graficzna reprezentacja ośrodka sterowania na diagramie sekwencji nie musi mieć charakteru ciągłego. Na konkretnej linii życia związanej z daną instancją klasyfikatora może wystąpić kilka **niezależnych ośrodków sterowania**. Oznacza to, że instancja ta jest aktywowana wielokrotnie w celu realizacji komunikatów wysyłanych przez inne instancje klasyfikatorów. W ten sposób instancja kilkakrotnie przechodzi w stany aktywne, jak zaprezentowano na rysunku 7.6, który przedstawia interakcję inwestora oraz systemu obsługi funduszu inwestycyjnego. Rysunek ten uwzględnia ponadto opcje w postaci możliwości numerowania kolejnych komunikatów oraz umieszczania na diagramie sekwencji parametrów operacji przy komunikatach.

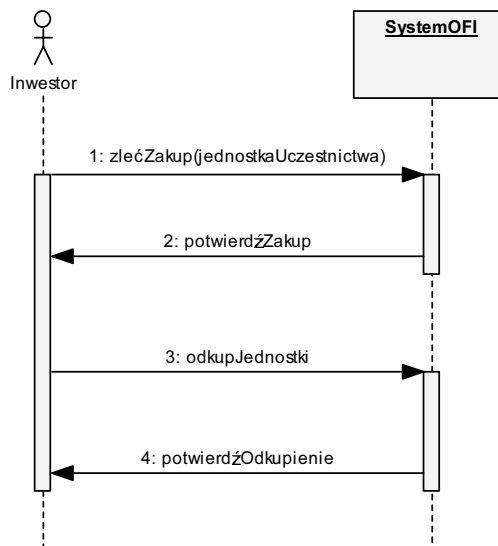
Zaawansowane składniki diagramu

Współczesne aplikacje informatyczne, zwłaszcza pracujące w czasie rzeczywistym, są bardzo złożone. Opisane w scenariuszach przypadków użycia interakcje aktor — system stanowią podstawę specyfikacji interakcji wewnątrz systemu informatycznego.

Rysunek 7.5.
Rezerwacja hotelowa



Rysunek 7.6.
Okres aktywności
instancji
klasyfikatorów



Przedstawione dotychczas kategorie modelowania diagramów sekwencji, tj. klasyfikator, komunikat, linia życia i ośrodek sterowania, nie są wystarczające do udokumentowania złożoności interakcji. Nie pozwalają na pełne opracowanie implementacyjnych diagramów sekwencji, będących podstawą generowania kodu systemów informatycznych. Dlatego diagramy sekwencji **wzbogacono** o szereg innych elementów, takich jak:

- ♦ rodzaje komunikatów,
- ♦ tworzenie i niszczenie obiektów,
- ♦ warunki,
- ♦ samowywoływanie,
- ♦ iteracja,
- ♦ rozgałęzienie,
- ♦ fragmenty wyodrębnione i operatory interakcji,
- ♦ przywoływane wystąpienia interakcji,
- ♦ bramy.

Wystąpieniowe diagramy sekwencji mogą korzystać z niektórych wymienionych kategorii pojęciowych, niemniej specyfika wystąpienia zawęża ich zakres.

Rodzaje komunikatów

W istocie komunikaty różnią się nie tylko treścią semantyczną, lecz także funkcjonalnością. Treść semantyczna, czyli polecenie wydane przez nadawcę, wyrażona jest pełną składnią komunikatu zdefiniowaną w rozdziale 6. Z kolei funkcjonalność instancji klasyfikatorów określa się dzięki wyborowi odpowiedniego rodzaju komunikatu. Specyfika tych rodzajów wyrażana jest graficznie poprzez wprowadzenie odmiennych oznaczeń. Wyróżnia się następujące **rodzaje komunikatów**:

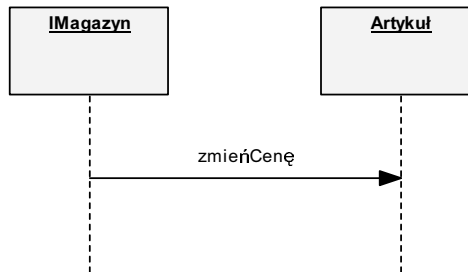
- ♦ synchroniczny,
- ♦ asynchroniczny,
- ♦ zwrotny,
- ♦ utracony,
- ♦ znaleziony,
- ♦ opcjonalny,
- ♦ oczekujący.

W rutynowej interakcji oba biorące w niej udział instancje klasyfikatorów — **klasyfikator-nadawca** oraz **klasyfikator-odbiorca** — są znane i ściśle zdefiniowane. Komunikaty biorące udział w takiej interakcji traktuje się jako kompletne. Należą do nich komunikaty synchroniczne, asynchroniczne, zwrotne, opcjonalne oraz oczekujące. W praktyce może zaistnieć także sytuacja, w której jedna z instancji, nadawca bądź odbiorca, jest nieznaną. Komunikaty niekompletne charakteryzujące opisaną sytuację nazywa się odpowiednio komunikatem znalezionym oraz komunikatem utraconym. Dzięki pojęciu stereotypu zestaw dostępnych rodzajów komunikatów jest rozszerzany przez różnych autorów i zespoły praktyków, użytkujących język UML.

Na zamieszczonych dotąd diagramach sekwencji przedstawiano jeden, w istocie dominujący, rodzaj komunikatu — komunikat synchroniczny. Wysłanie **komunikatu synchronicznego** (ang. *synchronous message*) oznacza przekazanie sterowania do klasyfikatora-odbiorcy. W momencie przesłania tego komunikatu aktualny przepływ sterowania klasyfikatora-nadawcy ulega przerwaniu — jest on wznowiany dopiero po wykonaniu przez klasyfikator-odbiorcę operacji inicjowanej przez ten komunikat (rysunek 7.7). Odbiorca aktywuje operację wskazaną w komunikacie, a zatem powinien występować merytoryczny związek pomiędzy nazwą komunikatu a nazwą operacji aktywowanej w klasyfikatorze-odbiorcy. Nazwy te, zgodnie z zasadą polimorfizmu obiektowości, mimo wskazywania na tę samą operację nie muszą być identyczne. Kwestia zgodności nazewnictwa w przypadku wykorzystania pakietów CASE do tworzenia diagramów sekwencji jest znacznie uproszczona. Nazwy nadawane komunikatom automatycznie przenoszone są jako nazwy operacji do odpowiednich klas.

Rysunek 7.7.

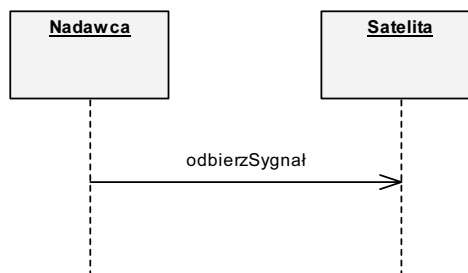
Komunikat synchroniczny



W odróżnieniu od scharakteryzowanych wyżej komunikatów synchronicznych, **komunikaty asynchroniczne** (ang. *asynchronous messages*) nie powodują przerwania aktualnego przepływu sterowania nadawcy. Instancja klasyfikatora wysyłająca komunikat nie oczekuje odpowiedzi, zarazem pozostając w stanie aktywności, co umożliwia jej dalsze przetwarzanie bądź wysyłanie komunikatów (rysunek 7.8).

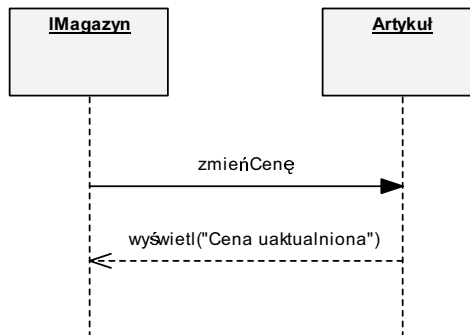
Rysunek 7.8.

Komunikat asynchroniczny



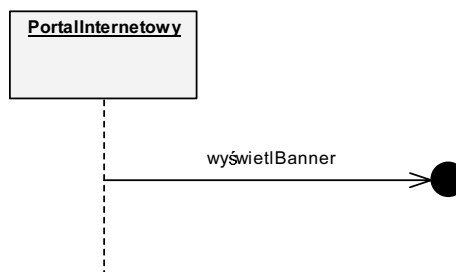
Komunikaty zwrotne (ang. *return messages*) wskazują na powrót sterowania do instancji klasyfikatora po wykonaniu komunikatu synchronicznego (rysunek 7.9). Komunikat ten nie jest umieszczany na diagramach sekwencji obligatoryjnie — projektant ma daleko posuniętą swobodę decyzji we wprowadzaniu komunikatów zwrotnych do diagramów. Funkcją komunikatu zwrotnego jest nie tylko przekazanie sterowania do danej instancji, ale też zainicjowanie jej określonej operacji.

Rysunek 7.9.
Komunikat zwrotny



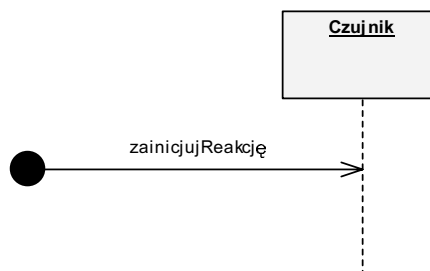
Komunikat określa się mianem **utraconego** (ang. *lost message*) w sytuacji, gdy klasyfikator-nadawca jest znany, natomiast odbiorca nieznany (rysunek 7.10). Komunikat utracony ma znaczenie w modelowaniu złożonych, składających się z wielu fragmentów interakcji. Na tym etapie tworzenia systemu jego projektant nie jest w stanie dokładnie sprecyzować odbiorcy komunikatu przesyłanego pomiędzy fragmentami interakcji. W dalszych iteracjach procesu tworzenia systemu, po integracji poszczególnych fragmentów, klasyfikator-odbiorca zostaje jednoznacznie zidentyfikowany.

Rysunek 7.10.
Komunikat utracony



Jeżeli odbiorca danego komunikatu jest znany w obrębie danego fragmentu interakcji, natomiast nadawca jest nieznany, komunikat taki nazywany jest **komunikatem znalezionym** (ang. *found message*). Komunikat znaleziony zamieszczono na rysunku 7.11.

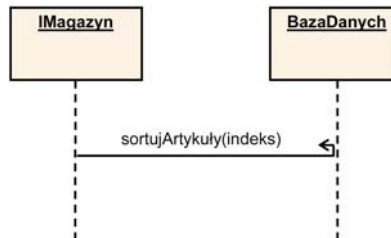
Rysunek 7.11.
Komunikat znaleziony



Komunikat znaleziony jest wysyłany spoza danego diagramu sekwencji. Impulsem do nadania opisywanego komunikatu może być przykładowo dym, ogień, hałas czy ruch, a zatem zjawiska, które nie poddają się opisowi w postaci standardowych elementów modelowania diagramów sekwencji języka UML.

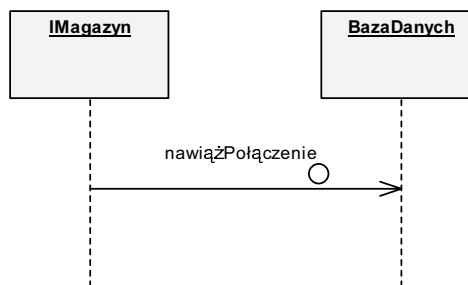
Komunikat opcjonalny (ang. *balking message*) oznacza, że nadawca wysyła do odbiorcy komunikat, oczekując gotowości do jego niezwłocznej obsługi przez odbiorcę. Jeżeli komunikat nie może zostać przyjęty bezpośrednio po wysłaniu, nadawca nie podejmuje dalszych prób jego przesyłania (rysunek 7.12). Tym samym komunikat opcjonalny nie w każdym przypadku jest obsługiwany.

Rysunek 7.12.
Komunikat opcjonalny



Komunikaty oczekujące (ang. *timeout messages*) wykazują duży stopień podobieństwa do komunikatów opcjonalnych. Analogicznie, nadawca wysyła komunikat do odbiorcy — z tym, że gotów jest czekać na jego obsłużenie przez określony odcinek czasu. Jeżeli odbiorca nie jest w stanie przyjąć komunikatu w wyspecyfikowanym czasie, nadawca rezygnuje z danej interakcji (rysunek 7.13).

Rysunek 7.13.
Komunikat oczekujący



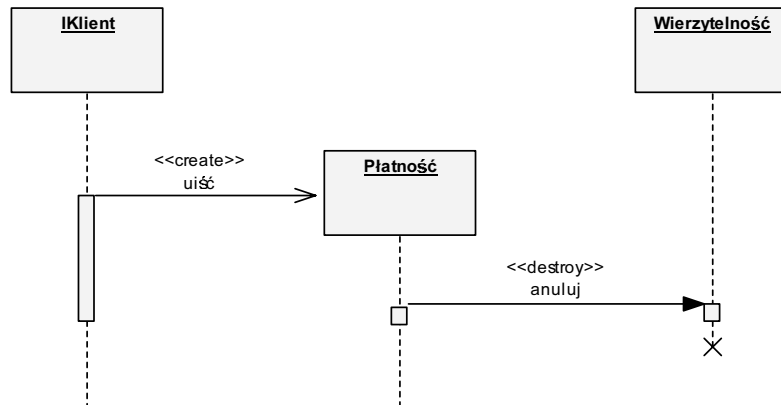
Komunikaty opcjonalny oraz oczekujący nie są elementami standardu UML 2.0, lecz stereotypami graficznymi zaproponowanymi przez korporację IBM Rational.

Tworzenie i niszczenie obiektów

Poszczególne instancje klasyfikatorów mogą za pośrednictwem operacji zarówno **tworzyć**, jak i **niszczyć** obiekty. Obiekt zostaje utworzony w wyniku przesłania komunikatu stereotypowanego «*create*». Obiekt ten umieszczany jest na diagramie sekwencji poniżej pierwotnie istniejących instancji klasyfikatorów, tak aby jego położenie było zgodne z czasem utworzenia (rysunek 7.14). Po słowie kluczowym «*create*» powinna być umieszczona nazwa operacji tworzącej obiekt, choć nie jest to obligatoryjne.

Obiekt przestaje istnieć wraz z odebraniem przezeń komunikatu stereotypowanego «*destroy*». Na diagramie sekwencji fakt ten jest dodatkowo oznaczony zdarzeniem niszczącym (ang. *destruction event*), graficznie reprezentowanym w postaci wielkiej litery **X**. Również w tym przypadku po słowie kluczowym opcjonalne jest podanie nazwy operacji.

Rysunek 7.14.
Tworzenie
i niszczenie obiektów



Warunki

Wykonanie komunikatu można uzależnić od spełnienia określonych **warunków** (ang. *guard conditions*).

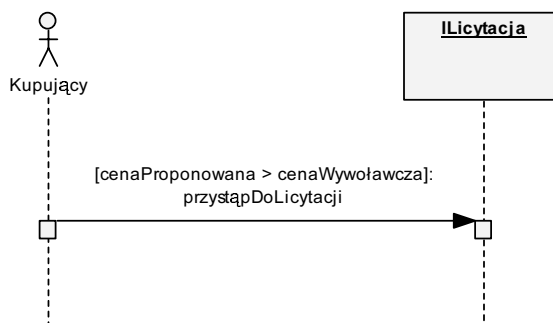


Warunek jest to związane z komunikatem kryterium, od którego spełnienia uzależnione jest wykonanie określonej operacji.

Konwencja zapisu warunków nie jest ściśle definiowana. W zależności od specyfiki projektu i rodzaju stosowanego diagramu sekwencji — konceptualnego, implementacyjnego lub wystąpieniowego — formalny zapis warunku może być różny. Najczęściej spotykane są warunki wyrażone w sposób tekstowy oraz z wykorzystaniem pseudo-kodu. Jeśli warunek dotyczący danego komunikatu nie zostanie spełniony, operacja wskazywana przez komunikat nie jest wykonywana.

Warunki umieszczane są w nawiasach kwadratowych przed nazwą komunikatu. Uwarunkowany komunikat zaprezentowano na rysunku 7.15, gdzie kupujący przystępuje do licytacji tylko w przypadku, gdy oferowana przez niego cena jest wyższa od ceny wywoławczej.

Rysunek 7.15.
Warunek realizacji
komunikatu



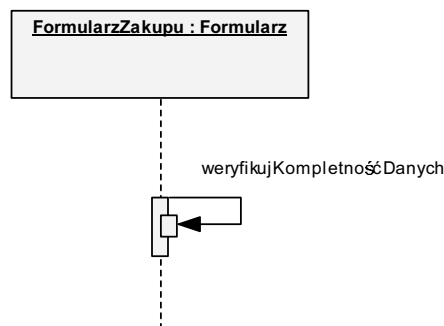
Na ogół z jednym komunikatem wiąże się tylko jeden warunek. Jednak jest również możliwe uzależnienie realizacji jednego komunikatu przez daną instancję klasyfikatora od spełnienia kilku warunków. Pomiędzy dwoma ośrodkami sterowania można umieścić dowolną liczbę uwarunkowanych komunikatów.

Warunki określane są także mianem wyrażień dozoru.

Samowywołanie

Kolejną kategorią pojęciową stosowaną w diagramach sekwencji jest **samowywołanie** (ang. *messages to self*). Oznacza ono sytuację, w której dana instancja klasyfikatora wywołuje własną operację (rysunek 7.16). Samowywołanie należy traktować jako szczególny przypadek iteracji.

Rysunek 7.16.
Samowywołanie



W efekcie samowywoływania na linii życia danej instancji klasyfikatora pojawia się zagnieżdżony ośrodek sterowania. Pojęcie zagnieżdżenia jest ściśle związane z kolejnością wzajemnie uzależnionego wywoływania się komunikatów i jest szczegółowo wyjaśnione w rozdziale 8.

Iteracja

W zastosowaniach biznesowych występuje wiele interakcji, w których ten sam komunikat wykonywany jest wielokrotnie, na zasadzie **iteracji**.

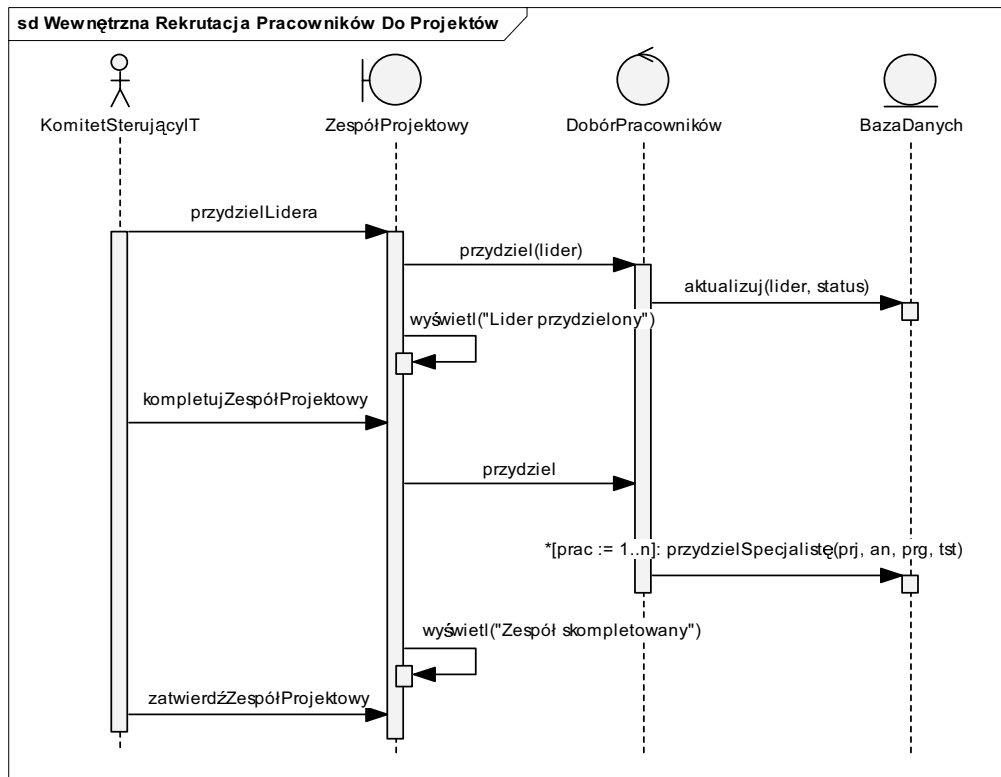


Iteracja to wielokrotne, policzalne powtórzenie jednostki zachowania w systemie.

Krotność wykonania iteracji określana jest w formie warunku poprzedzonego symbolem *. Podstawowa notacja iteracji jest następująca:

”*” | ”*” [” <specyfikacja-iteracji> ”]” <nazwa-operacji>

Iterację w klasycznym wydaniu przedstawia się jak na rysunku 7.17. Ilustruje on przydziałanie *Pracowników* zespołu projektowego do konkretnych projektów.



Rysunek 7.17. Iteracja w wewnętrznej rekrutacji pracowników do projektów

Firma programistyczna wykonuje równolegle wiele projektów. Obsługa kadrowa projektów wspomagana jest komputerowo. Odpowiedni system pozwala na adekwatne dopasowanie zawodowych kompetencji pracowników firmy do specyfiki potrzeb konkretnego projektu. Do realizacji każdego projektu powoływany jest *ZespółProjektowy*. W tym celu komitet sterujący wyznacza lidera zespołu poprzez wysłanie komunikatu *przydzielLidera*. Analityczny obiekt graniczny *ZespółProjektowy* przekazuje komunikat *przydziel* z identyfikatorem lidera do analitycznego obiektu sterującego *DobórPracowników*. W momencie przydzielenia do konkretnego projektu status lidera zostaje zmieniony w *BazieDanych*. Zmiana ta dokonywana jest za pośrednictwem komunikatu *aktualizuj* z parametrami *lider* oraz *status*. Po dokonaniu aktualizacji statusu interfejs *ZespółProjektowy* wyświetla potwierdzenie przydzielenia lidera. Kolejnym krokiem w powoływaniu nowego zespołu jest przydzielenie do projektu poszczególnych pracowników bezpośrednio zaangażowanych w proces tworzenia systemu informatycznego. Pojedynczy projekt jest realizowany przez ustaloną grupę specjalistów, w tym zazwyczaj kilku projektantów systemu, analityków systemowych, programistów oraz testujących. Poszczególne specjalizacje członków zespołu reprezentowane są przez parametry *prj*, *an*, *prg*, *tst* komunikatu *przydzielSpecjalistę*. Skompletowanie zespołu zakończone jest ponownym wyświetleniem potwierdzenia. Po wykonaniu procedury doboru zespołu *KomitetSterującyIT* zatwierdza *ZespółProjektowy*.

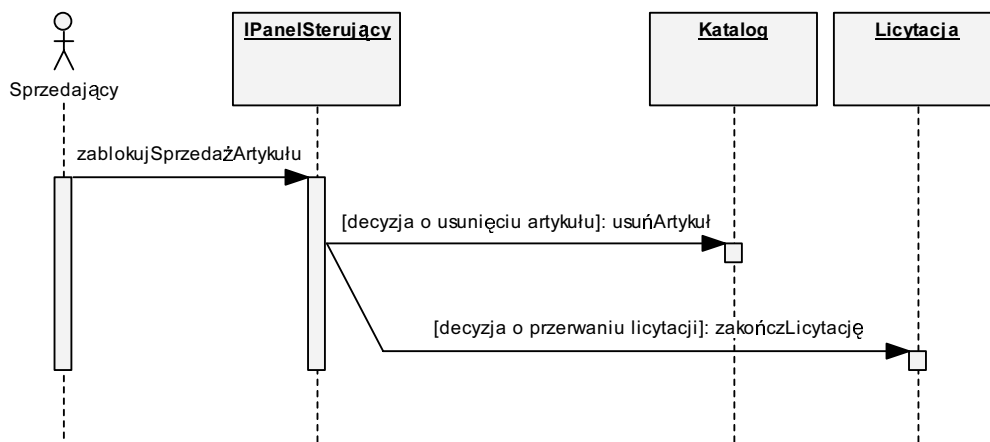
Począwszy od wersji 2.0 standardu UML, złożone interakcje można modelować z wykorzystaniem fragmentów wyodrębnionych, jak pokazano na rysunku 7.38.

Rozgałęzienie

Zweryfikowanie warunku może oznaczać wysłanie komunikatu skutkującego wykonaniem operacji odbiorcy, tak jak na rysunku 7.15. Jednak realizacja konkretnych operacji może opierać się na bloku decyzyjnym składającym się z przynajmniej dwóch wzajemnie wykluczających się warunków. W takim przypadku realizacja wykluczających się warunków na diagramie sekwencji oznacza konieczność przesyłania komunikatów, jak również przekazania sterowania do odpowiednich instancji klasyfikatorów. Funkcja obsługi tych warunków przybiera postać **rozgałęzienia** (ang. *branch*), jak na rysunku 7.18. A zatem:



Rozgałęzienie jest sposobem przekazania sterowania na diagramie sekwencji z linii życia (ośrodka sterowania) klasyfikatora-nadawcy do klasyfikatorów-odbiorców w zależności od spełnienia warunku przypisanego do przesyłanego komunikatu.



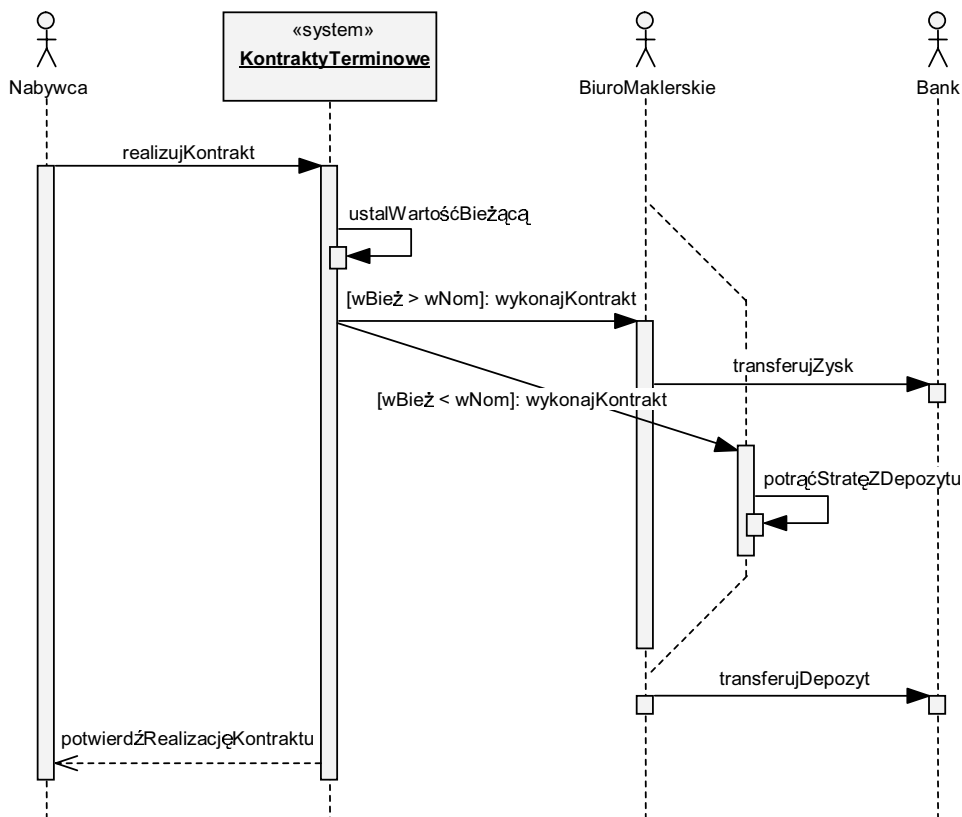
Rysunek 7.18. Rozgałęzienie

Rysunek 7.18 prezentuje wybrane interakcje dotyczące licytacji na aukcji internetowej, na którą *Sprzedający* może zgłosić szereg artykułów, umieszczając je w *Katalogu* aukcji. Bieżącą kontrolę nad przebiegiem aukcji umożliwia mu *IPanelSterujący*. W okresie między umieszczeniem artykułu w *Katalogu* a rozpoczęciem *Licytacji*, *Sprzedający* może być zainteresowany wycofaniem artykułu z aukcji. Skutkuje to wykreśleniem artykułu z *Katalogu* aukcji w wyniku przesłania komunikatu *usuńArtykuł*. Istnieje również drugi sposób zablokowania sprzedaży artykułu na *Licytacji*. Znajduje zastosowanie wtedy, gdy już w trakcie trwania *Licytacji* sprzedający zdecyduje się na wycofanie artykułu i przerwanie *Licytacji* bez podania przyczyn. Dokonuje się to w wyniku przesłania komunikatu *zakończLicytację*.

Na diagramach sekwencji występuje również inna odmiana rozgałęzienia. Przesłanie **alternatywnych komunikatów** wynikających ze spełnienia warunku może być także realizowane wyłącznie przez jedną instancję klasyfikatora. Wówczas konieczne jest rozszczępienie linii życia klasyfikatora-odbiorcy. Ponieważ komunikat jest wysyłany tylko do jednego odbiorcy, w momencie rozpoczęcia realizacji uwarunkowanych komunikatów rozszczępienie się linii życia klasyfikatora-odbiorcy, tworząc alternatywne linie życia (rysunek 7.19). Po wykonaniu stosownych operacji rozszczępienie linie życia mogą zostać ponownie połączone.

Zatem UML pozwala na wprowadzanie dwóch **odmian rozgałęzień**:

- ♦ klasyfikator-nadawca $\xrightarrow{[\text{warunki}]}$ kilka klasyfikatorów-odbiorców;
- ♦ klasyfikator-nadawca $\xrightarrow{[\text{warunki}]}$ jeden klasyfikator-odbiorca z rozszczępioną linią życia.



Rysunek 7.19. Realizacja kontraktu terminowego

Na rysunku 7.19 zaprezentowano drugi rodzaj rozgałęzienia na diagramach sekwencji. *Nabywca* decyduje się zrealizować uprzednio nabyty kontrakt terminowy. W tym celu wysła komunikat *realizujKontrakt* do systemu *KontraktyTerminowe*. *Nabywca* może spodziewać się zarówno zysków, jak i strat. System *ustalWartośćBieżącą* kontraktu. Następnie wartość ta jest porównywana z wartością nominalną równą cenie

zakupu. Po przeprowadzeniu porównania wysyłany jest komunikat *wykonajKontrakt*. Sposób wykonania kontraktu zależy od wyniku finansowego transakcji. W przypadku osiągnięcia zysków są one przelewane na konto *Nabywcy* kontraktu w wyniku wykonania komunikatu *transferujZysk*. Natomiast ewentualne straty są potrącane z jego depozytu w wyniku realizacji komunikatu *potrąćStratęZDepozytu*. W obu przypadkach następnym krokiem jest *transferDepozytu* (lub jego pozostałej części) przez *Biuro-Maklerskie* na konto *Nabywcy* w *Banku* prowadzącym rachunek bieżący *Nabywcy*. Transakcja kończy się *potwierdzeniemRealizacjiKontraktu*.

Notacja rozgałęzień traci obecnie na znaczeniu w związku z wprowadzeniem wraz z wersją UML 2.0 fragmentu wyodrębnionego z operatorem interakcji *alt*.

Fragmenty wyodrębnione i operatory interakcji

Złożoność diagramów sekwencji powoduje, że wyróżnia się w nich pewne specjalne obszary interakcji nazywane fragmentami interakcji. W praktyce przyjmują one postać:

- ♦ **fragmentów wyodrębnionych** (ang. *combined fragments*),
- ♦ przywoływanych wystąpień interakcji (ang. *interaction occurrences*).

Pod względem pojęciowym bardziej rozbudowana jest pierwsza z tych kategorii.



Fragment wyodrębniony jest to logicznie spójny obszar interakcji, część diagramu sekwencji charakteryzująca się specyficznymi właściwościami określonymi przez operator interakcji.

Stosowanie fragmentów wyodrębnionych umożliwia bardziej precyzyjne zobrazowanie istoty interakcji, co jest szczególnie ważne w przypadku systemów czasu rzeczywistego oraz wspomagających skomplikowane procesy biznesowe. Specyfika fragmentu wyodrębnionego jest ściśle uzależniona od charakteryzującego go **operatora interakcji** (ang. *interaction operator*).



Operator interakcji stanowi sprecyzowanie funkcjonalności realizowanej przez fragment wyodrębniony.

Fragment wyodrębniony występuje w formie obramowanej z wyszczególnieniem nagłówka. W nagłówku obligatoryjnie umieszczony jest operator charakteryzujący specyfikę danego fragmentu wyodrębnionego wraz z ewentualnymi parametrami, zapisany w następującej konwencji:

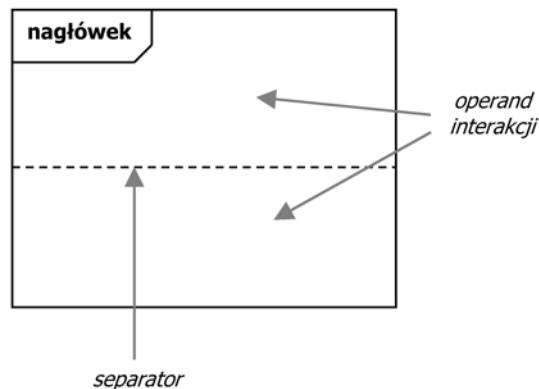
<operator-interakcji> [<parametry>]

Specyfikację fragmentu wyodrębnionego zamieszczono na rysunku 7.20.

Na diagramach sekwencji identyfikuje się fragmenty wyodrębnione, a następnie definiuje z wykorzystaniem następujących operatorów interakcji będących terminami angielskimi lub ich skrótami (por. tabela 7.3):

Rysunek 7.20.

Fragment
wyodrębniony
i jego elementy



- ♦ *alt* — alternatywa,
- ♦ *opt* — opcja,
- ♦ *break* — przerwanie,
- ♦ *loop* — iteracja,
- ♦ *neg* — funkcjonalność nieprawidłowa,
- ♦ *par* — współbieżność,
- ♦ *critical* — obszar krytyczny,
- ♦ *assert* — formuła,
- ♦ *consider* — istotność,
- ♦ *ignore* — nieistotność,
- ♦ *stricte* — ściśle uporządkowanie,
- ♦ *seq* — słabe uporządkowanie.

Fragment wyodrębniony składa się z jednego lub wielu **operandów interakcji** (ang. *interaction operands*).



Operand interakcji stanowi oddzielną część fragmentu wyodrębnionego; sub-fragment.

Reguły tworzenia i użytkowania operandów interakcji są zróżnicowane i uwarunkowane specyfiką konkretnego fragmentu wyodrębnionego, co zaznacza się odpowiednim operatorem interakcji w nagłówku fragmentu wyodrębnionego. I tak:

- ♦ we fragmentach wyodrębnionych oznaczonych operatorami *opt*, *loop* oraz *neg* występuje jeden i tylko jeden operand interakcji;
- ♦ dla fragmentów wyodrębnionych oznaczonych operatorem *alt* może wystąpić wiele operandów, jednak realizowany jest tylko jeden z nich, stosownie do wykonania warunku zapisanego w operandzie;

- ♦ jeśli fragment wyodrębniony oznaczony jest operatorem *par*, wszystkie składające się na niego operandy wykonywane są równolegle;
- ♦ we fragmentach wyodrębnionych oznaczonych pozostałymi operatorami poszczególne operandy interakcji wykonywane są sekwencyjnie.

Operandy interakcji umieszczane są w ramach fragmentu wyodrębnionego w układzie pionowym. Poszczególne operandy oddzielane są przez **separatory** w postaci linii przerywanej. Separatory naturalnie nie występują we fragmentach wyodrębnionych wykorzystujących operatory *opt*, *loop* oraz *neg*. W przypadku operatorów *alt* i *opt* w obszarze operandu należy zapisać **warunek**, który musi być spełniony, aby interakcja operandu została zrealizowana. Sposób specyfikowania warunków dla operandów zamieszczono na rysunku 7.21 oraz 7.22.

Tabela przedstawia **listę operatorów interakcji** (tabela 7.4). Każdy z nich jednoznacznie charakteryzuje specyficzne cechy opisywanych fragmentów wyodrębnionych. W kolejnych kolumnach tabeli zamieszczono:

- ♦ nazwę operatora interakcji,
- ♦ operator,
- ♦ interpretację istoty operatora interakcji,
- ♦ odwołania do rysunków ilustrujących fragmenty wyodrębnione dla adekwatnych operatorów.

Tabela 7.4. Operatory interakcji na diagramach sekwencji

Pełna nazwa operatora	Operator	Interpretacja	Rysunki
alternatywa	<i>alt</i>	Oznacza możliwość wyboru jednego i tylko jednego spośród wszystkich operandów danego fragmentu wyodrębnionego. Wybór ten jest dokonywany na podstawie warunku przypisanego do operandu. Warunki mogą być opisane w sposób jawny albo niejawny. Określenie warunku jawnego następuje poprzez jego umieszczenie w obszarze danego operandu. Pominięcie takiego zapisu oznacza, że warunek ten jest domyślny.	7.21
opcja	<i>opt</i>	Oznacza, że operand we fragmencie wyodrębnionym wystąpi lub zostanie pominięty.	7.22
przerwanie	<i>break</i>	Jest skróconą formą operatora <i>alt</i> . Występuje w sytuacji, gdy na diagramie istnieje tylko jeden ściśle zdefiniowany operand, natomiast otoczenie fragmentu wyodrębnionego potraktowane zostaje w kategoriach drugiego operandu. W momencie wykonania fragmentu wyodrębnionego oznaczonego operatorem <i>break</i> otaczające go interakcje są ignorowane.	7.23

Tabela 7.4. Operatory interakcji na diagramach sekwencji (ciąg dalszy)

Pełna nazwa operatora	Operator	Interpretacja	Rysunki
iteracja	<i>loop</i>	Oznacza powtarzanie operandu określoną ilość razy. Dolna i górna liczba iteracji może być wskazana w formie parametru, jak poniżej: $\text{loop ["("} \langle \text{minint} \rangle \text{"}, \text{"} \langle \text{maxint} \rangle \text{")"}]$ Parametr <i>minint</i> jest nieujemną liczbą całkowitą, natomiast <i>maxint</i> — liczbą całkowitą większą od <i>minint</i> . W przypadku wyspecyfikowania tylko jednego parametru pętla zostanie wykonana wskazaną ilość razy. Pominięcie parametrów jest domyślnie traktowane jako wykonanie pętli od zera do nieskończoności.	7.24, 7.25, 7.32
funkcjonalność nieprawidłowa	<i>neg</i>	Charakteryzuje operandy, które traktowane są w kategoriach działań nieprawidłowych. Operandy te wskazują na wyjątki, które muszą zostać obsłużone. Do takich wyjątków obsługiwanych w językach programowania (np. w języku JAVA) należą m.in. błąd operacji arytmetycznej, indeks poza zakresem, nieprawidłowy typ danych i przepełnienie stosu.	7.25
współbieżność	<i>par</i>	Wskazuje na równoległe wykonywanie wszystkich operandów danego fragmentu wyodrębnionego.	7.26, 7.28
obszar krytyczny	<i>critical</i>	Charakteryzuje fragment wyodrębniony stanowiący zamkniętą całość, który uzyskuje najwyższy priorytet w realizacji interakcji. Instancje klasyfikatorów uczestniczące w realizacji obszaru krytycznego nie mogą uczestniczyć w innych interakcjach. Są zablokowane wyłącznie dla potrzeb wykonania interakcji obszaru krytycznego. Interakcje występujące w otoczeniu obszaru krytycznego i nie ingerujące w jego funkcjonowanie mogą być wykonywane równocześnie.	7.28
formuła	<i>assert</i>	Reprezentuje wykonanie sformalizowanego twierdzenia, algorytmu. Jest często powiązany z operatorami <i>consider</i> lub <i>ignore</i> .	7.29
istotność	<i>consider</i>	Oznacza fragment wyodrębniony przedstawiający zestaw operandów zawierających komunikaty, które muszą zostać wykonane w ramach danego diagramu sekwencji. Składnia operatora jest następująca: $\text{consider " "{"} \langle \text{nazwy-komunikatów} \rangle \text{"} \text{"} \text{"} \text{"}$ W przypadku występowania we fragmencie wyodrębnionym kilku komunikatów są one oddzielane przecinkami; np. consider {wybierzWykładowcę, wybierzPracownię} consider {wstawArtykułNaAukcję}	7.30

Tabela 7.4. Operatory interakcji na diagramach sekwencji (ciąg dalszy)

Pełna nazwa operatora	Operator	Interpretacja	Rysunki
nieistotność	<i>ignore</i>	Oznacza fragment wyodrębniony specyfikujący komunikaty nie mające istotnego wpływu na całość interakcji. Komunikaty te mogą być wskazane w dowolnym miejscu na diagramie lub zostać pominięte. Po słowie kluczowym <i>ignore</i> w nawiasach klamrowych podaje się listę nieistotnych komunikatów: <i>ignore</i> ”{” <nazwy-komunikatów> ”}” W przypadku występowania we fragmencie wyodrębnionym kilku komunikatów są one oddzielane przecinkami; np. <i>ignore</i> {skasujPlikiTymczasowe, wyczyśćBufor} Operator <i>ignore</i> może być jednym z parametrów umieszczonych w nagłówku ramy diagramu sekwencji; np. <i>sd zarejestrujDokumenty ignore</i> {podajNarodowość}	7.31
ściśle uporządkowanie	<i>strict</i>	Wskazuje, że operandy muszą być wykonywane w niezmiennej, określonej na diagramie kolejności.	7.32
słabe uporządkowanie	<i>seq</i>	Wskazuje, że komunikaty na różnych liniach życia dla różnych operandów mogą wystąpić w dowolnym porządku.	7.33

Praktyczne zastosowanie wyróżnionych operatorów przedstawiają rysunki 7.21 – 7.33. Każdy z przykładów zawiera:

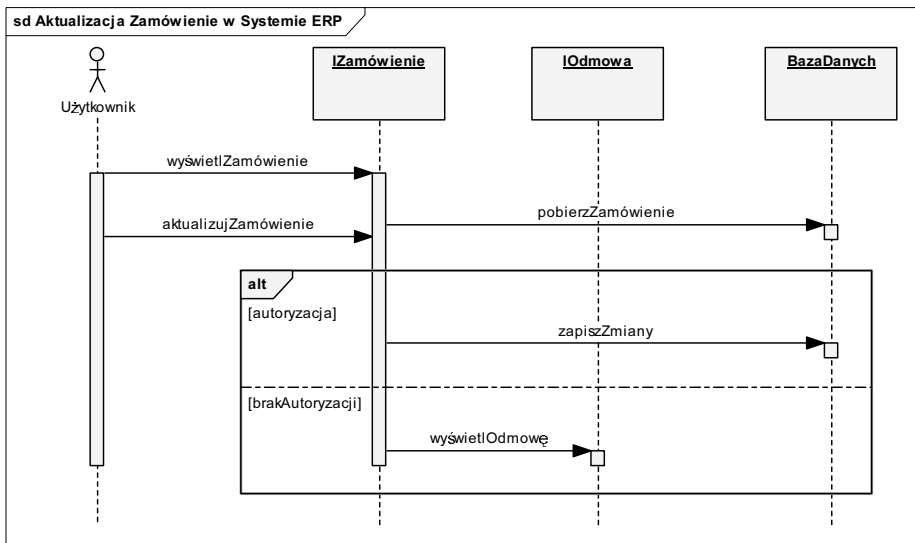
- ♦ nazwę operatora interakcji,
- ♦ szczegółową charakterystykę modelowanego przypadku biznesowego,
- ♦ diagram ilustrujący opisaną sytuację.

Alternatywa *alt*

System informatyczny użytkowany w firmie pozwala na aktualizację zamówień przechowywanych w bazie danych. Poprzez wysłanie komunikatu *pobierzZamówienie* użytkownicy systemu mogą przeglądać zamówienia przechowywane w bazie danych na formacie *IZamówienie* oraz aktualizować je. Aktualizacja zamówień jest czynnością, do wykonania której autoryzację mają wyłącznie pracownicy działu zamówień. Dostęp do zamówień przechowywanych w bazie danych mają również inni pracownicy firmy, w szczególności obsługujący system CRM. Ta druga grupa pracowników nie ma jednak uprawnień, nie ma autoryzacji do modyfikowania zamówień, tj. dokonywania operacji wprowadzania, skreślania i aktualizacji danych. W wyniku przesłania komunikatu *wyświetlZamówienie* system wyświetla obydwu grupom żądane zamówienia. W przypadku komunikatu *aktualizujZamówienie* system identyfikuje prawo dostępu do przeprowadzenia operacji aktualizacji danych w bazie. System realizuje komunikat *zapiszZmiany* dla pracowników działu zamówień, natomiast wszystkim pozostałym użytkownikom system wyświetla formatkę *IOdmowa*, realizując komunikat *wyświetlOdmowę*.

OPIS

DIAGRAM



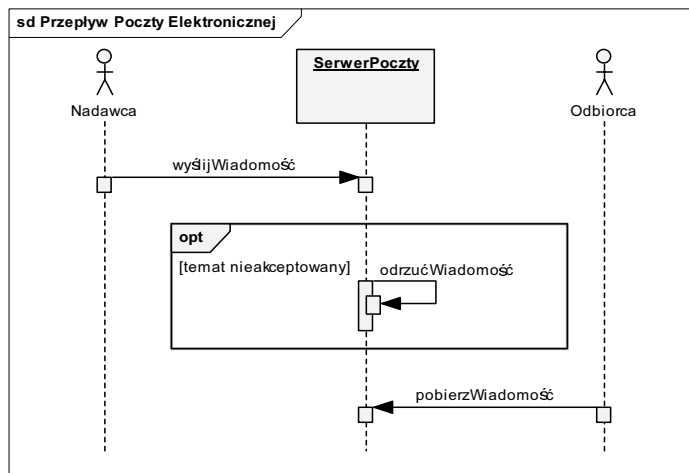
Rysunek 7.21. Aktualizacja zamówienia w systemie ERP

Opcja *opt*

SerwerPoczty obsługuje wielu *Nadawców* i *Odbiorców* poczty elektronicznej. *Odbiorcy* e-maili mogą zastrzec sobie prawo do nieprzyjmowania do swoich skrzynek wiadomości o określonych tematach. *SerwerPoczty* realizuje to zlecenie poprzez założenie odpowiednich filtrów na pocztę przychodzącą do odbiorców. Serwer przyjmuje listy od nadawców po otrzymaniu komunikatu *wyślijWiadomość*. W przypadku stwierdzenia otrzymania listu z tematem nieakceptowanym przez *Odbiorcę*, *SerwerPoczty* wywołuje komunikat *odrzućWiadomość*, co zaznaczono we fragmencie wydodrębnionym z zastosowaniem stosownego warunku. Serwer realizuje więc opcję przechowywania albo odrzucania wiadomości. W dogodnym momencie *Odbiorca* pobiera pocztę poprzez wysłanie komunikatu *pobierzPocztę* skierowanego do serwera. *Odbiorca* otrzymuje zatem listy wyłącznie o akceptowanych przez siebie tematach.

OPIS

DIAGRAM



Rysunek 7.22. Przepływ poczty elektronicznej

Przerwanie *break*

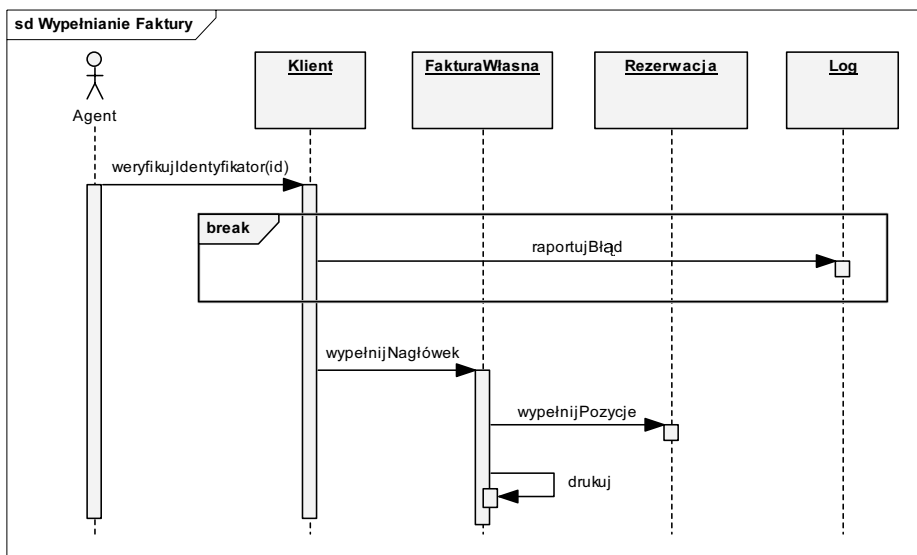
Agent firmy turystycznej po zrealizowaniu zlecenia związanego z zamówieniem wycieczki przystępuje do wygenerowania *FakturyWłasnej*. Na wstępie dokonuje weryfikacji klienta, porównując podany identyfikator z danymi zarejestrowanymi w obiekcie klasy *Klient*. Porównanie to polega na przesłaniu komunikatu *weryfikujIdentyfikator*. W przypadku wystąpienia niezgodności danych przechowywanych w systemie z podanymi przez *Agent*a, w *Logu* systemowym zapisywany jest fakt zaistnienia tej niezgodności. Dokonuje się to w systemie w efekcie przesłania komunikatu *raportujBłąd*. Tym samym wystawianie faktury zostaje przerwane. Natomiast jeśli dane podane przez *Agent*a i zawarte w obiekcie klasy *Klient* są zgodne, następują kolejne operacje:

- ♦ nagłówek faktury jest wypełniany na podstawie danych przechowywanych w obiekcie klasy *Klient* w wyniku przesłania komunikatu *wypełnijNagłówek*,
- ♦ obiekt klasy *FakturaWlasna* przesyła żądanie wypełnienia poszczególnych pozycji faktury w postaci komunikatu *wypełnijPozycje* do obiektu klasy *Rezerwacja*.

Pozycje faktury wypełniane są danymi przechowywanymi w obiekcie klasy *Rezerwacja*. Po wygenerowaniu w systemie faktury jest ona drukowana (komunikat *drukuj*).

OPIS

DIAGRAM



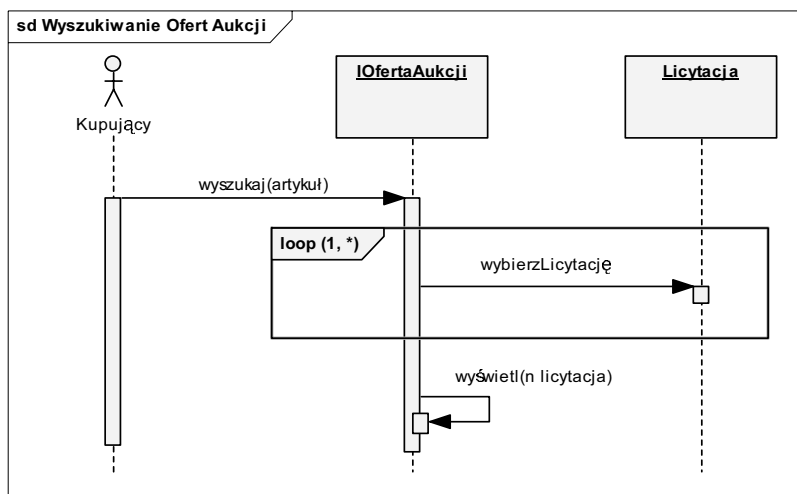
Rysunek 7.23. Wypełnianie faktury

Iteracja *loop*

Klient jest zainteresowany kupnem artykułu na aukcji internetowej. W tym celu wykorzystuje oprogramowanie znanej internetowej firmy aukcyjnej. Aby wyszukać interesujące go licytacje w danej kategorii, *Kupujący* przegląda ofertę wybranej firmy aukcyjnej. W tym celu wprowadza do formatki *IOfertaAukcji* nazwę poszukiwanego artykułu i zleca systemowi wyszukanie (przesłanie komunikatu *wyszukaj* z parametrem *artykuł*) wszystkich licytacji, w których przedmiotem sprzedaży jest poszukiwany artykuł. Instancja klasyfikatora *IOfertaAukcji* wywołuje operację *wybierzLicytację* obiektu klasy *Licytacja*. Operacja ta wykonywana jest jako pętla programowa (iteracja) i pozwala na wyszukanie wszystkich instancji klasy *Licytacja* będących licytacjami danego artykułu. Po wybraniu wszystkich licytacji zawierających dany artykuł ich zestawienie zostaje wyświetlone poprzez wykonanie komunikatu *wyświetl* na instancji klasyfikatora *Licytacja*.

OPIS

DIAGRAM



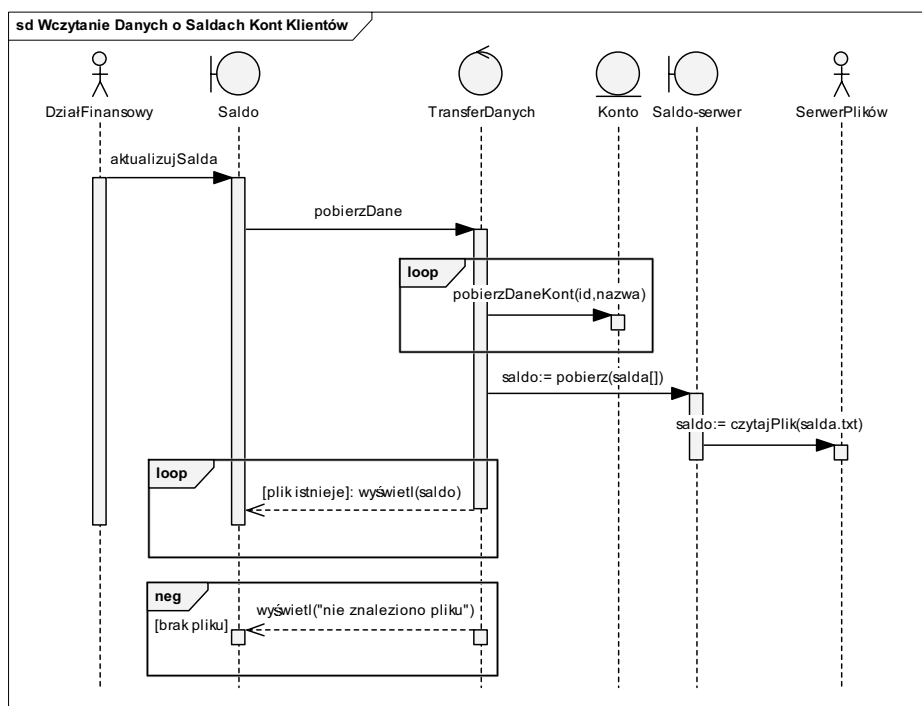
Rysunek 7.24. Wyszukiwanie ofert aukcji

Funkcjonalność nieprawidłowa *neg*

Bank prowadzi kilka osobnych kont firmy. Efektywne monitorowanie płynności finansowej przedsiębiorstwa wymaga dostępu do aktualnych sald kont bankowych. Dane te są przechowywane na serwerze plików w pliku tekstowym, który jest na bieżąco aktualizowany. *DziałFinansowy*, który oczekuje od systemu wyświetlenia aktualnych sald kont, inicjuje proces przesyłania danych o saldach kont bankowych przechowywanych na serwerze w pliku tekstowym o nazwie *salda.txt*. W tym celu wykorzystuje analityczny obiekt graniczny *Saldo*, który realizuje operację *aktualizujSalda*. Konkretna instancja analitycznej klasy sterującej *TransferDanych* zawiera logikę sterowania procesem importowania danych z pliku tekstowego. Interfejsem pośredniczącym jest obiekt klasy granicznej *Saldo-serwer*. Po uruchomieniu operacji instancji klasyfikatorów *Konto* oraz *SerwerPlików*, *TransferDanych* wysyła komunikat do instancji klasyfikatora *Saldo*, aby ten wyświetlił na ekranie systemu aktualne dane o saldach kont bankowych. W przypadku, gdy plik *salda.txt* nie zostaje znaleziony w określonym miejscu na serwerze, system informuje *DziałFinansowy* o tym nieprawidłowym zdarzeniu, które jako wyjątek musi zostać obsłużone przez system.

OPIS

DIAGRAM



Rysunek 7.25. Wczytanie danych o saldach kont klientów

Współbieżność *par*

Zadaniem systemu GPS, wykorzystywanego w ratownictwie miejskim i górskim, jest ustalanie pozycji śledzonego obiektu. *SystemInformatycznyGPS*, który jest częścią systemu GPS, obsługiwany jest przez *Operatora* inicjującego ustalanie pozycji obiektu. Elementy składowe systemu GPS, tj. *SystemInformatycznyGPS*, *OdbiornikGPS*, *SatelitaTelekomunikacyjny* i *NadajnikGPS* wykonują następujące funkcje równoległe (*par*):

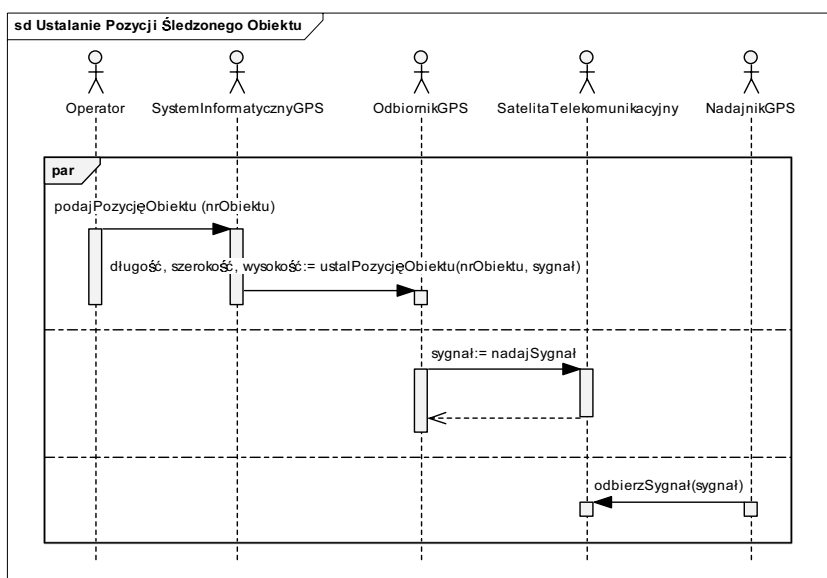
- ♦ *Operator* użytkuje *SystemInformatycznyGPS*, któremu przekazuje numer obiektu i zleca podaniePozycjiObiektu. Następnie *SystemInformatycznyGPS* aktualizuje pozycję śledzonego obiektu poprzez wywołanie operacji *podajPozycję* na *OdbiornikuGPS*. *OdbiornikGPS*, wykorzystując odebrany sygnał z *SatelityTelekomunikacyjnego*, ustala pozycję śledzonego obiektu (długość i szerokość geograficzną oraz wysokość nad poziomem morza).
- ♦ *SatelitaTelekomunikacyjny* w sposób nieprzerwany odbiera sygnały z *OdbiornikaGPS* w celu skutecznego zrealizowania operacji *nadawaniaSygnału*.

NadajnikGPS nieprzerwanie wysyła sygnały do *SatelityTelekomunikacyjnego*, aby ten odebrałSygnał, na podstawie którego *OdbiornikGPS* ustali lokalizację śledzonego obiektu.

Na tej zasadzie system GPS realizuje współbieżne funkcje.

OPIS

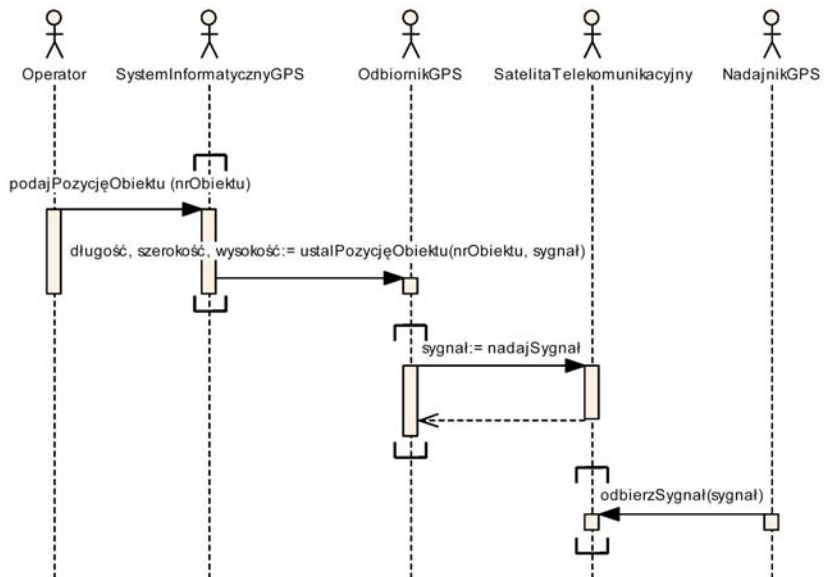
DIAGRAM



Rysunek 7.26. Ustalanie pozycji śledzonego obiektu

Istnieje uproszczona forma zapisu współbieżności operandów fragmentu wyodrębnionego na diagramach sekwencji. I tak odpowiednikiem rysunku 7.26 jest rysunek 7.27.

Rysunek 7.27.
*Ustalanie pozycji
śledzonego obiektu*
— skrócona
notacja
współbieżności

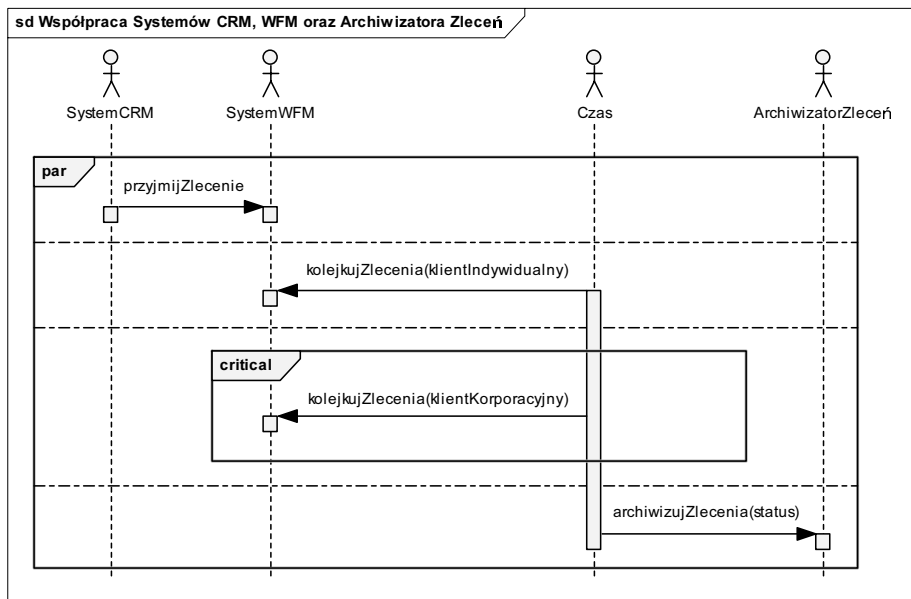


Obszar krytyczny *critical*

Integracja systemów informatycznych to rozwiązanie wykorzystywane przez dynamicznie rozwijające się przedsiębiorstwa. W celu zarządzania relacjami z klientem przedsiębiorstwa inwestują w systemy CRM (ang. *Customer Relationship Management*). Operatorzy takich systemów wprowadzają jakościowe i ilościowe dane o klientach oraz ich preferencjach. Systemy CRM pozwalają na wprowadzenie programów lojalnościowych, w których przypisuje się różny priorytet różnym klientom. Klienci, którzy generują wysokie obroty lub zyski, uzyskują najwyższy priorytet. W charakteryzowanym systemie priorytety przypisuje się każdemu klientowi korporacyjnemu. *SystemCRM* generuje zlecenie, które poprzez wywołanie operacji *przyjmijZlecenie* jest przekazywane do użytkowanego przez firmę systemu elektronicznego obiegu informacji WFM (ang. *Work Flow Management*). *SystemWFM* wykorzystuje generator kolejkowania. Określa on na podstawie danych z *SystemuCRM* priorytet zlecenia i drogę jego realizacji. Instancja klasyfikatora *Czas* inicjuje wykonanie operacji odświeżającej kolejkę realizacji zleceń wykonywanych w stałych odcinkach czasu oraz archiwizację zleceń w bazie danych *ArchiwizatorZleceń*. Wszystkie operandy mogą być realizowane równolegle, na co wskazuje operator *par* i oddzielone przerywanymi liniami poszczególne operandy. W momencie wykonania operacji *kolejkujZlecenia* z parametrem *klientKorporacyjny*, oznaczającej zidentyfikowane zlecenie klienta korporacyjnego, następuje zablokowanie realizowania operacji wszystkich instancji klasyfikatorów poza obrębem fragmentu wyodrębnionego oznaczonego operatorem *critical*. W praktyce powoduje to natychmiastowe wykonanie zlecenia klienta korporacyjnego o najwyższym priorytecie w kolejce przy jednoczesnym zawieszeniu pozostałych zleceń, które mają niższy priorytet i mogłyby przerwać wykonywanie interakcji w obszarze fragmentu wyodrębnionego oznaczonego operatorem *critical*. Po wykonaniu operacji *kolejkujZlecenia* dla klienta korporacyjnego systemy powracają do realizowania równoległych interakcji.

OPIS

DIAGRAM



Rysunek 7.28. Współpraca systemów CRM, WFM oraz archiwizatora zleceń

Formuła *assert*

W systemie sterowania zapasami w firmie wykorzystywany jest model zapasów z zakresu badań operacyjnych. W modelu tym monitorowany jest poziom zapasów wszystkich artykułów. Dla każdego towaru wyznaczony jest tzw. poziom zamówieniowy, po przekroczeniu którego należy złożyć zamówienie na kolejną partię towaru, aby sprostać prognozowanemu popytowi. Wielkość partii zakupu jest obliczana według reguły optymalizacyjnej zwanej regułą pierwiastka kwadratowego:

$$Q = \sqrt{\frac{2 * A * S}{R + (I * U)}}$$

gdzie:

Q — optymalna partia,

A — koszt złożenia zamówienia,

S — wielkość sprzedaży dla okresu prognozowanego,

R — jednostkowy koszt utrzymania zapasu,

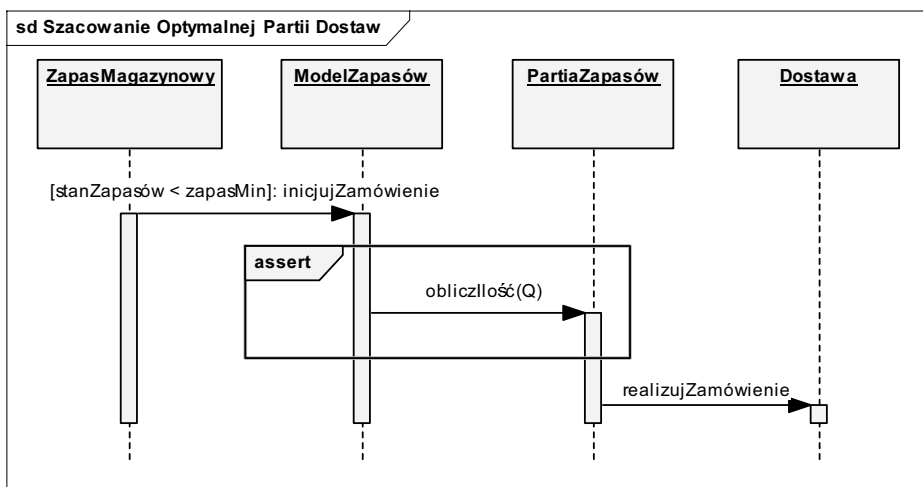
I — jednostkowy koszt braku zapasu,

U — współczynnik ryzyka wyrażony prawdopodobieństwem braku zapasów.

Po obliczeniu optymalnej partii zakupu według podanej formuły następuje natychmiastowe przesłanie zamówienia do dostawcy danego towaru.

OPIS

DIAGRAM



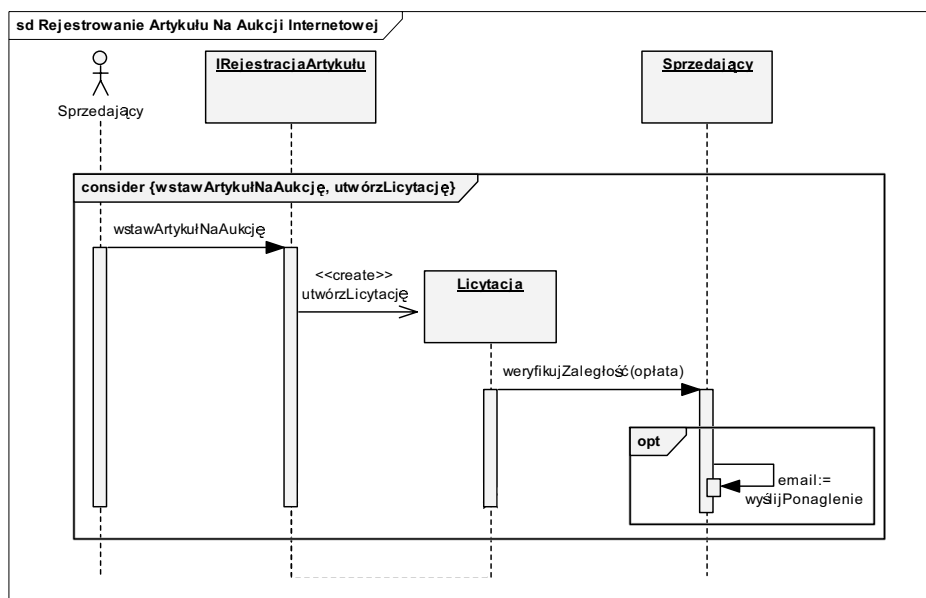
Rysunek 7.29. Szacowanie optymalnej partii dostaw

Istotność *consider*

Aukcje internetowe umożliwiają rejestrowanie artykułów, które później stają się przedmiotem licytacji. Uczestnicy licytacji tj. *Sprzedający* i *Kupujący*, wykorzystują przeglądarki internetowe do uczestnictwa w licytacjach czy też śledzenia przebiegu licytacji interesujących ich artykułów. Aby licytacja mogła się odbyć, *Sprzedający* musi zarejestrować artykuł w systemie aukcji internetowej. Czynność ta jest wykonywana za pośrednictwem formatki *IRejestracjaArtykułu* i polega na wprowadzeniu wymaganych informacji dotyczących cech artykułu oraz atrybutów licytacji, jak również na wywołaniu operacji *wstawArtykułNaAukcję*. Następnie system tworzy obiekt klasy *Licytacja* dzięki odebraniu od instancji klasyfikatora *IRejestracjaArtykułu* komunikatu stereotypowanego «create» *utwórzLicytację*. Utworzony obiekt w określonym przez *Sprzedającego* terminie i czasie zostanie aktywowany, co spowoduje rozpoczęcie licytacji. Podczas rejestracji artykułu instancja klasyfikatora *Licytacja* wysyła komunikat do instancji klasyfikatora *Sprzedający*, aby ta zweryfikowała, czy na koncie *Sprzedającego* nie występują zaległości finansowe z tytułu obsługi jego poprzednich licytacji. Jeśli zaległość występuje, system wysyła do *Sprzedającego* e-mail z ponagleniem. Realizuje się to w systemie poprzez wywołanie operacji *wyślijPonaglenie*. Należy zwrócić uwagę, że wystąpienie zaległości nie determinuje procesu rejestracji artykułu na aukcji internetowej i utworzenia licytacji. A zatem za komunikaty istotne (*consider*) uważa się *wstawArtykułNaAukcję* oraz *utwórzLicytację*. Konwencja notacji UML nakazuje tu zapisanie nazw operacji istotnych w nawiasach klamrowych, jak również umieszczenie nazw tych komunikatów w pentagramie fragmentu wyodrębnionego po nazwie operatora.

OPIS

DIAGRAM



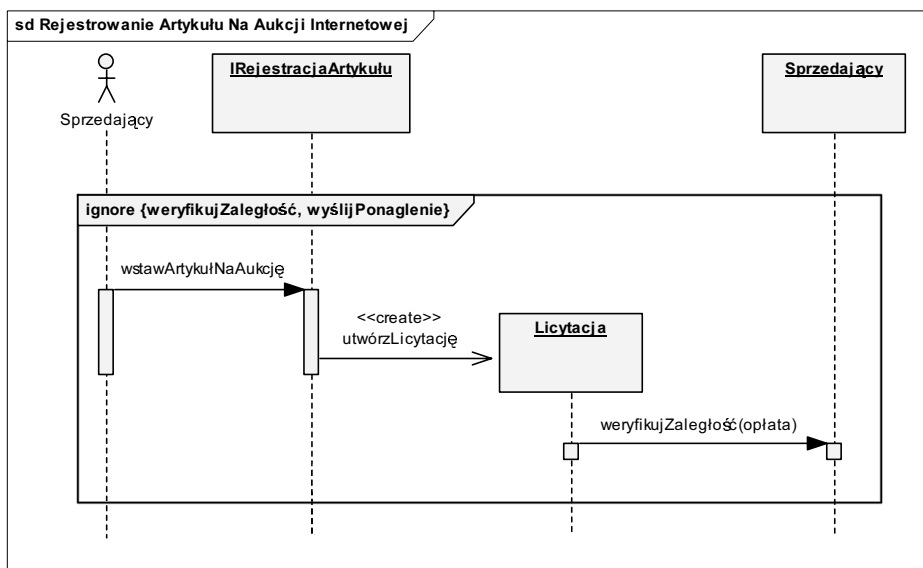
Rysunek 7.30. Rejestrowanie artykułu na aukcji internetowej — operator *consider*

Nieistotność *ignore*

Przykład podany na poprzednim rysunku w odniesieniu do operatora istotności *consider* pozwala również doskonale zilustrować znaczenie operatora *ignore*. Wyjaśniono na nim znaczenie poszczególnych instancji klasyfikatorów (*Sprzedający* — jako aktor, *IRejestracjaArtykułu*, *Sprzedający* — jako obiekt klasy) oraz dwóch komunikatów: *wstawArtykułNaAukcję* oraz *utwórzLicytację*. Dwie pierwsze operacje są uznawane za istotne. Inne wywoływane operacje nie mają znaczącego wpływu na zarejestrowanie artykułu oraz utworzenie obiektu klasy *Licytacja*. Hierarchia ważności tych komunikatów wynika z reguł biznesowych e-licytacji. Wskazują one, że wystąpienie jakiegokolwiek zaległości na koncie *Sprzedającego* z tytułu obsługi poprzednich licytacji nie determinuje zainicjowania procesu rejestracji artykułu i utworzenia licytacji w systemie. Zatem komunikaty wywołujące operacje *weryfikujZaległość* oraz *wyślijPonaglenie* określa się jako nieistotne w stosunku do dwóch podstawowych, wyżej wymienionych komunikatów. Konwencja notacji operatora *ignore* polega na umieszczeniu po słowie kluczowym *ignore* nazw komunikatów nieistotnych w nawiasach klamrowych. Komunikaty te mogą się pojawić na diagramie lub można je pominąć w fragmencie wyodrębnionym. Nie oznacza to jednak wyeliminowania tych komunikatów — wprost przeciwnie, są one każdorazowo wykonywane w systemie. Stosownie do tych zasad na niniejszym rysunku jeden z tych komunikatów, *weryfikujZaległość*, przedstawiono explicite, natomiast *wyślijPonaglenie* pominięto.

OPIS

DIAGRAM



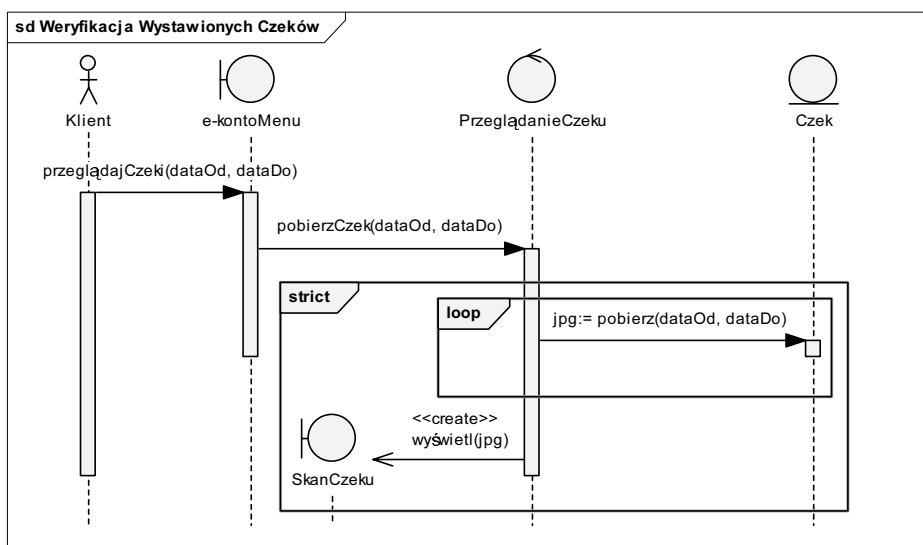
Rysunek 7.31. Rejestrowanie artykułu na aukcji internetowej — operator *ignore*

Ścisłe uporządkowanie *strict*

Klient posiadający konto bankowe może korzystać z kilku form płatności. Jedną z nich jest uregulowanie należności poprzez wystawienie czeku bankowego. W późniejszym czasie *Klient* może obejrzeć w formie elektronicznej wystawione przez siebie чеки. W tym celu korzysta z konta elektronicznego za pośrednictwem przeglądarki internetowej. Funkcjonalność tego konta pozwala na wyszukanie wszystkich czeków, które *Klient* wystawił w podanym przedziale czasu. Wprowadza on dwie daty — *datęOd* oraz *datęDo* — jako parametry operacji *przeglądajCzeki*, wykonywanej w wyniku przesłania komunikatu na obiekcie granicznym *e-kontoMenu*. W efekcie ostatnia z wymienionych instancji klasyfikatora inicjuje komunikat *pobierzCzek* przesłany do obiektu sterującego *PrzeglądanieCzeku*. Komunikat ten przysyłany jest z podanymi wcześniej datami jako parametrami. Wykonywanie kolejnych operacji jest ściśle określone (nie można zmienić ich kolejności), toteż zostały one ujęte we fragmencie wyodrębnionym z operatorem *strict*. Obejmuje on pętlę pobierania kopii kolejnych czeków w formacie *jpg* i umieszczanie ich w nowym obiekcie granicznym *SkanCzeku*. Obiekt ten jest tworzony poprzez wysłanie komunikatu stereotypowanego «create» *wyświetl* ze stosownym parametrem.

OPIS

DIAGRAM



Rysunek 7.32. Weryfikacja wystawionych czeków

Stabe uporządkowanie seq

Użytkownik Internetu tworzy na *IPulpicie* swojego systemu informatycznego ikony reprezentujące skróty do najczęściej odwiedzanych przez niego serwisów. Należą do nich:

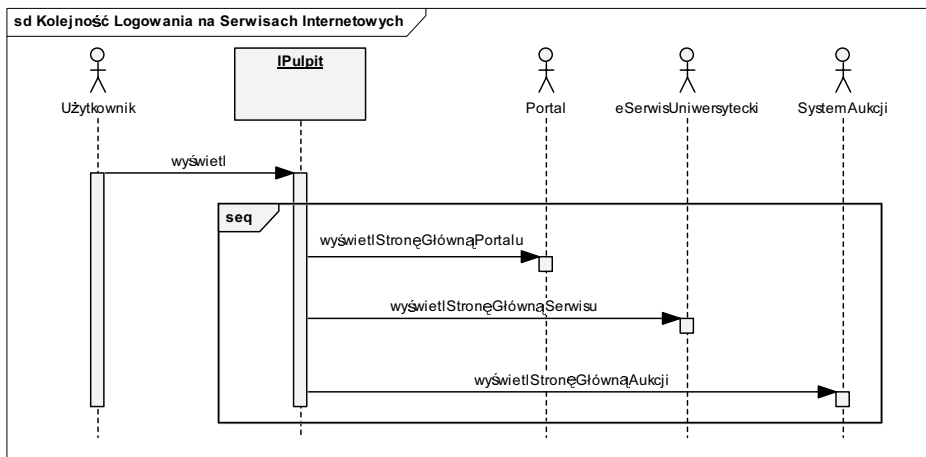
- ♦ *Portal* internetowy,
- ♦ *eSerwisUniwersytecki*,
- ♦ *SystemAukcji* internetowej.

Dzięki temu użytkownik systemu może wywoływać w dowolnej kolejności poszczególne serwisy, wykorzystując odpowiednio:

- ♦ komunikat *wyświetlStronęGłównąPortalu*,
- ♦ komunikat *wyświetlStronęGłównąSerwisu*,
- ♦ komunikat *wyświetlStronęGłównąAukcji*.

OPIS

DIAGRAM



Rysunek 7.33. Kolejność logowania na serwisach internetowych

Przywoływane wystąpienia interakcji

W złożonych systemach zaprojektowanie realizacji danego przypadku użycia na jednym diagramie sekwencji jest trudne. Problem ten rozwiązuje się poprzez przedstawienie danej interakcji na kilku ściśle, logicznie powiązanych ze sobą diagramach. Całość interakcji syntetycznie przedstawia diagram sterowania interakcją lub implementacyjny diagram sekwencji, który z punktu widzenia powiązanych diagramów ma charakter **bazowy**. Związane diagramem bazowym, spójne tematycznie wystąpienia interakcji mogą być udokumentowane na oddzielnych, lecz ściśle powiązanych z diagramem bazowym diagramach sekwencji. Stąd na diagramach sekwencji, obok fragmentów wyodrębnionych opisywanych przez operatory interakcji, można modelować **przywołane wystąpienia interakcji** (ang. *interaction occurrences*).



Przywołane wystąpienie interakcji jest zamieszczanym na diagramie bazowym odwołaniem do ściśle z nim powiązanego diagramu interakcji.

Wystąpienia te mają duże znaczenie w przypadku rozbudowanych diagramów sekwencji, w których przywołuje się merytorycznie diagramy sekwencji zdefiniowane wcześniej w projekcie.

Zastosowanie przywoływanych wystąpień interakcji na diagramach sekwencji pozwala na zstępujące (ang. *top-down*) modelowanie interakcji w systemie. Wystąpienia te oznacza się **operatorem przywołania *ref***, zamieszczonym w pentagramie diagramu. Istnieją dwa sposoby zainicjowania wystąpień interakcji w systemie:

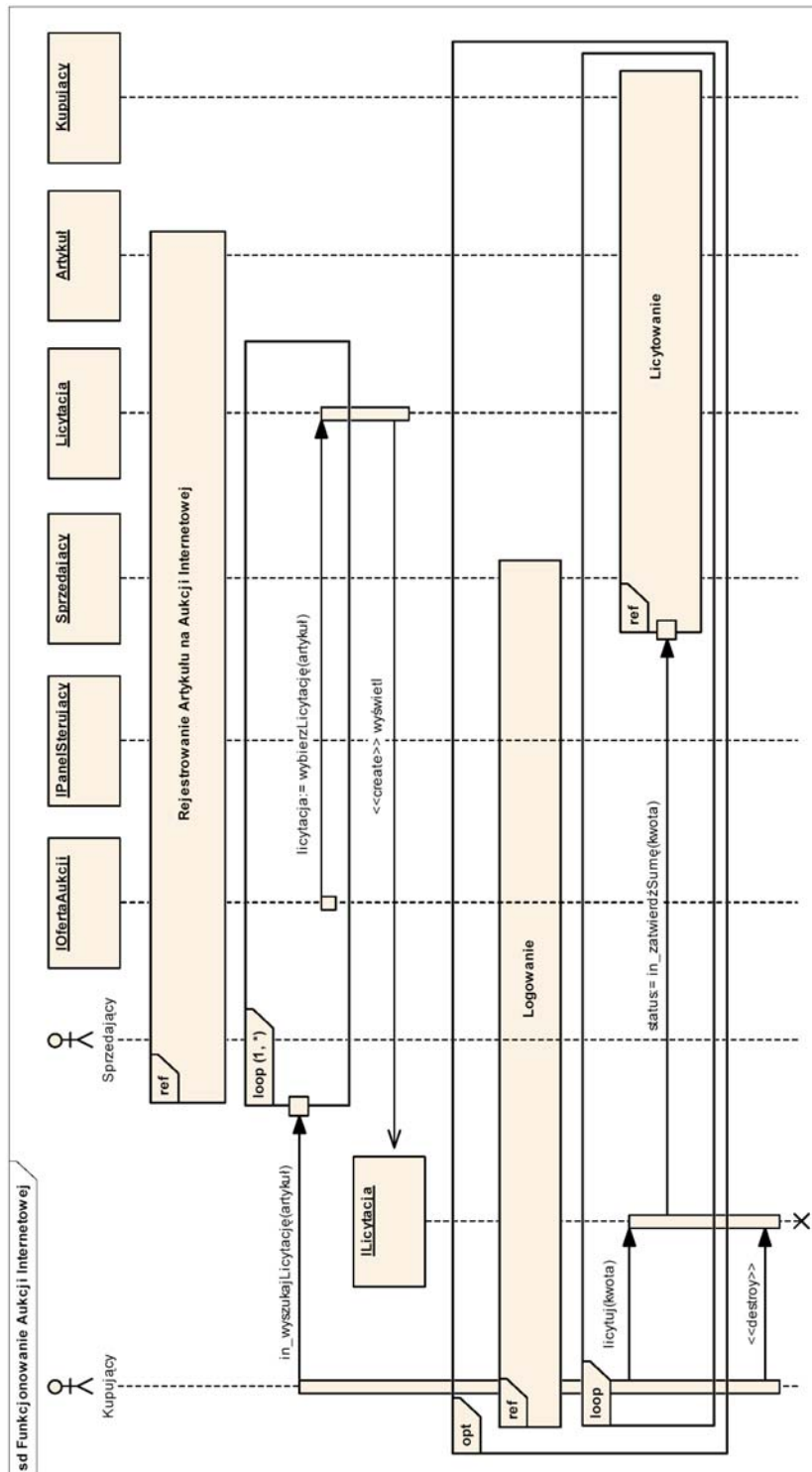
- ♦ poprzez komunikat,
- ♦ poprzez czynnik czasu.

W pierwszym przypadku odwołanie się do wystąpienia interakcji *ref* dokonywane jest za pośrednictwem komunikatu przesyłanego z diagramu bazowego przez bramę do pierwszej instancji klasyfikatora przywoływanego wystąpienia interakcji. W drugim przypadku czynnikiem inicjującym jest sekwencja występowania elementów diagramu bazowego, w tym obszarów przywoływanych wystąpień interakcji. Odwołanie się do wystąpienia interakcji oznacza zawieszenie wykonywania funkcjonalności bazowego diagramu sekwencji. Po wykonaniu pełnej funkcjonalności przywołanego wystąpienia interakcji następuje przekazanie sterowania do bazowego diagramu sekwencji (rysunek 7.34).

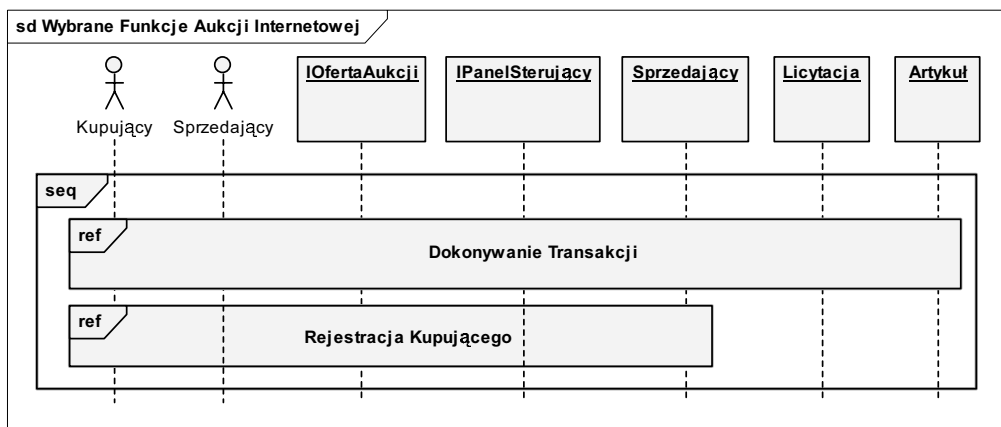
Na aukcji internetowej wystawionych jest na licytację wiele artykułów. W licytacji biorą udział aktorzy *Kupujący* i *Sprzedający* oraz obiekty klas przechowujące informacje związane z przebiegiem danej aukcji: *IOfertaAukcji*, *IPanelSterujący*, *Sprzedający*, *Licytacja*, *Artykuł*, *Kupujący*. Funkcjonalność aukcji internetowej jest złożona, toteż niektóre wystąpienia interakcji precyzyjnie udokumentowano na osobnych diagramach sekwencji. Zatem w syntetycznym, bazowym diagramie sekwencji istnieje możliwość przywoływania tych udokumentowanych interakcji w postaci wystąpień interakcji oznaczonych operatorem *ref*. W przypadku aukcji internetowej zdefiniowano wcześniej trzy wystąpienia interakcji: *Rejestrowanie Artykułu na Aukcji Internetowej*, *Logowanie* oraz *Licytowanie*. W pierwszym i drugim przypadku czynnikiem inicjującym jest sekwencja występowania elementów diagramu, natomiast w trzecim komunikat *in_zatwierdźSumę*. Przywołanie każdego z tych wystąpień interakcji powoduje jego aktywację i zawieszenie interakcji wykonywanych na bazowym diagramie sekwencji.

Graficznie obszar wystąpienia interakcji oznaczony operatorem *ref* **przysłania** linie życia instancji klasyfikatorów, które uczestniczą w interakcjach przywoływanego diagramu. W modelowaniu wystąpień interakcji może zaistnieć problem techniczny związany ze wskazywaniem instancji biorących udział w przywoływanym diagramie. Dotyczy to sytuacji, w której dla zachowania **logiki interakcji** nieuniknione jest umieszczenie pomiędzy instancjami klasyfikatorów, biorącymi udział w przywoływanym wystąpieniu interakcji, instancji, które nie uczestniczą w tym wystąpieniu. W takim przypadku linia życia instancji klasyfikatora powinna być ciągła, a zatem przykrywać obszar wystąpienia interakcji. Tym samym jednoznacznie wskazuje się zestaw instancji klasyfikatorów z diagramu bazowego, które uczestniczą w przywoływanym wystąpieniu interakcji.

Rysunek 7.34.
Funkcjonowanie aukcji internetowej



Na rysunku 7.35 zamieszczono diagram zawierający odwołania do diagramów ilustrujących dokonywanie transakcji oraz rejestrację kupującego. Pierwsze z przywoływanych wystąpień interakcji wykorzystuje wszystkie instancje klasyfikatorów zamieszczonych na wspomnianym rysunku, natomiast drugie — wyłącznie *Kupującego*, *IOfertęAukcji* oraz klasę *Sprzedającego*.



Rysunek 7.35. Wybrane funkcje aukcji internetowej

Bramy

Diagramy sekwencji, fragmenty wyodrębnione oraz przywoływane wystąpienia interakcji kontaktują się ze swoim otoczeniem. Otoczenie to wysyła komunikaty do wymienionych obszarów interakcji lub odbiera je z tych obszarów poprzez **bramy** (ang. *gates*).



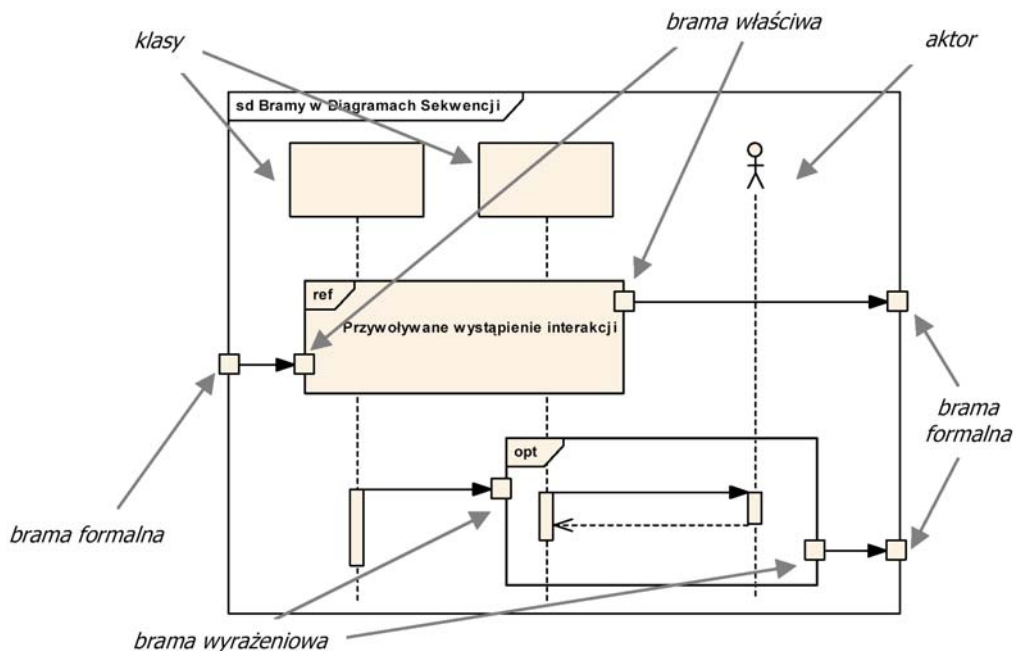
Brama jest punktem przejścia komunikatów z (do) diagramów sekwencji, przywoływanych wystąpień interakcji oraz fragmentów wyodrębnionych do (z) ich otoczenia.

W związku z tym można wyróżnić następujące rodzaje bram,:

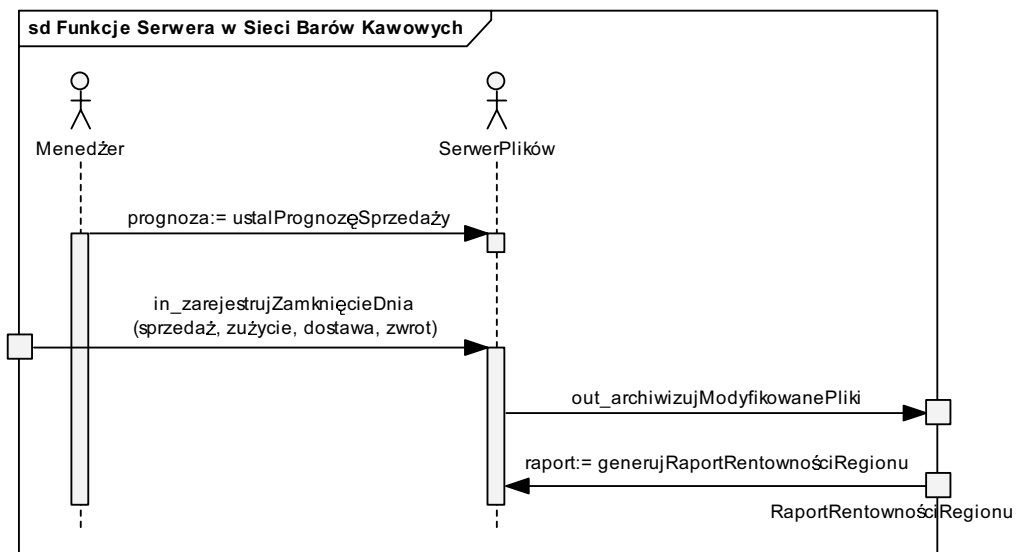
- ♦ **bramy formalne** (ang. *formal gates*) — dotyczą całego diagramu sekwencji;
- ♦ **bramy właściwe** (ang. *actual gates*) — dotyczą przywoływanego wystąpienia interakcji;
- ♦ **bramy wyrażeniowe** (ang. *expression gates*) — dotyczą operandów interakcji danego fragmentu wyodrębnionego.

Zależności pomiędzy pojęciami związanymi z bramami ilustruje rysunek 7.36. Oznaczenie bramy występuje na nim w formie stereotypu.

Praktyczne wykorzystanie pojęcia bramy formalnej na diagramie sekwencji przedstawia rysunek 7.37. Pozostałe rodzaje bram, tj. bramę właściwą i wyrażeniową, zaprezentowano na rysunku 7.34.



Rysunek 7.36. Bramy w diagramach sekwencji



Rysunek 7.37. Funkcje serwera w sieci barów kawowych

Rozległe geograficzne rozmieszczenie sieci barów kawowych może spowodować, że przetwarzanie danych dziennej sprzedaży, obliczanie rentowności firmy czy ustalanie prognoz sprzedaży będzie niemożliwe do zrealizowania bez wsparcia systemu informatycznego. Aby uniknąć takiej sytuacji, w centrali firmy wykorzystano duży komputer

SerwerPlików. Dzięki odpowiedniemu oprogramowaniu komputer, poprzez uruchomienie operacji *zarejestrujZamknięcieDnia* z parametrami: *sprzedaż*, *zużycie*, *dostawa*, *zwrot*, codziennie rejestruje dane z zamknięcia dnia każdego baru kawowego firmy. Dla bezpieczeństwa *SerwerPlików* codziennie przesyła komunikatem *out_archiwizujModyfikowanePliki* wszystkie pliki zmodyfikowane w danym dniu. Ponadto *SerwerPlików* dzięki dużej mocy obliczeniowej obsługuje generator raportów, który umożliwia *Menedżerom* generowanie prognoz sprzedaży. W tym celu *Menedżer* wysyła komunikat do *SerweraPlików*, aby ten wykonał operację *ustalPrognozęSprzedaży*. Aby generator dostarczył raport rentowności regionu sprzedaży, nadawca uruchamia na *SerwerzePlików* operację *generujeRaportRentownościRegionu*, która powoduje utworzenie w pamięci komputera wspomnianego raportu. Komunikat ten przechodzi przez bramę nazwaną *RaportRentownościRegionu*.

Bramy można nazywać dwojako:

- ♦ bezpośrednio, poprzez umieszczenie **nazwy bramy** przy jej graficznym stereotypie; nazwa bramy powinna być zbieżna z nazwą przechodzącego przez nią komunikatu, ponieważ przez każdą bramę przechodzi tylko jeden typ komunikatu;
- ♦ pośrednio, poprzez dodanie do nazwy komunikatu przechodzącego przez bramę **przedrostka** wskazującego kierunek jego przepływu,

np. *in_<nazwa-komunikatu>*,

out_<nazwa-komunikatu>.

Oba wymienione sposoby zaprezentowano na rysunku 7.34. Nazwa komunikatu przesyłanego przez bramę właściwą do przywoływanego wystąpienia interakcji *ref* musi być identyczna z nazwą komunikatu przesyłanego przez bramę formalną powiązane-go diagramu.

Proces tworzenia diagramu sekwencji

Punktem wyjścia w tworzeniu diagramu sekwencji jest przypadek użycia oraz jego scenariusze. Kluczowe znaczenie w tworzeniu diagramu sekwencji mają wymienione poniżej **etapy**:

1. analiza adekwatnego przypadku użycia oraz scenariuszy tego przypadku;
2. identyfikacja klasyfikatorów, których instancje uczestniczą w interakcji;
3. opracowanie diagramu konceptualnego, zawierającego:
 - ♦ zidentyfikowane instancje klasyfikatorów, umieszczane na osi poziomej;
 - ♦ komunikaty, uporządkowane na osi pionowej;
 - ♦ ośrodki sterowania, zlokalizowane na liniach życia;

4. opracowanie implementacyjnego diagramu sekwencji, na podstawie opracowanego na trzecim etapie konceptualnego diagramu sekwencji, poprzez wprowadzenie adekwatnych do specyfikacji danego scenariusza zaawansowanych kategorii pojęciowych, takich jak:
 - ♦ rodzaje komunikatów,
 - ♦ tworzenie i niszczenie obiektów,
 - ♦ warunki,
 - ♦ samowywoływanie,
 - ♦ iteracje,
 - ♦ rozgałęzienia,
 - ♦ fragmenty wyodrębnione i operatory interakcji,
 - ♦ przywoływane wystąpienia interakcji,
 - ♦ bramy;
5. opcjonalne sporządzenie dla danego implementacyjnego diagramu sekwencji wybranych wystąpieniowych diagramów sekwencji.

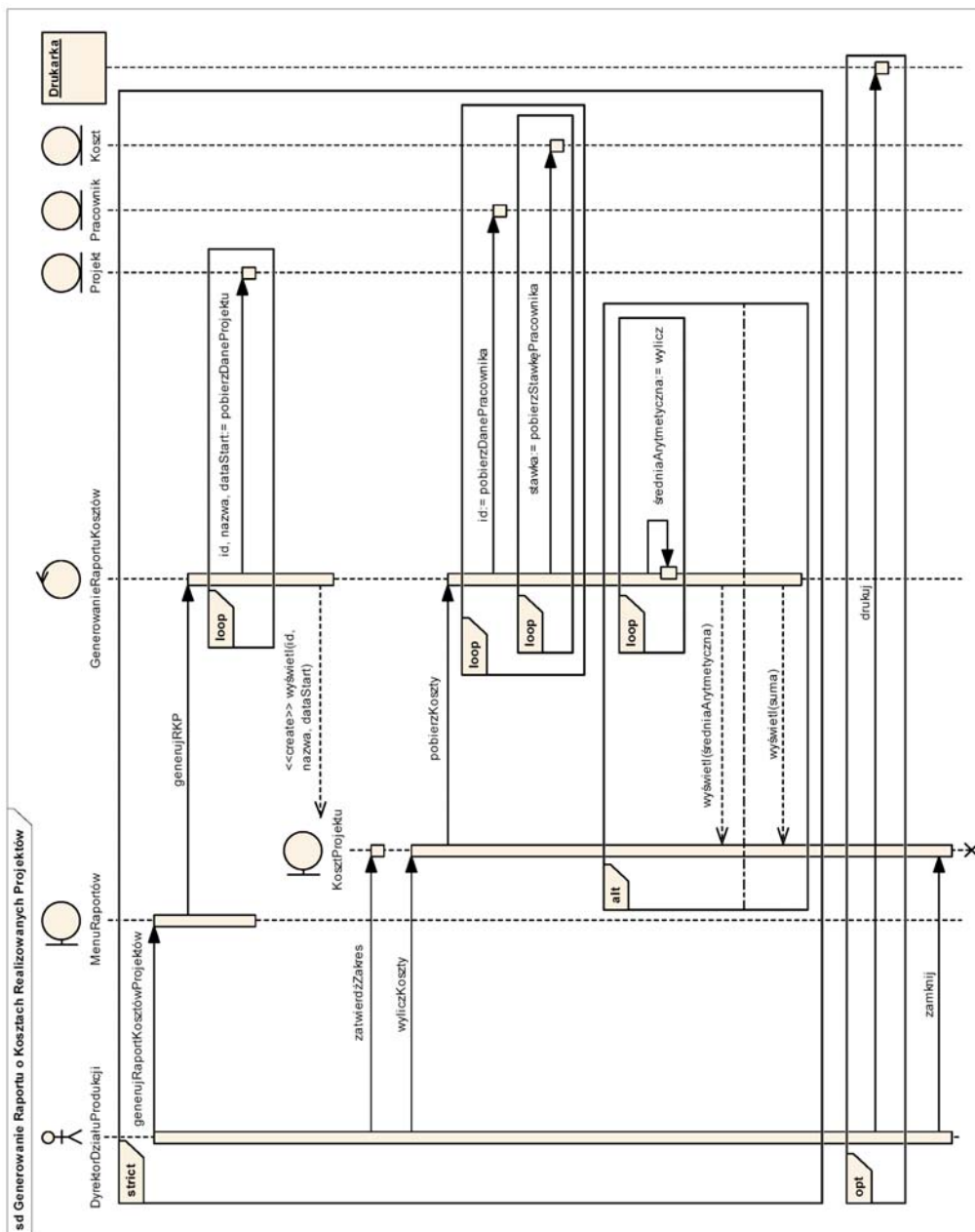
Podobnie jak inne diagramy, również diagram sekwencji podlega regułom iteracyjno-przyrostowego cyklu życia systemu.

Studium diagramu sekwencji

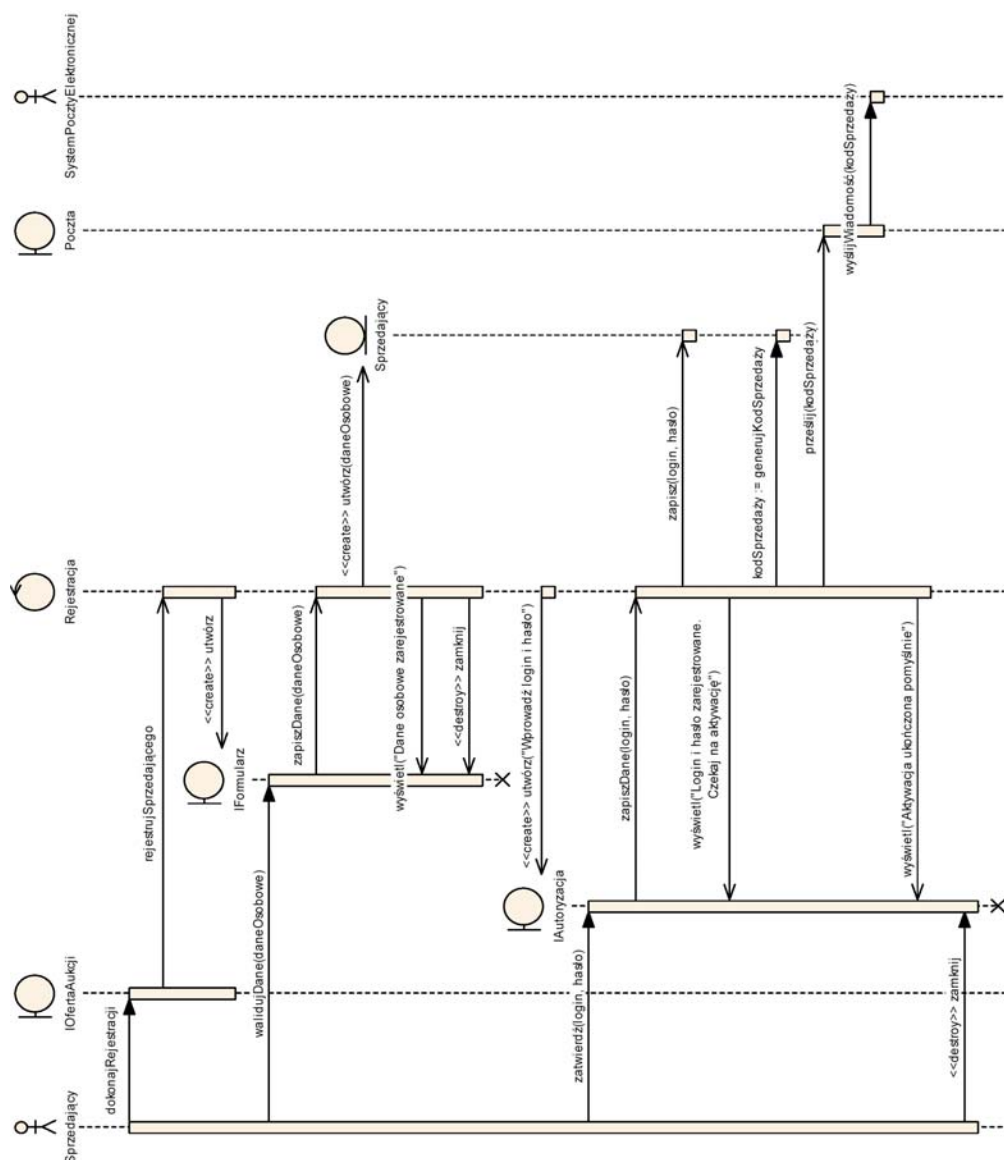
Niniejsze studium zawiera złożone **przykłady** diagramów sekwencji. Zaprezentowano na nich zasady użytkowania elementów diagramu i współzależności między nimi. Poszczególne kategorie całej gamy elementów zostały szczegółowo omówione w punktach 7.1 – 7.3. Poniżej przeanalizowano i zaprezentowano diagramy sekwencji:

- ♦ diagram implementacyjny — „Generowanie raportu o kosztach realizowanych projektów” (rysunek 7.38),
- ♦ diagram wystąpieniowy — „Rejestracja sprzedającego na aukcji internetowej” (rysunek 7.39).

Dynamicznie rozwijające się firmy bardzo szczegółowo monitorują i analizują koszty związane z prowadzonymi projektami. W tym celu system informatyczny powinien dostarczać kadrze menedżerskiej aktualnych danych o poniesionych nakładach. Zamieszczony na rysunku 7.38 diagram sekwencji przedstawia generowanie raportu o kosztach realizowanych projektów. Na przedstawionym diagramie *DyrektorDziałuProdukcji* firmy oczekuje od systemu, że ten w pierwszej kolejności dostarczy mu informacji o prowadzonych projektach. Dla zainicjowania analizy kosztów posługuje się on interfejsem *MenuRaportów*, przekazując komunikat *generujRaportKosztówProjektów*.



Rysunek 7.38. Generowanie raportu o kosztach realizowanych projektów



Rysunek 7.39. Rejestracja sprzedającego na aukcji internetowej

Po otrzymaniu komunikatu *generujRKP* z instancji klasyfikatora *MenuRaportów* analityczny obiekt sterujący *GenerowanieRaportuKosztów* wywołuje operację *pobierzDaneProjektu* analitycznego obiektu przechowującego *Projekt*. Operacja ta jest realizowana w pętli i polega na pobieraniu podstawowych danych o projektach: *id*, *nazwy*, *datyStart*. Po wykonaniu opisanej pętli analityczny obiekt sterujący *GenerowanieRaportuKosztów* tworzy poprzez komunikat stereotypowany «create» interfejs — obiekt graniczny *KosztProjektu*. Interfejsowi przekazywane są parametry *id*, *nazwa*, *dataStart*.

Następnie *DyrektorDziałuProdukcji* wybiera projekty do analizy finansowej kosztów osobowych tych przedsięwzięć za pośrednictwem komunikatu *zatwierdźZakres*. Zleca on następnie przeprowadzenie analizy finansowej tych projektów za pośrednictwem komunikatu *zatwierdźZakres* *wyliczKoszty*. Dla wyselekcjonowanych projektów system kolejno:

- ♦ iteracyjnie wybiera pracowników przypisanych do projektu za pośrednictwem komunikatu *pobierzDanePracownika*;
- ♦ następnie iteracyjnie pobiera przypisaną do wybranego *Pracownika* stawkę z instancji klasyfikatora *Koszt* za pośrednictwem komunikatu *pobierzStawkęPracownika*;
- ♦ alternatywnie:
 - ♦ dla każdego z tych projektów wylicza stawkę średnią i *wyświetla* ją na formacie *KosztProjektu*,
 - ♦ wylicza całkowite nakłady na poszczególne projekty i *wyświetla* je na wspomnianej formacie.

Wskazana wyżej kolejność interakcji poszczególnych operandów nie może ulec zmianie. Przed *zamknięciem* projektu *DyrektorDziałuProdukcji* może opcjonalnie wydrukować zestawienie kosztów. Posługuje się w tym celu komunikatem *drukuj* wysłanym do *Drukarki*.

Zaprezentowane dotąd diagramy sekwencji miały charakter albo conceptualnych, albo implementacyjnych diagramów sekwencji. Diagram zamieszczony na rysunku 7.39 stanowi przykład wystąpieniowego diagramu sekwencji służącego do rejestracji sprzedającego na aukcji internetowej.

Sprzedanie artykułu na aukcji internetowej musi zostać poprzedzone rejestracją *Sprzedającego* oraz rejestracją artykułu będącego przedmiotem licytacji. W celu zarejestrowania *Sprzedającego* wykorzystywana jest strona główna aukcji *IOfertaAukcji*. Za pośrednictwem obiektu klasy sterującej *Rejestracja* wywoływany jest interfejs rejestracji danych osobowych *IFormularz*. Następnie *Sprzedający* wprowadza wymagane dane osobowe i wywołuje operację *walidujDane*, która weryfikuje poprawność danych osobowych wprowadzonych do *IFormularza*. Następnie obiekt sterujący wysyła komunikat *utwórz*, tworząc obiekt klasy *Sprzedający*. Jako parametr w postaci tablicy przekazywane są dane osobowe, będące atrybutami utworzonego obiektu. Utworzenie obiektu klasy *Sprzedający* skutkuje wyświetleniem komunikatu z potwierdzeniem rejestracji, który po krótkiej chwili wyświetlania zostaje wykasowany z pamięci komputera. Takie zachowanie systemu jest realizowane przez komunikaty *wyświetl* z parametrem *Dane osobowe zarejestrowane* oraz *zamknij*.

W momencie zniszczenia interfejsu *IFormularz* instancja klasyfikatora *Rejestracja* tworzy za pośrednictwem komunikatu *utwórz* interfejs *IAutoryzacja*. Interfejs wyświetla pobraną jako parametr treść *Wprowadź login i hasło*. Kolejnym krokiem rejestracji wymagającej aktywności aktora *Sprzedający* jest wprowadzenie przez niego loginu oraz hasła. Obydwa parametry są niezbędne do autoryzowanego korzystania z pełnej oferty aukcji internetowej. Przekazanie parametrów autoryzacji odbywa się za pośrednictwem

komunikatu *zatwierdź* z parametrami login oraz hasło, wysłanego przez *Sprzedającego* do interfejsu *IAutoryzacja*. Parametry autoryzacji zostają zapisane jako atrybuty obiektu klasy *Sprzedający* po wywołaniu przez instancję klasyfikatora *IAutoryzacja* operacji *zapiszDane* z parametrami login oraz hasło (a w konsekwencji operacji *zapisz* z tymi samymi parametrami). W odpowiedzi obiekt klasy *Rejestracja* wysła komunikat *wyświetl* z parametrem *Login i hasło zarejestrowane. Czeka na aktywację*, który powoduje wyświetlenie parametru operacji.

Następnie obiekt klasy *Rejestracja* pobiera kod poprzez wysłanie do obiektu przechowującego *Sprzedający* komunikatu *generujKodSprzedaży*. Wygenerowany i użyty później kod pozwoli na dostęp do odpowiednich modułów aukcji internetowej, które umożliwiają tworzenie licytacji i dokonywanie transakcji online. Wygenerowany *kodSprzedaży* jest przekazywany jako parametr *SystemowiPocztyElektronicznej*, który wykonuje operację *wyślijWiadomość*. Odbywa się to za pośrednictwem interfejsu *Poczta*. Po wykonaniu operacji na interfejsie *IAutoryzacja* wyświetlana jest następująca treść: *Aktywacja ukończona pomyślnie*. Treść ta jest przekazywana jako parametr będący częścią komunikatu *wyświetl* przesłanego od obiektu klasy *Rejestracja*. Osoba rejestrująca się na aukcji może teraz — mając pewność poprawnej rejestracji z prawami sprzedającego — zamknąć interfejs *IAutoryzacja*. Robi to, klikając odpowiedni element interfejsu, co na diagramie przedstawione jest jako przesłanie przez *Sprzedającego* komunikatu *zamknij* do interfejsu *IAutoryzacja*.

Podstawowe pojęcia

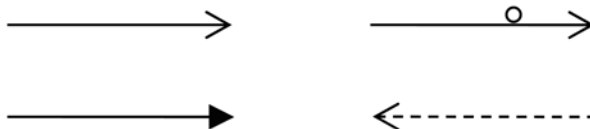
Brama	Fragment interakcji
Formalna	Fragment wyodrębniony
Właściwa	Przywoływane wystąpienie interakcji
Wyrażeniowa	Generator kodu źródłowego
CASE	Interakcja
Diagram sekwencji	Iteracja
Definicja	Klasyfikator
Implementacyjny	Aktor
Konceptualny	Artefakt
Pentagram	Interfejs
Proces tworzenia	Klasa
Rama	Analityczna
Wystąpieniowy	Komponent

Pakiet	Operator interakcji
Przypadek użycia	<i>alt</i>
Sygnał	<i>assert</i>
Węzeł	<i>break</i>
Współdziałanie	<i>consider</i>
Komunikat	<i>critical</i>
Asynchroniczny	<i>ignore</i>
Oczekujący	<i>loop</i>
Opcjonalny	<i>neg</i>
Synchroniczny	<i>opt</i>
Utracony	<i>par</i>
Znaleziony	<i>seq</i>
Zwrotny	<i>strict</i>
Linia życia	Ośrodek sterowania
Obiekt	Aktywacja
Niszczenie	Dezaktywacja
Tworzenie	Sterowanie
Operacja	Parametr
Operand interakcji	Rozgałęzienie
	Samowywoływanie
	Warunek

Pytania i zadania

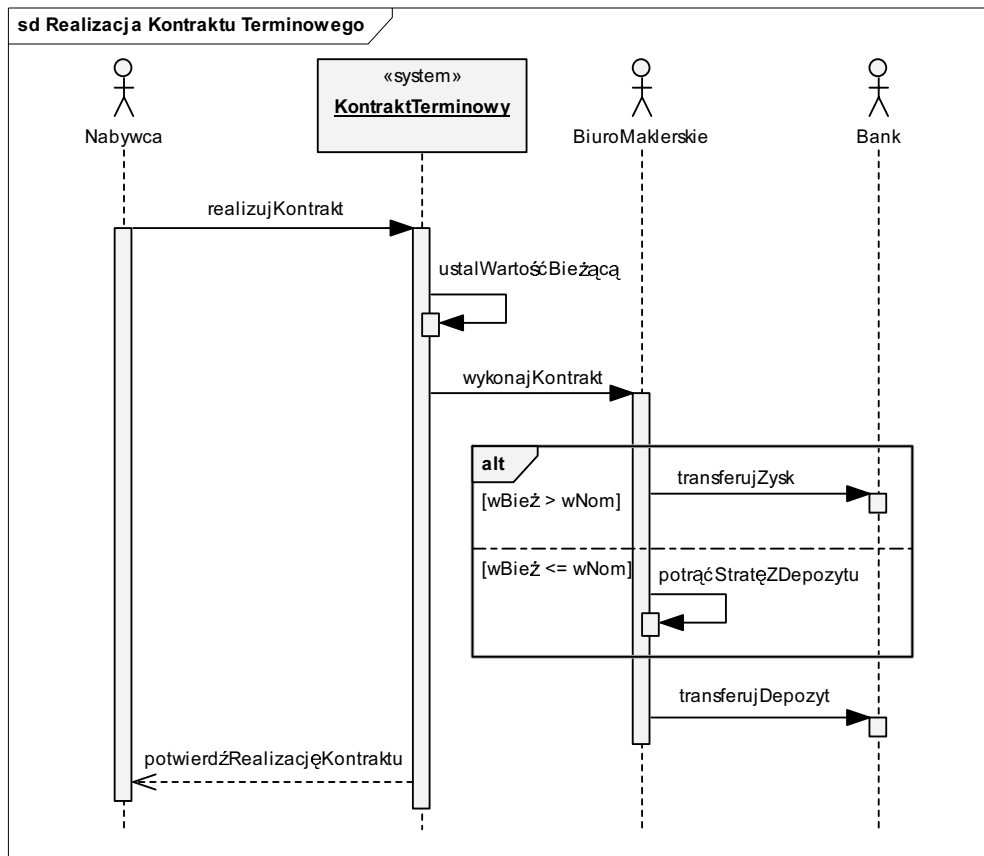
1. Wyjaśnij związek pomiędzy diagramami przypadków użycia a diagramami sekwencji.
2. Dlaczego diagramy sekwencji ściśle odzwierciedlają koncepcję obiektowego modelowania systemów informatycznych?
3. Uzasadnij potrzebę zróżnicowania poziomów uogólnienia diagramów sekwencji. Jakie grupy zawodowe powinny być zainteresowane tworzeniem i użytkowaniem diagramu konceptualnego?
4. Który z rodzajów diagramów sekwencji jest podstawową dokumentacją programistyczną? Dlaczego?

5. Jakie są uwarunkowania przejścia z diagramu implementacyjnego do wystąpieniowego?
6. Tabela 7.1 wskazuje, że poszczególne rodzaje diagramów sekwencji wykorzystują różny zakres kategorii pojęciowych odnoszących się do diagramów sekwencji. Wyjaśnij, które dokładnie kategorie pojęciowe stosowane są na diagramach poszczególnych rodzajów.
7. Wyjaśnij istotę poszczególnych kategorii pojęciowych konceptualnego diagramu sekwencji.
8. Które z kategorii pojęciowych diagramu sekwencji mają podstawowe znaczenie i dlaczego?
9. Jaki rodzaj klasyfikatorów najczęściej występuje w praktyce na diagramach sekwencji i dlaczego? Wskaż inne powszechnie używane na tych diagramach rodzaje klasyfikatorów.
10. Dlaczego klasyfikator jest konstrukcją abstrakcyjną?
11. Wskaż, jaka jest zależność pomiędzy komunikatem a operacją. W odniesieniu do rysunku 7.3 naszkicuj klasę reprezentującą *IBankomat*, uwzględniając operacje i przykładowe atrybuty.
12. Opisz za pomocą konceptualnego diagramu sekwencji interakcje w systemie obsługi wyścigów konnych. Uwzględnij obstawienie konkretnego konia przez gracza zawierającego zakład i wysokość stawki zakładu. Dołącz inne interakcje, które uznasz za istotne dla funkcjonowania systemu.
13. Opisz trzy sposoby graficznego oznaczania obiektów na diagramach sekwencji. Jakie informacje o klasie, której instancją jest dany obiekt, wiążą się z poszczególnymi sposobami oznaczania?
14. Jaka jest rola ośrodka sterowania? Jaki jest jego związek z linią życia?
15. Jak należy interpretować aktywację i dezaktywację?
16. Uzupełnij rysunek 7.3 o ośrodki sterowania. Zastanów się, na których liniach życia wystąpi więcej niż jeden ośrodek sterowania.
17. Jakie są powody wzbogacenia podstawowego zestawu pojęć na diagramach sekwencji o kategorie zaawansowane?
18. Nazwij poniższe typy komunikatów:



19. Które z komunikatów określić można jako kompletne?
20. Wyjaśnij różnicę pomiędzy komunikatami utraconymi a komunikatami znalezionymi.

21. Jako użytkownik portalu internetowego zilustruj poszczególne rodzaje komunikatów występujące w trakcie jego użytkowania. Skorzystaj z notacji zaprezentowanej na rysunkach 7.7 – 7.13.
22. Dlaczego umieszczanie przez projektanta komunikatów zwrotnych na diagramach sekwencji nie jest obligatoryjne?
23. Jakim stereotypem oznaczamy komunikat tworzący obiekt? Podaj przykład z zakresu użytkowania komunikatorów internetowych.
24. Jakie dwa oznaczenia wprowadzane są do diagramu sekwencji w celu poinformowania o zakończeniu istnienia danego obiektu?
25. Przedstaw składnię komunikatów warunkowych oraz komunikatów wykonywanych na zasadzie iteracji.
26. Podaj przykład warunku w odniesieniu do systemu obsługi dziekanatu. Zaproponuj komunikat z kilkoma warunkami.
27. Jakim mianem określa się wywołanie operacji przez instancję klasyfikatora, jeżeli ta operacja jest jednym z elementów składowych tej instancji?
28. Naskicuj implementacyjny diagram sekwencji dla przypadku użycia „Generuj listę studentów”. Funkcjonalność tego przypadku użycia pozwala prowadzącemu przedmiot wydrukować listę studentów, którzy zdali egzamin w pierwszym obowiązkowym terminie i uzyskali wynik powyżej 4.0.
29. Wyjaśnij pojęcie rozgałęzienia. Jakiego rodzaju rozgałęzienia można wyróżnić? W jakich sytuacjach się je stosuje?
30. W systemie stypendialnym uczelni przyznaje się stypendia w zależności od wysokości średniej ocen uzyskanej w danym roku akademickim. Opracuj diagram sekwencji z rozgałęzieniami opisujący ten system stypendialny dla twojego wydziału.
31. Wyjaśnij zależności pomiędzy kategoriami:
 - ♦ diagram sekwencji,
 - ♦ interakcja,
 - ♦ fragment wyodrębniony,
 - ♦ operand interakcji.
32. W jaki sposób identyfikowane są fragmenty wyodrębnione?
33. Podaj składnię nagłówka fragmentu wyodrębnionego.
34. Wymień operatory interakcji charakteryzujące fragmenty wyodrębnione. Zinterpretuj podstawowe reguły tworzenia i użytkowania operandów interakcji w odniesieniu do konkretnych operatorów.
35. W odniesieniu do których operatorów interakcji zasadne jest zastosowanie separatorów? Dlaczego?
36. Wskaż w niniejszym rozdziale rysunek ilustrujący sytuację analogiczną do przedstawionej na rysunku 7.40, lecz z wykorzystaniem innej, równoznacznej notacji. Oceń podobieństwa i różnice pomiędzy tymi notacjami.



Rysunek 7.40. Realizacja kontraktu terminowego

37. Opracuj trzy diagramy wyjaśniające zastosowanie operatora interakcji *opt* w branży telekomunikacyjnej.
38. Oceń, czy następujące sytuacje mogą być udokumentowane przez operatory interakcji *break*, *critical* czy też *neg*:
- ♦ wykrycie wirusa w systemie,
 - ♦ pojawienie się zagrożenia terrorystycznego na stadionie miejskim w kontekście systemu ratownictwa miejskiego,
 - ♦ zidentyfikowanie włamania do systemu bankowego.
39. Firma posiada system CRM, który pozwala inicjować i monitorować programy lojalnościowe. Aby wykorzystać tę funkcjonalność, należy, przyjmując kryterium rocznego obrotu, podzielić klientów firmy na trzy segmenty:
- ♦ powyżej miliona €,
 - ♦ pomiędzy 200 tys. a milionem €,
 - ♦ poniżej 200 tys. €.

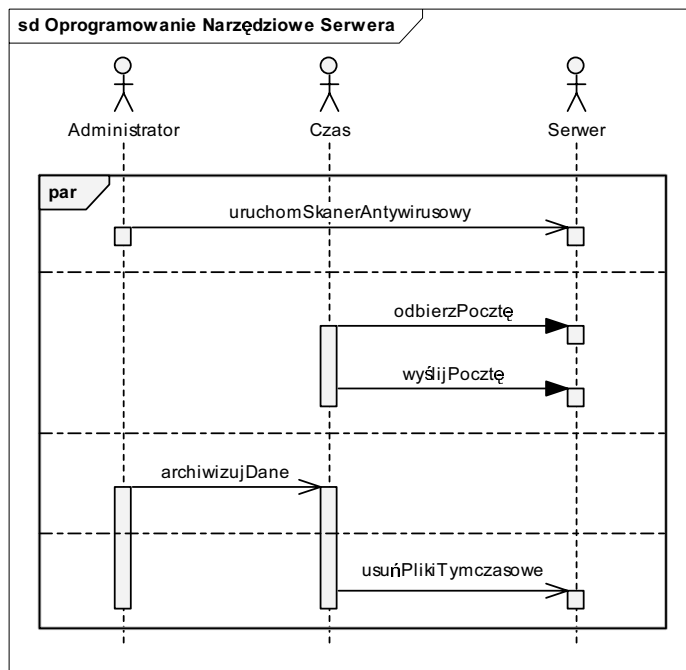
Do pełnego wdrożenia systemu niezbędne jest uporządkowanie klientów firmy w kolejności malejącej w ramach poszczególnych przedziałów wielkości obrotów. Na podstawie operatora interakcji *loop* opracuj diagram sekwencji realizujący powyższe założenia w systemie CRM.

40. Posługując się źródłami internetowymi, wskaż wyjątki standardowo obsługiwane w języku programowania VB .NET. Zobrazuj je, stosując odpowiedni operator interakcji.
41. Wykorzystując operator interakcji *neg*, przedstaw reakcje serwera poczty elektronicznej na przesłanie wiadomości z błędnie wpisanym adresem odbiorcy.
42. Duża biblioteka uniwersytecka użytkuje zautomatyzowany system obsługi wypożyczeń. W systemie występują następujące instancje klasyfikatorów:
 - ♦ aktor — *Czytelnik*;
 - ♦ interfejsy — *IZamówienieKsiążki*, *IKatalog*;
 - ♦ klasy — *Czytelnik*, *Egzemplarz*, *Książka*, *Wypożyczenie*.

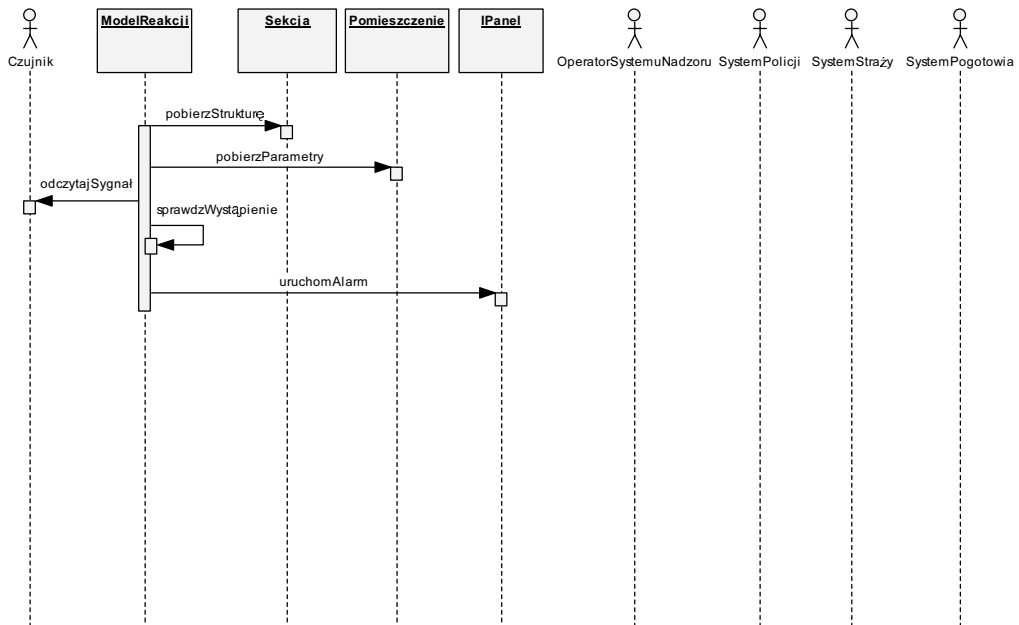
Wskaż interakcje wykonywane równolegle w tym systemie, stosując adekwatne operatory interakcji.

43. Wykorzystując dane publikowane przez Eurostat (<http://europa.eu.int/comm/eurostat/>), zaproponuj wskaźniki internetyzacji mierzące poziom rozwoju społeczeństwa wiedzy w regionie. Wykorzystując język UML, zdefiniuj te wskaźniki na diagramie sekwencji, wykorzystując adekwatny operator interakcji.
44. Wyjaśnij zależności pomiędzy operatorami interakcji *consider* oraz *ignore*. Zilustruj je na przykładzie oprogramowania komunikatora internetowego.
45. Które z interakcji zaprezentowanych na rysunku 7.17, przedstawiającym wewnętrzną rekrutację pracowników do konkretnych projektów, powinny być scharakteryzowane operatorem *seq*, a które *strict*?
46. Jak można określić zależność pomiędzy poszczególnymi operandami zamieszczonymi na rysunku 7.41? Sprawdź diagram pod kątem poprawności. Popraw ewentualne błędy.
47. Jakie jest uzasadnienie stosowania bram w diagramach sekwencji?
48. Czym uwarunkowane jest zróżnicowanie bram?
49. Wskaż i wyjaśnij sposoby nazywania bram. Zmodyfikuj diagram zaprezentowany na rysunku 7.31 poprzez dodanie bram i nadanie im nazw.
50. W jaki sposób rozwiązuje się problem dokumentowania złożonych diagramów sekwencji?
51. Czy poprawne jest umieszczanie przywoływanych wystąpień interakcji wewnątrz fragmentów wyodrębnionych?
52. Uzasadnij twierdzenie, że przywoływanie wystąpień interakcji pozwala na zstępujące modelowanie systemów informatycznych.

Rysunek 7.41.
Oprogramowanie narzędziowe serwera



- 53.** Wykorzystując instancje klasyfikatorów z rysunku 7.28, opracuj dwa przywoływane wystąpienia interakcji — pierwsze inicjowane za pośrednictwem komunikatu, a drugie uwzględniające czynnik czasu.
- 54.** Analityk modeluje interakcje w systemie nadzoru pomieszczeń kompleksu kinowego. Menedżerowie kompleksu kinowego chcą, aby w sytuacji zagrożenia projektowany system informował odpowiednie służby publiczne w celu umożliwienia szybkiej interwencji. Analityk ustalił kompletną listę instancji klasyfikatorów biorących udział w interakcji. Pięć pierwszych instancji klasyfikatorów to elementy składowe systemu wbudowanego, a więc zestawu sprzętu oprogramowanego. Firma specjalizująca się w monitorowaniu bezpieczeństwa pomieszczeń dostarczyła dokumentację, na podstawie której zidentyfikowano podstawowe operacje klas systemu wbudowanego. Analityk nie zidentyfikował dotychczas dalszych komunikatów i operacji. Na podstawie rysunku 7.42 zaproponuj dalsze interakcje i uzupełnij diagram o fragmenty wydzielone.



Rysunek 7.42. *System nadzoru pomieszczeń kompleksu kinowego*

Rozdział 8.

Diagram komunikacji

Znaczenie diagramów komunikacji

Diagramy interakcji ewoluowały wraz z rozwojem języka UML. W wersjach UML 1.x istniały tylko dwa rodzaje **diagramów interakcji**:

- ♦ diagramy sekwencji,
- ♦ diagramy współdziałania (kooperacji).

W wersji 2.0 standardu UML statyczny aspekt współdziałania instancji klasyfikatorów rozbudowano, tworząc nowy rodzaj diagramu — diagram struktur połączonych. W związku z tym dla zaakcentowania aspektu **komunikowania się** instancji klasyfikatorów zreorganizowano architekturę języka UML poprzez zmianę nazwy „diagramy współdziałania” na „diagramy komunikacji”. Po poprzedniku zachowały one semantykę oraz notację kategorii pojęciowych.

Diagramy komunikacji są stosowane do określania przepływów zdarzeń wyszczególnionych w scenariuszach adekwatnego przypadku użycia oraz zależnych wewnętrznych interakcjach systemu.



Diagram komunikacji jest rodzajem diagramu interakcji, specyfikującym strukturalne związki pomiędzy instancjami klasyfikatorów biorącymi udział w interakcji oraz wymianę komunikatów pomiędzy tymi instancjami.

A zatem diagram komunikacji zawiera dwa ściśle powiązane ze sobą składniki:

- ♦ **strukturalną organizację** klasyfikatorów wyrażoną asocjacjami;
- ♦ **interakcję** między ich instancjami, realizowaną za pośrednictwem komunikatów — w ujęciu graficznym przyporządkowanych do asocjacji łączących klasyfikatory.

Diagramy komunikacji są izomorficzne w stosunku do diagramów sekwencji, co z praktycznego punktu widzenia oznacza możliwość przekształcenia jednego rodzaju diagramu w drugi bez utraty informacji stanowiących o istocie interakcji. Funkcja transformacji diagramów sekwencji w diagram komunikacji i na odwrót jest automatycznie realizowana przez niektóre narzędzia CASE. Diagramy komunikacji mogą być również wykorzystywane w iteracyjnym uszczegóławianiu adekwatnych diagramów klas, obiektów i pozostałych diagramów interakcji.

Podstawowe kategorie pojęciowe oraz notacja graficzna

Podstawowymi **elementami** diagramów komunikacji są:

- ♦ klasyfikator,
- ♦ asocjacja,
- ♦ komunikat.

Pojęcia klasyfikatora oraz komunikatu wyjaśniono szczegółowo w rozdziale 7., natomiast pojęcie asocjacji w rozdziale 3. Pojęciowy **izomorfizm** diagramów komunikacji oraz diagramów sekwencji wyraża się podobieństwem występujących w nim kategorii pojęciowych. Na diagramach komunikacji można zaprezentować większość elementów charakterystycznych dla diagramów sekwencji (tabela 8.1).

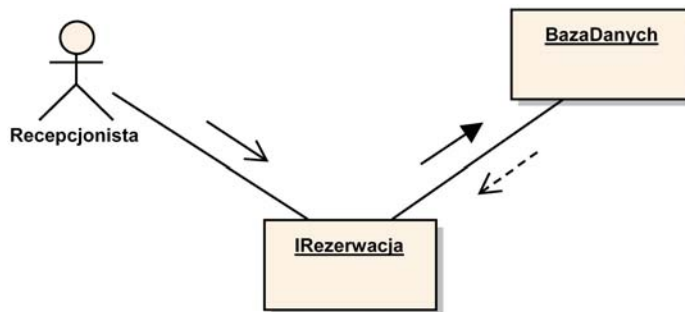
Tabela 8.1. Kategorie pojęciowe diagramów komunikacji i sekwencji

Kategoria pojęć	Pojęcie występujące na diagramach obydwu rodzajów	Pojęcie występujące tylko na diagramach sekwencji	Pojęcie występujące tylko na diagramach komunikacji
Podstawowe	Klasyfikator	Linia życia	Asocjacja
	Komunikat	Ośrodek sterowania	
Zaawansowane	Komunikat synchroniczny	Komunikat znaleziony	Obiekt wielokrotny
	Komunikat asynchroniczny	Komunikat utracony	
	Komunikat zwrotny	Rozgałęzienie	
	Komunikat opcjonalny	Fragmenty wyodrębnione i operatory interakcji	
	Komunikat oczekujący		
	Tworzenie obiektów	Brama	
	Niszczenie obiektów	Przywoływane wystąpienia interakcji	
	Warunek		
	Samowywoływanie		
	Iteracja		
	Zagnieżdżenie		

Ustanowienie jakiejkolwiek komunikacji pomiędzy instancjami **klasyfikatorów** wymaga uprzedniej specyfikacji związków między nimi. Po identyfikacji klasyfikatorów następuje ich powiązanie **asocjacjami**. Następnie do diagramu wprowadzane są **symbole komunikatów**, umieszczane równoległe do linii asocjacji (rysunek 8.1) lub według konwencji zastosowanej na rysunku 8.2. Grot symbolu komunikatu wskazuje kierunek przepływu komunikatu.

Rysunek 8.1.

Komunikaty w diagramach komunikacji



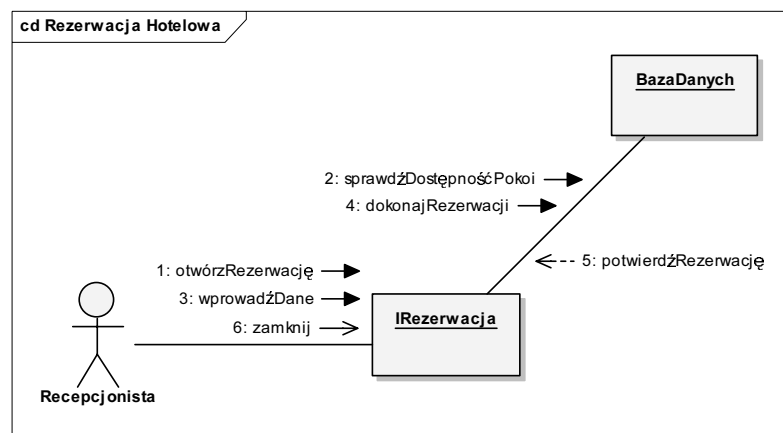
Numerowanie komunikatów

Na diagramach komunikacji kolejność wysyłania komunikatów nie wynika z ich naturalnego uporządkowania na osi czasu, tak jak na diagramach sekwencji. Konieczne jest zatem wprowadzenie **numeracji porządkowej**. Wyróżnić można następujące rodzaje numeracji:

- ♦ numerację prostą w postaci kolejnych liczb naturalnych (rysunek 8.2),

Rysunek 8.2.

Proste numerowanie komunikatów



- ♦ numerację kwalifikowaną — system klasyfikacji dziesiętnej Deweya.

Rysunek 8.2 opisuje komunikację obiektów podczas dokonywania rezerwacji hotelowej. Numeracja wskazuje, że pierwszy komunikat *otwórzRezerwację* przesyłany jest pomiędzy obiektami *Recepcjonista* oraz *IRezerwacja*. Następnie *IRezerwacja* komunikuje się z obiektem *BazaDanych*, przysyłając komunikat *sprawdźDostępnośćPokoji*.

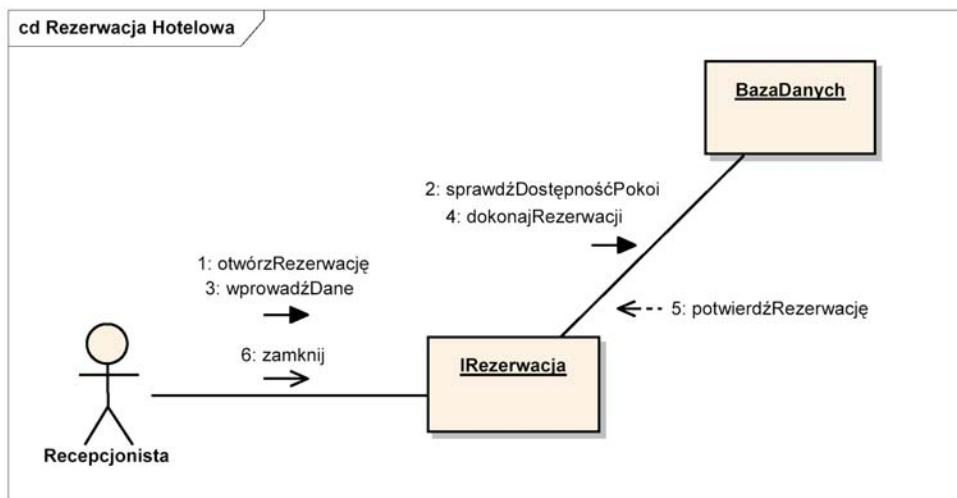
W dalszej kolejności przesyłane są komunikaty *sprawdźDane*, *dokonajRezerwacji*, *potwierdźRezerwację* oraz *zamknij*, sekwencyjnie według kolejnych numerów przypisanych do tych komunikatów.

Zasady wprowadzania komunikatów do diagramu

Podczas wprowadzania komunikatów do diagramów komunikacji należy kierować się następującymi **zasadami**:

1. Każdy komunikat umieszczany na diagramie komunikacji musi mieć numer porządkowy oraz nazwę wskazującą na operację klasyfikatora-odbiorcy. Numer porządkowy i nazwa są oddzielone dwukropkiem. Numer komunikatu może być rozszerzany o małe litery alfabetu łacińskiego w przypadku współbieżności.
2. Obligatoryjnymi cechami komunikatów na diagramie komunikacji jest wskazanie **rodzaju komunikatu** oraz kierunku jego przepływu. Jeżeli pomiędzy instancjami klasyfikatorów przesyłana jest większa liczba komunikatów, każdy z nich musi być opisany przez symbol komunikatu, a także przypisany mu numer porządkowy i nazwę.
3. W sytuacji, gdy diagram komunikacji jest przeładowany oznaczeniami komunikatów i przez to mało czytelny, można dokonać **zagregowanego** opisu diagramu poprzez grupowanie operacji z uwzględnieniem rodzaju komunikatu i kierunku przepływu. Nad takim symbolem-agregatem zapisuje się numery i nazwy odpowiednich komunikatów. Jest to czysto techniczny, porządkujący zabieg, nie zmieniający treści merytorycznej diagramu opisanego w punkcie drugim.

Na rysunku 8.3, zgodnie z wyżej wymienionymi zasadami, dokonano **agregacji** komunikatów tego samego rodzaju przesyłanych w tym samym kierunku.



Rysunek 8.3. Zasady umieszczania komunikatów

Zaawansowane składniki diagramu

Diagramy komunikacji stwarzają możliwość wzbogacenia ich zawartości merytorycznej poprzez włączenie koncepcji **zaawansowanych**. Należą do nich:

- ♦ izomorfizm diagramów sekwencji i komunikacji;
- ♦ zagnieżdżenie;
- ♦ poprzednik;
- ♦ współbieżność;
- ♦ obiekty wielokrotne;
- ♦ klasy aktywne;
- ♦ inne kategorie zaawansowane, takie jak warunki, tworzenie obiektów, niszczenie obiektów, samowywoływanie czy też iteracje.

Izomorfizm

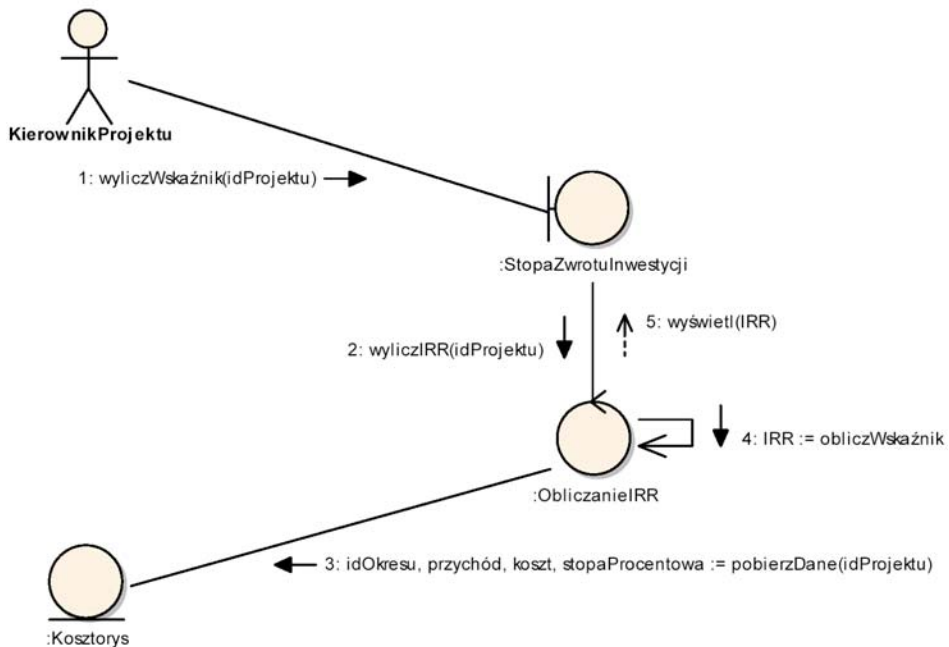
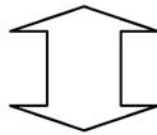
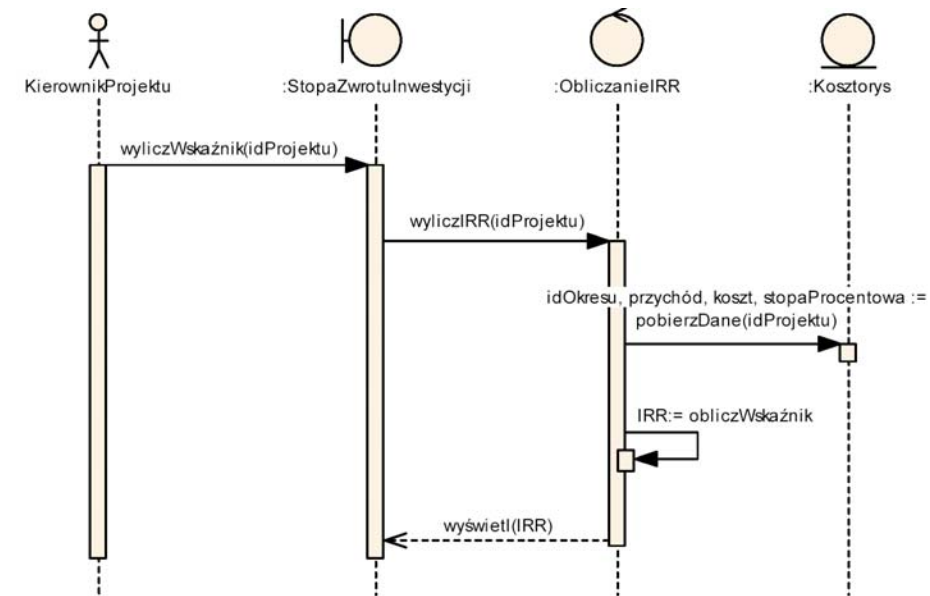
Jak wspomniano, diagramy sekwencji oraz diagramy komunikacji można uznać za izomorficzne, tzn. jednoznacznie wzajemnie przekształcalne. Przykłady izomorficznych, zastępowalnych diagramów sekwencji i komunikacji przedstawiono na rysunku 8.4. Dotyczy on przeprowadzenia studium wykonalności przedsięwzięcia w firmie. Jednym z jego elementów jest próba kwantyfikowania przyszłych nakładów oraz zysków. Uzyskanie tych danych pozwala kierownikowi projektu na wyliczenie tzw. wewnętrznej stopy zwrotu inicjowanego przedsięwzięcia. W tym celu *KierownikProjektu* wykorzystuje analityczny obiekt graniczny *StopaZwrotuInwestycji*, przesyłając do niego komunikat *wyliczWskaźnik* z parametrem *idProjektu*. Analityczny obiekt sterujący *ObliczanieIRR* po odebraniu od interfejsu komunikatu *wyliczIRR* wysyła komunikat *pobierzDane* do *Kosztorysu*. Wysłany komunikat powoduje przekazanie — jako wartości zwrotnych — atrybutów niezbędnych do zastosowania formuły *obliczWskaźnik*, wyliczającej wewnętrzną stopę zwrotu inwestycji. Komunikat zwrotny wywołujący operację *wyświetl* analitycznego obiektu granicznego *StopaZwrotuInwestycji* powoduje wyświetlenie wskaźnika IRR.

Zagnieżdżenie

Diagramy komunikacji to szczególnie użyteczna technika obrazowania **zagnieżdżeń** (ang. *nesting*) komunikatów w modelowanym systemie. Dokonuje się tego z wykorzystaniem notacji Deweya w numerowaniu komunikatów. I tak:



Zagnieżdżenie określa logicznie powiązany ciąg komunikatów wywołujących się wzajemnie w ustalonej kolejności.



Rysunek 8.4. Izomorfizm diagramów sekwencji i komunikacji

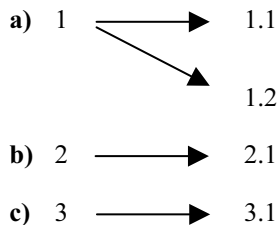
Kolejność wywoływania komunikatów w zagnieżdżeniu określona jest kolejnymi liczbami naturalnymi i pozycjami dziesiętnymi, tak jak w **systemie klasyfikacji dziesiętnej Deweya**. W praktyce programowania zagnieżdżenie jest równoznaczne z pojęciem wątku (ang. *thread*).



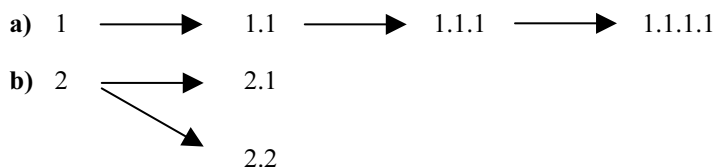
Wątek to pojedyncza ścieżka przepływu sterowania wykonywana przez program.

Głębokość zagnieżdżenia nie podlega ograniczeniom. W przypadku transformacji diagramu komunikacji w diagram sekwencji, kolejne zagnieżdżenia obrazowane są na diagramie docelowym poprzez wprowadzenie dodatkowych ośrodków sterowania na liniach życia odpowiednich instancji klasyfikatorów. Jeżeli na danej linii życia znajduje się już ośrodek (ośrodek) sterowania, fakt wystąpienia zagnieżdżenia można zaakcentować poprzez przesunięcie dodatkowego ośrodka sterowania w prawo. Notacja Deweya wnosi do diagramu dodatkową wartość semantyczną i wzbogaca dokumentację programistyczną danej aplikacji.

Tabela 8.2 wskazuje zagnieżdżenia wyrażone w systemie klasyfikacji dziesiętnej. Jak wynika z tabeli, wprowadzanie do diagramu kolejnych zagnieżdżeń jest zależne od logiki przedstawianej interakcji. W czwartym bloku tabeli zaznaczono na diagramie poszczególne wątki stanowiące sekwencje zagnieżdżonych komunikatów. Tak więc kolejność wzajemnego przywoływania komunikatów w wątkach tego diagramu jest następująca:

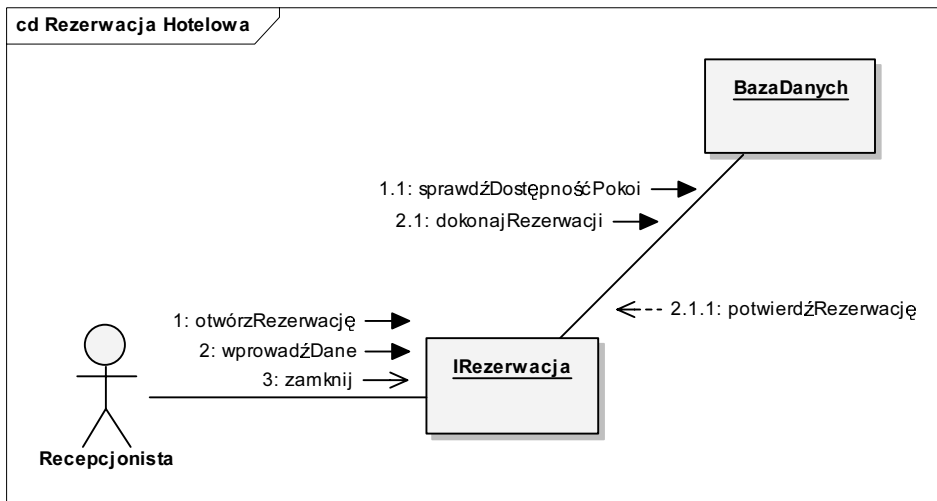
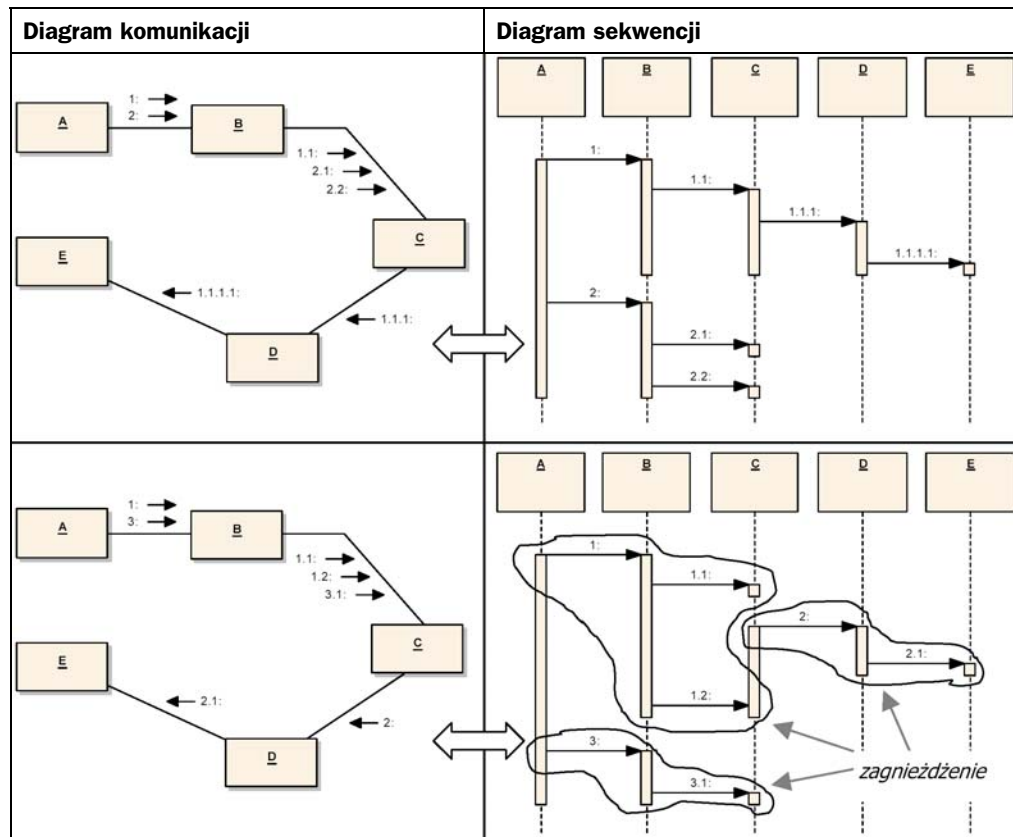


Z kolei w drugim z diagramów sekwencji zamieszczonych w tabeli wyodrębnić można dwa wątki:



Zasadę dokumentowania zagnieżdżeń na diagramach komunikacji przy zastosowaniu notacji Deweya ilustruje rysunek 8.5. Nawiązuje on do rysunku 8.2. I tak inicjowany przez *Recepcjonistę* komunikat *otwórzRezerwację* (1) powoduje wywołanie komunikatu zagnieżdżonego *sprawdźDostępnośćPokoi* (1.1). Analogicznie, operacja wywoływana przez komunikat *wprowadźDane* (2) inicjuje w konsekwencji operację *dokonajRezerwacji* (2.1), natomiast ta wywołuje operację *potwierdźRezerwację* (2.1.1).

Tabela 8.2. Zagnieżdżenia



Rysunek 8.5. Zagnieżdżenie komunikatów w notacji Deweya

Poprzednik

W opisie komunikatów istnieje możliwość uwzględnienia ich **poprzedników**. Interpretacja notacji prostej oraz systemu klasyfikacji dziesiętnej domyślnie wyznacza kolejność wysyłanych komunikatów. W drugiej z tych notacji numer komunikatu wywołującego znajduje się o jedną pozycję dziesiętną wyżej w hierarchii w stosunku do komunikatu wywołwanego. W związku z tym na ogół nie jest wymagane dodatkowe zaznaczanie poprzedników na diagramie komunikacji. Jednak w określonych sytuacjach należy jednoznacznie wskazać komunikat lub komunikaty, które muszą poprzedzać komunikat bieżący. Dokonuje się tego poprzez rozszerzenie składni komunikatu o numery komunikatów poprzedzających. Numery poprzedników w nazwie komunikatu są wówczas oddzielone symbolem ukośnika:

1.1, 3.3.2/ 4.4: realizujTransakcjęOnLine

Pełną specyfikację składni komunikatu zamieszczono w rozdziale 6.

Współbieżność

Opis komunikowania się instancji klasyfikatorów w systemie na ogół realizuje się sekwencyjnie, w postaci kolejno następujących po sobie komunikatów. Jednak interakcje systemów obejmują sytuacje, w których instancje klasyfikatorów mogą wykonywać jednocześnie wiele wątków i operacji. Sytuacje te wiążą się z cechą **współbieżności** (ang. *concurrency*).

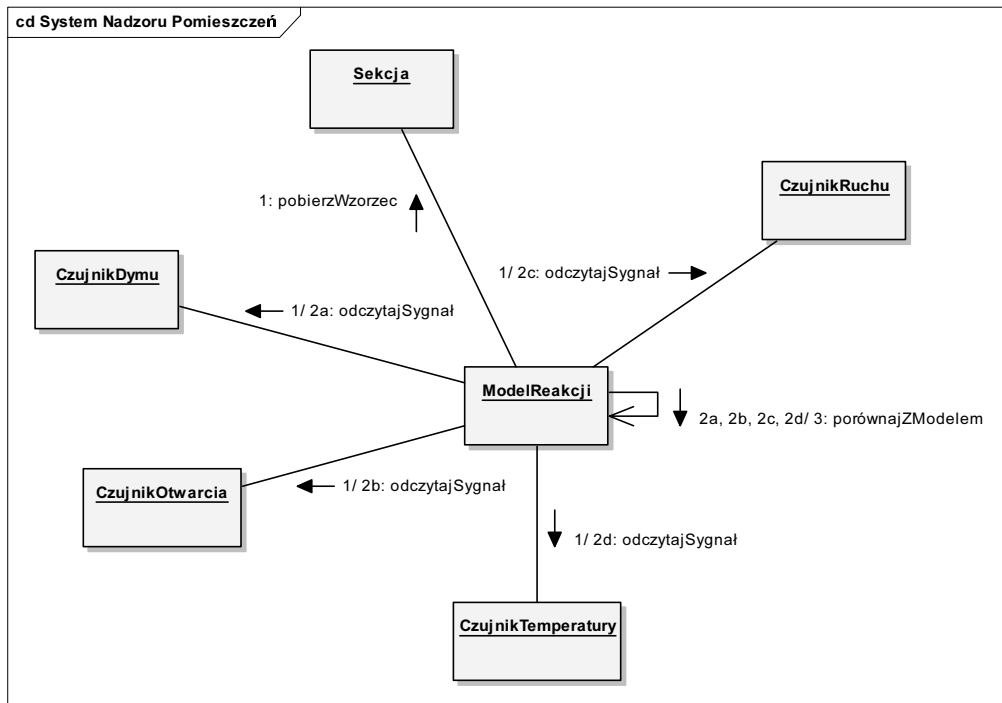


Współbieżność to wystąpienie dwóch lub więcej czynności w tym samym czasie, co pozwala na przeplatanie lub równoczesne wykonywanie wątków.

Cechę współbieżności na diagramach komunikacji zilustrowano na rysunku 8.6. W celu jednoznacznego zaznaczenia porządku wykonywania operacji przez kilka instancji klasyfikatorów funkcjonujących współbieżnie można w sposób jawny określić w składni komunikatu poprzednika (poprzedników) tego komunikatu.

Administratorzy kompleksów biurowych wykorzystują system nadzoru pomieszczeń, który monitoruje wystąpienia niepożądanych zdarzeń zagrażających bezpieczeństwu pracowników. Elementem sterującym systemem jest centrala alarmowa, czyli urządzenie oprogramowane. Do centrali podłączone są następujące czujniki:

- ♦ CzujnikDymu,
- ♦ CzujnikOtwarcia,
- ♦ CzujnikRuchu,
- ♦ CzujnikTemperatury.



Rysunek 8.6. Współbieżność oraz poprzednicy

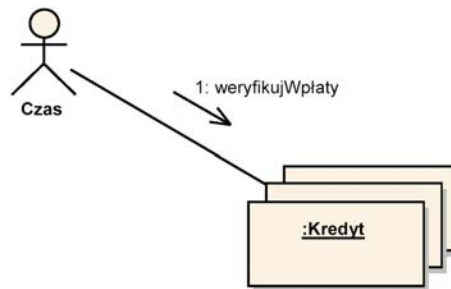
System nadzoru przechowuje w klasie *ModelReakcji* operacje, które są wykonywane w przypadku odebrania przez jakikolwiek czujnik sygnału alarmowego. W celu zachowania integralności monitorowanego obszaru *ModelReakcji* pobiera parametry kompleksu biurowego poprzez uruchomienie operacji *pobierzWzorzec* instancji klasyfikatora *Sekcja*. Następnie instancja klasyfikatora *ModelReakcji* wysyła komunikaty *odczytajSygnał* jednocześnie do wszystkich rodzajów czujników. Po odebraniu sygnału z czujników *ModelReakcji* porównuje go ze wzorcem. System nadzoru został tak zaprojektowany, aby wykonanie operacji *porównajZModelem* było możliwe dopiero po wykonaniu równoległych operacji *odczytaniaSygnału* przez wszystkie czujniki. Z kolei *odczytanieSygnału* jest wykonywane dopiero po *pobranieWzorca* sekcji kompleksu biurowego.

Obiekty wielokrotne

Pojedyncza instancja klasyfikatora może przysyłać komunikat do wszystkich instancji danej klasy. W tym celu może odwoływać się do konkretnych egzemplarzy obiektów. Wiązałoby się to jednak z koniecznością zaprezentowania każdego z tych odwołań na osobnym diagramie. Bardziej efektywnym sposobem jest wprowadzenie tzw. **obiektów wielokrotnych** (ang. *multiple objects*), będących symboliczną reprezentacją zbioru anonimowych obiektów, a więc obiektów z wyspecyfikowaną wyłącznie nazwą klasy. Dla

takich przypadków zarezerwowano odmienny sposób notacji obiektów. Obiekty te reprezentowane są przez nachodzące na siebie symbole obiektów, jak pokazano na rysunku 8.7.

Rysunek 8.7.
*Obiekty wielokrotne
na diagramach
komunikacji*



Na przedstawionym przykładzie aktor *Czas* inicjuje operację *weryfikujWpłaty*. Przesyła on w tym celu asynchroniczny komunikat do *Kredytu*. Zastosowanie obiektów wielokrotnych oznacza, że następuje weryfikacja wszystkich obiektów klasy *Kredyt*.

Odwoływanie się do wielokrotnych obiektów ma charakter **iteracyjny**, jednak zaznaczanie tego faktu w postaci znacznika iteracji (*) przy komunikacie w połączeniu z zastosowaniem notacji obiektów wielokrotnych jest nadmiarowe.

Klasy aktywne

W aspekcie występowania i dokumentowania w systemie interakcji klasyfikatory można podzielić na:

- ♦ pasywne,
- ♦ aktywne.

Aktywność jest ściśle uwarunkowana przez uczestnictwo w interakcjach. Każda instancja klasyfikatora uczestnicząca w interakcji odbiera lub wysyła komunikaty. Zazwyczaj jednak interakcja ta jest konsekwencją zlecenia klasyfikatorowi-odbiorcy wykonania określonych operacji przez klasyfikator-nadawcę. Klasyfikatory te nazywane są pasywnymi i dominują w aplikacjach biznesowych. Obok nich w języku UML występują klasyfikatory **aktywne** (ang. *active*), mogące zainicjować sterowanie w systemie. Największe znaczenie spośród klasyfikatorów aktywnych mają klasy aktywne.



Klasa aktywna to klasa, której instancje stanowią aktywne obiekty, mogące autonomicznie inicjować własne wątki i sterować wykonaniem tych wątków.

Klasa aktywna ma specyficzne oznaczenie. Ilustruje się ją na diagramie poprzez dodanie dwóch linii równoległych do pionowych krawędzi klasy.

Pełne wyrażenie dynamicznego aspektu klas aktywnych jest możliwe przy zastosowaniu diagramów komunikacji. Klasy aktywne mają instancje w postaci aktywnych obiektów. Mogą one **autonomicznie** inicjować wątki, operacje własne oraz operacje innych

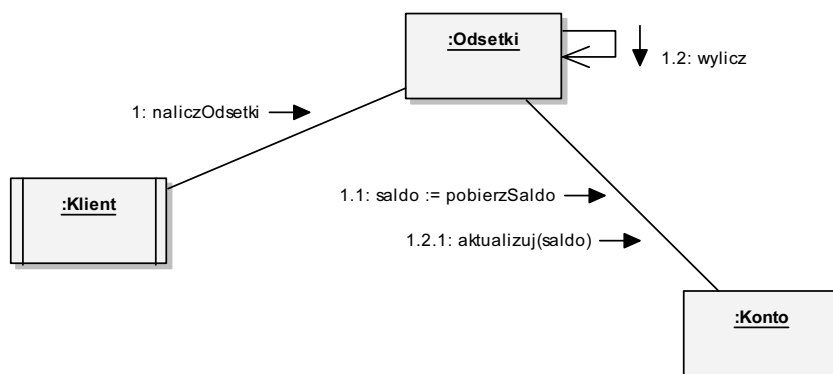
instancji klasyfikatorów. Naturalnie, klasy i obiekty aktywne mogą także realizować wątki i operacje wywołane przez inne instancje klasyfikatorów, w tym klasy aktywne. Podstawą identyfikowania klas aktywnych jest:

- ♦ zdolność danej klasy do inicjowania wątku poprzez wysłanie pierwszego komunikatu w ciągu komunikatów zagnieżdżonych,
- ♦ samoistne wysyłanie komunikatów w określonych odcinkach czasu,
- ♦ wykonywanie złożonych procedur obliczeniowych na podstawie danych przechowywanych przez obiekty innych klas.

Klasy aktywne, funkcjonując w trybie ciągłym, mają szczególne znaczenie w systemach czasu rzeczywistego. Przykładem obiektów aktywnych w tych systemach są programowe sterowniki urządzeń oraz czujniki. Innym przykładem obiektów aktywnych, często stosowanych podczas projektowania aplikacji biznesowych, są analityczne obiekty sterujące. Rolę klasy aktywnej na diagramie komunikacji przedstawia rysunek 8.8.

Rysunek 8.8.

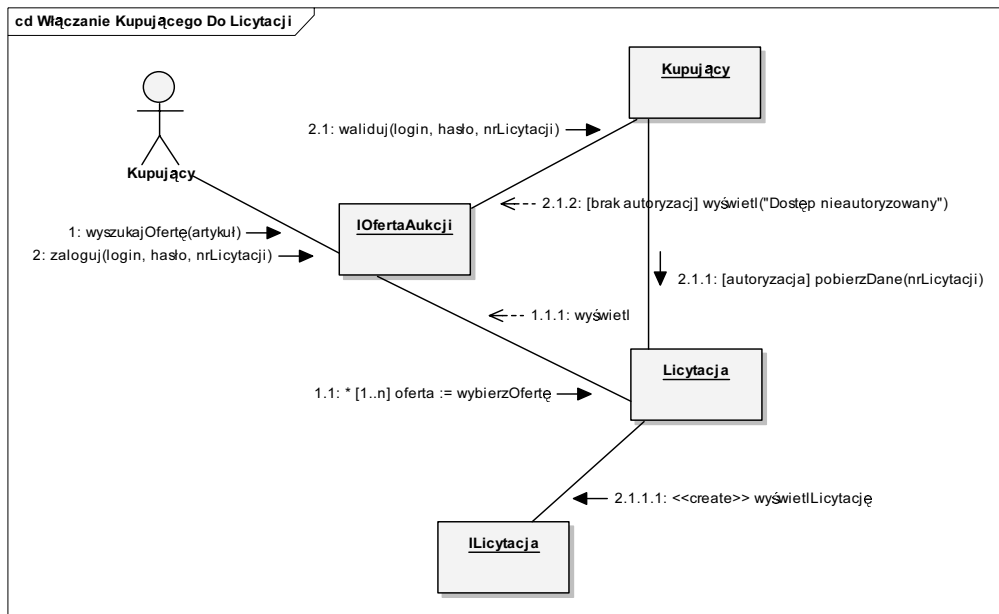
Obiekty klasy aktywnej w naliczaniu odsetek



Na życzenie *Klienta* bank dokonuje wyliczenia odsetek należnych z tytułu lokat. W tym celu w systemie bankowym przesyłany jest komunikat *naliczOdsetki* (1). W wyniku jego odebrania obiekt *Odsetki* wysyła do obiektu *Konto* komunikat *pobierzSaldo* (1.1) i dokonuje *wyliczenia* (1.2) wielkości odsetek na podstawie zwróconej wysokości salda. Po naliczeniu odsetek następuje *aktualizacja* salda. W kontekście opisanego przykładu *Klient* jest obiektem aktywnej klasy.

Inne kategorie zaawansowane

Jak zaznaczono w tabeli 8.1, w przypadku diagramów sekwencji i komunikacji wiele pojęć jest równoważnych. Z tego względu opis wspólnych dla obu rodzajów diagramów kategorii zaawansowanych, takich jak warunki czy iteracje, można odnieść bezpośrednio do konwencji przyjętych na diagramach sekwencji. Zależności te znajdują odzwierciedlenie na rysunku 8.9. Zaprezentowano na nim włączanie się klienta do licytacji w systemie aukcji internetowej.



Rysunek 8.9. Włączanie kupującego do licytacji

W opisywanej interakcji bierze udział pięć instancji klasyfikatorów:

- ♦ aktor *Kupujący*,
- ♦ *IOfertaAukcji*,
- ♦ *Licytacja*,
- ♦ obiekt *Kupujący*,
- ♦ *ILicytacja*.

Na pierwszy z dwóch wątków, inicjowany przez aktora *Kupujący*, składają się:

- ♦ komunikat *wyszukajOfertę*, przekazujący jako parametr konkretny artykuł do interfejsu systemu;
- ♦ wykonywany iteracyjnie komunikat *wybierzOfertę*, wskazujący wszystkie licytacje, w których przedmiotem sprzedaży jest poszukiwany artykuł;
- ♦ komunikat *wyświetl*, prezentujący na interfejsie wszystkie licytacje, których przedmiotem jest poszukiwany artykuł.

Drugi wątek jest związany z logowaniem do systemu. Podobnie jak miało to miejsce w przypadku wyszukiwania licytacji, jest on inicjowany przez aktora *Kupujący*. Wątek ten jest wykonywany w następującej sekwencji:

- ♦ komunikat *zaloguj*, przekazujący do strony głównej aukcji login, hasło *Kupującego* oraz *nrLicytacji* jako parametry;
- ♦ komunikat *waliduj*, który porównuje zgodność przekazanych parametrów z przechowywanymi w obiekcie klasy *Kupujący*;
- ♦ komunikat *pobierzDane* z parametrem *nrLicytacji* przesyłany do instancji klasyfikatora *Licytacja* tylko i wyłącznie w przypadku zidentyfikowania autoryzowanego użytkownika; wysłany komunikat dostarcza szczegółowych informacji o wybranej licytacji w postaci wartości zwrotnych; następnie obiekt klasy *Licytacja* wywołuje operację «create» *wyświetlLicytację*, przekazując dane licytacji do utworzonego interfejsu *ILicytacja*;
- ♦ w alternatywnej sytuacji system nie pozwala *Kupującemu* na uczestnictwo w licytacji, traktując go jako użytkownika nieautoryzowanego; obiekt *Kupujący* wywołuje operację *wyświetl* z parametrem „*Dostęp nieautoryzowany*”, która powoduje wyświetlenie na interfejsie informacji o niedopuszczeniu *Kupującego* do interesującej go licytacji.

Proces tworzenia diagramu komunikacji

Analogicznie jak w przypadku diagramów sekwencji, punktem wyjścia w tworzeniu diagramu komunikacji jest przypadek użycia oraz jego scenariusze. W tworzeniu opisywanego diagramu mają miejsce następujące **etapy**:

1. identyfikacja klasyfikatorów;
2. powiązanie zidentyfikowanych klasyfikatorów asocjacjami;
3. identyfikacja i nadanie nazw komunikatom przesyłanym pomiędzy instancjami klasyfikatorów;
4. określenie typów nazwanych komunikatów;
5. numerowanie komunikatów z wykorzystaniem prostej numeracji;
6. identyfikowanie klas aktywnych i wątków przy użyciu numeracji kwalifikowanej;
7. wzbogacenie opisu diagramu z zastosowaniem zaawansowanych kategorii, takich jak współbieżność, obiekty wielokrotne, warunki, rozgałęzienia oraz iteracje;
8. opcjonalne dokonanie agregacji komunikatów.

Powyższa procedura w odniesieniu do diagramów komunikacji ma charakter autonomiczny. Można jednak generować je **automatycznie** z wykorzystaniem narzędzi CASE na podstawie adekwatnych diagramów sekwencji. Sposób ten funkcjonuje także w przeciwną stronę na zasadzie konwersji diagramów komunikacji na diagramy sekwencji. Należy jednak zaznaczyć, że istnieje tu potencjalne zagrożenie utraty części informacji w związku ze wzbogaceniem diagramu sekwencji w wersji 2.0 języka UML o takie kategorie modelowania jak fragmenty wyodrębnione oraz przywoływane fragmenty

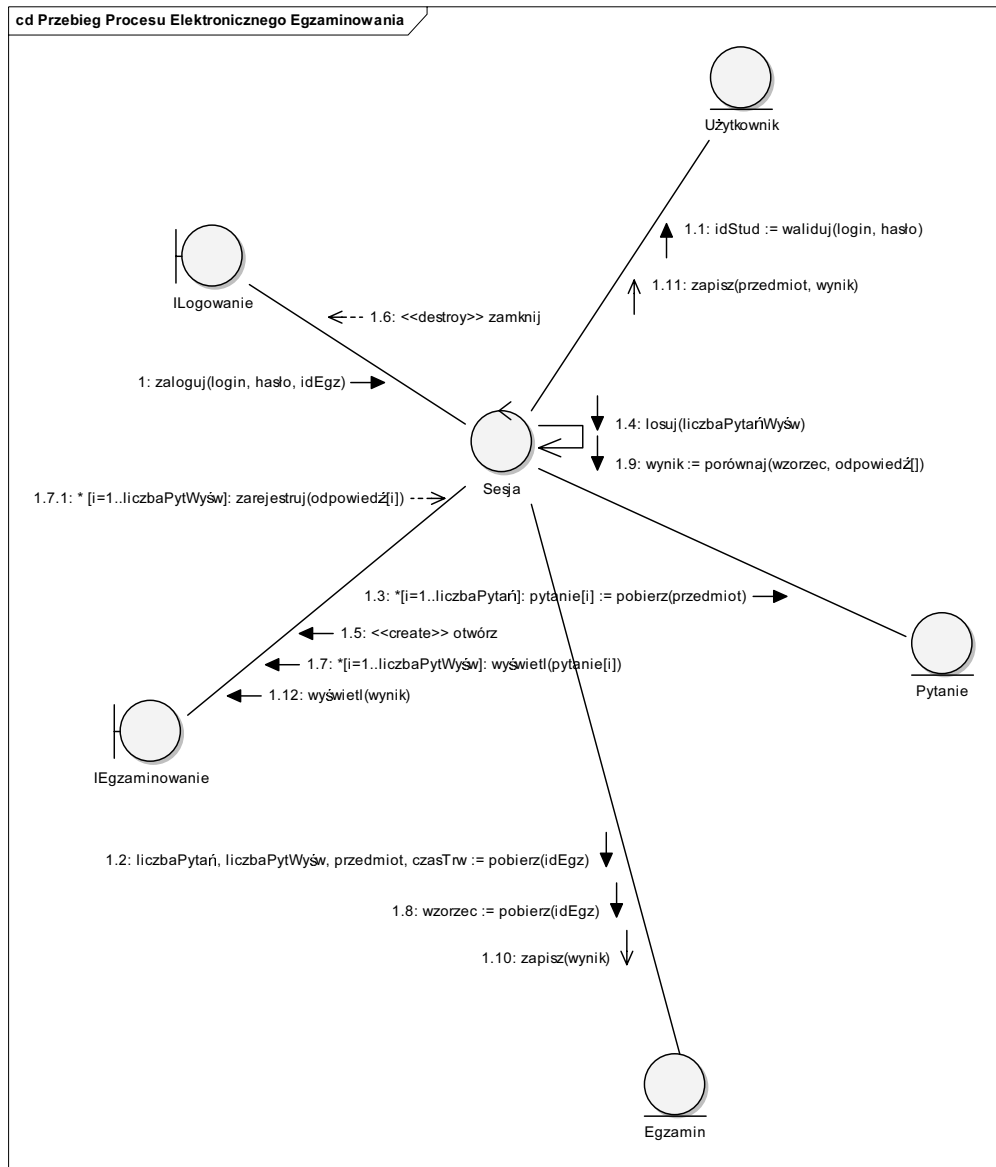
interakcji; przedstawienie znaczenia tych elementów z wykorzystaniem kategorii modelowania diagramów komunikacji jest technicznie wykonalne, chociaż kłopotliwe i wymagające wykorzystania w szerokim zakresie składni komunikatu. Różnice kategorii pojęciowych diagramów komunikacji i sekwencji określa szczegółowo tabela 8.1. Należy ponadto rozważyć właściwości i parametry poszczególnych narzędzi CASE dokonujących tych transformacji.

Studium diagramu komunikacji

Niniejsze studium charakteryzuje przebieg procesu egzaminowania studenta w elektronicznym systemie egzaminacyjnym. Opiera się na podstawowych i zaawansowanych elementach diagramów komunikacji zaprezentowanych odpowiednio w punktach 8.2 oraz 8.3 w tym rozdziale.

Jak pokazano na rysunku 8.10, w procedurze egzaminowania studenta wyróżniono następujące, ściśle związane ze sobą kroki:

- ♦ pobranie z formatek interfejsu loginu, hasła oraz identyfikatora egzaminu, wpisanych przez osobę egzaminowaną;
- ♦ weryfikacja danych niezbędnych do otwarcia sesji, połączona ze zwróceniem identyfikatora studenta;
- ♦ pobranie parametrów egzaminu, wyspecyfikowanych przez egzaminatora w ramach innego przypadku użycia;
- ♦ iteracyjne pobranie puli pytań przewidzianych dla danego przedmiotu;
- ♦ wylosowanie z ogólnej puli pytań wskazanej przez egzaminatora ustalonej liczby pytań, z których składać się będzie konkretny egzamin;
- ♦ wyświetlenie ekranu egzaminowania;
- ♦ zamknięcie ekranu logowania;
- ♦ iteracyjne wyświetlanie na interfejsie kolejnych pytań;
- ♦ zarejestrowanie zestawu wskazanych przez egzaminowanego odpowiedzi odnoszących się do konkretnych pytań;
- ♦ pobranie wzorca odpowiedzi z bazy danych;
- ♦ porównanie odpowiedzi ze wzorcem;
- ♦ fizyczne zapisanie zarejestrowanych wyników egzaminu;
- ♦ odnotowanie wyniku egzaminu w profilu użytkownika;
- ♦ wyświetlenie uprzednio zapisanych wyników na interfejsie.



Rysunek 8.10. Egzaminowanie studenta w systemie elektronicznego egzaminowania

Operacja *waliduj* (1.1) wywoływana jest przez operację *zaloguj* (1). Kolejne komunikaty, zgodnie z systemem klasyfikacji dziesiętnej Deweya, również są zależne od operacji *zaloguj*. Głębokość zagnieżdżenia nie podlega ograniczeniom — komunikat *wyświetl* (1.7) ma dalsze zagnieżdżenie w postaci komunikatu *zarejestruj* noszącego numer 1.7.1. W praktyce proces egzaminowania z punktu widzenia studenta kończy się na *wyświetleniu* wyników. Zarządzanie egzaminem realizowane jest w ramach odrębnego przypadku użycia.

Podstawowe pojęcia

Agregacja komunikatów	Obiekt
Asocjacja	Niszczenie
Diagramy interakcji	Tworzenie
Diagram harmonogramowania	Wielokrotny
Diagram komunikacji	Numeracja komunikatów
Definicja	Kwalifikowana
Proces tworzenia	Prosta
Diagram sekwencji	System klasyfikacji dziesiętnej Deweya
Diagram sterowania interakcją	Poprzednik
Iteracja	Samowywołanie
Izomorfizm	Symbol komunikatu
Klasa	Warunek
Aktywna	Wątek
Pasywna	Współbieżność
Klasyfikator	Zagnieżdżenie
Aktywny	
Pasywny	
Komunikat	
Asynchroniczny	
Oczekujący	
Opcjonalny	
Synchroniczny	
Zwrotny	

Pytania i zadania

1. Zinterpretuj zmiany dokonane w zakresie diagramów interakcji w wersji 2.0 standardu UML w stosunku do wersji 1.x.
2. Diagramem jakiego rodzaju można zastąpić diagram komunikacji, aby nie nastąpiła istotna utrata treści przekazywanej przez diagram?

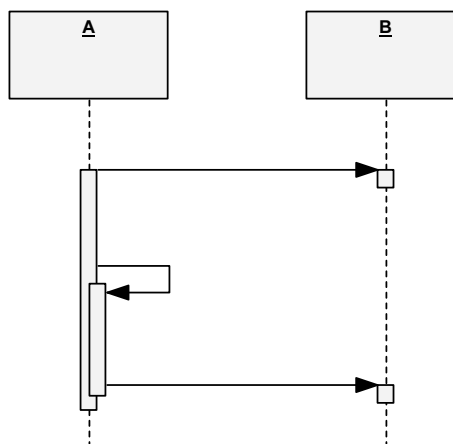
3. Jakie aspekty interakcji obiektów podkreślane są na diagramach komunikacji?
4. Strukturalna organizacja instancji klasyfikatorów i komunikacja między nimi to ściśle powiązane składniki diagramu komunikacji. W jaki sposób pojęcia podstawowe diagramu komunikacji powiązane są z wymienionymi składnikami?
5. Zdefiniuj pojęcie izomorfizmu diagramów komunikacji i sekwencji.
6. Uzupełnij tabelę 8.3, wpisując odpowiedniki pojęć, o ile istnieją.

Tabela 8.3. *Diagram komunikacji oraz sekwencji — odpowiedniki pojęć*

Diagram sekwencji	Diagram komunikacji
Linia życia	Komunikat synchroniczny
Komunikat utracony	
Niszczenie obiektów	
Rozgałęzienie	Warunek
Brama	Zagnieżdżenie
Fragment wyodrębniony	

7. W jaki sposób na diagramach komunikacji przedstawiane jest przesłanie komunikatu?
8. Wymień sposoby przedstawiania kolejności przesyłania komunikatów na diagramach komunikacji. Jakie znaczenie ma system klasyfikacji dziesiętnej Deweya?
9. Na podstawie sekwencyjnej zależności pomiędzy instancjami klasyfikatorów *A* i *B* zaproponuj i odpowiednio ponumeruj komunikaty na diagramie pokazanym na rysunku 8.11 i na analogicznym diagramie komunikacji.

Rysunek 8.11.
Interakcje pomiędzy instancjami klasyfikatorów



10. Podaj różnicę pomiędzy numeracją prostą a kwalifikowaną.
11. W jakich przypadkach należy poszerzyć numer komunikatu o małe litery alfabetu łacińskiego?
12. Na czym polega agregowanie opisu komunikatów w diagramie?
13. Wskaż zależności pomiędzy kategoriami:
 - ♦ zagnieżdżenie,
 - ♦ wątek,
 - ♦ ośrodek sterowania.
14. Zaznacz wątki na drugim z przedstawionych w tabeli 8.2 diagramów sekwencji.
15. Wyjaśnij zasadność stosowania poprzedników. Czy wprowadzenie do diagramu poprzedników może mieć związek ze współbieżnością?
16. Przedstaw sposób oznaczania faktu wysłania przez instancję klasyfikatora komunikatu do wielu obiektów danej klasy.
17. Scharakteryzuj pojęcie klasy aktywnej. Wyjaśnij związki pomiędzy klasami aktywnymi, pasywnymi oraz interakcją.
18. W jaki sposób dokonasz identyfikacji klas aktywnych w opisie systemu?
19. Zaproponuj pełną specyfikację komunikatów dla diagramu komunikacji zamieszczonego na rysunku 8.10. Dokonaj szczegółowego opisu zaprezentowanej na nim sytuacji.
20. Przedstaw i uzasadnij kolejność etapów w procesie tworzenia diagramów komunikacji.
21. Stwórz diagram komunikacji będący odpowiednikiem diagramu opisującego sprawdzanie stanu konta za pośrednictwem bankomatu, przedstawionego w poprzednim rozdziale na rysunku 7.3.

Rozdział 9.

Diagram harmonogramowania

Znaczenie diagramów harmonogramowania

Analizowane i projektowane interakcje w systemie realizowane są w określonym **czasie**. Spełnienie kryterium czasu przez interakcję jest ważnym czynnikiem oceny tych systemów — zwłaszcza systemów wbudowanych i systemów czasu rzeczywistego. Wychodząc naprzeciw tym oczekiwaniom, zaproponowano diagramy harmonogramowania.



Diagram harmonogramowania jest rodzajem diagramu interakcji, reprezentującym na osi czasu zmiany dopuszczalnych stanów, jakie może przyjmować instancja klasyfikatora uczestnicząca w interakcji.

Diagramy te stosuje się w celu sporządzenia **harmonogramów** interakcji, a więc specyfikacji interakcji instancji klasyfikatorów w aspekcie zmian czasu trwania ich stanów. Punktem wyjścia tych diagramów są podstawowe kategorie diagramów sekwencji oraz maszyn stanowych. Terminowa realizacja interakcji wymaga niekiedy wielkiej dokładności czasowej. W związku z tym na diagramach harmonogramowania można przedstawiać kolejność występowania stanów instancji klasyfikatorów oraz czas ich trwania. Wprowadzenie diagramów harmonogramowania w UML 2.0 jest istotną zmianą w stosunku do poprzednich wersji. Ich tworzenie i użytkowanie jest szczególnie zalecane w systemach o rozbudowanej dynamice.

Podstawowe kategorie pojęciowe oraz notacja graficzna

Diagramy harmonogramowania dokumentują aspekt czasu interakcji. Stąd w diagramie harmonogramowania na osi poziomej zaznacza się **skalę czasu** w postaci ustalonych odcinków. Natomiast na osi pionowej przedstawia się poszczególne instancje klasyfikatorów biorące udział w interakcji, a przy każdej z nich jej stany. Z zasady diagramy harmonogramowania tworzy się po opracowaniu diagramów sekwencji lub komunikacji. Na diagramach tych bowiem precyzyjnie wskazane są instancje klasyfikatorów występujące w systemie, a wyszczególnienie linii życia, ośrodków sterowania i komunikatów ułatwia ustalenie jednostek na skali czasu. W ten sposób osiąga się harmonizację interakcji pomiędzy instancjami klasyfikatorów w układzie czasowym. Akceptowalne jest również autonomiczne podejście do tworzenia wspomnianych diagramów, bez uwzględniania powiązanych diagramów sekwencji czy komunikacji, na podstawie innych posiadanych specyfikacji systemu.

W podstawowej postaci diagramu harmonogramowania ilustrowane są jego podstawowe **kategorie pojęciowe** (por. rysunek 9.1):

- ♦ klasyfikator,
- ♦ nazwa stanu,
- ♦ linia zmiany stanów instancji klasyfikatora.

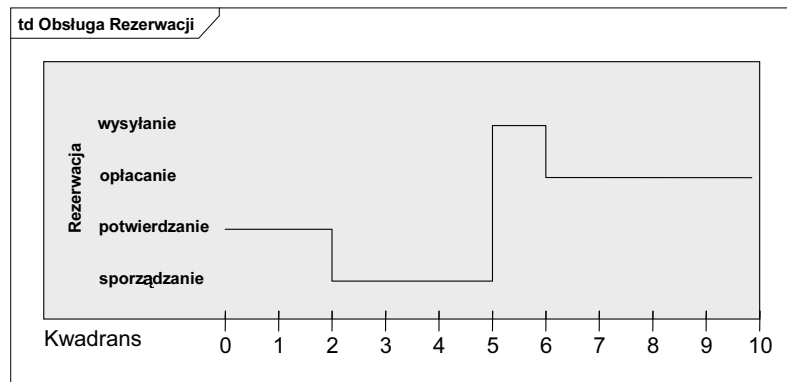
Pojęcie klasyfikatora wprowadzono i omówiono w rozdziale 1., podczas gdy powszechnie stosowane rodzaje klasyfikatorów języka UML 2 wskazano w rozdziale 7.

Lista możliwych stanów jest specyficzna dla każdej instancji klasyfikatora, jednak można wyróżnić kilka typowych **stanów** (ang. *states*), takich jak:

- ♦ bezczynność,
- ♦ czuwanie,
- ♦ oczekiwanie,
- ♦ wykonywanie,
- ♦ obliczanie.

Linia zmiany stanów (ang. *timeline*) może przedstawiać stany instancji klasyfikatora lub określonej, mierzalnej zmiennej — np. natężenia hałasu lub poziomów wybranego indeksu giełdowego. Diagram harmonogramowania w odniesieniu do instancji klasyfikatora *Rezerwacja* ze stanami *sporządzanie*, *potwierdzanie*, *opłacanie* oraz *wysyłanie* przedstawia rysunek 9.1.

Rysunek 9.1.
Diagram
harmonogramowania
dla obiektu klasy
Rezerwacja



Zaawansowane składniki diagramu

Przedstawiony na rysunku 9.1 diagram harmonogramowania jest użyteczny w dokumentowaniu stosunkowo prostych interakcji. Istnieje jednak możliwość jego rozszerzenia o szereg **zaawansowanych kategorii**, takich jak:

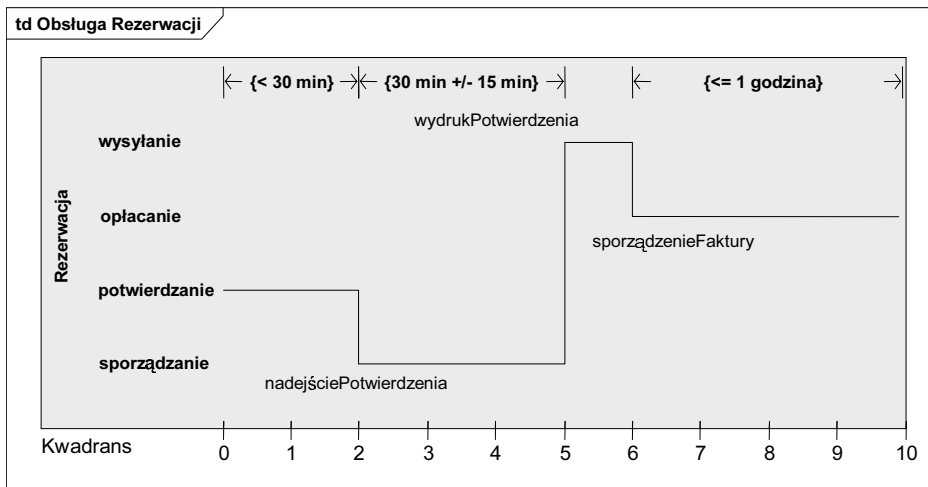
- ♦ zdarzenia,
- ♦ ograniczenia czasowe,
- ♦ alternatywne sposoby prezentacji stanów,
- ♦ harmonizacja linii zmiany stanów dla kilku instancji klasyfikatorów biorących udział w interakcji,
- ♦ przesyłanie komunikatów.

Zdarzenia i ograniczenia czasowe

Załamanie linii zmiany stanów instancji klasyfikatora oznacza wystąpienie **zdarzenia** powodującego zainicjowanie nowego stanu tej instancji. Zdarzenia w harmonogramie rezerwacji przedstawia rysunek 9.2. Sprecyzowano na nim stany instancji klasyfikatora *Rezerwacja* i czas ich trwania poprzez określenie **ograniczeń czasowych** (ang. *constraints*) w odniesieniu do stanów:

- ♦ *potwierdzanie* — {<30 min},
- ♦ *sporządzanie* — {30 min +/- 15 min},
- ♦ *opłacanie* — {<=1 godzina}.

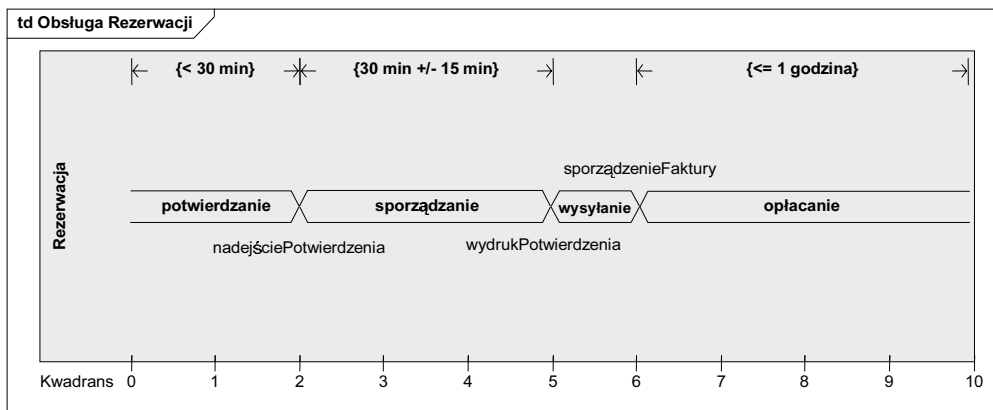
Jeśli skala czasu w sposób jednoznaczny określa długość trwania stanu, nie zaznacza się ograniczeń czasowych, co zilustrowano na rysunku 9.2 dla stanu *wysyłanie*.



Rysunek 9.2. Diagram harmonogramowania ze zdarzeniami i ograniczeniami czasowymi

Alternatywne sposoby prezentacji stanów

Poza podstawową formą prezentacji diagramów harmonogramowania istnieje **alternatywna** konwencja ich dokumentowania. Obie notacje mogą być stosowane zamiennie i wzajemnie przekształcane. Diagram harmonogramowania utworzony z wykorzystaniem alternatywnej konwencji, będący odpowiednikiem rysunku 9.2, zaprezentowano na rysunku 9.3.

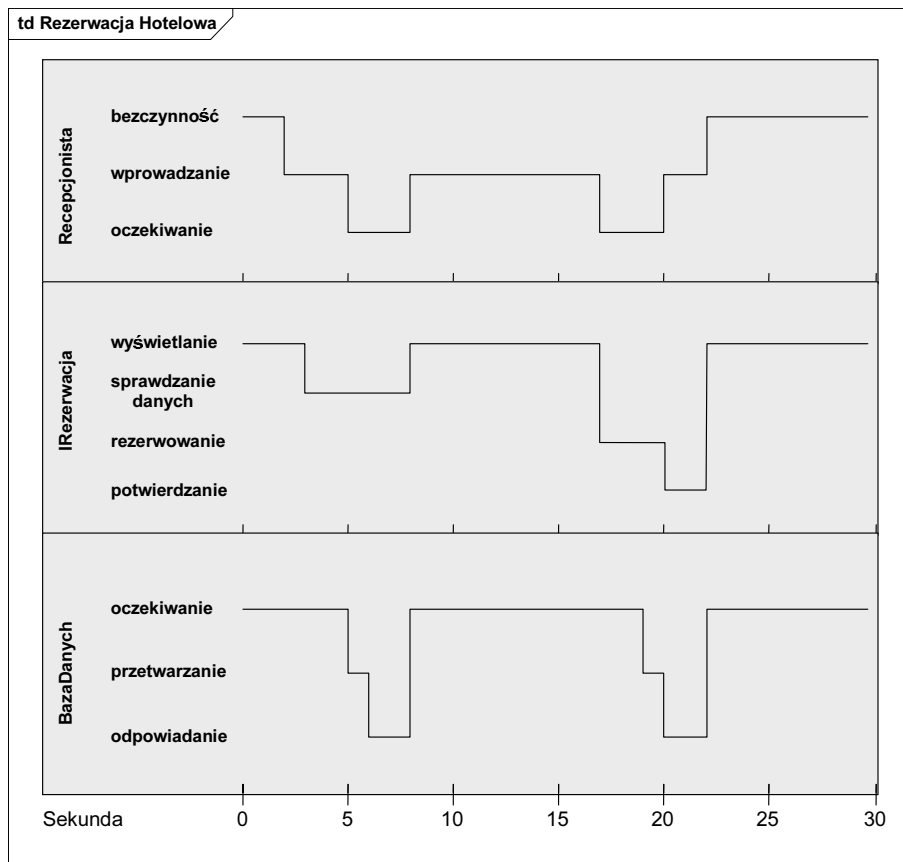


Rysunek 9.3. Alternatywna notacja diagramów harmonogramowania

Harmonizacja linii zmiany stanów

W interakcji w praktyce uczestniczy kilka, kilkanaście, a w niektórych sytuacjach nawet więcej instancji klasyfikatorów. Każda z nich może przyjmować określone stany. Diagramy harmonogramowania umożliwiają przedstawienie interakcji w pełnym

wymiarze, tzn. ze wszystkimi współpracującymi instancjami klasyfikatorów w horyzoncie czasowym harmonogramu. W tak opracowanym diagramie harmonogramowania osiąga się nie tylko specyfikację zmian stanu instancji klasyfikatora. Umożliwia on również przedstawienie interakcji instancji klasyfikatorów, które biorą udział w tej interakcji w każdym momencie okresu, dla którego jest opracowany harmonogram. Stwarza to możliwość **harmonizacji** współdziałania instancji klasyfikatorów w czasie. Harmonizacja poszczególnych instancji klasyfikatorów i ich stanów na osi czasu może być właściwie oddana wyłącznie z wykorzystaniem podstawowej notacji. Diagram na rysunku 9.4 jest przykładem harmonizacji instancji klasyfikatorów. Przedstawia przypadek rejestrowania rezerwacji w bazie danych hotelu.



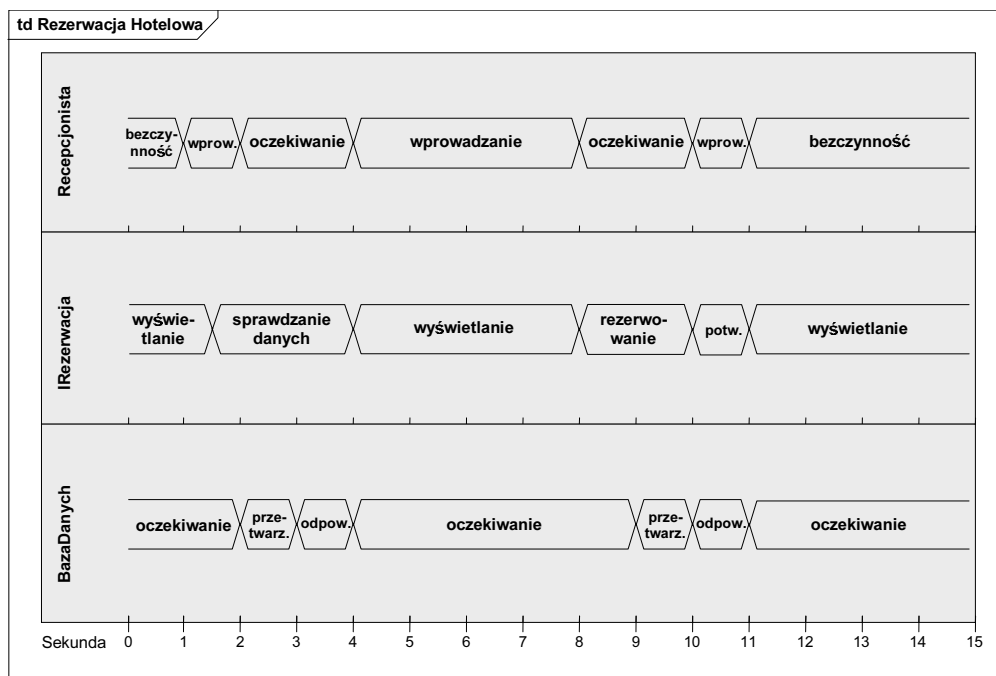
Rysunek 9.4. Harmonogramowanie rejestrowania rezerwacji hotelowej

Przypadek rejestrowania rezerwacji w bazie danych hotelu jest elementem scenariusza wyspecyfikowanego na diagramie sekwencji przedstawionym na rysunku 7.5. W przypadku tym występują trzy instancje klasyfikatorów:

- ♦ *Recepcjonista*,
- ♦ *IRezerwacja*,
- ♦ *BazaDanych*.

Na wstępie instancje te są odpowiednio w stanach: *bezczynności*, *wyświetlania* oraz *oczekiwania*. W wyniku wystąpienia konkretnego zdarzenia, tj. przyjęcia zgłoszenia klienta, recepcjonista otwiera rezerwację. Posługuje się w tym celu wyświetlaną na ekranie formatką *IREzerwacja*. *Recepcjonista* wprowadza dane, których kompletność oraz poprawność formalna jest sprawdzana przez *IREzerwację*. Następnie realizowana jest operacja przetwarzania danych — fakt dokonania rezerwacji pokoju albo pokoi jest odnotowywany w *BazieDanych*. W ramach *IREzerwacji* wyświetlane jest potwierdzenie dokonania rezerwacji. W efekcie zakończenia realizacji całej transakcji poszczególne instancje klasyfikatorów przechodzą do stanów wyjściowych.

Prezentowanie stanów kilku instancji klasyfikatorów w systemie jest możliwe także z wykorzystaniem opisanej notacji alternatywnej. Notacja ta pozwala na specyfikację stanów poszczególnych instancji względem czasu trwania tych stanów. Nie zawiera ona jednak elementu harmonizacji poszczególnych instancji klasyfikatorów i ich stanów na osi czasu (rysunek 9.5).



Rysunek 9.5. Rejestrowanie rezerwacji w bazie danych hotelu — notacja alternatywna

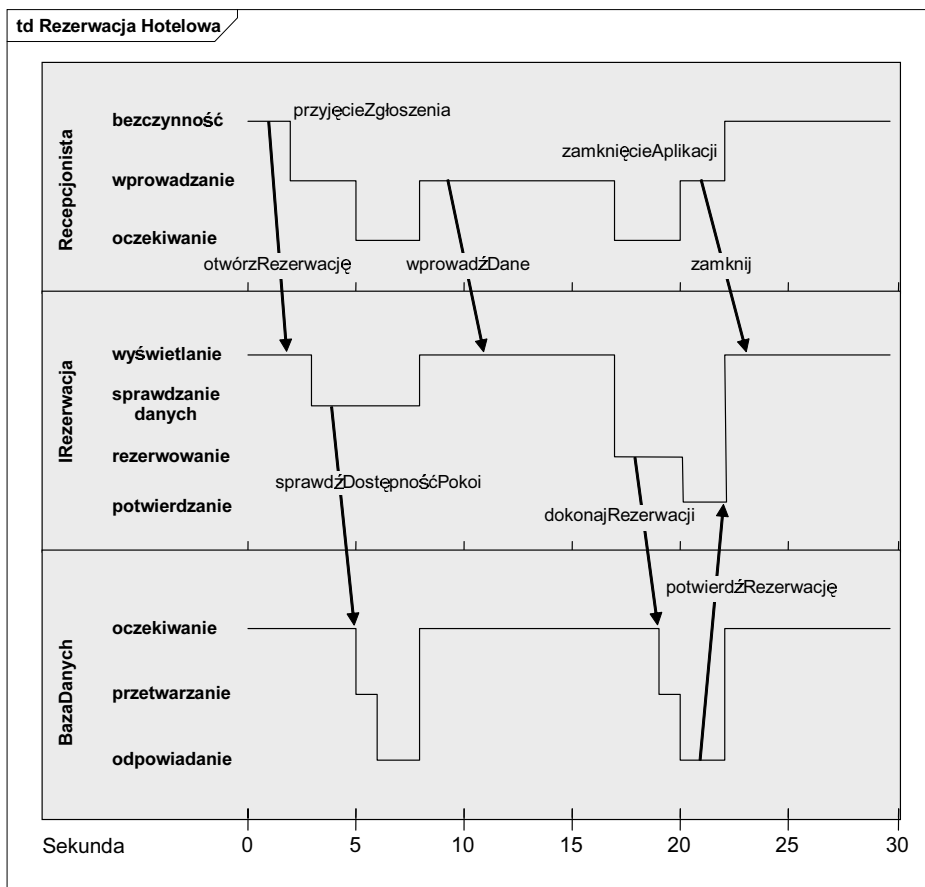
Przesyłanie komunikatów

Diagramy harmonogramowania można wzbogacić o dokumentowanie interakcji w postaci komunikatów przesyłanych między instancjami klasyfikatorów. Źródłem tej informacji są odpowiednie diagramy sekwencji lub komunikacji. W związku z tym na diagramach harmonogramowania można przedstawić wszystkie rodzaje komunikatów

omówionych w punkcie *Zaawansowane składniki diagramu* z wyjątkiem komunikatu utraconego oraz znalezionego. Na podstawie diagramu harmonogramowania przedstawionego na rysunku 9.4 wyspecyfikowano następujące komunikaty:

- ♦ *otwórzRezerwację*,
- ♦ *sprawdźDostępnośćPokoi*,
- ♦ *wprowadźDane*,
- ♦ *dokonajRezerwacji*,
- ♦ *potwierdźRezerwację*,
- ♦ *zamknij*.

Zostały one wprowadzone do diagramu harmonogramowania zilustrowanego na rysunku 9.6.



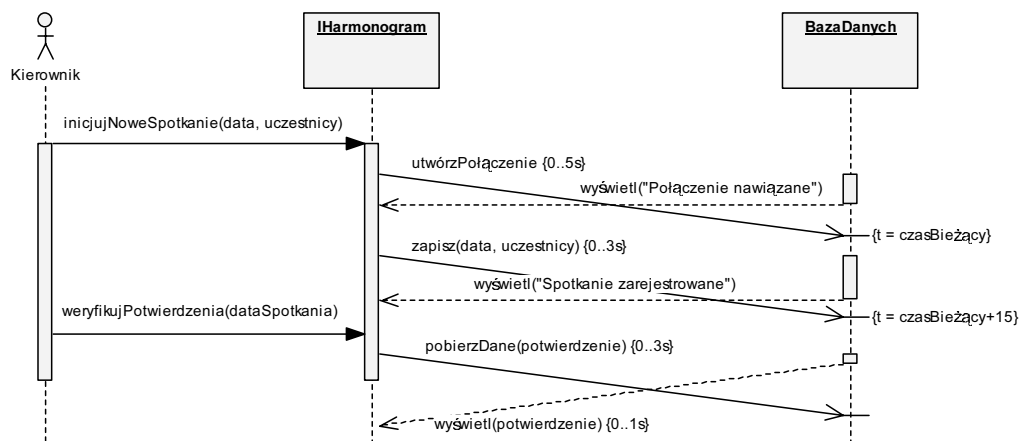
Rysunek 9.6. Komunikaty na diagramach harmonogramowania

Diagramy sekwencji a harmonogramowanie

Pewne elementy harmonogramowania, w szczególności ograniczenia czasowe, można przedstawić na diagramach sekwencji. Wprowadza się je:

- ♦ nad symbolem komunikatu pomiędzy dwoma instancjami klasyfikatorów — w przypadku wskazywania czasu wykonania operacji inicjowanej przez komunikat;
- ♦ równolegle do linii życia instancji klasyfikatora pomiędzy dwoma komunikatami — w przypadku wskazywania przedziału czasowego pomiędzy tymi komunikatami.

Określanie ograniczeń czasowych w wersjach 1.x języka UML było możliwe przede wszystkim na diagramach sekwencji. Na diagramach harmonogramowania ograniczenia przedstawia się w sposób jawny. Jednak poważnym, technicznym utrudnieniem zapisu wspomnianych ograniczeń na diagramie sekwencji jest stosowanie przy ich tworzeniu pełnej, obszernej składni komunikatu. W połączeniu z zapisem ograniczeń czasowych nadmiernie komplikuje to diagram, czyniąc go tym samym mało przejrzystym. W związku z wprowadzeniem diagramów harmonogramowania poprzednia forma zapisu ograniczeń czasowych może mieć jedynie charakter wstępny lub uproszczony. Ograniczenia czasowe na diagramach sekwencji przedstawia rysunek 9.7.



Rysunek 9.7. Ograniczenia czasowe w systemie harmonogramowania spotkań

Rysunek ten ilustruje harmonogramowanie projektów. I tak *Kierownik*, chcąc zainicjować nowe spotkanie zespołu projektowego, określa datę oraz uczestników spotkania z wykorzystaniem interfejsu *IHarmonogram*. W momencie, gdy *Kierownik* potwierdzi wybór, instancja klasyfikatora *IHarmonogram* tworzy połączenie z *BaząDanych*. Zastosowane ograniczenie czasowe wskazuje, że wykonanie operacji *utwórzPołączenie* może potrwać maksymalnie 5 sekund. W momencie uzyskania połączenia na interfejsie *IHarmonogram* wyświetlane jest stosowne potwierdzenie. Następnie realizowana jest operacja *zapisz*. Jej wykonanie powoduje zapisanie w *BazieDanych* daty spotkania

i jego składu osobowego, wcześniej określonego przez *Kierownika*. Limit czasu wykonania wspomnianej operacji wynosi 3 sekundy. Z kolei pomiędzy wywołaniem operacji *utwórzPołączenie* oraz *zapisz* upływa w systemie 15 sekund.

Po zapisaniu danych wyświetlane jest potwierdzenie zarejestrowania spotkania. Asynchronicznie w stosunku do inicjowania przez siebie nowych spotkań, *Kierownik* może weryfikować potwierdzenia obecności uczestników na innych spotkaniach projektowych. W tym celu *Kierownik* określa konkretną datę spotkania. Następnie *IHarmonogram* pobiera potwierdzenia zarejestrowane tego dnia. W efekcie realizowana jest operacja *wyświetl*, udostępniająca *Kierownikowi* potwierdzenia poszczególnych uczestników spotkań projektowych. Na wyświetlenie potwierdzeń przewidziano 1 sekundę.

Proces tworzenia diagramu harmonogramowania

Informacje przedstawiane na diagramach harmonogramowania są w znacznej mierze pochodną kategorii pojęciowych diagramów sekwencji, komunikacji oraz diagramów maszyny stanowej. Stąd w procesie tworzenia diagramów harmonogramowania wykorzystywane mogą być inne diagramy języka UML. Kluczowymi **etapami** tego procesu są:

1. identyfikacja interakcji udokumentowanej diagramem sekwencji lub diagramem komunikacji,
2. przeniesienie lub dobór klasyfikatorów,
3. identyfikacja stanów każdej instancji klasyfikatora z wykorzystaniem diagramów maszyny stanowej,
4. ustalenie horyzontu czasowego diagramu,
5. wyspecyfikowanie linii zmiany stanu instancji klasyfikatorów,
6. wprowadzenie ograniczeń czasowych dla poszczególnych stanów instancji klasyfikatora,
7. nazwanie i wprowadzenie odpowiednich zdarzeń na podstawie diagramów maszyny stanowej,
8. harmonizacja linii zmiany stanu wszystkich instancji klasyfikatorów interakcji prezentowanych na diagramie,
9. przeniesienie lub wprowadzenie komunikatów przesyłanych pomiędzy instancjami klasyfikatorów uczestniczącymi w interakcji.

Ze względu na wykorzystywanie w procesie tworzenia diagramów harmonogramowania szeregu kategorii pojęciowych innych diagramów, bardzo istotne jest zachowywanie spójności pomiędzy nimi. Proces tworzenia diagramów harmonogramowania podlega regułom iteracyjno-przyrostowego cyklu życia systemu.

Podstawowe pojęcia

Diagram harmonogramowania	Harmonizacja
Definicja	Interakcja
Klasyfikator	Iteracja
Linia zmiany stanów	Komunikat
Nazwa stanu	Przesyłanie
Notacja	Asynchroniczny
Klasyczna	Oczekujący
Alternatywna	Opcjonalny
Proces tworzenia	Synchroniczny
Diagram komunikacji	Zwrotny
Izomorfizm	Ograniczenie czasowe
Stan	Skala czasu
Bezczynność	Stan
Czuwanie	System
Obliczanie	Czasu rzeczywistego
Oczekiwanie	Wbudowany
Wykonywanie	Warunek
Diagram maszyny stanowej	Zdarzenie
Diagram sekwencji	
Ośrodek sterowania	
Linia życia	

Pytania i zadania

1. Wyjaśnij zależności pomiędzy poszczególnymi diagramami interakcji.
2. Uzasadnij potrzebę sporządzania harmonogramów interakcji.
3. Jakie rodzaje diagramów języka UML można uznać za źródłowe dla diagramów harmonogramowania? Wyjaśnij dlaczego.
4. Wymień przykłady instancji klasyfikatorów, których stany mogą być odzwierciedlane na diagramie harmonogramowania. Wskaż typowe stany dla każdego z nich.

5. Czym jest linia zmiany stanu instancji klasyfikatora? Co oznaczają jej załamania?
6. Opierając się na diagramach 7.24 oraz 5.13, sporządź diagram harmonogramowania opisujący funkcjonowanie:
 - ♦ *Licytacji*,
 - ♦ *Egzaminu*.
7. Jakich aspektów dotyczą zaawansowane kategorie pojęciowe diagramów harmonogramowania?
8. Na diagramie harmonogramowania instancji klasyfikatora *Licytacja* (pytanie 6.) zaznacz i nazwij zdarzenia.
9. Uzupełnij tak sporządzony diagram o ograniczenia czasowe. Przekształć stosowaną na nim notację w alternatywną.
10. W jakim celu stosuje się harmonizację? Jakie warunki muszą być spełnione, aby zaprezentowanie harmonizacji na diagramie było możliwe?
11. Opierając się na rysunku 7.17, sporządź diagram harmonogramowania. Obligatoryjnie uwzględnij komunikaty.
12. Odwołaj się do strony internetowej *www.bankier.pl*. Wybierz produkty finansowe o notowaniach skokowych. Na ich podstawie opracuj stosowny diagram harmonogramowania.
13. Porównaj użyteczność i przejrzystość zapisu ograniczeń czasowych na diagramach harmonogramowania i diagramach sekwencji. Zilustruj wnioski graficznie, korzystając z diagramu zamieszczonego na rysunku 9.2.
14. Jak przebiega proces tworzenia diagramów harmonogramowania? Uzasadnij jego poszczególne etapy.

Rozdział 10.

Diagram sterowania interakcją

Znaczenie diagramów sterowania interakcją

Specyfikację dokumentacji interakcji realizuje się poprzez opracowanie szeregu logicznie powiązanych diagramów sekwencji, komunikacji oraz harmonogramowania. Szczegółowo scharakteryzowano je w rozdziałach 7. – 9. Diagramy te dokumentują całościowo interakcje dla danego przypadku użycia lub ich części zwane fragmentami interakcji. Wymagania specyfikacji złożonych procesów biznesowych i systemowych oznaczają konieczność powiązania tych fragmentów w logiczny ciąg. Poszczególne diagramy powinny być zatem połączone przepływami sterowania, a następnie przedstawione na osobnym, syntetycznym diagramie. Taką oczekiwaną rolę, wychodzącą naprzeciw **udokumentowaniu logicznych związków między fragmentami interakcji**, mogą spełniać diagramy czynności. Jednak w diagramach tych rolę czynności i akcji pełnić powinny odwołania do poszczególnych, osobnych diagramów lub fragmentów interakcji osadzonych bezpośrednio na danym diagramie. A zatem:



Diagram sterowania interakcją jest rodzajem diagramu interakcji, dokumentującym przepływ sterowania pomiędzy logicznie powiązanymi diagramami i fragmentami interakcji z wykorzystaniem kategorii modelowania diagramów czynności.

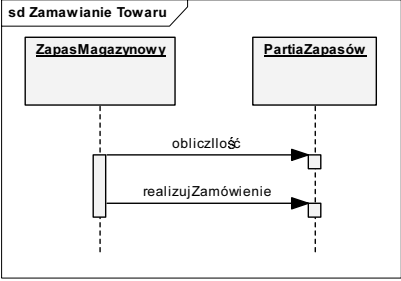
Podstawowe kategorie pojęciowe oraz notacja graficzna

Zgodnie z definicją, na diagramie sterowania interakcją zastosowanie znajdują:

- ♦ podstawowe oraz wybrane zaawansowane elementy diagramów czynności;
- ♦ wybrane kategorie pojęciowe poszczególnych diagramów interakcji, tj. diagramów sekwencji, komunikacji, harmonogramowania.

W związku z tym podstawowe **kategorie pojęciowe** diagramów sterowania interakcją przedstawia tabela 10.1.

Tabela 10.1. Podstawowe kategorie pojęciowe diagramu sterowania interakcją

Nazwa	Notacja	Interpretacja
Przepływ sterowania		Przepływ sterowania pomiędzy fragmentami interakcji.
Początek		Punkt rozpoczęcia przepływu sterowania inicjujący funkcjonowanie diagramu sterowania interakcją.
Koniec		Punkt zatrzymania wszystkich przepływów sterowania na diagramie sterowania interakcją.
Zakończenie przepływu		Punkt zatrzymania wybranego przepływu sterowania na diagramie sterowania interakcją.
Fragment interakcji		Odwołanie do diagramu sekwencji (<i>sd</i>), komunikacji (<i>cd</i>) lub harmonogramowania (<i>td</i>), opisanego całościowo w danym fragmencie interakcji. W nagłówku diagramu sterowania interakcją wyróżnik jest obligatoryjny, natomiast nazwa fragmentu interakcji opcjonalna.
Przywoływane wystąpienie interakcji		Odwołanie do diagramu interakcji (<i>ref</i>) oznaczonego nazwą wymienioną w przywoływanym wystąpieniu interakcji.

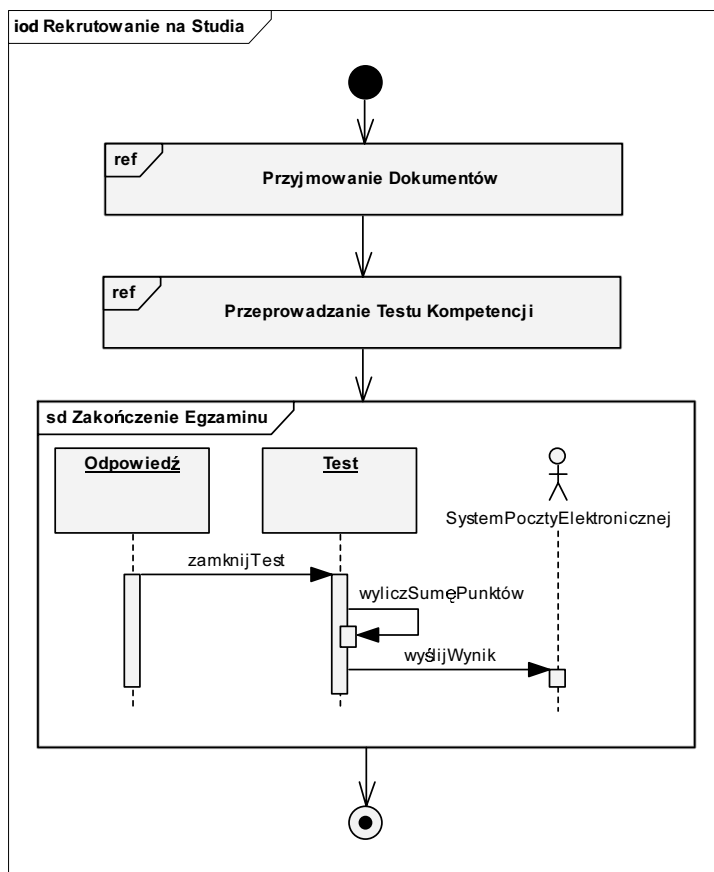
Jak zaznaczono w tabeli 10.1, **fragmenty interakcji** na diagramach sterowania interakcją można oznaczyć:

- ♦ jako przywoływane wystąpienia interakcji (*ref*), czyli swoiste odsyłacze do innych diagramów interakcji; są one wyspecyfikowane wyłącznie z wykorzystaniem nazwy i szczególnie użyteczne w przypadku złożonych diagramów interakcji, do których należą np. implementacyjne diagramy sekwencji;

- ♦ poprzez wyróżniki poszczególnych rodzajów diagramów interakcji — *sd*, *cd*, *td* — w sytuacji, gdy diagram sterowania interakcją w całości przedstawia poszczególne, niezbyt obszerne i złożone diagramy składowe.

Graficznie diagram sterowania interakcją przedstawia się w formie obramowanej, z wyróżnikiem diagramu *id*, jego nazwą oraz opcjonalną listą instancji klasyfikatorów biorących udział w interakcji. Przykład diagramu sterowania interakcją, który zawiera przywoływane wystąpienia interakcji *ref* opisujące proces rekrutacji na studia, przedstawia rysunek 10.1.

Rysunek 10.1.
Diagram sterowania
interakcją



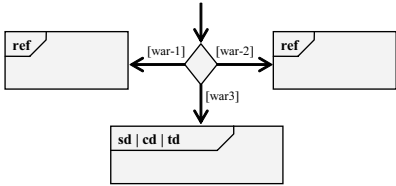
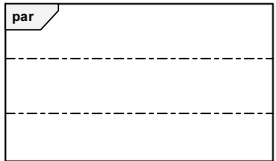
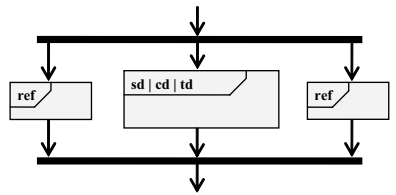
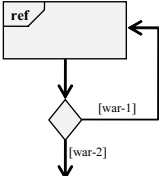
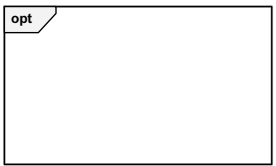
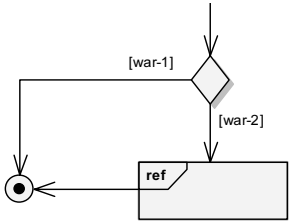
Proces rekrutacji kandydatów na studia jest wspomagany systemem informatycznym. Przedstawienie całego procesu na jednym diagramie możliwe jest przy wykorzystaniu diagramu sterowania interakcją. Na pierwszym etapie rekrutacji kandydaci za pośrednictwem przeglądarek internetowych zgłaszają swoją kandydaturę i przesyłają wymagane dokumenty. W odpowiedzi system generuje kody dostępu niezbędne do uczestnictwa w teście kompetencji. Wymienione interakcje są częścią przywołania *ref* o nazwie *PrzyjmowanieDokumentów*. Następnie w ramach przywołanej interakcji *PrzeprowadzenieTestuKompetencji* kandydaci logują się do systemu i uczestniczą w teście. System wyświetla kolejne pytania, a kandydaci zaznaczają odpowiedzi. Następnie, po upływie

przewidzianego czasu trwania testu, system automatycznie go kończy, blokując możliwość wprowadzania czy modyfikowania odpowiedzi. Interakcje składające się na pokazaną finalizację procedury egzaminacyjnej przedstawia całościowo diagram sekwencji *ZakończenieEgzaminu*.

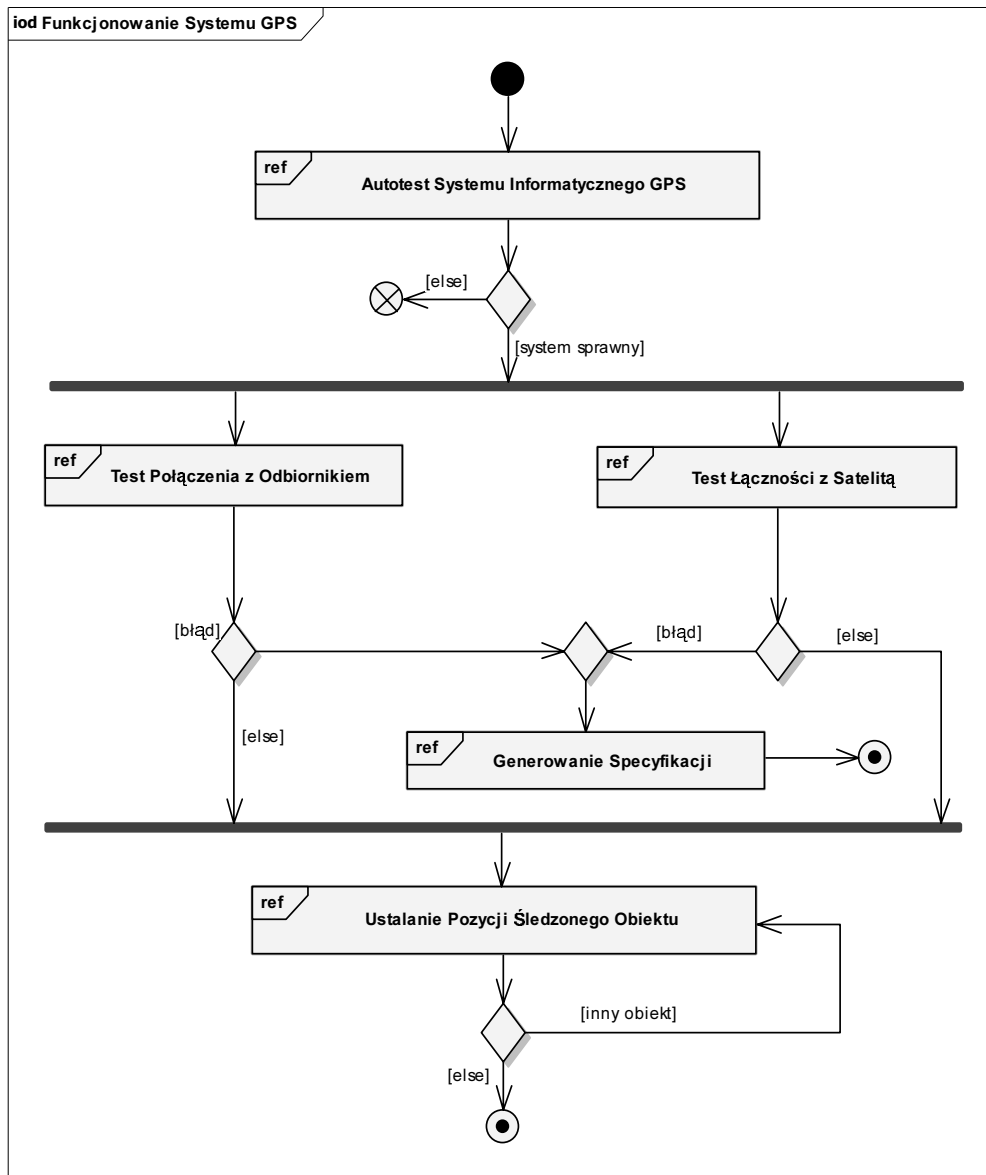
Zaawansowane składniki diagramu

W praktyce diagramy sterowania interakcją mogą wykorzystywać znaczny zakres kategorii pojęciowych **diagramów czynności**. Dzięki temu możliwe jest tworzenie szczegółowych, rozbudowanych diagramów sterowania interakcją. Szczególną rolę w tej współzależności pełnią zaawansowane kategorie pojęciowe diagramów czynności i sekwencji. **Relacje** pomiędzy wybranymi, istotnymi zaawansowanymi kategoriami wymienionych rodzajów diagramów w odniesieniu do diagramów sterowania interakcją przedstawia tabela 10.2.

Tabela 10.2. Zestawienie notacji diagramów czynności i diagramów sekwencji

Kategorie diagramów sekwencji		Odpowiedniki na diagramach czynności	
Nazwa	Oznaczenie graficzne	Nazwa	Oznaczenie graficzne
alternatywa		decyzja	
współbieżność		rozwidlenie scalenie	
iteracja		decyzja iteracja	
opcja		decyzja opcja	

Praktyczne zastosowanie wymienionych w tabeli 10.2 zaawansowanych kategorii pojęciowych diagramu sterowania interakcją zilustrowano na rysunku 10.2.



Rysunek 10.2. Diagram sterowania interakcją z procesami współbieżnymi i warunkowymi

Rysunek przedstawia funkcjonowanie systemu GPS. W momencie uruchomienia systemu informatycznego GPS automatycznie uruchamiają się skrypty weryfikujące poprawne funkcjonowanie oprogramowania. Jeśli testy nie wykazują niewłaściwego zachowania systemu, system informatyczny GPS inicjuje równoległe wykonanie dwóch przywoływanych wystąpień interakcji. Są to: *Test Połączenia z Odbiornikiem* oraz *Test*

Łączności z Satelitą. Jednak w momencie wykrycia podczas autotestu nieprawidłowości przepływ ulega zakończeniu. Wykonywane równolegle wystąpienia interakcji również mogą zakończyć się błędem. W takim przypadku następuje *Generowanie Specyfikacji błędów*. W sytuacji poprawnego zakończenia wszystkich testów system GPS jest gotowy do *Ustalania Pozycji Śledzonych Obiektów*. Interakcje związane z tą funkcjonalnością są wykonywane iteracyjnie dla wszystkich śledzonych obiektów.

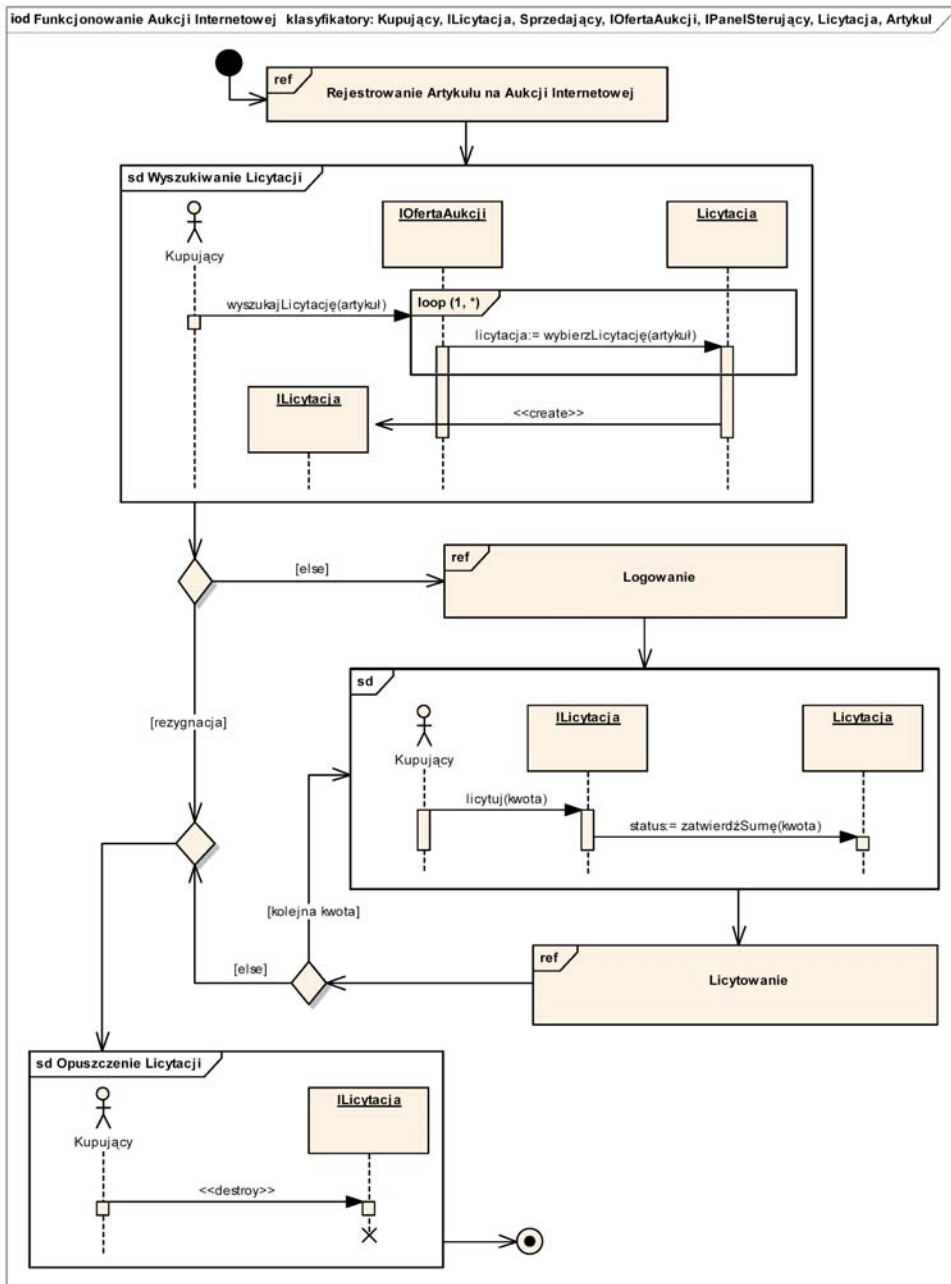
Proces tworzenia diagramu sterowania interakcją

Diagramy sterowania interakcją, podobnie jak inne diagramy języka UML, są tworzone zgodnie z zasadami cyklu iteracyjno-przyrostowego. Zalecany kierunek ich tworzenia jest kierunek wstępujący. Wynika to z faktu wcześniejszego opracowania poszczególnych diagramów interakcji, których wzajemne zależności mogą być jednoznacznie przedstawione. A zatem w procesie tworzenia diagramu sterowania interakcją można wyróżnić następujące **etapy**:

1. zgromadzenie wszystkich opracowanych diagramów interakcji — sekwencji, komunikacji oraz harmonogramowania — dla danego przypadku użycia;
2. przeanalizowanie wzajemnych zależności pomiędzy tymi diagramami;
3. zakwalifikowanie poszczególnych diagramów do:
 - ♦ przywoływanych wystąpień interakcji (*ref*) w przypadku diagramów złożonych,
 - ♦ fragmentów interakcji (*cd*), (*sd*), (*td*) dla mniej obszernych diagramów;
4. ustalenie logicznej kolejności wykonywania poszczególnych fragmentów interakcji;
5. opracowanie kompletnego diagramu sterowania interakcją z wykorzystaniem podstawowych i zaawansowanych kategorii pojęciowych;
6. opcjonalne wymienienie w nagłówku diagramu sterowania interakcją nazw instancji klasyfikatorów biorących udział w interakcji.

Studium diagramu sterowania interakcją

Funkcjonowanie aukcji internetowej z wykorzystaniem zarówno poszczególnych diagramów sekwencji (*Wyszukiwanie Licytacji* oraz *Opuszczanie Licytacji*), jak i przywoływanych wystąpień interakcji (*Wstawianie Artykułu na Aukcję Internetową* oraz *Licytowanie*) przedstawia rysunek 10.3. Ponieważ konwencja diagramów sterowania



Rysunek 10.3. Diagram sterowania interakcją — funkcjonowanie aukcji internetowej

interakcją nie pozwala na bezpośrednie zaprezentowanie w obszarze diagramu poszczególnych instancji klasyfikatorów biorących udział w interakcji, listę tych instancji zamieszczono w nagłówku diagramu w formie parametrów po słowie kluczowym *klasyfikatory*.

Przedstawiony na rysunku 10.3 diagram sterowania interakcją ilustruje funkcjonowanie aukcji internetowej. Diagram ten jest izomorficzny w stosunku do diagramu sekwencji o takiej samej nazwie (rysunek 7.34). Interakcje aukcji internetowej zostały podzielone na trzy przywoływane wystąpienia interakcji: *Rejestrowanie Artykułu na Aukcji Internetowej*, *Logowanie* i *Licytowanie* oraz trzy fragmenty interakcji typu *sd*. Dwa z nich, tj. *Wyszukiwanie Licytacji* i *Opuszczenie Licytacji*, są nazwane. Elementami diagramu są również dwie decyzje, złączenie i dwie iteracje. Podczas *Rejestrowania Artykułu na Aukcji Internetowej* osoby zarejestrowane jako *Sprzedawcy* mają uprawnienia do rejestrowania artykułów, które później stają się przedmiotem licytacji. Aby zainicjować licytację, *Sprzedawcy* musi wypełnić stosowny formularz. Po wprowadzeniu wszystkich wymaganych danych system tworzy w bazie danych obiekt *Licytacja*. W obrębie fragmentu interakcji (*sd*) *Wyszukiwanie Licytacji* fragment wyodrębniony jest realizowany na zasadzie pętli programowej wyświetlającej jako wynik wszystkie spełniające kryterium licytacji.

Zrealizowanie opcji przystąpienia *Kupującego* do licytacji artykułu w pierwszej kolejności wymaga jego zalogowania do systemu aukcji. Sterowanie zostaje wtedy przekazane do przywoływanego fragmentu interakcji (*ref*) nazwanego *Logowanie*. Dopiero po zalogowaniu *Kupujący* ma prawo do pełnego uczestnictwa w licytacji, w tym proponowania i zwiększania kwot. *Sprzedawcy* wprowadza odpowiednią kwotę do interfejsu licytacji. Następnie jest ona przekazywana przez interfejs do odpowiedniego obiektu klasy *Licytacja*. Porównanie zaproponowanej kwoty z propozycjami innych uczestników licytacji oraz odświeżenie listy kwot i *Kupujących* jest realizowane w ramach przywoływanego wystąpienia interakcji (*ref*) *Licytowanie*. Jak pokazano zarówno na diagramie sterowania interakcją, jak i na odpowiadającym mu diagramie sekwencji, nienazwany fragment interakcji (*sd*) oraz przywołane wystąpienie interakcji mogą być realizowane na zasadzie pętli programowej. Iteracja jest wykonywana, jeśli *Sprzedawcy* decyduje się zaproponować kolejne kwoty. W sytuacji, gdy po wyszukaniu interesujących go artykułów *Kupujący* podejmuje decyzję o nieprzystępowaniu do ich licytowania, sterowanie jest przekazywane do fragmentu interakcji *Opuszczenie Licytacji*. *Sprzedawcy* zamyka wtedy interfejs z wyszukanymi licytacjami, po czym przepływ sterowania ulega zakończeniu. Również w przypadku, gdy *Kupujący* zdecydował się przystąpić do licytacji, w momencie jej zakończenia czy też rezygnacji z dalszego uczestnictwa sterowanie zostaje przekazane do fragmentu interakcji *Opuszczenie Licytacji*. Na diagramie przepływ ten oznaczono słowem kluczowym *else*.

Podstawowe pojęcia

Diagram czynności

Koniec

Początek

Warunek

Zakończenie przepływu

Diagram interakcji

Diagram harmonogramowania

Diagram komunikacji

Diagram sekwencji

Diagram sterowania interakcją	Opcja
Definicja	Współbieżność
Nagłówek	Przywoływane wystąpienie interakcji (ref)
Proces tworzenia	Interakcja
Wyróżnik	Przeływ decyzyjny
Fragment interakcji	Decyzja
(cd)	Złączenie
(sd)	Przeływ sterowania
(td)	Rozwidlenie
Fragment wyodrębniony	Scalenie
Alternatywa	
Iteracja	

Pytania i zadania

- Wyjaśnij zależności pomiędzy:
 - ♦ diagramami sterowania interakcją,
 - ♦ innymi diagramami interakcji,
 - ♦ diagramami czynności.
- Uzasadnij dobór podstawowych kategorii modelowania diagramów sterowania interakcją.
- Wymień rodzaje fragmentów interakcji. Wyjaśnij ich rolę.
- Kiedy stosuje się przywoływane wystąpienia interakcji *ref*, a kiedy fragmenty interakcji *sd/cd/td*?
- Opracuj diagram sterowania interakcją z wykorzystaniem fragmentów interakcji typu *ref* na podstawie diagramu „Generowanie raportów o kosztach realizowanych projektów” (rysunek 7.38).
- Wskaż i zinterpretuj zaawansowane kategorie diagramów sterowania interakcji.
- Opierając się na diagramie „System nadzoru pomieszczeń kompleksu kinowego” (rysunek 7.42), wyszczególnij istniejące interakcje w postaci fragmentu interakcji typu *sd*. Następnie wprowadź fragmenty interakcji różnego typu opisujące proponowany dalszy przebieg interakcji opisanego systemu i wyspecyfikuj szczegółowo przepływy sterowania pomiędzy nimi.

- 8.** Wykorzystując podstawowe i zaawansowane kategorie diagramu sterowania interakcją, opracuj adekwatny diagram dla firmy ubezpieczeniowej. W odniesieniu do zidentyfikowanych klasyfikatorów i interakcji ich instancji, zaproponuj odpowiednie fragmenty interakcji. Dziedzina przedmiotowa przedstawia się następująco:

Firma oferuje na rynku produkty ubezpieczeniowe kierowane do potencjalnych segmentów klientów. Oferta ubezpieczenia wraz z prognozą składek ubezpieczeniowych może być przedstawiana klientom wyłącznie przez przedstawicieli tzw. kanałów dystrybucyjnych, tj. agencji ubezpieczeniowych oraz biur brokerskich. Prognozy składek są generowane w dedykowanym systemie informatycznym. Jego funkcjonalność umożliwia tworzenie prognoz dla różnych wariantów kwot i okresów ubezpieczenia. Po analizie prognozy klient może podjąć decyzję o podpisaniu wniosku ubezpieczeniowego, który na dalszym etapie staje się polisą. Obsługa wykupionych polis jest realizowana w odrębnym systemie.

- 9.** Przeanalizuj relacje pomiędzy tymi operatorami interakcji języka UML 2, które nie zostały wyszczególnione w tabeli 10.2, a kategoriami bądź zestawami kategorii modelowania stosowanymi na diagramie sterowania interakcją.
- 10.** Jak przebiega proces tworzenia diagramów sterowania interakcją?

Rozdział 11.

Diagramy wdrożeniowe

Znaczenie diagramów wdrożeniowych

Pełny projekt informatyczny musi zawierać opis struktury i dynamiki systemów obiektowych. Aspekty fizyczne struktury systemu specyfikuje się z wykorzystaniem **diagramów wdrożeniowych**. Wyróżnia się dwa rodzaje tych diagramów, a mianowicie:

- ♦ diagramy komponentów,
- ♦ diagramy rozlokowania.

Istotność obu z nich należy podkreślić w przypadku, gdy założenia projektowe obejmują daleko posuniętą specyfikację struktury kodu źródłowego. Użyteczność diagramów wdrożeniowych przejawia się w możliwości modelowania konfiguracji tworzonego systemu. I tak **diagramy komponentów** pozwalają na modelowanie elementów oprogramowania i związków między nimi, natomiast **diagramy rozlokowania** — na modelowanie rozmieszczenia infrastruktury sprzętowej oraz platform użytkowania systemu. A zatem pojęcie diagramów wdrożeniowych, podobnie jak pojęcie diagramów interakcji, jest kategorią abstrakcyjną.

Diagram komponentów

W ostatnich kilku latach jedną z dominujących tendencji w dziedzinie tworzenia oprogramowania oraz — w szerszym rozumieniu — systemów informatycznych było oparcie ich struktury na wzajemnie powiązanych komponentach, które świadczą określone usługi. Zainteresowanie tym rozwiązaniem oraz jego praktyczne wdrożenie wiązało się z większą elastycznością i możliwością wymiany poszczególnych komponentów składających się na system. Komponenty mogły być łatwo wymieniane, aktualizowane, a także włączane do innych aplikacji. Wiąże się to z koncepcją ponownego wykorzystania (ang. *reusable components*), czyli jedną z podstawowych koncepcji obiektowości.

Koncepcja komponentów została zaadoptowana w języku UML od pierwszych jego wersji, a znacząco zmieniona i rozwinięta w wersji 2.0. Naturalnie podstawowym pojęciem w tym kontekście jest **komponent** (ang. *component*).



Komponent to hermetyczny, wymienny moduł oprogramowania systemu, realizujący określone jego usługi za pośrednictwem interfejsów.

W języku UML komponenty dokumentowane są na diagramach komponentów.



Diagram komponentów to rodzaj diagramu wdrożeniowego, który wskazuje organizację i zależności między komponentami.

Kategorie modelowania diagramów komponentów przedstawia tabela 11.1.

Kategorią pojęciową wykorzystywaną na diagramach komponentów są również artefakty, omówione w punkcie *Diagram rozlokowania*. Artefakt jest pojęciem bardziej ogólnym od komponentu. W związku z tym w niniejszym punkcie skoncentrowano się na komponentach. Obecnie dominuje zapis komponentów w postaci klasy ze stereotypem graficznym lub tekstowym. Wynika to z wprowadzenia możliwości specyfikowania atrybutów oraz operacji komponentu. W wersjach 1.x języka UML używano symbolu komponentu w postaci prostokąta z dwoma bolcami w lewej jego części (por. tabela 11.1, komponent *Klient.db*). Symbol ten ciągle występuje w wielu opracowaniach.

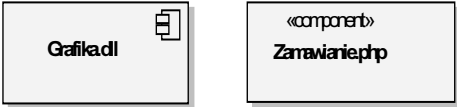
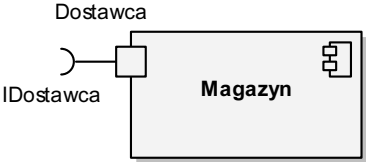
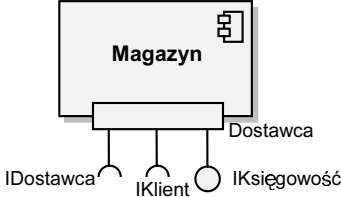
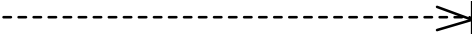
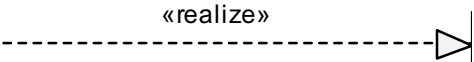
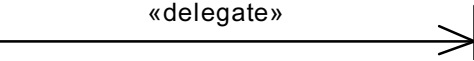
Przykładami komponentów są podstawowe moduły systemu informatycznego. Można przypisywać im **stereotypy**, jak poniżej:

- ♦ programy wykonywalne — «*executable*» (rysunek 11.1a),
- ♦ biblioteki programów — «*library*» (rysunek 11.1b),
- ♦ fizyczne bazy danych i tabele baz danych — «*table*» (rysunek 11.1c),
- ♦ podsystemy — «*subsystem*» (rysunek 11.1d),
- ♦ komponenty przetwarzające — «*service*» (rysunek 11.1e).

Komponenty są podstawowym składnikiem diagramów komponentów. Po ich określeniu można ustalić wzajemne zależności pomiędzy komponentami poprzez opracowanie **konceptualnego diagramu komponentów**. Taki diagram w odniesieniu do systemu rejestracji referatów na konferencji naukowej przedstawia rysunek 11.2.

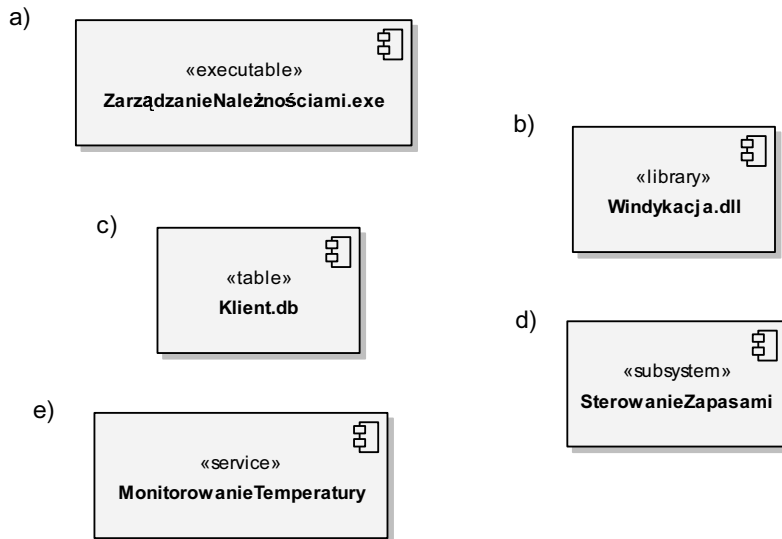
Funkcjonalność współczesnych systemów obsługi konferencji naukowej musi obejmować moduł przyjmowania oraz recenzowania referatów zgłoszonych przez autorów zainteresowanych konferencją. Tego typu systemy wykorzystują technologię internetową. Autor referatu, wykorzystując przeglądarkę internetową, rejestruje referat za pośrednictwem komponentu *RejestracjaReferatu*. Strona wyświetlana w przeglądarce została opisana w języku XML, na co wskazuje rozszerzenie nazwy komponentu. Zapewnienie opisanej funkcjonalności umożliwia skrypt wykonany w języku PHP o nazwie *IndeksKonferencji*. Ten ostatni komponent wykorzystuje plik bazy danych o nazwie

Tabela 11.1. Kategorie modelowania diagramów komponentów

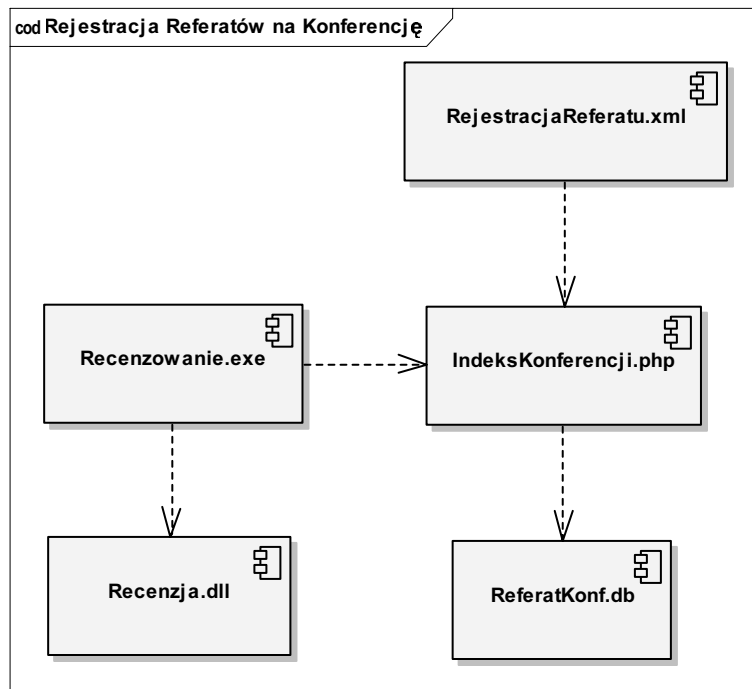
Nazwa	Notacja
Komponent	
Interfejs udostępniający	
Interfejs pozyskujący	
Port	
Port złożony	
Port złożony	
Zależność	
Realizacja	
Konektor delegowany	
Konektor składany	

ReferatKonf. Rejestrowane referaty zgodnie z regułami konferencji podlegają procedurom recenzowania. System musi pozwalać na recenzowanie przez kilku recenzentów każdego zgłoszonego referatu. Funkcjonalność umożliwiającą rejestrację recenzji zgłoszonych referatów w bazie danych zawiera komponent *Recenzowanie*, który jest plikiem wykonywalnym. Komponent ten korzysta z zawartości dynamicznie dołączanej biblioteki o nazwie *Recenzja*, na co wskazuje związek zależności pomiędzy tymi komponentami. Wspomniana biblioteka obsługuje m.in. wspólne reguły recenzowania referatów.

Rysunek 11.1.
Komponenty
i ich stereotypy



Rysunek 11.2.
Konceptualny
diagram
komponentów
systemu rejestracji
referatów



Konceptualny diagram komponentów staje się podstawą opracowania implementacyjnego diagramu komponentów, co wymaga identyfikacji i rozumienia podstawowych pojęć:

- ♦ interfejsów,
- ♦ specyfikacji komponentów,
- ♦ portów,
- ♦ konektorów.

Interfejsy

Na diagramach komponentów elementem umożliwiającym korzystanie z usług poszczególnych komponentów i pośredniczącym pomiędzy komponentami jest **interfejs** (ang. *interface*).

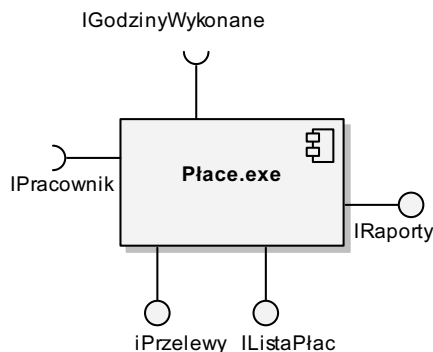


Interfejs to zestaw operacji, które wyznaczają usługi oferowane przez klasę lub komponent.

Interfejsy bądź zestawy interfejsów, poprzez które komponenty oferują swoje usługi, są nazywane **interfejsami udostępniającymi** (ang. *provided interfaces*). Obok interfejsów udostępniających istnieją **interfejsy pozyskujące** (ang. *required interfaces*), które pozwalają komponentom na korzystanie z usług innych komponentów. Zależności zarówno pomiędzy komponentem a interfejsem udostępniającym — tj. dostarczającym określone swoje usługi — jak i komponentem a interfejsem pozyskującym przedstawiono na rysunku 11.3.

Rysunek 11.3.

*Zewnętrzna
perspektywa
komponentu Place*



Rysunek ten przedstawia moduł systemu *Place.exe*. Zaimplementowano w nim funkcjonalność niezbędną do rozliczenia płac w systemie księgowym. Komponent udostępnia trzy interfejsy:

- ♦ *IRaporty*,
- ♦ *IListaPłac*,
- ♦ *iPrzelewy*.

Korzysta on natomiast z usług innych komponentów poprzez następujące interfejsy pozyskujące:

- ♦ *IPracownik*,
- ♦ *IGodzinyWykonane*.

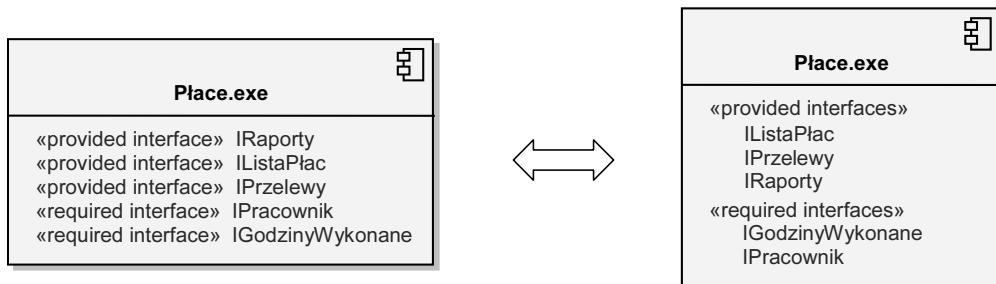
Specyfikacja komponentów

Specyfikacja zawartości komponentu może mieć charakter:

- ♦ czarnej skrzynki,
- ♦ białej skrzynki.

W pierwszym przypadku przedstawia się komponent wraz z jego interfejsami w sposób ogólny, czyli ilustruje jego perspektywę zewnętrzną. Przykłady komponentów wyspecyfikowanych w postaci **czarnej skrzynki** (ang. *black box*) zaprezentowano na rysunkach 11.3 i 11.4. Specyfikacje interfejsów przedstawione na rysunku 11.4 są odpowiednikami sposobu dokumentowania interfejsów przedstawionego na rysunku 11.3. Specyfikacja interfejsów komponentów na rysunku 11.4 została utworzona z wykorzystaniem stereotypów tekstowych. Są to odpowiednio stereotypy:

- ♦ «*provided interface*» — dla interfejsów udostępniających,
- ♦ «*required interface*» — dla interfejsów pozyskujących.



Rysunek 11.4. Specyfikacja interfejsów w postaci stereotypów tekstowych

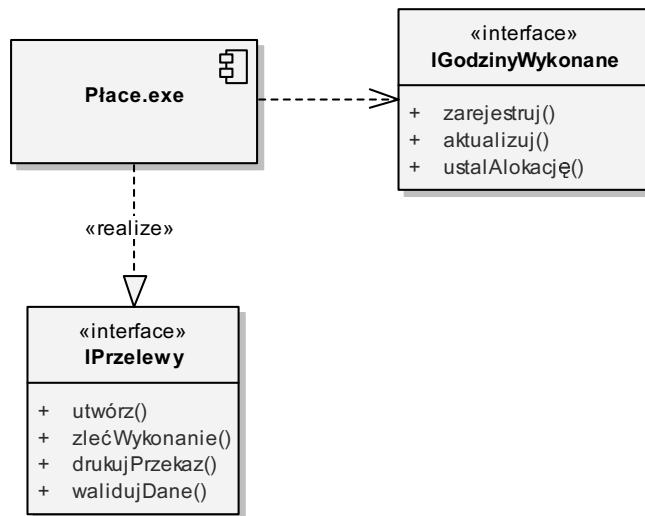
Specyfikacje interfejsów zamieszczone na opisywanym rysunku mogą być stosowane zamiennie.

Interfejsy mogą być również przedstawione w postaci instancji klasyfikatorów ze wskazanymi **operacjami** i **zdarzeniami**. Instancje te łączy się wówczas z komponentami za pośrednictwem związków zależności i realizacji (rysunek 11.5), przy czym:

- ♦ związki zależności wskazują na interfejsy pozyskujące,
- ♦ natomiast związki realizacji na interfejsy udostępniające.

Rysunek 11.5.

Interfejsy
w postaci instancji
klasyfikatorów
w systemie HRM



I tak komponent *Place.exe*, będący modulem systemu zarządzania zasobami ludzkimi HRM, udostępnia część swoich usług realizowanych przez interfejs *IPrzelewy* w postaci czterech operacji wyspecyfikowanych na interfejsie:

- ♦ *utwórz*,
- ♦ *zlećWykonanie*,
- ♦ *drukujPrzekaz*,
- ♦ *walidujDane*.

Opisywany komponent jest automatycznie zobowiązany do realizowania tych usług, na co wskazuje związek realizacji oznaczony dodatkowo stereotypem «realize». W przypadku interfejsu pozyskującego *IGodzinyWykonane* usługi w postaci operacji:

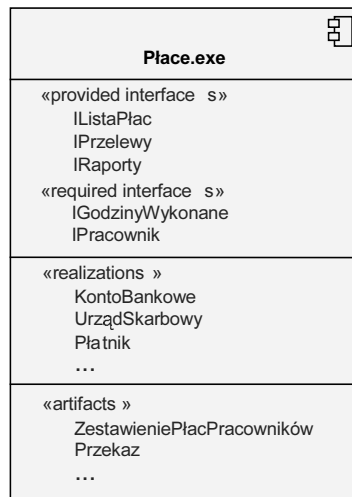
- ♦ *zarejestruj*,
- ♦ *aktualizuj*,
- ♦ *ustalAlokację*

są pozyskiwane z innego komponentu. Przykładowo mógłby to być komponent *Karta-Pracy.exe*, domyślnie połączony stereotypowanym związkiem realizacji z interfejsem *IGodzinyWykonane*. Z perspektywy użytkownika systemu HRM interfejs *IGodzinyWykonane* umożliwiałby przekazywanie do modułu *Place* informacji zawartych w kartach pracy poszczególnych pracowników.

W odróżnieniu od czarnej skrzynki **biała skrzynka** (ang. *white box*) zawiera specyfikację zawartości komponentu, czyli jego perspektywę wewnętrzną. Zachowanie komponentu jest realizowane przez instancje jego wewnętrznych klasyfikatorów. Specyfikacji udziału klasyfikatorów można dokonać poprzez wprowadzenie dodatkowych sekcji komponentu, opisanych stereotypami tekstowymi (rysunek 11.6). Tak więc poza standardową sekcją interfejsów pozyskujących i udostępniających specyfikacja ta zawiera sekcje:

Rysunek 11.6.

Specyfikacja
komponentu
w postaci
białej skrzynki



- ♦ klasyfikatorów — «realizations»,
- ♦ artefaktów — «artifacts».

W sekcji klasyfikatorów wskazuje się nazwy instancji klasyfikatorów, które są odpowiedzialne za realizację zachowania komponentu. Pełna specyfikacja komponentu może również zawierać sekcję artefaktów, omówionych w punkcie *Diagram rozlokowania*.

Porty

Symbol komponentu można uzupełniać o **porty** (ang. *ports*), czyli wyróżnialne punkty związane ściśle z interfejsami, przez które komponent komunikuje się z otoczeniem. Poprzez interfejs oraz port komponent oferuje swoje usługi i oczekuje realizacji konkretnych usług przez otoczenie. Umieszczanie portów na diagramie znajduje uzasadnienie w przypadku jawnego wskazywania klasyfikatorów wewnętrznych danego komponentu. Za jego pośrednictwem można wówczas jednoznacznie przyporządkować interfejsy do instancji klasyfikatorów. Jeżeli z danym portem związane jest kilka interfejsów, określa się go mianem **portu złożonego**. Przykłady portów zawiera tabela 11.1 oraz rysunek 11.7.

Konektory

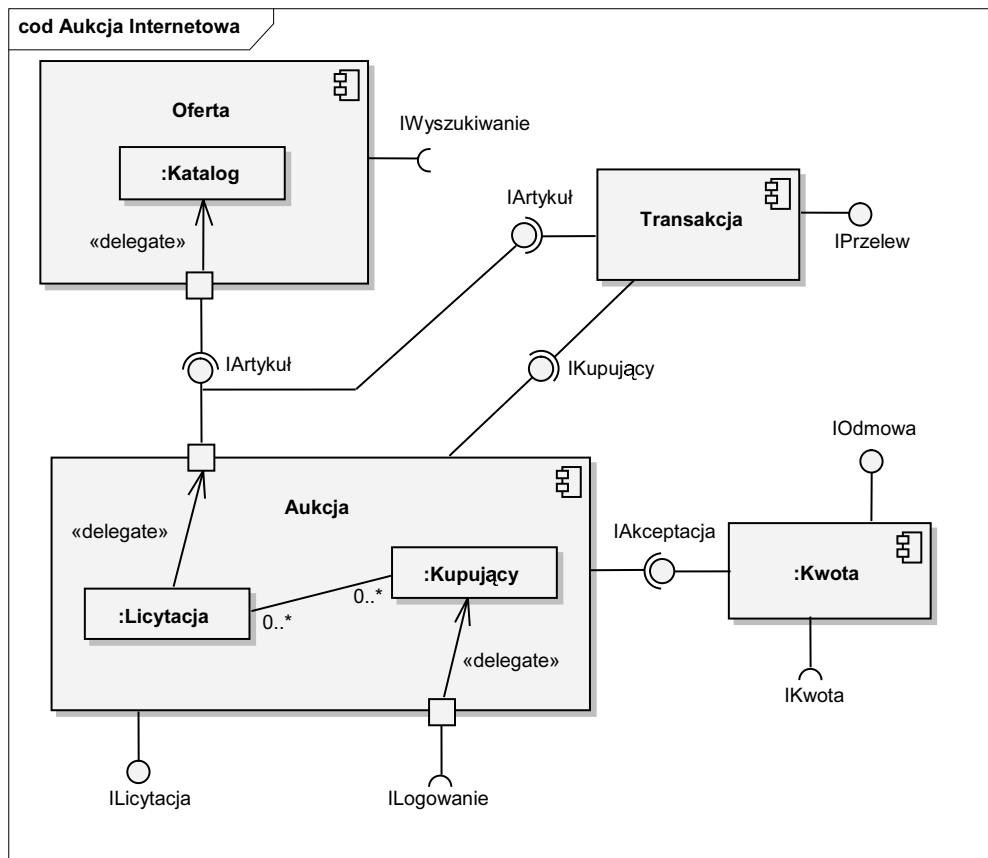
Komponenty na diagramach komponentów łączone są za pośrednictwem związków zależności lub — w przypadku wyspecyfikowania interfejsów udostępniających oraz pozyskujących — za pośrednictwem **konektorów** (ang. *connectors*). Wyróżnia się dwa rodzaje konektorów:

- ♦ delegowany,
- ♦ składany.

Pierwszy z wymienionych, stereotypowany jako *«delegate»*, łączy poprzez port instancje klasyfikatorów znajdujące się na zewnątrz komponentu z wewnętrzną realizacją interfejsu przez odpowiednią instancję klasyfikatora komponentu. Oznacza to przekazanie sygnału do portu lub odpowiedniej instancji klasyfikatora. Konektor składany oznacza powiązanie dwóch komponentów, z których jeden udostępnia poprzez interfejs usługę pozyskiwaną przez drugi komponent. Konektor składany przybiera więc postać połączenia dwóch interfejsów: udostępniającego i pozyskującego, kolokwialnie określanego mianem *ball & socket*. Przykłady konektorów zawiera tabela 11.1 oraz rysunek 11.7.

Implementacyjny diagram komponentów

Implementacyjne diagramy komponentów przedstawiają zaawansowane kategorie pojęciowe diagramów komponentów — takie jak interfejsy, porty, specyfikacje komponentów czy też konektory — w sposób kompleksowy. Dobłą podstawą opracowania takiego diagramu jest konceptualny diagram komponentów. Implementacyjny diagram komponentów systemu aukcji internetowej ilustruje rysunek 11.7.



Rysunek 11.7. Implementacyjny diagram komponentów aukcji internetowej

Na rysunku zamieszczono cztery komponenty składające się na oprogramowanie aukcji. Zawiera on również wszystkie opisane powyżej kategorie modelowania. Tak więc komponent *Aukcja* przechowuje klasy *Licytacja* oraz *Kupujący*. Korzysta on z usług komponentu *Kwota*. Zastosowanie składni nazewnictwa typowej dla klas wskazuje, że w tym drugim komponencie zakodowana jest klasa o takiej samej nazwie. Komponent *Kwota* przechowuje proponowane sumy za artykuł będący przedmiotem licytacji. Ponadto wdraża on metody pozwalające rejestrować proponowane kwoty zgodnie z regułami aukcji. Interfejs graficzny *IKwota* jest interfejsem pozyskującym, co pozwala proponować coraz to większe sumy za artykuł. Natomiast drugi, również graficzny interfejs udostępniający *IOdmowa* informuje o odrzuceniu proponowanych kwot. Interfejs ten jest wyświetlany w sytuacjach niezgodności licytowania z regulaminem aukcji — np. w przypadku zadeklarowania kwoty niższej od aktualnie zarejestrowanej w systemie. Najwyższa uzyskana kwota przekazywana jest poprzez interfejs *IAkceptacja* do komponentu *Aukcja*. *ILogowanie* umożliwia zalogowanie się uczestnika licytacji do systemu. Z kolei *ILicytacja* zawiera szczegółowe informacje o licytacji. Interfejs pozyskujący jest połączony z klasą *Kupujący*, co umożliwia weryfikację loginu i hasła, gdyż zgodnie z regułami aukcji, aby aktywnie uczestniczyć w licytacjach, wymagana jest identyfikacja użytkownika.

Komponent *Aukcja* jest połączony z dwoma konektorami składanymi, nazwanymi odpowiednio *IArtykuł* oraz *IKupujący*. Pierwszy z nich przekazuje dane o licytacjach (sprzedawanych artykułach) do komponentu *Oferta* — gdzie są delegowane do klasy *Katalog* — oraz komponentu *Transakcja*. Drugi umożliwia przekazanie danych kupującego do komponentu *Transakcja*. Dane artykułu oraz kupującego są niezbędne do przeprowadzenia transakcji. Zatem interfejsem udostępniającym komponentu *Transakcja* jest *IPrzelew*. Jest on kandydatem na interfejs graficzny lub interfejs przeznaczony dla systemu bankowego umożliwiającego transakcję online. W tym drugim przypadku interfejs graficzny nie będzie wymagany, a dane zostaną przesłane bezpośrednio do systemu bankowego. Wówczas zachowanie interfejsu określić należy jako wewnętrzne. Komponent *Oferta* przechowuje klasę *Katalog*. Interfejs pozyskujący *IWyszukiwanie* jest interfejsem graficznym, który umożliwia użytkownikom przeszukiwanie katalogu aukcji.

Diagram rozlokowania

Po wykonaniu w języku UML logicznej specyfikacji systemu niezbędne staje się udokumentowanie jego fizycznego **rozlokowania** (ang. *deployment*). Eksploatacja współczesnego systemu informatycznego dokonuje się przez użytkowanie różnorodnego sprzętu, oprogramowania i sieci komputerowych. Głównymi elementami specyfikacji logicznej, tworzonej w trakcie trzech pierwszych dyscyplin metodyki RUP, są diagramy przypadków użycia, klas, obiektów, czynności, maszyny stanowej oraz interakcji. Specyfikacja na poziomie fizycznym powinna pozwolić na udokumentowanie architektury eksploatowanego w systemie sprzętu oraz związków użytkowanego oprogramowania z tą infrastrukturą informatyczną. Narzędziem realizacji tego zadania są diagramy rozlokowania. Dzięki diagramom rozlokowania język UML pozwala na wyspecyfikowanie rozmieszczenia modułów użytkowanej aplikacji na połączonych węzłach.



Rozlokowanie to alokacja artefaktów lub instancji artefaktów w określonym węźle.

Kluczowe znaczenie dla diagramów rozlokowania mają więc artefakty (ang. *artifacts*).



Artefakt oznacza każdy sztucznie wytworzony produkt.

W języku UML **artefakt** należy odnieść do wytwarzanych i użytkowanych produktów informatycznych. A zatem artefakt w informatyce to każdy produkt informatyczny wytworzony lub użyty w trakcie cyklu życia systemu. Przykładami artefaktów w systemach informatycznych są:

- ♦ pakiety oprogramowania,
- ♦ bazy danych,
- ♦ arkusze kalkulacyjne,
- ♦ specyfikacje wymagań systemu,
- ♦ klasy,
- ♦ diagramy,
- ♦ modele systemowe i biznesowe,
- ♦ scenariusze przypadku użycia,
- ♦ plany testów,
- ♦ komponenty.

Artefakty mogą być stereotypowane. Dotyczą ich wszystkie stereotypy związane z komponentami wymienione w punkcie *Diagram komponentów*. Ponieważ pojęcie artefaktu jest szersze od pojęcia komponentu, w odniesieniu do artefaktów występują również takie stereotypy jak:

- ♦ «*script*» — skrypty systemowe,
- ♦ «*document*» — dokumenty wykorzystywane w systemie,
- ♦ «*file*» — pliki,
- ♦ «*source*» — pliki zawierające kod źródłowy.

Zarówno artefakty, jak i komponenty rozmieszczane są na węzłach **diagramu rozlokowania**. Stąd diagramy rozlokowania są ściśle związane z diagramami komponentów.



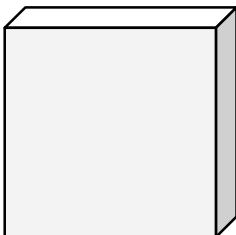
Diagram rozlokowania to rodzaj diagramu wdrożeniowego, który przedstawia sieć połączonych ścieżkami komunikowania węzłów z ułożonymi na nich artefaktami.

Podstawowymi elementami diagramów rozlokowania są:

- ♦ węzły,
- ♦ ścieżki komunikowania.

Oznaczenia graficzne podstawowych kategorii pojęciowych diagramu rozlokowania przedstawia tabela 11.2.

Tabela 11.2. Podstawowe kategorie pojęciowe diagramów rozlokowania

Nazwa	Notacja graficzna
Węzeł	
Ścieżka komunikowania	

Poniżej szczegółowo scharakteryzowano podstawowe i zaawansowane kategorie diagramów rozlokowania, tj.:

- ♦ węzły,
- ♦ ścieżki komunikowania,
- ♦ zależności,
- ♦ osadzone artefakty i komponenty,
- ♦ manifestowanie,
- ♦ specyfikację rozlokowania,
- ♦ diagramy rozlokowania na poziomie fizycznym.

Węzły

Kluczową rolę w diagramach rozlokowania odgrywa pojęcie **węzła** (ang. *node*), który jest kolejnym rodzajem klasyfikatora.



Węzeł to fizyczny lub logiczny zasób przetwarzający, na którym są osadzone artefakty użytkowanego systemu.

Węzeł może przybierać postać jednostki sprzętu komputerowego lub platformy użytkowania systemu. Pierwszy z wymienionych rodzajów komponentów charakteryzuje się zazwyczaj posiadaniem zdolności obliczeniowej oraz pamięci. Najbardziej typowymi przykładami **węzłów sprzętowych** (ang. *devices*) są:

- ♦ serwery;
- ♦ urządzenia wejścia-wyjścia, takie jak drukarki czy też monitory;
- ♦ routery, bramy, switche, koncentratory i inne urządzenia sieciowe.

Platformy użytkowania systemu (ang. *execution environments*) obejmują:

- ♦ systemy operacyjne,
- ♦ systemy zarządzania bazą danych,
- ♦ platformy systemów elektronicznego obiegu informacji WFM,
- ♦ platformy e-learningowe,
- ♦ systemy CRM oraz BIS.

W związku z dużą różnorodnością węzłów, diagramy rozlokowania charakteryzują się największą różnorodnością stereotypów wśród diagramów języka UML. Mogą to być zarówno stereotypy **tekstowe**, jak i **graficzne**. Stereotypy graficzne umieszcza się w prawym górnym rogu, natomiast tekstowe nad nazwą węzła lub artefaktu. Węzły sprzętowe standardowo oznaczane są na diagramie stereotypem «*device*», natomiast platformy użytkowania systemu — stereotypem «*execution env*». W ramach wyżej wymienionych rodzajów węzły sprzętowe i platformy użytkowania systemu można oznaczać szeregiem innych stereotypów, które pozwalają na bardziej szczegółową specyfikację węzłów. Pojedynczemu węzłowi można przypisać tylko jeden stereotyp. I tak dla węzłów sprzętowych będą to przykładowo:

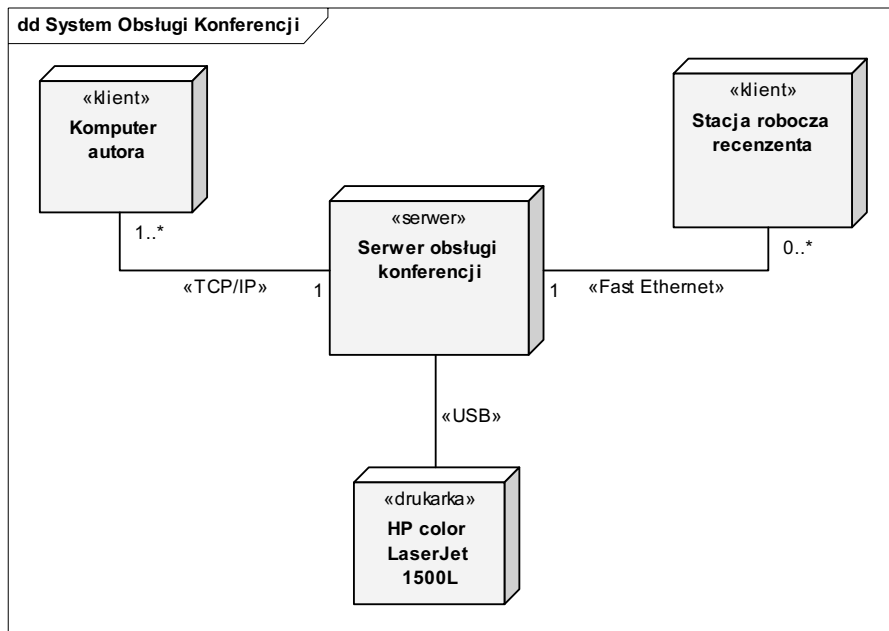
- ♦ «*server*»,
- ♦ «*drukarka*»,
- ♦ «*klient*»,
- ♦ «*PC*»,
- ♦ «*switch*».

Platformom użytkowania systemu przypisuje się m.in. następujące stereotypy:

- ♦ «*UNIX*»,
- ♦ «*DBMS*»,
- ♦ «*DB2*»,
- ♦ «*Lotus*».

Ścieżki komunikowania

Rolę **ścieżek komunikowania** (ang. *communication paths*) między węzłami pełnią związki asocjacji. Zazwyczaj są one stereotypowane. Typowy diagram rozlokowania, opisujący architekturę sprzętu niezbędnego do poprawnego, stabilnego działania systemu obsługi konferencji, przedstawia rysunek 11.8.



Rysunek 11.8. Diagram rozlokowania systemu obsługi konferencji

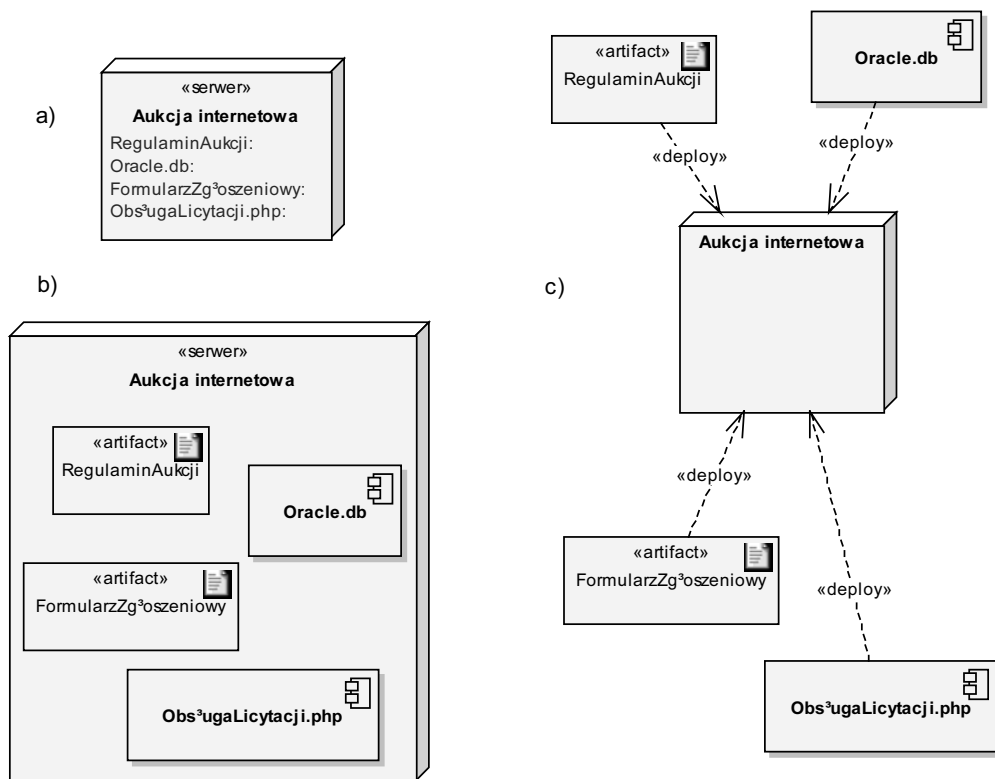
Dla funkcjonowania zilustrowanego systemu niezbędny jest komputer osobisty pełniący rolę serwera oraz wydajna drukarka laserowa. Podłącza się ją do serwera za pośrednictwem portu USB. Autorzy łączą się z serwerem za pośrednictwem Internetu z wykorzystaniem protokołu TCP/IP. Z kolei komputery recenzentów, znajdujące się w sieci lokalnej, do komunikowania się wykorzystują technologię Fast Ethernet.

Osadzone artefakty i komponenty

Po zidentyfikowaniu węzłów można im przyporządkować **osadzone** na nich artefakty. Wyróżnia się trzy alternatywne sposoby określania rozmieszczenia artefaktów na węźle. Przybierają one postać:

- ♦ tekstowego wyliczenia artefaktów w węźle (rysunek 11.9a),
- ♦ artefaktów bezpośrednio ulokowanych w węźle (rysunek 11.9b),
- ♦ artefaktów połączonych stereotypowaną zależnością «*deploy*» z węzłem (rysunek 11.9c).

Konsekwencją wprowadzania do diagramu rozlokowania informacji typowych dla diagramu komponentów jest stosowanie związków **zależności**. Szczególnym rodzajem związków zależności na diagramach rozlokowania są zależności stereotypowane «*deploy*» oraz «*manifest*». Wymieniona zależność stereotypowana «*deploy*» oznacza osadzenie zależnego artefaktu na węźle oraz wskazanie, że artefakt używa tego węzła do realizacji swojej funkcjonalności.



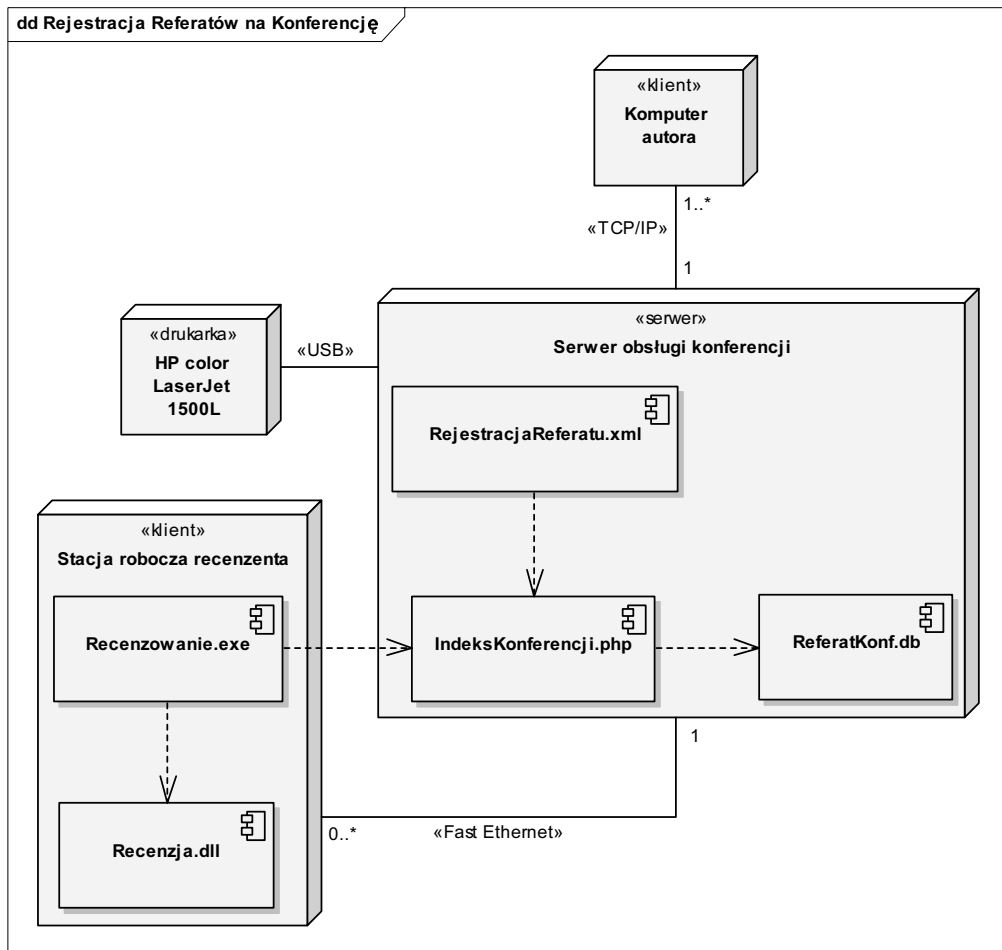
Rysunek 11.9. Alternatywne sposoby osadzania artefaktów na węzłach

Rysunek 11.10 stanowi odwzorowanie rozlokowania komponentów realizujących rejestrację referatów w systemie obsługi konferencji na węzłach tego systemu. W ten sposób zintegrowano elementy tego systemu przedstawione uprzednio na rysunkach 11.8 oraz 11.2.

Manifestowanie

Na diagramach rozlokowania można modelować kategorię **manifestowania**, czyli zawierania i reprezentowania przez artefakt swoich elementów składowych. Cechę tę oznacza się związkiem zależności ze stereotypem «manifest» pomiędzy danym artefaktem a jego składnikami (składnikami). Element manifestowany, składnik artefaktu, dostępny jest poprzez interfejs artefaktu. Manifestowanie przedstawia rysunek 11.11.

Zamieszczony na nim diagram dotyczy fragmentu rozlokowania węzłów oraz komponentów infrastruktury firmy prowadzącej kursy e-learningowe. Pierwszy zilustrowany na nim węzeł to serwer, na którym osadzono komponent oznaczony stereotypem «OS», co wskazuje na system operacyjny. W tym konkretnym przypadku jest to dystrybucja Red Hat systemu Linux. Do komponentów osadzonych na serwerze należy *Czat* umożliwiający wymianę poglądów z trenerami oraz uczestnikami kursów. Program ten jest uruchamiany w formie apletu. Drugim artefaktem osadzonym na serwerze



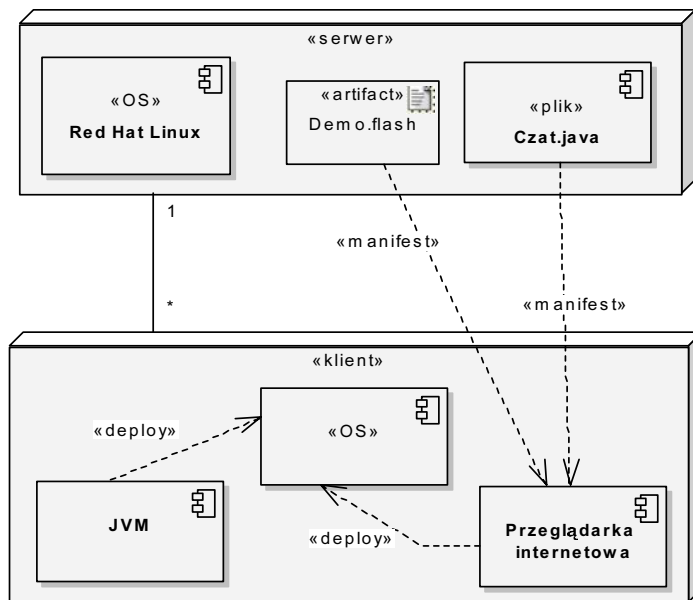
Rysunek 11.10. Diagram rozlokowania systemu obsługi konferencji z komponentami

jest demonstracja kursu e-learningowego, zapisana w pliku o nazwie *Demo*. Została ona przygotowana w technologii Flash. Zarówno czat, jak i demonstracja są manifestowane za pośrednictwem *Przeglądarki internetowej*. Na komputerze klienta osadzone są trzy komponenty:

- ♦ system operacyjny,
- ♦ przeglądarka internetowa,
- ♦ wirtualna maszyna JAVY.

Dwa ostatnie z wymienionych komponentów są zależne od systemu operacyjnego. Przeglądarka internetowa jest niezbędna do oglądania wersji demonstracyjnej oraz rozmowy w ramach czatu. Wirtualna maszyna JAVY (*JVM*) umożliwia natomiast poprawne wyświetlanie apletów JAVY.

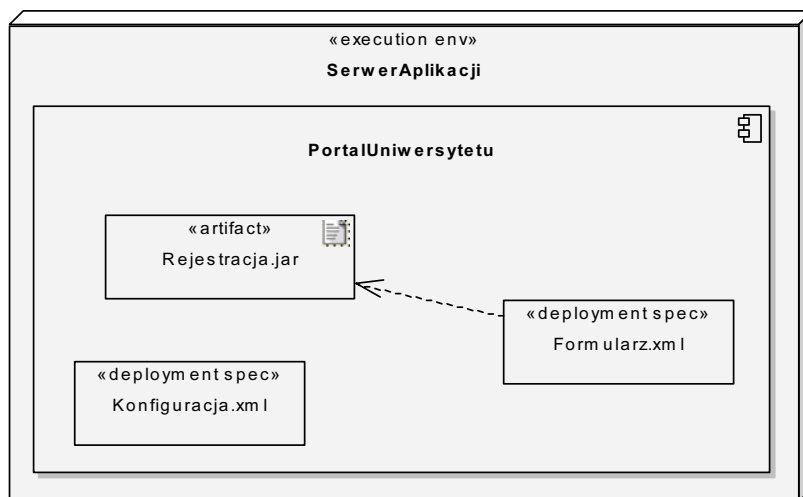
Rysunek 11.11.
Manifestowanie
w platformie
e-learningowej



Specyfikacja rozlokowania

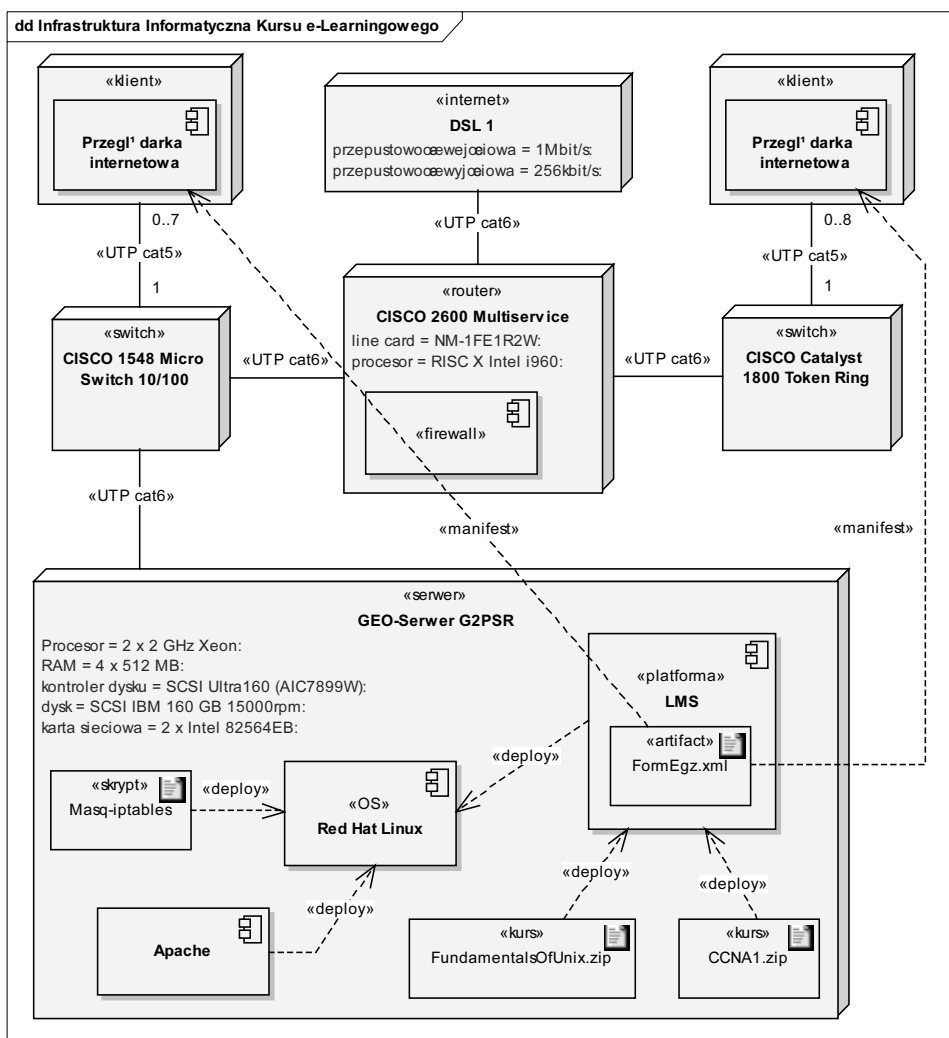
Specyfikacja rozlokowania (ang. *deployment specification*) zawiera zestaw cech artefaktu lub komponentu. Określa parametry użytkowania artefaktu osadzonego na określonym węźle. W języku JAVA odnosi się do pojęcia «*container*». Specyfikacje mogą być dalej rozwijane w kodzie komponentu. Przykład specyfikacji rozlokowania zaprezentowano na rysunku 11.12. Zastosowano na nim stereotyp «*deployment spec*» w odniesieniu do dwóch artefaktów — *Formularza* oraz pliku *Konfiguracji* — specyfikujących odpowiednio: artefakt *Rejestracja.tar* oraz komponent *PortalUniwersytetu*.

Rysunek 11.12.
Specyfikacja
rozlokowania
portalu
uniwersyteckiego



Diagramy rozlokowania na poziomie fizycznym

Diagramy rozlokowania można opracować na poziomie **logicznym** oraz **fizycznym**. Pierwszy rodzaj diagramu rozlokowania (np. rysunki 11.8 i 11.10) zawiera wyłącznie ogólne informacje o węzłach i ścieżkach komunikowania. Dozwolone jest jednak wprowadzanie do ich oznaczeń na diagramie bardziej szczegółowego opisu charakteryzujących je parametrów. Przykłady diagramów rozlokowania na poziomie logicznym przedstawiono na rysunkach 11.10 i 11.11. Bardziej konkretny jest diagram rozlokowania na poziomie fizycznym, który szczegółowo specyfikuje atrybuty węzłów. Przykładowo — w odniesieniu do serwera — wskazuje on nie tylko parametry techniczne, ale i adres czy numer pokoju, w którym jest ulokowany. Przykładem takiego podejścia do specyfikacji węzłów jest rysunek 11.13.



Rysunek 11.13. Diagram rozlokowania na poziomie fizycznym

Jest on rozbudowaną wersją rysunku 11.11. Treścią rysunku jest zatem fizyczne rozlokowanie infrastruktury firmy prowadzącej kursy na odległość. Do stworzenia infrastruktury sieciowej w firmie wykorzystano zarówno technologię Ethernet, jak i Token Ring. Zgodnie z tym założeniem zainstalowano odpowiedni osprzęt sieciowy. Szczegółowe parametry wskazano w przypadku trzech węzłów, w tym serwera i routera. Pierwszy z nich zawiera informacje o procesorze, pamięci RAM, kontrolerze dysku twardego, dysku twardym oraz karcie sieciowej. Natomiast drugi obejmuje dane o procesorze i karcie routera. Na serwerze osadzono siedem elementów — cztery artefakty oraz trzy komponenty. Artefakty to:

- ♦ skrypt *masq-iptables*, zawierający konfigurację tzw. maskarady;
- ♦ dwa archiwa z danymi kursów na odległość;
- ♦ formularz egzaminacyjny zawierający pytania dotyczące każdego rozdziału kursu.

Kursy zarchiwizowane są w popularnym formacie *.zip*. Pliki kursów pobierane są przez użytkownika. Przeprowadzenie egzaminu wymaga natomiast wyświetlenia formularza egzaminacyjnego przez przeglądarkę internetową zainstalowaną na komputerze klienta. Tym samym artefakt w formacie *.xml* jest manifestowany po stronie klienta. Komponenty osadzone na serwerze firmy prowadzącej kursy na odległość obejmują:

- ♦ system operacyjny *Red Hat Linux*,
- ♦ serwer *Apache*,
- ♦ platformę e-learningową *LMS*.

Związki zależności pomiędzy wymienionymi elementami, stereotypowane jako «*deploy*», wskazują na fizyczną i logiczną zależność. I tak poszczególne kursy są zależne od platformy e-learningowej, natomiast sama platforma, serwer Apache oraz skrypt — od systemu operacyjnego, na którym są osadzone. Będący kolejnym węzłem sieci Router CISCO, model 2600 Multiservice, dzięki odpowiedniej karcie umożliwia podłączenie dwóch switchy firmy CISCO. Pierwszy z nich, Catalyst 1800, pracuje w technologii Token Ring, natomiast drugi, 1548 Micro Switch 10/100, został oparty na technologii Ethernet. Oba switchy umożliwiają podłączenie ośmiu komputerów. Router podłączony jest do Internetu przy wykorzystaniu łącza o przepustowości 1Mbit/s na wejściu i 256 kbit/s na wyjściu. Poszczególne węzły sieci są połączone skrętką (UTP) piątej bądź szóstej kategorii.

Proces tworzenia diagramów wdrożeniowych

Ponieważ diagramy wdrożeniowe występują w postaci ściśle ze sobą powiązanych diagramów komponentów oraz rozlokowania, **proces** ich tworzenia jest zintegrowany, jak przedstawiono poniżej:

1. zidentyfikowanie komponentów systemu;
2. wyspecyfikowanie rodzajów komponentów i ich stereotypowanie;
3. utworzenie konceptualnego diagramu komponentów poprzez powiązanie poszczególnych komponentów zależnościami;
4. zdefiniowanie perspektywy zewnętrznej komponentu — czarnej skrzynki — czyli usług udostępnianych oraz pozyskiwanych przez poszczególne komponenty poprzez interfejsy;
5. opcjonalne uszczegółowienie specyfikacji komponentów do poziomu białej skrzynki poprzez dodanie sekcji klasyfikatorów i artefaktów;
6. utworzenie implementacyjnego diagramu komponentów poprzez uwzględnienie konektorów, portów oraz opracowanych specyfikacji komponentów;
7. wyspecyfikowanie węzłów na podstawie dokumentu wizji systemu oraz innych sformalizowanych dokumentacji;
8. wskazanie ścieżek komunikowania pomiędzy węzłami, ich stereotypowanie oraz określenie liczebności;
9. opcjonalne wyspecyfikowanie węzłów na poziomie fizycznym;
10. osadzenie komponentów i artefaktów na odpowiednich węzłach;
11. uwzględnienie na diagramie artefaktów manifestowanych oraz specyfikacji rozlokowania.

Diagramy wdrożeniowe wynikają z wizji systemu oraz wybranej technologii jego realizacji. W procesie modelowania systemu diagramy komponentów oraz rozlokowania stanowią treść perspektyw o analogicznych nazwach.

Podstawowe pojęcia

Artefakt	Diagram rozlokowania
Skrzynka	Definicja
Biała	Na poziomie fizycznym
Czarna	Na poziomie logicznym
Diagramy wdrożeniowe	Proces tworzenia
Diagram komponentów	Interfejs
Definicja	Udostępniający
Implementacyjny	Pozyskujący
Konceptualny	
Proces tworzenia	

Język	Port
JAVA	Złożony
PHP	Technologia
XML	E-learning
Komponent	Ethernet
Konektor	Internet
Delegowany	Klient-serwer
Składany	LAN
Manifestowanie	Token Ring
Osadzenie	WAN
Rozlokowanie	Węzeł
Specyfikacja	Platforma użytkownika systemu
Stereotyp	Sprzętowy
Graficzny	Związek
Tekstowy	Asocjacja
Ścieżka komunikowania	Zależność

Pytania i zadania

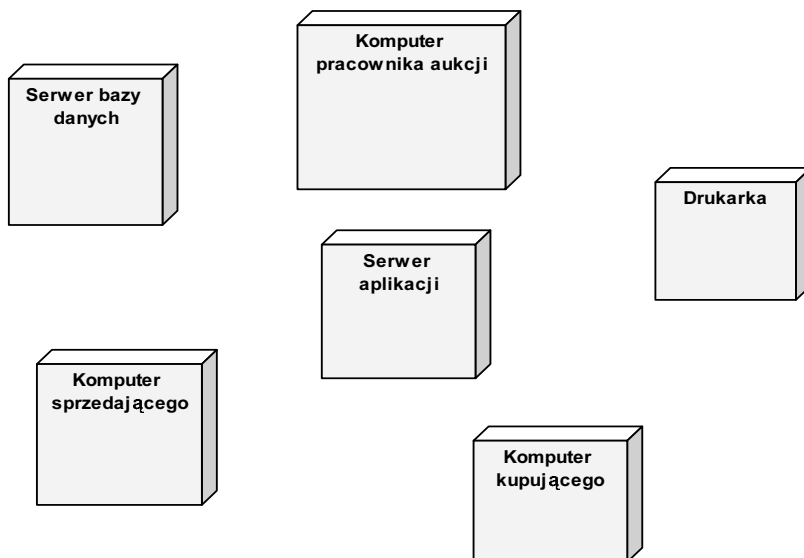
1. Wymień diagramy, które stosuje się do prezentacji aspektów związanych z wdrożeniem systemu informatycznego.
2. Na modelowanie jakich aspektów pozwalają poszczególne rodzaje diagramów wdrożeniowych?
3. Z jakich przesłanek rozwoju oprogramowania wynika znaczenie diagramów komponentów?
4. Odnosząc się do definicji komponentu, wymień i wyjaśnij jego cechy.
5. Wskaż kategorie pojęciowe diagramów komponentów i określ ich zastosowanie.
6. Jakie są alternatywne oznaczenia komponentu?
7. Wymień przykładowe rodzaje komponentów z dziedziny oprogramowania i zinterpretuj znaczenie poszczególnych stereotypów komponentów.
8. Konceptualny a implementacyjny diagram komponentów.

9. Opierając się na przykładzie aplikacji MS Outlook, zidentyfikuj komponenty i wskaż zależności pomiędzy nimi.
10. O jakie elementy należy wzbogacić konceptualny diagram komponentów, aby otrzymać implementacyjny diagram komponentów?
11. W jaki sposób na diagramie komponentów prezentuje się interfejsy?
12. Jaka jest różnica pomiędzy interfejsami udostępniającymi a pozyskującymi?
13. Przedstaw komponent *ZarządzanieNależnościami* w postaci:
 - ♦ czarnej skrzynki,
 - ♦ białej skrzynki.
14. Przedstaw specyfikację interfejsów *ZarządzaniaNależnościami* w postaci klas.
15. Opracuj konceptualny diagram komponentów dla instalacyjnej wersji systemu operacyjnego.
16. Jaka jest rola portu w diagramie komponentów? Wskaż różnice pomiędzy portem a portem złożonym.
17. Jaka jest różnica pomiędzy konektorem delegowanym a składanym? Podaj przykłady.
18. Na podstawie konceptualnego diagramu komponentów zaprezentowanego na rysunku 11.2 utwórz diagram na poziomie implementacyjnym. Skorzystaj ze wszystkich przedstawionych w punkcie *Diagram komponentów* kategorii modelowania.
19. Opracuj implementacyjny diagram komponentów dla wybranego zintegrowanego systemu klasy ERP (np. SAP/R3, IFS, BAAN).
20. Jak należy rozumieć rozlokowanie?
21. Czym jest artefakt? Podaj przykłady.
22. Podaj i określ rolę głównych składników diagramu rozlokowania.
23. Jakie są rodzaje węzłów? Podaj przykłady każdego z nich.
24. Jakie możliwości stwarzają stereotypy w zakresie oznaczania rodzajów węzłów i artefaktów na diagramach rozlokowania?
25. Zakwalifikuj zgodnie z terminologią diagramu rozlokowania następujące pojęcia:
 - ♦ platforma systemu CRM,
 - ♦ biblioteka *graficzna.dll*,
 - ♦ tablica relacyjnej bazy danych,
 - ♦ sformalizowana wizja systemu informatycznego,
 - ♦ okienkowy system operacyjny,

- ♦ światłowód,
 - ♦ port USB,
 - ♦ listwa zasilająca.
26. W jaki sposób można zaznaczyć na węzłach osadzone na nich artefakty i komponenty?
27. Wyjaśnij znaczenie stereotypu «*deploy*».
28. Na przykładzie węzła *Serwer obsługi portalu* opracuj trzy różne sposoby zapisu jego artefaktów.
29. Jakie jest znaczenie stereotypu «*manifest*»?
30. Czy zależność może pełnić rolę ścieżki komunikowania?
31. Do czego służy specyfikacja rozlokowania? Podaj przykład.
32. Wyjaśnij różnice pomiędzy rozlokowaniem a osadzeniem.
33. Opracuj diagram rozlokowania dla przynajmniej dwóch znanych Ci topologii sieciowych.
34. Uzupełnij rysunek 11.14, przedstawiający wstępnie zidentyfikowane węzły w systemie aukcji internetowej, o ścieżki komunikowania i nadaj im odpowiednie stereotypy. Uwzględnij na tworzonym diagramie liczebności poszczególnych węzłów.

Rysunek 11.14.

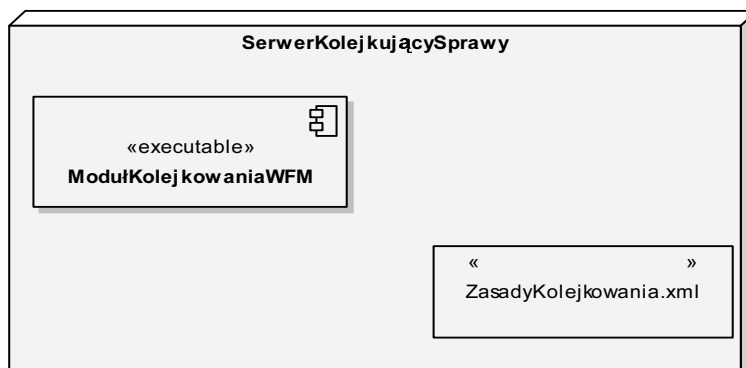
Węzły aukcji
internetowej



35. Rozrysuj na diagramie rozlokowania fragmenty sieci TASK. Skorzystaj z witryny internetowej www.task.gda.pl.
36. Z diagramu zamieszczonego na rysunku 11.15 usunięto jeden ze stereotypów. Jaki?

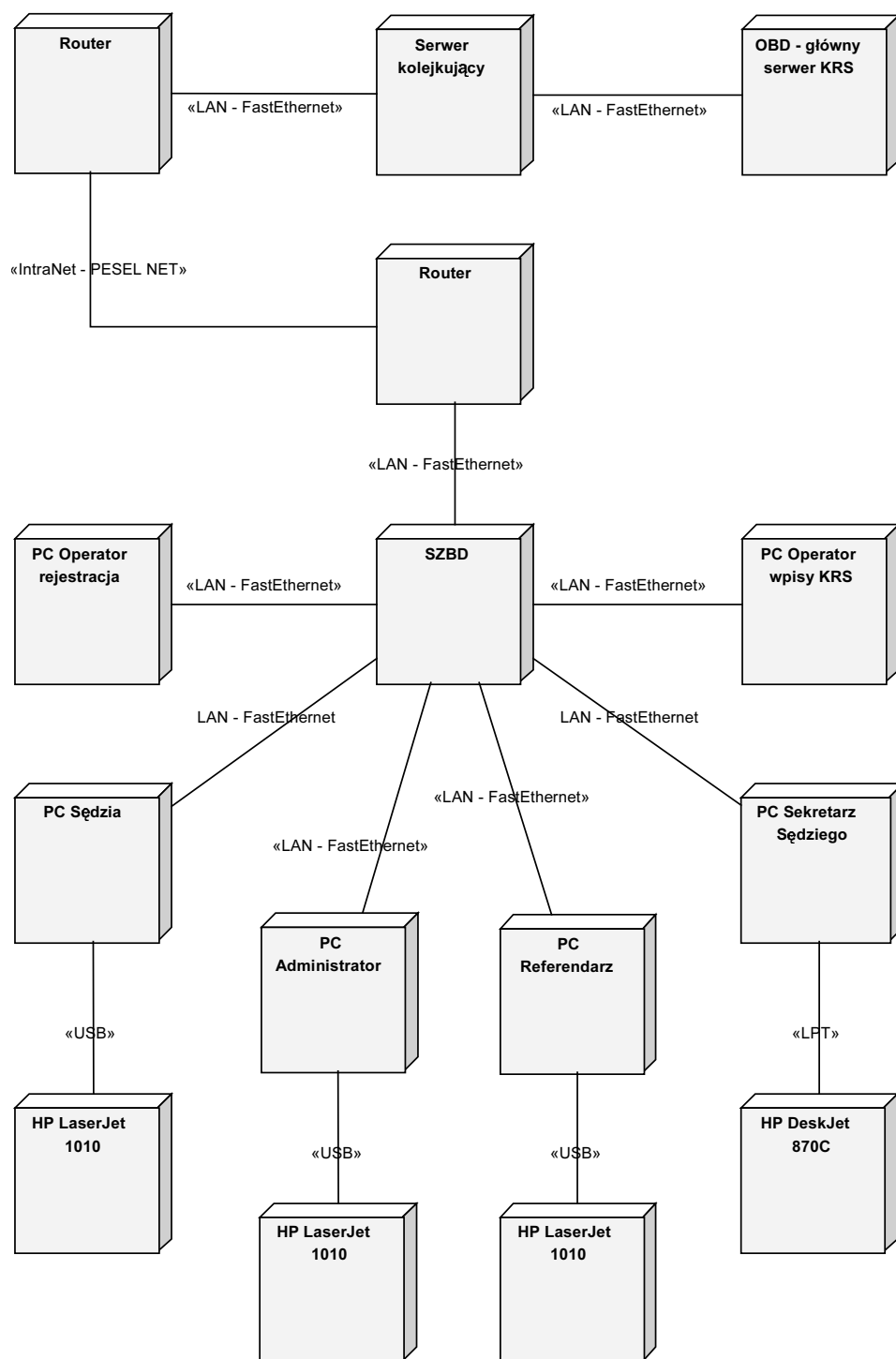
Rysunek 11.15.

Definiowanie
zasad
kolejkowania
w module
systemu WFM



- 37.** Rozbuduj o odpowiednie urządzenia diagram rozlokowania przedstawiony na rysunku 11.13, tak aby po wdrożeniu projektu sieć lokalna zapewniała przepustowość 100 Mbit/s (w routerze należy zainstalować dodatkową kartę pracującą w technologii Fast Ethernet — w celu określenia modelu karty skorzystaj ze strony www.cisco.com). Ponadto:
- ♦ skonfiguruj węzły oznaczone stereotypem «*klient*», obligatoryjnie zamieszczając specyfikację rozlokowania;
 - ♦ zainstaluj na serwerze komponenty, które umożliwią realizację usługi FTP;
 - ♦ zaproponuj odpowiednie artefakty w formacie *.xml*, oraz *.pdf*.
- 38.** Opisz i uzasadnij logikę procesu tworzenia diagramów wdrożeniowych.
- 39.** Odwołując się do rysunku 11.16, przedstawiającego schemat sieci Centrum Ogólnopolskich Rejestrów Sądowych (CORS) i wydziałów Krajowego Rejestru Sądowego (KRS), oraz zamieszczonego poniżej opisu będącego rezultatem wstępnych prac analityka:
- ♦ zaproponuj stosowne komponenty oprogramowania osadzone na poszczególnych węzłach,
 - ♦ określ zależności pomiędzy wyspecyfikowanymi komponentami,
 - ♦ dokonaj uszczegółowienia specyfikacji istniejących węzłów,
 - ♦ zaproponuj specyfikację stacji klienta umożliwiającą zdalne rejestrowanie podmiotów gospodarczych.

Działalność *Wydziału KRS* polega na obsłudze tzw. spraw, którymi mogą być np. rejestracja podmiotu, rejestracja zmian podmiotu czy realizacja postępowania upadłościowego. Pracownicy Wydziału KRS zapewniają obsługę rejestracji spółek (cywilnych, jawnych, komandytowych, z ograniczoną odpowiedzialnością, akcyjnych itd.), jak również spółdzielni, przedsiębiorstw państwowych, jednostek badawczo-rozwojowych, towarzystw ubezpieczeń wzajemnych, fundacji, stowarzyszeń i innych organizacji społecznych lub zawodowych podejmujących działalność gospodarczą. Rejestrację podmiotu zgłasza *Klient* w *Wydziale KRS*, gdzie otrzymuje i może wypełnić odpowiedni wniosek w postaci formularza.



Rysunek 11.16. Diagram rozlokowania wydziału KRS oraz centrum CORS

Następnie *Klient* przekazuje wniosek *Pracownikowi Administracji Wydziału KRS*, który przydziela sprawie osobę odpowiedzialną za jej prowadzenie. Może to być *Sędzia* lub *Referendarz*. Po weryfikacji dokumentów sprawy *Sędzia* orzeka o dalszej drodze jej realizacji. W przypadku braku wymaganych danych *Sędzia* prosi *Sekretarza* o prowadzenie korespondencji z *Klientem* w celu uzyskania niezbędnych danych umożliwiających dalsze prowadzenie sprawy. Po skompletowaniu i zweryfikowaniu wszystkich danych sprawy *Sędzia* wydaje zarządzenie o wpisie projektu do Centrum Ogólnopolskich Rejestrów Sądowych — w skrócie CORS. Za realizację tej decyzji odpowiedzialny jest *Pracownik Administracji*, który przesyła dokumenty sprawy do CORS. W Centrum Ogólnopolskich Rejestrów Sądowych podmiotowi zostaje przydzielony unikalny numer KRS, który zostaje przesłany do *Wydziału KRS*, do *Sekretarza Sędziego*. *Sekretarz* po otrzymaniu korespondencji z numerem KRS jest odpowiedzialny za przesłanie numeru KRS *Klientowi*.

Rozdział 12.

Diagram struktur połączonych

Znaczenie diagramów struktur połączonych

Użytkowanie złożonych systemów informatycznych wiąże się ze specyficznymi potrzebami realizacji częściowych zadań i celów. Diagramy struktury zazwyczaj odpowiadają pełnemu zakresowi dokumentacji statyki danego systemu. W trakcie użytkowania systemu realizowane są funkcje określone przez przypadki użycia, wątki lub operacje. W realizacji tych **specyficznych funkcji** uczestniczą różne kategorie modelowania statyki. Istnieje w związku z tym potrzeba dokumentowania ich współdziałania dla wycinka systemu. I tak dla sprecyzowanego wycinka, modułu systemu:

- ◆ w systemie bankowym opisywanym współdziałaniem spłaty kredytów mogą być monitorowane,
- ◆ w systemie szpitalnym operacje mogą być harmonogramowane,
- ◆ w systemie ratownictwa miejskiego wypadek może być monitorowany.

Do realizacji tego wycinkowego zadania dobiera się zespół współdziałających elementów systemu. Takie rozwiązania istniały w wersjach 1.x języka UML w ramach diagramów współdziałania, charakteryzując się ograniczonym zestawem kategorii pojęciowych. W wersji 2.0 omawianą koncepcję rozwinięto w postaci nowego, **autonomicznego** rodzaju diagramów UML — diagramów struktur połączonych.

Podstawowe kategorie pojęciowe oraz notacja graficzna

Diagramy struktur połączonych ściśle wiążą się z istotą pojęcia **współdziałanie** (ang. *cooperation*).



Współdziałanie jest specyfikacją pożądanego funkcjonalności zestawu powiązanych i współpracujących klasyfikatorów, z których każdy pełni dedykowaną rolę.

Współdziałanie dokumentuje się na **diagramach struktur połączonych**.



Diagram struktur połączonych to graficzne przedstawienie wzajemnie współdziałających części dla osiągnięcia pożądanego funkcjonalności współdziałania.

Kategoriemi pojęciowymi diagramu struktur połączonych są:

- ♦ współdziałanie,
- ♦ część,
- ♦ połączenie,
- ♦ interfejs,
- ♦ konektor składany,
- ♦ port.

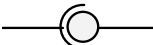
Ich notację graficzną przedstawia tabela 12.1.

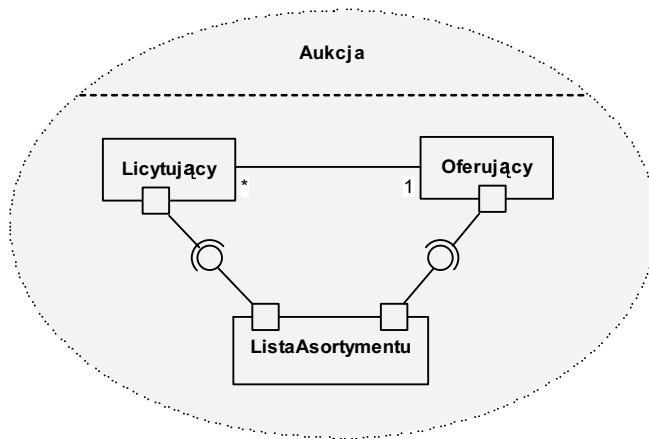
Pojęcie części współdziałania (ang. *part*) obejmuje i uogólnia klasy, obiekty, atrybuty, operacje, przypadki użycia, komponenty, węzły oraz inne współdziałania. Części te pełnią we współdziałaniu określone **role**. Z kolei połączenia (ang. *connections*) mogą przybrać postać asocjacji (najpowszechniej), konektora składanego czy też zależności. Diagramy struktur połączonych wykorzystują alternatywne notacje odpowiadające poziomom:

- ♦ konceptualnemu — rysunek 12.1,
- ♦ implementacyjnemu — rysunek 12.2,
- ♦ wystąpieniowemu — rysunek 12.3.

Przykład diagramu struktur połączonych specyfikującego wewnętrzną strukturę współdziałania zaprezentowano na rysunku 12.1. Przedstawia on współdziałanie *Licytowanie*, będące wycinkiem funkcjonalności aukcji internetowej.

Tabela 12.1. Kategorie modelowania diagramów struktur połączonych

Nazwa	Notacja
Współdziałanie	
Część	
Połączenie	
Interfejs	
Konektor składany	
Port	

Rysunek 12.1.
Diagram struktur połączonych

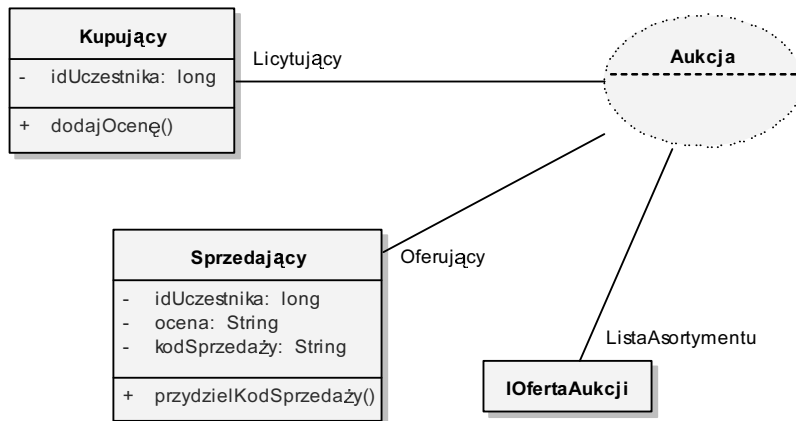
Współdziałanie to składa się z trzech części — *Licytującego*, *Oferującego* oraz *Listy Asortymentu*, powiązanych związkiem asocjacji oraz konektorami składanymi, zawierającymi interfejsy i porty.

W praktyce w systemie:

- ♦ rolę *Licytującego* spełnia obiekt klasy *Kupujący*,
- ♦ rolę *Oferującego* — obiekt klasy *Sprzedający*,
- ♦ rolę *Listy Asortymentu* — interfejs *IOfertaAukcji*.

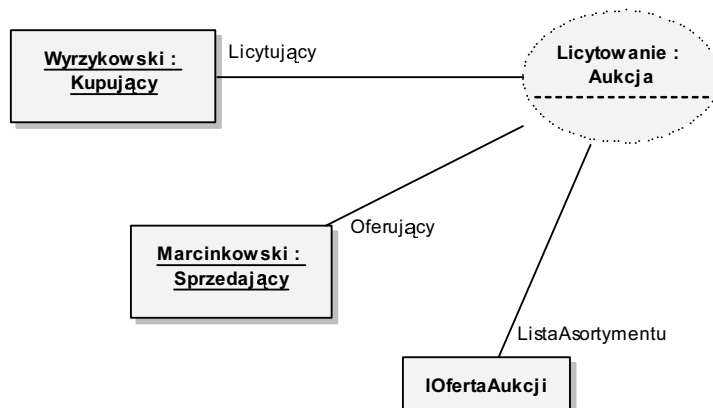
Opisane wykorzystanie konkretnego współdziałania ilustruje rysunek 12.2.

Rysunek 12.2.
Współdziałanie
z przypisanymi
klasyfikatorami
systemu



Tak wyspecyfikowany diagram struktur połączonych jest użytkowany w systemie w postaci konkretnych **instancji** współdziałań, części oraz połączeń. Można to oznaczać na wystąpieniowym diagramie struktur połączonych poprzez podanie nazw ich instancji, np. *Licytowanie : Aukcja* (rysunek 12.3).

Rysunek 12.3.
Licytowanie
jako wystąpienie
współdziałania



Proces tworzenia diagramu struktur połączonych

Proces tworzenia diagramów struktur połączonych można podzielić na następujące **etapy**:

1. określenie zakresu i nazwy współdziałania;
2. identyfikacja poszczególnych części: klas, obiektów, przypadków użycia itd., jak również ich ról;
3. wyspecyfikowanie połączeń;

4. zsyntetyzowanie implementacyjnego diagramu struktur połączonych poprzez przypisanie klasyfikatorów systemu do opracowanego współdziałania oraz wskazanie konkretnych jego wystąpień.

Podstawowe pojęcia

Część	Diagram komponentów
Atrybut	Diagram obiektów
Klasa	Diagram pakietów
Komponent	Diagram rozlokowania
Obiekt	Diagram struktur połączonych
Operacja	Definicja
Przypadek użycia	Proces tworzenia
Węzeł	Połączenie
Diagram dynamiki	Asocjacja
Diagram czynności	Konektor składany
Diagram harmonogramowania	Zależność
Diagram komunikacji	Interfejs
Diagram maszyny stanowej	Udostępniający
Diagram przypadków użycia	Pozyskujący
Diagram sekwencji	Port
Diagram sterowania interakcją	Rola
Diagram struktury	Wątek
Diagram klas	Współdziałanie

Pytania i zadania

1. Uzasadnij tworzenie diagramów struktur połączonych.
2. Jak należy rozumieć pojęcie współdziałania w aspekcie diagramu struktur połączonych?
3. Wskaż kategorie pojęciowe diagramu struktur połączonych.

4. Przedstaw na diagramie współdziałania systemu wspomagającego funkcjonowanie uczelni związane z:
 - ♦ rejestracją studenta,
 - ♦ wypożyczeniem książki z biblioteki uczelnianej,
 - ♦ rozliczeniem indeksów.
5. Na podstawie wybranych interakcji przedstawionych na rysunku 4.26 opracuj stosowny diagram struktur połączonych.
6. Wymień kluczowe etapy procesu tworzenia diagramu struktur połączonych.
7. Opracuj współdziałanie w ramach systemu bankowego dla zadania otwarcia sejf. Koniecznie uwzględnij kwestie bezpieczeństwa oraz konieczność jednoczesnego uzyskania trzech potwierdzeń tego zadania.

Rozdział 13.

Diagram pakietów

Znaczenie diagramów pakietów

Język UML umożliwia tworzenie złożonych, rozbudowanych systemów informatycznych. W tym celu dysponuje szerokim instrumentarium zaawansowanych kategorii modelowania. Ich zastosowanie w odniesieniu do konkretnego projektu powoduje powstanie obszernej i skomplikowanej dokumentacji w postaci różnorodnych modeli i diagramów. Są one połączone licznymi związkami. W dotychczas zaprezentowanym zestawie dwunastu diagramów UML nie ma takiego rodzaju diagramu, który pozwoliłby na **porządkowanie**, grupowanie tej dokumentacji. Dlatego w języku UML zaproponowano specjalny rodzaj diagramów służący do tego celu. Są to diagramy pakietów. Zostały one oficjalnie zdefiniowane w wersji 2.0 języka UML, chociaż były często używane także we wcześniejszych wersjach. Diagramy pakietów mogą być stosowane we wszystkich pięciu perspektywach architektury systemu (por. rysunek 1.4).

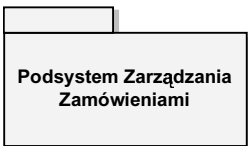
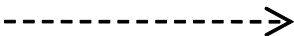
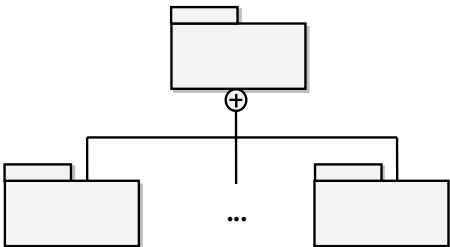
Podstawowe kategorie pojęciowe oraz notacja graficzna

Podstawowe pojęcia diagramów pakietów to:

- ◆ pakiet,
- ◆ zależność,
- ◆ zagnieżdżenie pakietów.

Wymienione pojęcia zilustrowano w tabeli 13.1.

Tabela 13.1. Podstawowe elementy diagramu pakietów

Nazwa	Notacja graficzna
Pakiet	
Zależność	
Zagnieżdżenie pakietów	

Pakiet

Punktem wyjścia w tworzeniu diagramów pakietów jest identyfikacja poszczególnych **pakietów** (ang. *packages*).



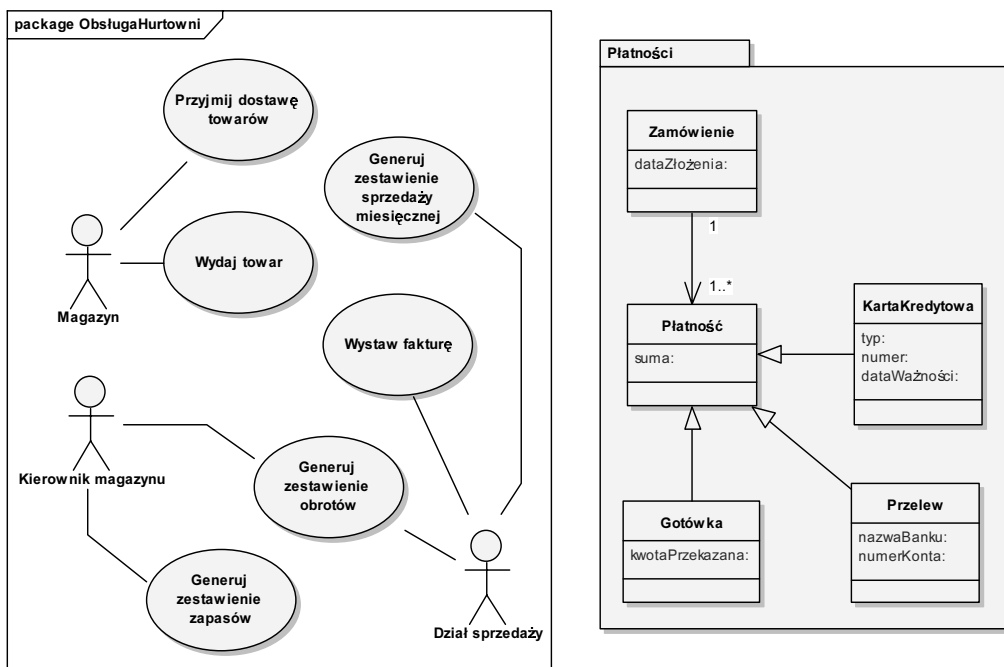
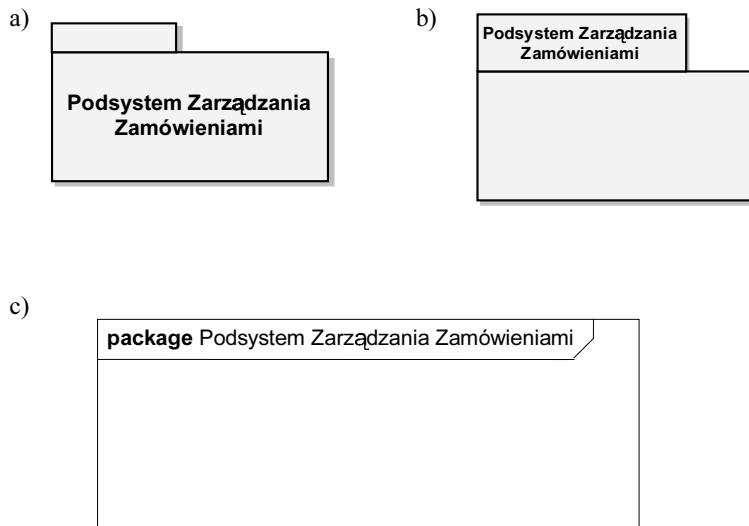
Pakiet jest mechanizmem ogólnego zastosowania, służącym do organizowania elementów w grupy.

Nazwę pakietu umieszcza się wewnątrz symbolu pakietu (rysunek 13.1a), w jego zakładce, jeżeli w symbolu pakietu zamieszczono wiele elementów (rysunek 13.1b), lub w nagłówku ramy zastosowanej do opisu pakietu (rysunek 13.1c). Dokumentowanie pakietów w postaci ramy diagramu umożliwiono wraz z wprowadzeniem wersji 2.0 języka UML, przy czym nazwa pakietu musi być w takim przypadku umieszczana w pentagramie w wolnym miejscu po wyróżniku pakietu. Wykorzystując notację ramy na diagramach pakietów, należy kierować się następującymi zasadami:

- ♦ pojedynczym pakietom przypisuje się wyróżnik *package*;
- ♦ kompletnym diagramom pakietów odpowiednio wyróżnik angielszczyzny *pd* bądź polskojęzyczny *pkt*.

Pakiet może **zawierać** różnorodne klasyfikatory języka UML, kompletne diagramy, biblioteki klas, podsystemy, modele czy też szablony. I tak pakiety zawierające przypadki użycia składające się na system obsługi hurtowni oraz strukturę podsystemu płatności przedstawia rysunek 13.2.

Rysunek 13.1.
Nazwy pakietów



Rysunek 13.2. Pakiety i ich zawartość

Zależność

Podstawowym sposobem łączenia pakietów są związki zależności. Połączenie pakietów w logiczną strukturę z wykorzystaniem zależności skutkuje utworzeniem **diagramu pakietów**.

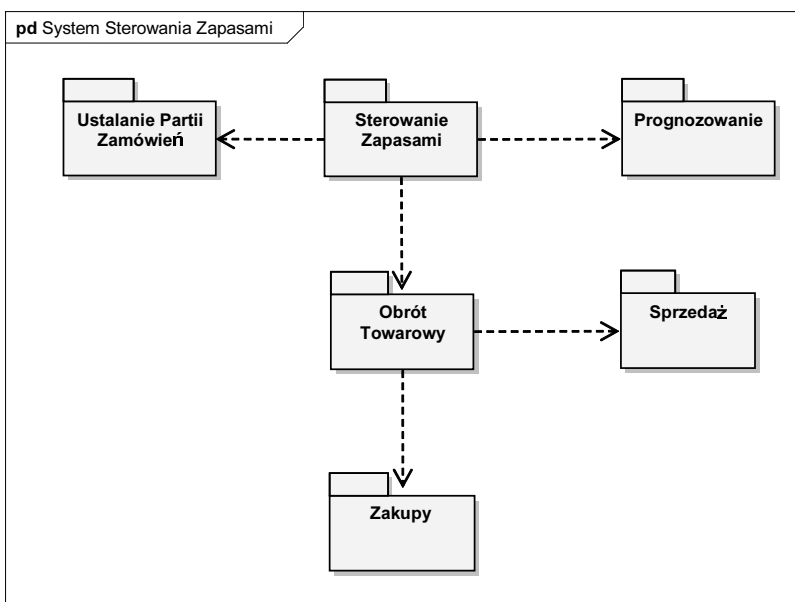


Diagram pakietów to diagram przedstawiający logiczną strukturę dokumentacji systemu w postaci zestawu pakietów połączonych zależnościami i zagnieżdżeniami.

Przykład diagramu pakietów związany z systemem sterowania zapasami przedstawia rysunek 13.3.

Rysunek 13.3.

Diagram pakietów systemu sterowania zapasami

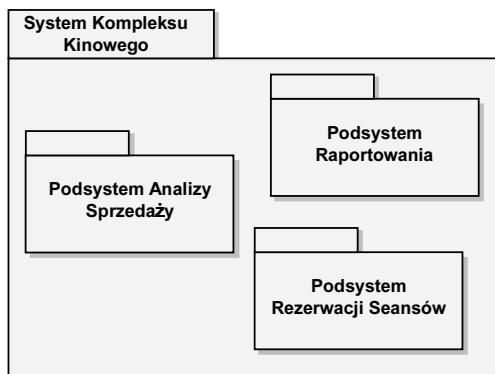


Zagnieżdżenie pakietów

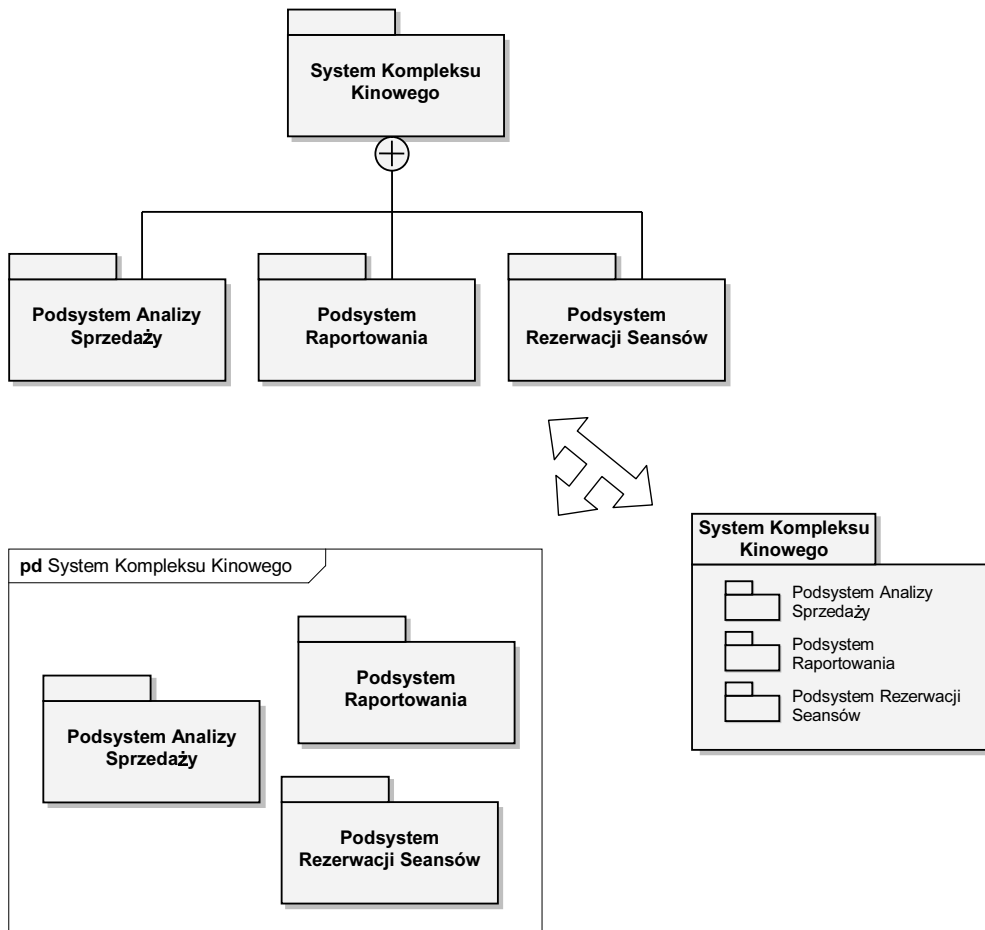
Grupowanie poszczególnych elementów składowych pakietów odbywa się na takiej samej zasadzie jak organizowanie plików w foldery czy katalogi w systemie operacyjnym. Stąd też naturalnym sposobem organizowania pakietów jest ich **zagnieżdżanie** (rysunek 13.4).

Rysunek 13.4.

Zagnieżdżenie pakietów systemu kompleksu kinowego



Alternatywne sposoby dokumentowania zamieszczonego na rysunku 13.4 zagnieżdżenia pakietów pokazano na rysunku 13.5. Zagnieżdżenia pakietów mogą mieć charakter wielopoziomowy. Przykładowo w odniesieniu do rysunku 13.4 zagnieżdżonym pakietem *Podsystemu Rezerwacji Seansów* mógłby być *Podsystem Generowania Oferty dla Klienta Grupowego*.



Rysunek 13.5. Alternatywne notacje dokumentowania zagnieżdżeń pakietów

Zaawansowane składniki diagramu

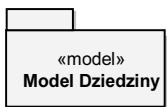
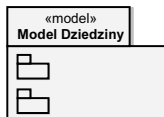
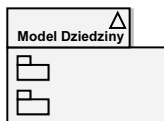
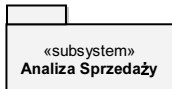
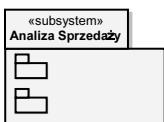
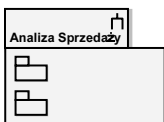
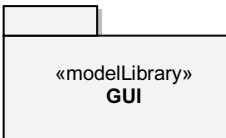
Zarówno pakiety, jak i zależności różnią się w swej funkcjonalności, toteż w diagramach pakietów mogą one być **stereotypowane**. Do zaawansowanych kategorii pojęciowych diagramu pakietów zalicza się tym samym:

- ♦ stereotypowanie pakietów,
- ♦ stereotypowanie zależności.

Stereotypowanie pakietów

Powszechnie stosowanymi stereotypami pakietów są stereotypy modelu, podsystemu, szablonu i biblioteki klas (tabela 13.2).

Tabela 13.2. Stereotypy pakietów

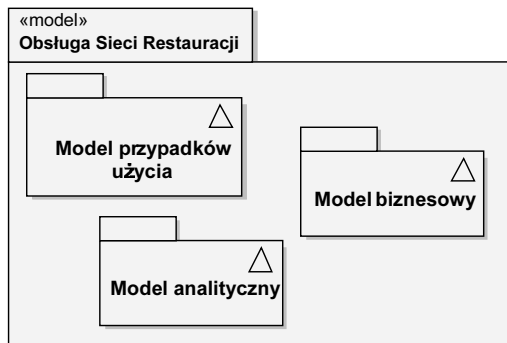
Nazwa	Stereotyp	Interpretacja	
Model			Stereotyp «model» opisuje pakiety będące modelami, czyli odwzorowaniami, abstrakcjami rzeczywistości. Modele te koncentrują się na całościowym przedstawieniu systemu z określonego punktu widzenia, np. użytkownika, menedżera czy też analityka. Typowe stereotypowane pakiety modeli reprezentują modele przypadków użycia, biznesowe bądź analityczne. Pakiet ten można też stereotypować graficznie za pomocą ikony .
			
Podsystem			Stereotyp «subsystem» jest szczególnie użyteczny w modelowaniu systemów zdekomponowanych na hierarchicznie uporządkowane pakiety. Alternatywnym graficznym stereotypem podsystemu jest ikona .
			
Zrąb			Stereotyp «framework» opisuje pakiet będący wzorcem architektonicznym, czyli elastycznym szablonem rozwiązań problemów z danej dziedziny. Zręby typowo obejmują klasy, wzorce i szablony. Zrąb specyficzny dla określonego typu zastosowań nazywany jest zrębem zastosowań.
Biblioteka klas			Stereotyp «modelLibrary» opisuje pakiet grupujący elementy systemu informatycznego, które z założenia mają być użytkowane przez inne pakiety. Pakiet taki jest analogią biblioteki klas w obiektowym języku programowania.

Obok stereotypu podsystemu duże znaczenie praktyczne dla wszystkich uczestników procesu tworzenia systemu informatycznego ma stereotyp modelu. Na rysunku 13.6, który odnosi się do pakietu *Obsługa sieci restauracji*, przedstawiono stereotypowane pakiety modeli:

- ♦ przypadków użycia,
- ♦ analitycznego,
- ♦ biznesowego.

Rysunek 13.6.

Modele obsługi
sieci restauracji
jako pakiety
stereotypowane



Stereotypowanie zależności

Zależności pomiędzy pakietem źródłowym a docelowym opisywane są z wykorzystaniem stereotypów «import», «merge» oraz «access». Ich interpretację przedstawia tabela 13.3.

Tabela 13.3. Stereotypy zależności w diagramach pakietów

Nazwa	Stereotyp	Interpretacja
Import		Stereotyp «import» opisuje powszechnie stosowaną zależność, oznaczającą włączanie klasyfikatorów, będących zawartością pakietu docelowego (<i>Sprzedaż</i>) do pakietu źródłowego (<i>Obsługa Sieci Restauracji</i>) w trakcie użytkowania systemu.
Scalenie		Zależność stereotypowana «merge» prowadzi do scalenia klasyfikatorów, tworzących zawartość pakietu docelowego, z klasyfikatorami będącymi zawartością pakietu źródłowego. Klasyfikatory pakietu źródłowego (<i>Lokalna Baza Danych</i>) dziedziczą po klasyfikatorach pakietu docelowego (<i>Rozproszona Baza Danych</i>). Scalenie jest realizowane wyłącznie w przypadku klasyfikatorów tego samego typu o tożsamyh nazwach, mających widoczność <i>public</i> (+).
Dostęp		Zależność stereotypowana «access» oznacza możliwość dostępu pakietu źródłowego (<i>Udzielanie Kredytów</i>) do zawartości pakietu docelowego (<i>Obsługa Należności</i>) bez fizycznego pobierania danych.

W wersjach 1.x języka UML istniała możliwość stosowania związku uogólnienia pomiędzy pakietami. W wersji 2.0 ta kategoria pojęciowa została w pełni zastąpiona przez zależności stereotypowane «import» oraz «merge».

Istnieje możliwość odwoływania się do **wybranych** klasyfikatorów pakietu docelowego wyspecyfikowanych z wykorzystaniem ograniczenia. Ograniczenie to jest umieszczane po nazwie pakietu i alternatywne wobec stereotypowanego związku zależności. Ma ono następującą składnię:

```
{import <nazwa-ścieżkowa>}  
{access <nazwa-ścieżkowa>}
```

lub

```
{merge <nazwa-ścieżkowa>}
```

I tak przykład diagramu pakietów, ilustrującego opisane rodzaje zależności, został zaprezentowany w punkcie *Studium diagramu pakietów* na rysunku 13.7.

Proces tworzenia diagramu pakietów

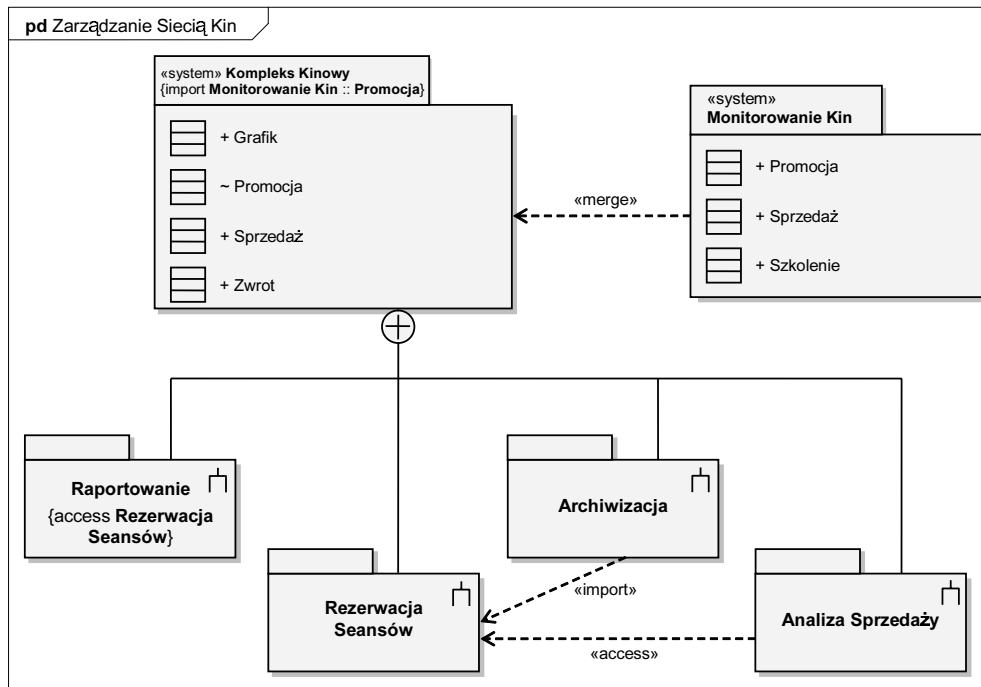
Proces tworzenia diagramów pakietów przebiega w następującej sekwencji:

1. identyfikacja i nazwanie pakietów;
2. zakwalifikowanie zidentyfikowanych pakietów z zastosowaniem odpowiednich stereotypów graficznych lub tekstowych: *subsystem*, *framework*, *model*;
3. określenie zagnieżdżeń pakietów;
4. powiązanie pakietów z wykorzystaniem związku zależności;
5. wyspecyfikowanie zależności poprzez nadanie im stosownych stereotypów: *import*, *merge*, *access*.

Naturalnie proces tworzenia diagramów pakietów powinien być realizowany zgodnie z koncepcją iteracyjno-przyrostowego cyklu życia systemu informatycznego.

Studium diagramu pakietów

Studium diagramu pakietów, zawierające podstawowe oraz zaawansowane kategorie modelowania diagramu pakietów, zaprezentowano na rysunku 13.7. Przykład ilustruje zarządzanie siecią kin. W praktyce system *Kompleks Kinowy* oraz jego podsystemy są zainstalowane na serwerach poszczególnych kompleksów kinowych, natomiast system *Monitorowanie Kin* jest aplikacją zainstalowaną w centralnym oddziale firmy. Wszystkie wymienione aplikacje umożliwiają efektywne zarządzanie siecią kin. Częściami pakietu *Kompleks Kinowy* są następujące podsystemy:



Rysunek 13.7. System zarządzania siecią kin

- ♦ *Raportowanie*,
- ♦ *Rezerwacja Seansów*,
- ♦ *Archiwizacja*,
- ♦ *Analiza Sprzedaży*.

Graficzny stereotyp zamieszczony w prawym górnym narożniku poszczególnych pakietów lub ich zakładki jest charakterystyczny dla podsystemów. Alternatywnie można stosować stereotyp tekstowy *«podsystem»*. Zależność stereotypowana *«import»* pozwala podsystemowi *Archiwizacja* tworzyć kopie zapasowe wszystkich klas pakietu *Rezerwacja Seansów*. Alternatywną formą wyrażenia tej zależności jest zamieszczenie przy nazwie pakietu *Archiwizacja* ograniczenia zgodnie z adekwatną składnią. Formę tę zastosowano przy imporcie klasy *Promocja* z pakietu *Monitorowanie Kin* do pakietu *Komplex Kinowy*.

W praktyce specjaliści, analitycy sprzedaży, przygotowują oferty promocyjne w aplikacji *Monitorowania Kin* dla poszczególnych *Kompleksów Kinowych*. Oferty te są przechowywane jako obiekty klasy *Promocja*. Opisana konwencja zapisu importowania wskazuje, że pozostałe klasy pakietu *Monitorowanie Kin*, tj. *Sprzedaż* i *Szkolenie*, nie są importowane. Związek zależności stereotypowany *«access»* umożliwia dostęp do wszystkich elementów podsystemu. I tak podsystem *Analiza Sprzedaży* jest aplikacją wykorzystującą technologię OLAP. Zatem niezbędny jest dostęp do wszystkich danych podsystemu *Rezerwacja Seansów*, który przechowuje informacje będące podstawą przeprowadzanych analiz. Podobnie jak w przypadku importowania, istnieje możliwość

zapisu dostępu do elementów pakietu w formie ograniczenia. Składnia jest identyczna jak w przypadku alternatywnej formy zapisu importu. Dostęp pakietu *Raportowanie* do *Rezerwacji Seansów* to stosowny przykład takiego zapisu. W zarządzaniu siecią kin wykorzystano ponadto związek zależności stereotypowany «merge». Umożliwia on scalenie klasy *Sprzedaż* systemu *Monitorowania Kin* z klasą o tej samej nazwie, będącą elementem pakietu *Kompleks Kinowy*. Scalenie tych klas jest niezbędne, aby system *Monitorowania Kin* mógł realizować strategiczne decyzje menedżerów wyższego szczebla. Decyzje te są podejmowane na podstawie szczegółowych analiz danych sprzedaży wszystkich *Kompleksów Kinowych*. W momencie odwołania się do klasy *Sprzedaż* dowolnego systemu *Kompleks Kinowy* klasa o tej samej nazwie w systemie *Monitorowania Kin* jest scalana i rozszerzana o atrybuty i (lub) operacje.

Należy zauważyć, że oba wspomniane pakiety przechowują inną klasę o zgodnej nazwie, tj. *Promocja*. Podczas scalenia klasa ta nie zostaje jednak uwzględniona, ponieważ widoczność klasy *Promocja* w systemie *Kompleksu Kinowego* jest inna niż publiczna. Naturalnie, scaleniu nie podlegają również klasy nie mające odpowiedników o tej samej nazwie w zależnych pakietach, tj. *Szkolenie*, *Grafik* oraz *Zwrot*.

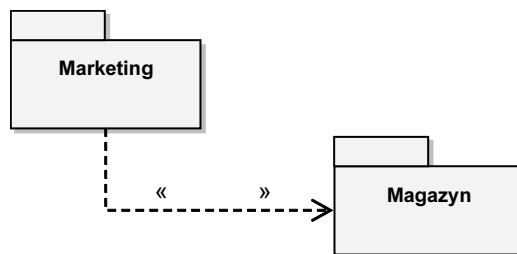
Podstawowe pojęcia

Diagram pakietów	Pakietu
Definicja	Biblioteka klas «modelLibrary»
Proces tworzenia	Model «model»
Klasyfikator	Podsystem «subsystem»
Model	Szablon «framework»
Analityczny	Tekstowy
Biznesowy	Zależności
Dziedziny	Dostęp «access»
Przypadków użycia	Import «import»
Pakiet	Scalenie «merge»
Zagnieżdżanie	Zależność
Stereotyp	
Graficzny	

Pytania i zadania

1. Wyjaśnij celowość zastosowania diagramów pakietów.
2. Co może stanowić zawartość pakietu?
3. Jak oznacza się graficznie pakiety i ich zagnieżdżenia? Podaj przykłady.

4. Wykorzystując różne notacje graficzne, sporządź zagnieżdżoną strukturę pakietów na podstawie rysunku 13.3. W tym celu wprowadź dodatkowe pakiety i przegrupuj zawartość pakietów zaprezentowanych na diagramie.
5. Uzasadnij znaczenie związku zależności w odniesieniu do diagramów pakietów.
6. Przeanalizuj funkcjonowanie uczelni wyższej. Zidentyfikuj przypadki użycia systemu informatycznego wspomagającego jej funkcjonowanie oraz pogrupuj je w pakiety.
7. Jakiego rodzaju stereotypy można odnieść do pakietów? Wyjaśnij różnice pomiędzy modelem a podsystemem.
8. Zaproponuj stosowne modele dla pakietu *Sterowanie Zapasami* (rysunek 13.3).
9. Przedstaw zagnieżdżenie pakietów z rysunku 13.4, używając znanych Ci stereotypów podsystemu.
10. Zaproponuj na podstawie rysunku 13.4 zagnieżdżenia dalszego poziomu.
11. Zamierzasz stworzyć aplikację w języku JAVA wykorzystującą funkcje graficzne i operacje dyskowe. Jakie zastosowanie w tym kontekście mają pakiety stereotypowane «*modelLibrary*»?
12. Jakie rodzaje stereotypów zależności występują na diagramach pakietów? Połącz zidentyfikowane pakiety systemu informatycznego wspomagania uczelni z wykorzystaniem stereotypowanych zależności.
13. Przedstaw stereotypy związków zależności «*import*» i «*access*», wykorzystując nazwy ścieżkowe.
14. Poniżej zamieszczono związek zależności pomiędzy dwoma pakietami systemu informatycznego dużego przedsiębiorstwa. Rozważ, które ze znanych Ci stereotypów związku zależności stosowanego na diagramie pakietów znalazłyby zastosowanie w konkretnych sytuacjach. Wypowiedź uzasadnij.



15. Wyjaśnij, dlaczego w wersji 2.0 języka UML nie występuje konieczność stosowania związku uogólnienia pomiędzy pakietami.
16. Wymień etapy procesu tworzenia diagramu pakietów.

Część II

Metodyki, modele i narzędzia wspomagające UML 2

Rozdział 14.

Metodyka RUP

Znaczenie iteracyjno-przyrostowego procesu projektowania systemów

Zagadnienia metodologiczne tworzenia systemów informatycznych stanowiły jeden z głównych nurtów dyscypliny analizy i projektowania systemów informatycznych już od lat 80. Na bazie różnych definicji można stwierdzić, że **metodyka** (ang. *methodology*) tworzenia systemów informatycznych (TSI) stanowi spójny, logicznie uporządkowany zestaw metod i procedur o charakterze technicznym i organizatorskim, pozwalających zespołowi projektowemu realizować cykl życia systemu. Ogólnie metodyki można podzielić na strukturalne, obiektowe i społeczne [Wrycza-1999].

Podobnie jak w przypadku metodyk strukturalnych, pojawił się szereg propozycji związanych z modelowaniem obiektowym. Największe uznanie zdobyły podejścia OMT, OOAD i OOSE, inkorporujące pewne elementy metodyki. Pierwsza próba unifikacji obiektowego procesu tworzenia systemu nosiła nazwę USDP (*Unified Software Development Process*). Autorami USDP są I. Jacobson, G. Booch oraz J. Rumbaugh [Booch-1998]. Jest to metodyka rodzajowa (ang. *generic*), stwarzająca możliwość opracowywania różnych jej konfiguracji i implementacji. Do głównych **cech** tego podejścia należą:

- ◆ ukierunkowanie na przypadki użycia,
- ◆ potraktowanie architektury systemu jako centralnego zagadnienia w procesie tworzenia oprogramowania,
- ◆ iteracyjno-przyrostowy charakter procesu tworzenia systemu.

Implementacją tego podejścia jest metodyka **RUP** (*Rational Unified Process*), merytorycznie i pojęciowo związana z kategoriami USDP. Metodyka RUP czerpie obszernie ze zbioru praktyk korporacji Rational (*Rational Approach*) oraz podejścia *Objectory*, zainicjowanego przez I. Jacobsona. Stopniowo udoskonalana i rozbudowywana, w roku 2003 stała się własnością firmy IBM. Metodyka RUP zdobyła znaczne uznanie i obecnie dominuje w obszarze obiektowych metodyk tworzenia systemów.



RUP to metodyka tworzenia systemów informatycznych oparta na paradygmacie obiektowości i języku UML, zaproponowana przez korporację Rational Software.

Jako kompletna metodyka RUP zawiera następujące **elementy**:

- ♦ przyrostowo-iteracyjny cykl życia systemu;
- ♦ pojęcia, metody i techniki z zakresu języka UML i inne, w tym heurystyczne;
- ♦ zintegrowany pakiet narzędzi CASE, wspomagający stosowanie elementów metodyki;
- ♦ role zdefiniowane w zespole projektowym i procedury zarządzania projektem;
- ♦ kryteria oceny i monitorowania postępu prac;
- ♦ hipertekstową bazę wiedzy;
- ♦ internetowy serwis wspomagający metodykę i jej użytkowników.

Metodyka RUP jest ukształtowana w sposób umożliwiający **dopasowanie procesu tworzenia systemu** do potrzeb konkretnego przedsięwzięcia na podstawie pełnego spektrum swoich możliwości. W efekcie RUP jest elastyczny — znajduje zastosowanie zarówno przy mniejszych, jak i dużych projektach informatycznych.

Ścieżkę ewolucyjną rozwoju koncepcji metodyki RUP przedstawia rysunek 14.1.

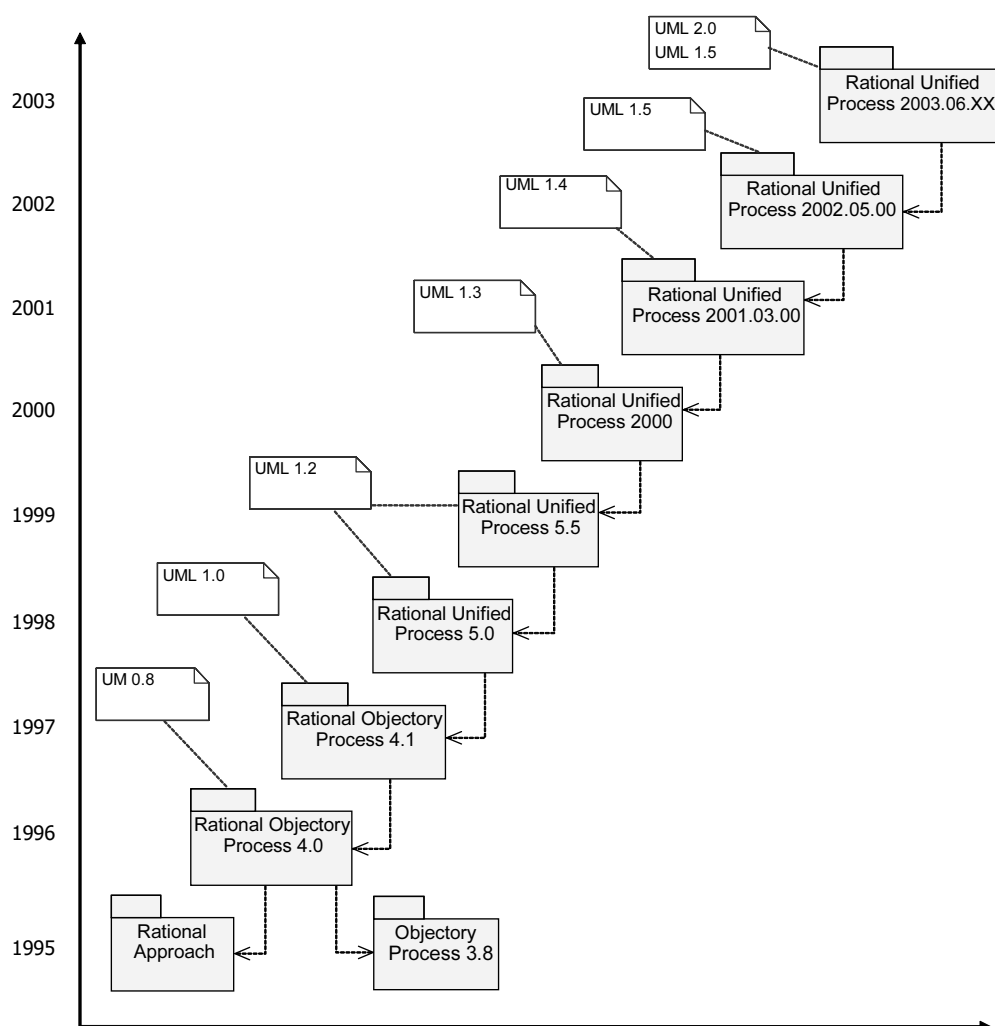
Najważniejsze **obszary zmian** w zakresie metodyki związane były z:

- ♦ rozwijaniem poszczególnych dyscyplin, w tym przepływów pracy oraz dotyczących ich dokumentów;
- ♦ wprowadzeniem modułów dodatkowych (ang. *plug-ins*) RUP oraz integracją i włączaniem nowych narzędzi CASE, wspomagających metodykę RUP;
- ♦ reorganizacją układu hipertekstowej bazy wiedzy;
- ♦ zdefiniowanymi w metodyce artefaktami — modelami i dokumentami.

Założenia rodzajowej metodyki USDP, iteracyjno-przyrostowy cykl życia systemu, a także kategorie modelowania języka UML wykorzystywane są przez inne, aktualnie rozwijane metodyki obiektowe. Do najbardziej interesujących należą:

- ♦ XP (*Extreme Programming*),
- ♦ Agile,
- ♦ Scrum,
- ♦ DSDM (*Dynamic Systems Development Method*).

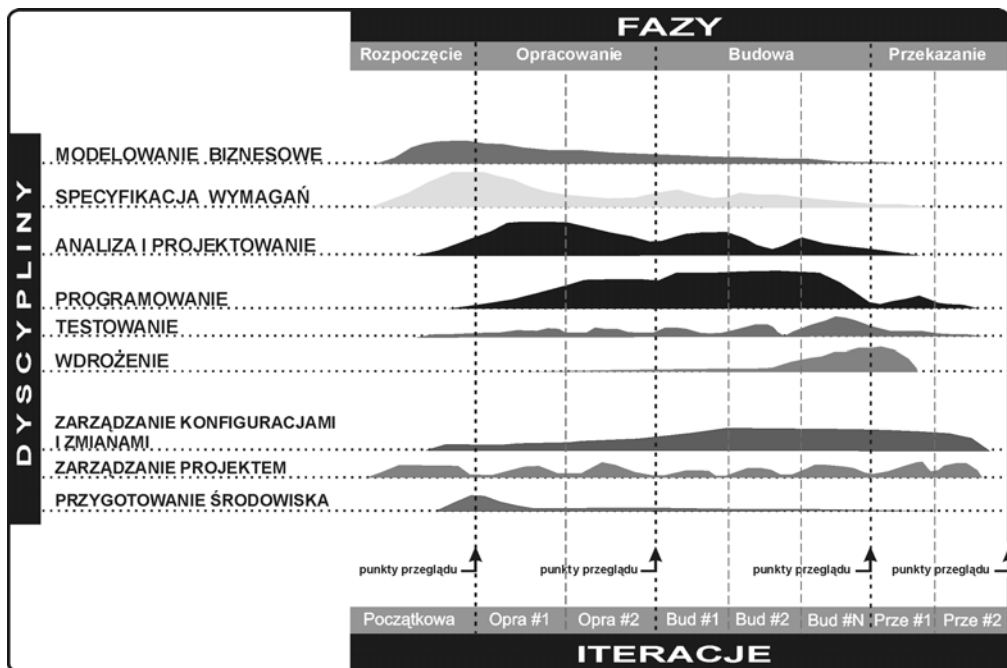
Istotną cechą wymienionych metodyk jest ograniczenie czasu przeznaczonego na modelowanie systemu i przygotowywanie dokumentacji analityczno-projektowej w celu szybkiego przejścia do tworzenia kodu źródłowego przyszłego systemu. Z tego powodu określane są one czasami mianem metodyk „zwinnych” lub „żwawych”. Metodyki te omówione są w licznych opracowaniach, np. [Dennis-2005] oraz [Boehm-2004].



Rysunek 14.1. Ewolucja metodyki RUP

Struktura RUP

Przez kilka dekad dominującym wzorcem cyklu życia systemu był cykl liniowy, a później spiralny. Metodyka RUP opiera się na zasadniczym novum — **iteracyjno-przyrostowym** (ang. *iterative-incremental*) cyklu życia systemu. Ma on postać dwuwymiarową (rysunek 14.2).



Źródło: opracowanie własne na podstawie [RUP-2003]

Rysunek 14.2. Iteracyjno-przyrostowy cykl życia systemu w metodyce RUP

Linia pozioma reprezentuje czas, a zatem **dynamiczny aspekt** procesu TSI, uwzględniając:

- ♦ fazy,
- ♦ iteracje,
- ♦ punkty przeglądu.

Z kolei linia pionowa odzwierciedla **statyczny aspekt** procesu TSI, tj. jego opis w kategoriach:

- ♦ dyscyplin,
- ♦ czynności,
- ♦ artefaktów,
- ♦ ról,
- ♦ przepływów pracy.

Tak więc linia pozioma odpowiada fazom cyklu życia systemu, natomiast pionowa jego dyscyplinom. W nawiązaniu do kategorii linii pionowej pojęcie dyscypliny scharakteryzowano poniżej — głównie w tabeli 14.1. Pojęcie czynności i artefaktu wyjaśniono już w rozdziałach 4. oraz 11.

Rola oznacza obowiązki i kompetencje osoby lub zespołu zaangażowanego w proces tworzenia systemu informatycznego lub jego wycinka. W wyniku realizacji roli powstają oczekiwane artefakty. Spektrum ról, odpowiadających często współczesnym zawodom informatycznym, jest bardzo szerokie i wiąże się z możliwościami zaawansowanych technologii informatycznych. Przykładami ról w metodyce RUP są:

- ♦ menedżer projektu,
- ♦ inżynier procesu,
- ♦ analityk procesów biznesowych,
- ♦ analityk systemowy,
- ♦ projektant interfejsu użytkownika,
- ♦ projektant bazy danych,
- ♦ architekt oprogramowania,
- ♦ programista,
- ♦ projektant testów,
- ♦ testujący,
- ♦ specjalista ds. narzędzi CASE.

Ostatnim ze strukturalnych elementów opisu iteracyjno-przyrostowego procesu jest **przebieg pracy**, który stanowi sekwencję czynności, w wyniku której powstaje lub jest przetwarzany artefakt.

Dyscypliny

W metodyce RUP wyróżnia się dziewięć **dyscyplin** (ang. *disciplines*), które mają charakter podstawowy lub wspomagający.



Dyscyplina jest kolekcją powiązanych czynności, artefaktów, ról oraz przebiegów pracy odpowiadających tematycznie głównym obszarom tworzenia systemów informatycznych.

Dyscypliny podstawowe stanowią rdzeń procesu tworzenia systemu. Należą do nich:

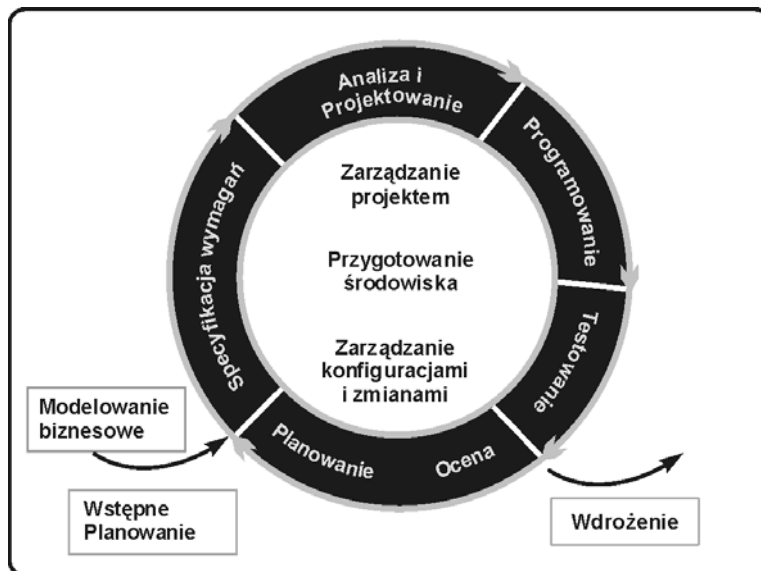
- ♦ modelowanie biznesowe,
- ♦ specyfikacja wymagań,
- ♦ analiza i projektowanie,
- ♦ programowanie,
- ♦ testowanie,
- ♦ wdrożenie.

Dyscypliny wspomagające realizują funkcje zarządcze i konfiguracyjne w procesie tworzenia systemu. Można do nich zaliczyć:

- ♦ zarządzanie konfiguracjami i zmianami,
- ♦ zarządzanie projektem,
- ♦ przygotowanie środowiska.

Układ i zależności pomiędzy wymienionymi dyscyplinami w metodyce Rational Unified Process w odniesieniu do każdej iteracji przedstawia rysunek 14.3.

Rysunek 14.3.
Układ dyscyplin
cyklu życia RUP
w iteracji



Z realizacją każdej iteracji wiążą się kwestie wstępnego planowania jej przebiegu oraz oceny wyników i ryzyka. Cztery fazy cyklu iteracyjno-przyrostowego zawierają iteracje. Każda z tych iteracji prowadzi do opracowania kolejnej **wykonuwalnej wersji systemu**. Wersja ta w następnych iteracjach jest uzupełniana o kolejne elementy wynikające z dyscyplin, czynności i przepływów pracy. Poszczególne czynności są wykonywane przez role, które tworzą czy też rozwijają artefakty. Poziom zaawansowania kolejnych wersji systemu określany jest przez nazewnictwo iteracji. I tak iteracja *Opra#1* oznacza system w stanie pierwszej iteracji z fazy opracowania, a iteracja *Bud#N* oznacza system w stanie n-tej iteracji fazy budowy. Dominującym czynnikiem, decydującym o kierunku dokonywanych zmian w kolejnych iteracjach, jest uwzględnienie i zapobieganie ryzyku w tworzonego systemie. Ogólną charakterystykę zakresu tematycznego poszczególnych dyscyplin zawiera tabela 14.1.

Tabela 14.1. Podstawowy zakres merytoryczny dyscyplin w procesie RUP

Dyscyplina	Zakres
Modelowanie biznesowe (ang. <i>business modeling</i>)	Zawiera wizję nowej, docelowej organizacji, definicje występujących w jej ramach procesów, ról i zakresów odpowiedzialności; informacje te przedstawia się w postaci biznesowych diagramów UML (por. rozdział 15.).
Specyfikacja wymagań (ang. <i>requirements</i>)	Oznacza opracowanie wizji systemu, modelu przypadków użycia i zdefiniowanie wymagań нефункциональных.
Analiza i projektowanie (ang. <i>analysis & design</i>)	Zawiera analizę i projekt systemu z wykorzystaniem całego spektrum diagramów UML.
Programowanie (ang. <i>implementation</i>)	Pozwala na opracowanie kodu źródłowego w wybranym języku programowania oraz kompilację kodu i integrację komponentów.
Testowanie (ang. <i>test</i>)	Oznacza planowanie testów oraz ocenę systemu poprzez wykonanie szeregu testów.
Wdrożenie (ang. <i>deployment</i>)	Dotyczy instalacji oprogramowania systemu i formalną akceptację kolejnych wersji systemu przez klienta czy użytkownika.
Zarządzanie konfiguracjami i zmianami (ang. <i>configuration & change management</i>)	Obejmuje kontrolę wersji artefaktów opracowanych podczas kolejnych iteracji.
Zarządzanie projektem (ang. <i>project management</i>)	Oznacza planowanie i kontrolę realizacji projektu.
Przygotowanie środowiska (ang. <i>environment</i>)	Obejmuje przygotowanie infrastruktury dla skutecznej realizacji projektu.

Źródło: opracowanie własne na podstawie [RUP-2003]

Fazy

Istota cyklu iteracyjno-przyrostowego opiera się więc na zależnościach pomiędzy dyscyplinami (scharakteryzowanymi w tabeli 14.1) a fazami (ang. *phases*).



Fazą w metodyce RUP jest okres między kolejnymi punktami przeglądu cyklu iteracyjno-przyrostowego, w którym zrealizowano niezbędne czynności i opracowano adekwatne artefakty.

Metodyka RUP wprowadza cztery **fazy** (rysunek 14.2):

- ♦ rozpoczęcie (ang. *inception*),
- ♦ opracowanie (ang. *elaboration*),
- ♦ budowa (ang. *construction*),
- ♦ przekazanie (ang. *transition*).

System informatyczny jest rozwijany w kolejnych **iteracjach** (ang. *iterations*), występujących w ramach każdej z wymienionych faz. Przejście pomiędzy iteracjami poprzedza przyrostowa integracja artefaktów otrzymanych we wszystkich dotychczasowych iteracjach.



Iteracja w metodyce RUP to pojedynczy cykl w ramach fazy, polegający na realizacji czynności poszczególnych dyscyplin; jej efektem jest kolejny przyrost systemu.

W miarę realizacji kolejnych iteracji powstają kolekcje powiązanych ze sobą artefaktów stanowiących wersje projektu systemu. Początkowo mają ogólny charakter, są wzbogacane i udoskonalane w kolejnych iteracjach. Zastosowanie cyklu iteracyjno-przyrostowego znacząco zmniejsza ryzyko niepowodzenia projektu w porównaniu z cyklem liniowym. Wynika to z możliwości przeprowadzania oceny kolejnych wersji systemu nie tylko pod kątem technicznym, lecz również pod kątem zgodności z wymaganiami użytkownika. Iteracyjne podejście umożliwia ponadto szybkie wprowadzenie nowych wymagań i sprawną modyfikację istniejących, a także niezwłoczną reakcję na obie te zmiany. Liczba iteracji uzależniona jest od złożoności tworzonego systemu oraz czasochłonności poszczególnych faz. I tak dla **kryterium czasu** tworzenia typowego systemu informatycznego przyjmuje się uśrednione szacunki [Ariadne-2002]:

- ♦ na fazę rozpoczęcia poświęca się 10% czasu,
- ♦ na fazę opracowania — 30%,
- ♦ na fazę budowy — 50%,
- ♦ na fazę przekazania — 10%.

W związku z tym liczba niezbędnych iteracji w fazie budowy jest znacząco większa niż liczba iteracji rozpoczęcia czy przekazania. Mechanizm funkcjonowania cyklu iteracyjno-przyrostowego polega na tym, że w każdej z tych iteracji użytkowane są adekwatne artefakty z poszczególnych dyscyplin. W każdej iteracji mamy tym samym do czynienia z **mini-cykiem liniowym**. W odróżnieniu od klasycznego liniowego cyklu życia systemu, w fazach cyklu życia RUP czynności poszczególnych dyscyplin często wykonywane są równolegle. Dzięki powstającym artefaktom określone funkcje systemu mogą być wdrożone i ocenione po wykonaniu kolejnych iteracji. Zmniejsza to radykalnie ryzyko niepowodzenia projektu, gdyż w tradycyjnym, liniowym podejściu gotowy system pojawia się na końcu pełnego cyklu życia systemu.

Cele poszczególnych faz sprecyzowano w tabeli 14.2.

W wyniku oceny związanej z użytkowaniem systemu następuje jego modyfikacja. W ten sposób po następujących po sobie cyklach użytkowania powstają kolejne **generacje** systemu. I tak przykładowe oznaczenie systemu w postaci „2.7” wskazuje na siódmą wersję w ramach drugiej generacji systemu.

Tabela 14.2. *Fazy w metodyce RUP i ich cele*

Faza	Cele
Rozpoczęcie (ang. <i>inception</i>)	Wypracowanie ogólnej wizji przedsięwzięcia oraz osiągnięcie zrozumienia i akceptacji projektu ze strony wszystkich jego uczestników.
Opracowanie (ang. <i>elaboration</i>)	Ustalenie architektury systemu, stworzenie planu projektu oraz wyeliminowanie elementów wysokiego ryzyka z projektu.
Budowa (ang. <i>construction</i>)	Stworzenie systemu na podstawie przyjętej architektury.
Przekazanie (ang. <i>transition</i>)	Dostarczenie gotowego systemu użytkownikom czy klientom.

Podstawowe pojęcia

Architektura systemu	Faza
Artefakt	Budowa
CASE	Opracowanie
Cykl życia systemu	Przekazanie
Iteracyjno-przyrostowy	Rozpoczęcie
Liniowy	Generacja
Spiralny	Iteracja
Czynność	Punkt przeglądu
Dyscyplina	Mini-cykl liniowy
Podstawowa	Przepływ pracy
Analiza i projektowanie	Rola
Modelowanie biznesowe	RUP
Programowanie	Ewolucja
Specyfikacja wymagań	Metodyka
Testowanie	Agile
Wdrożenie	DSDM
Wspomagająca	Scrum
Zarządzanie konfiguracjami i zmianami	USDP
Zarządzanie projektem	XP
Przygotowanie środowiska	Tworzenie systemów informatycznych
	Zespół projektowy

Pytania i zadania

1. Opisz strukturę RUP.
2. Wymień dyscypliny wspomagające RUP.
3. Korzystając z literatury przedmiotu, określ podobieństwa i różnice pomiędzy modelami cyklu życia:
 - ♦ liniowym,
 - ♦ spiralnym,
 - ♦ iteracyjno-przyrostowym.
4. Czy w swojej istocie RUP jest bardziej zbliżony do modelu liniowego, czy raczej spiralnego?
5. Określ, czy następujące pojęcia dotyczą aspektu dynamicznego metodyki RUP, czy statycznego:
 - ♦ artefakt,
 - ♦ rola,
 - ♦ punkt przeglądu,
 - ♦ dyscyplina,
 - ♦ przepływ pracy,
 - ♦ iteracja.
6. Wymień przykładowe metodyki „zwinne”. Jaka jest ich podstawowa cecha?
7. Wskaż przykładowe role w zespole projektowym.
8. Wymień różnice pomiędzy dyscypliną modelowania biznesowego a dyscypliną specyfikacji wymagań.
9. Korzystając z wiedzy zaprezentowanej w niniejszej książce, przyporządkuj diagramy UML do odpowiednich dyscyplin.
10. Czym różni się zarządzanie projektem od zarządzania konfiguracjami i zmianami?
11. Jak rozumiesz iteracyjność oraz przyrostowość procesu tworzenia oprogramowania? Wypowiedz się na temat liczby oraz długości poszczególnych iteracji.
12. Porównaj i zinterpretuj cele faz metodyki RUP.
13. Korzystając z hipertekstowej bazy wiedzy RUP, zaproponuj diagramy czynności dla faz metodyki RUP.
14. Uzasadnij czasochłonność realizacji faz wyrażoną w ujęciu procentowym.
15. Wymień poszczególne artefakty tworzone w ramach poszczególnych faz RUP.

Rozdział 15.

Modelowanie biznesowe

Znaczenie modelowania biznesowego

Oficjalny kanon elementów języka UML, obejmujący 13 rodzajów diagramów, zaprezentowany został w części pierwszej niniejszej książki. Diagramy te pozwalają zbudować i efektywnie opisać model systemu informatycznego, nie narzucając przy tym projektantowi jednego, konkretnego sposobu rozwiązania danego problemu.

Zrozumienie istoty procesów biznesowych jest podstawą specyfikacji wymagań oraz analizy i projektowania systemów informatycznych. Wiele metodyk przypisuje temu zagadnieniu wysoki priorytet. W procesie tworzenia systemu zgodnym z metodyką RUP pierwszą dyscypliną iteracyjno-przyrostowego cyklu życia systemu jest właśnie **modelowanie biznesowe** (ang. *business modeling*).



Modelowanie biznesowe jest sposobem odwzorowywania i dokumentowania procesów biznesowych.

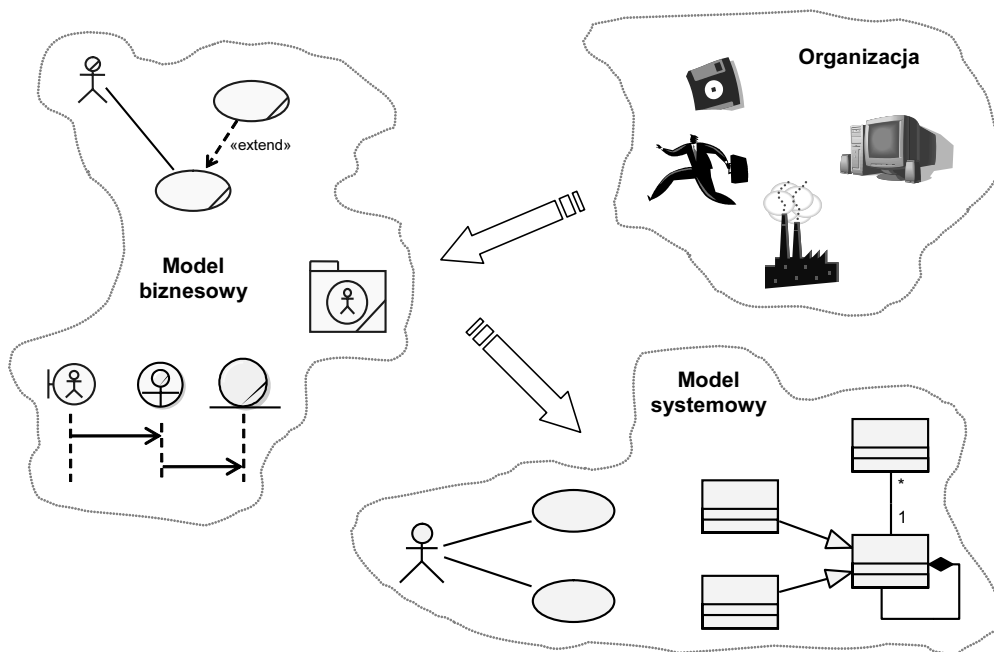
Tworzenie modeli biznesowych istotnie przyczynia się do lepszego zrozumienia sposobu funkcjonowania organizacji poprzez precyzyjny opis procesów biznesowych. Koncepcja modelowania biznesowego uległa znaczącemu rozwinięciu w związku z szeregiem rozczarowań związanych z zakończonymi niepowodzeniem wdrożeniami aplikacji biznesowych, szczególnie dotkliwymi dla użytkowników i integratorów.

A zatem, w nawiązaniu do kategorii modelowania języka UML, wyróżnić można:

- ◆ modelowanie systemowe — ukierunkowane na tworzenie systemu informatycznego, jego oprogramowania oraz infrastruktury sprzętowej;
- ◆ modelowanie biznesowe.

Zdefiniowane w dyscyplinie modelowania biznesowego procesy biznesowe są w naturalny sposób wspomagane przez system informatyczny, a ich część na pewnym etapie tworzenia systemów informatycznych ulega transformacji w **procesy systemowe**.

Modelowanie biznesowe można połączyć z **reengineeringiem procesów biznesowych** (*Business Process Reengineering*) — w skrócie BPR — w przypadku, gdy zamierza się dokonać radykalnej zmiany funkcjonowania organizacji i opracować modele nowych procesów biznesowych. Modele biznesowe znajdują zastosowanie przede wszystkim w pierwszej fazie cyklu życia RUP, fazie rozpoczęcia, a w mniejszym zakresie także w kolejnych fazach (opracowaniu, budowie oraz przekazaniu). Opracowany w tych fazach model biznesowy stanowi podstawę przyszłego modelowania systemu za pomocą różnorodnych diagramów UML omówionych w niniejszym opracowaniu. Opiswane relacje przedstawiono na rysunku 15.1.



Rysunek 15.1. Modelowanie biznesowe w procesie tworzenia systemu

Poza proponowaną w niniejszym rozdziale notacją, nawiązującą bezpośrednio do języka UML, istnieje szereg innych technik realizujących to zadanie. Wykorzystanie kategorii modelowania pochodzących z tego samego źródła jest jednak czynnikiem sprzyjającym sprawnej współpracy wielu grup zawodowych uczestniczących w procesie tworzenia systemu.

Podstawowe kategorie pojęciowe oraz notacja graficzna

Model biznesowy, podobnie jak model systemu informatycznego, jest przedstawiany w postaci **diagramów**. Diagramy biznesowe są w ujęciu czysto technicznym odpowiednikami diagramów systemowych. Tak więc możliwe jest przystosowanie do potrzeb

modelowania biznesowego większości diagramów wymienionych w specyfikacji języka UML 2.0, czyli:

- ♦ diagramów przypadków użycia,
- ♦ diagramów klas,
- ♦ diagramów obiektów,
- ♦ diagramów czynności,
- ♦ diagramów maszyny stanowej,
- ♦ diagramów interakcji,
- ♦ diagramów struktur połączonych,
- ♦ diagramów pakietów.

Ze względu na specyfikę funkcjonowania rzeczywistych organizacji największe **praktyczne zastosowanie** mają następujące rodzaje diagramów biznesowych:

- ♦ biznesowe diagramy przypadków użycia,
- ♦ biznesowe diagramy klas,
- ♦ biznesowe diagramy czynności,
- ♦ biznesowe diagramy sekwencji,
- ♦ biznesowe diagramy pakietów.

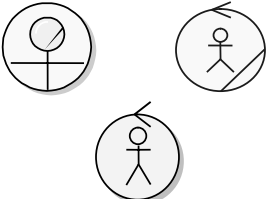
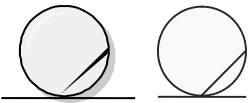
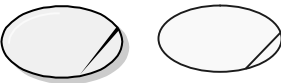
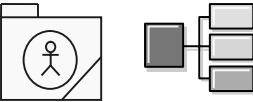
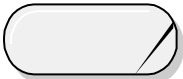
W przypadku modelowania biznesowego niezbędne stają się **kategorie modelowania**, które nie są elementami diagramów opisujących oprogramowanie. Kategorie te mają charakter stereotypów graficznych. Najczęściej stosowane stereotypy i ich interpretację przedstawia tabela 15.1.

Biznesowe diagramy stworzone w ramach modelowania biznesowego są transformowane w kolejnych fazach iteracyjno-przyrostowego cyklu RUP w analityczne lub systemowe diagramy języka UML. Transformacja diagramów biznesowych w systemowe nie zachodzi automatycznie. Do osiągnięcia ostatecznego efektu modelowania systemowego niezbędna jest praca analityczna i projektowa twórców systemu, stosownie do zaleceń UML i RUP.

Studium modelu biznesowego

W niniejszym punkcie przeprowadzono analizę biznesu na przykładzie systemu biznesowego księgarni. Analizę rozpoczęto od ustalenia kontekstu biznesowego (rysunek 15.2). Pozwoliło to na skompletowanie listy aktorów biznesowych związanych z funkcjonowaniem księgarni. Należą do nich:

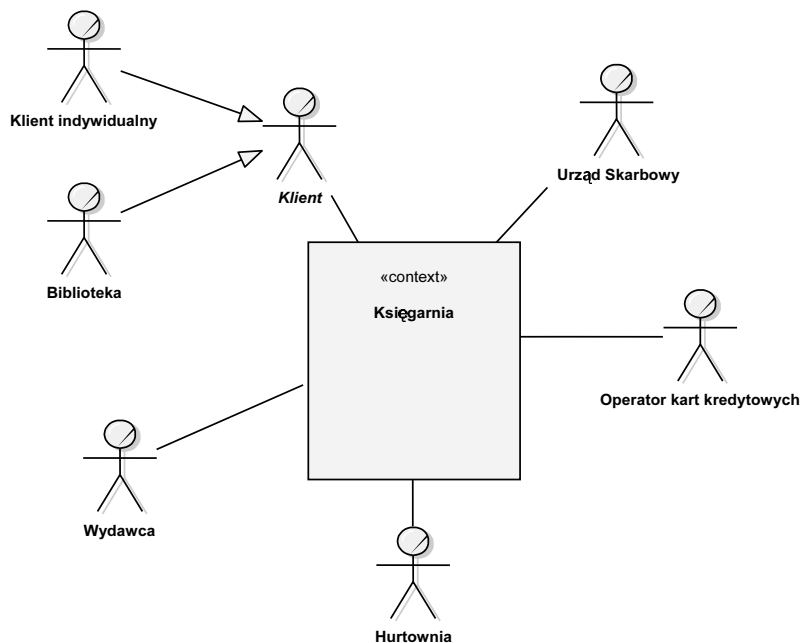
Tabela 15.1. Podstawowe kategorie modelowania diagramów biznesowych

Nazwa	Stereotypy	Charakterystyka
Aktor biznesowy		Rola pełniona przez użytkownika działającego w otoczeniu organizacji.
Pracownik biznesowy		Funkcjonujący w ramach organizacji pracownik lub system, pełniący określoną rolę lub zestaw ról wewnątrz procesu biznesowego. Współpracuje z innymi pracownikami biznesowymi i wykonuje działania oparte na obiektach biznesowych klas przechowujących.
Biznesowa klasa przechowująca		Fizycznie istniejący byt użytkowany i przetwarzany przez pracowników biznesowych oraz pracowników kontaktu. Operacje wykonywane na biznesowych klasach lub (i) obiektach przechowujących obejmują dostęp, aktualizację, tworzenie, monitorowanie oraz kasowanie.
Pracownik kontaktu		Pracownik biznesowy obsługujący proces biznesowy i jednocześnie będący w bezpośredniej interakcji z aktorami biznesowymi. Pracownik kontaktu nie może być systemem.
Biznesowy przypadek użycia		Proces biznesowy, dostarczający wyników istotnych z punktu widzenia aktora biznesowego.
Współdziałanie biznesowe		Zestaw diagramów przedstawiających elementy organizacji wspomagające proces biznesowy, takie jak pracownicy biznesowi czy biznesowe klasy lub obiekty przechowujące.
Jednostka organizacyjna		Grupa pracowników biznesowych, biznesowych jednostek przechowujących, związków między nimi, współdziałań biznesowych i innych składników organizacji. Składniki te są użytkowane w celu opracowania modelu biznesowego poprzez jego dekompozycję na mniejsze części w konwencji diagramów pakietów.
Czynność biznesowa		Określone zachowanie złożone z logicznie uporządkowanych ciągów podczynności, akcji oraz obiektów w celu wykonania pewnego procesu biznesowego.

Źródło: opracowanie własne na podstawie [RUP-2003] oraz [OMG-2003a].

- ♦ Klient — jako Klient indywidualny oraz Biblioteka,
- ♦ Wydawca,
- ♦ Hurtownia,
- ♦ Operator kart kredytowych,
- ♦ Urząd Skarbowy.

Rysunek 15.2.
Biznesowy kontekst systemu księgarni

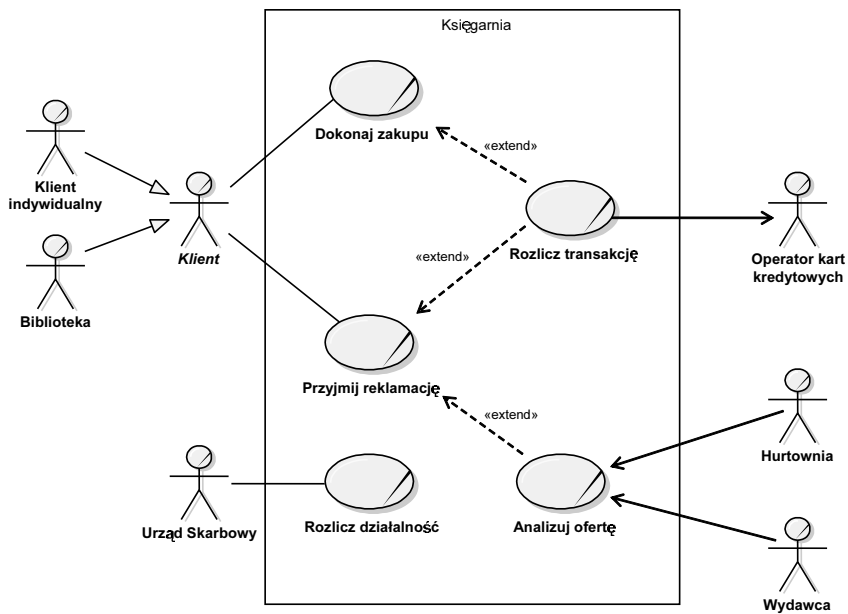


Następnie rozbudowano zidentyfikowany kontekst systemu rzeczywistej organizacji o biznesowe przypadki użycia (rysunek 15.3). Istotnej wartości dla aktorów organizacji dostarczają następujące biznesowe przypadki użycia:

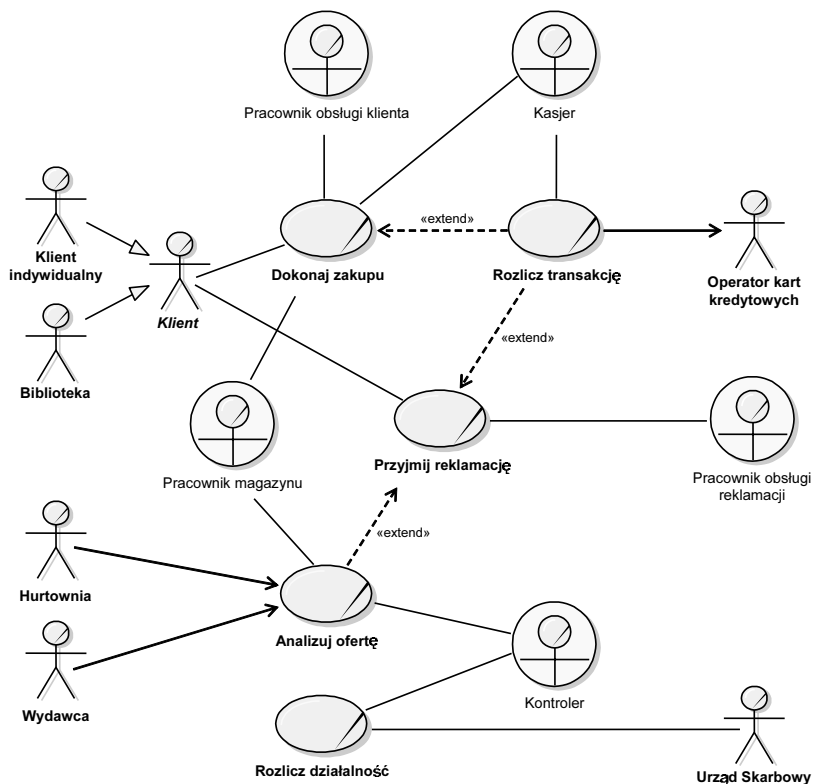
- ♦ *Dokonywanie zakupu* wybranych książek;
- ♦ *Przyjmowanie reklamacji*, dotyczące produktów wadliwych;
- ♦ *Analiza oferty* wydawcy, realizowana standardowo oraz w przypadku pozytywnego rozpatrzenia reklamacji połączonego z wyborem ekwiwalentu książkowego za wadliwy produkt w miejsce zwrotu gotówki;
- ♦ inne niż gotówkowe *Rozliczanie transakcji* zakupu;
- ♦ okresowe *Rozliczanie działalności* — w szczególności kwestii podatkowych — z *Urzędem Skarbowym*.

Wszystkie zidentyfikowane biznesowe przypadki użycia zostały szczegółowo przeanalizowane i udokumentowane. Dokumentację scenariusza głównego i scenariuszy alternatywnych wybranego przypadku użycia (*Przyjmij reklamację*) przedstawia tabela 15.2.

Biznesowy diagram przypadków użycia może być wykorzystany jako mapa procesów biznesowych związanych z funkcjonowaniem księgarni. W tym celu wskazuje się, którzy pracownicy biznesowi uczestniczą w realizacji danego przypadku użycia. Należy zauważyć, że pracownicy biznesowi są integralną częścią systemu i współdziałania biznesowego; powiązanie ich z przypadkami użycia w konwencji przedstawiania aktorów w tym przypadku wnosi do diagramu dodatkową wartość semantyczną (rysunek 15.4).



Rysunek 15.3. Biznesowy diagram przypadków użycia systemu księgarni



Rysunek 15.4. Mapa procesów biznesowych funkcjonowania księgarni

Tabela 15.2. Dokumentacja przypadku użycia „Przyjmij reklamację”

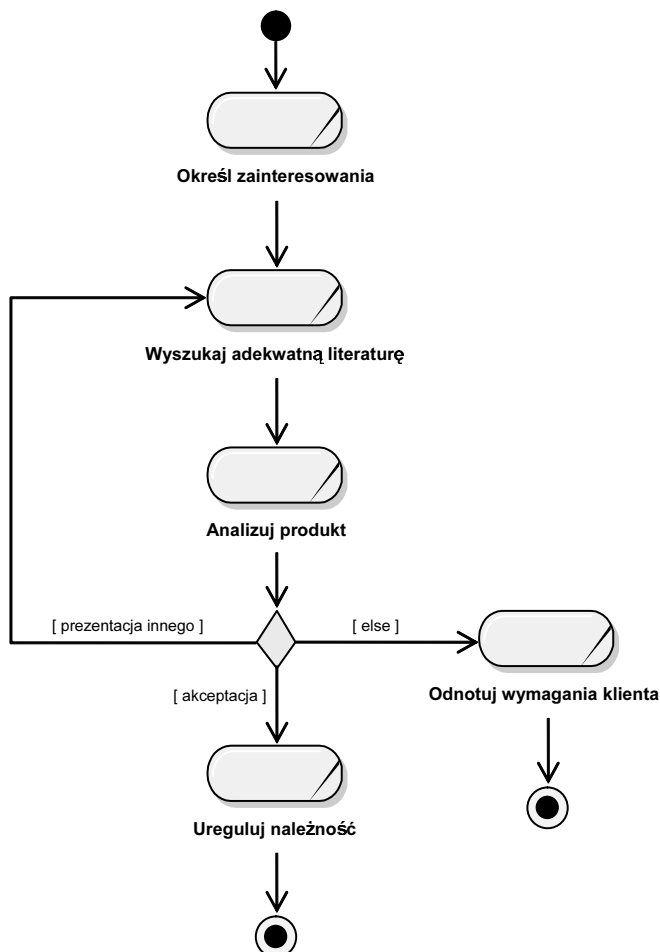
Nazwa przypadku użycia:	Przyjmij reklamację
Numer:	3
Twórca:	Stanisław Wrycza — Analityk procesów biznesowych
Aktorzy:	Klient indywidualny, Biblioteka
Krótki opis:	Przyjęcie produktu do reklamacji
Warunki wstępne:	Wymagane dostarczenie produktu oraz dowodu zakupu (paragon lub faktura)
Warunki końcowe:	Wydanie nowego produktu lub nieuwzględnienie reklamacji
Główny przepływ zdarzeń:	1. Klient zgłasza reklamację i przekazuje pracownikowi obsługi reklamacji dowód zakupu (paragon lub fakturę) oraz reklamowany produkt 2. Pracownik obsługi reklamacji weryfikuje dowód zakupu oraz produkt 3. Klient uzasadnia reklamację 4. Pracownik obsługi reklamacji uwzględnienia reklamację i przygotowuje kartę reklamacji dla dostarczonego produktu 5. Klient udostępnia szczegółowe dane do karty reklamacji 6. Pracownik obsługi reklamacji dostarcza nowy produkt 7. Klient odbiera nowy produkt
Alternatywne przepływy zdarzeń:	2a Brak dowodu zakupu — odrzucenie przyjęcia reklamacji 2b Brak produktu — odrzucenie przyjęcia reklamacji 4a Pracownik obsługi reklamacji nie uwzględnia reklamacji i zwraca klientowi dowód zakupu oraz reklamowany produkt 6a Brak nowego produktu — wydanie pieniędzy klientowi
Specjalne wymagania:	Termin dostarczenia nowego produktu klientowi nie może być dłuższy niż 14 dni
Notatki i kwestie:	1. Obsługa reklamacji odbywa się zgodnie z procedurą zamieszczoną w dokumencie <i>Procedura reklamacji.pdf</i> 2. Miejsca rozszerzenia: pozytywne rozpatrzenie reklamacji, żądanie zwrotu gotówki.

W celu wysokopoziomowego scharakteryzowania logiki konkretnego przypadku użycia można posłużyć się biznesowym diagramem czynności. Analityk opisujący funkcjonowanie biznesu aukcji wykorzystał ten diagram do wyspecyfikowania biznesowego przypadku użycia „Dokonaj zakupu”. Diagram ten został przedstawiony na rysunku 15.5.

Scenariuszom przypadków użycia towarzyszą stosowne diagramy interakcji. Szczególnie często stosowaną techniką ich specyfikowania są diagramy sekwencji. I tak rysunek 15.6 przedstawia specyfikację przypadku użycia „Przyjmij reklamację”, opierającą się na jego dokumentacji (tabela 15.2).

Rysunek 15.5.

Czynności przypadku
użycia „Dokonaj
zakupu”

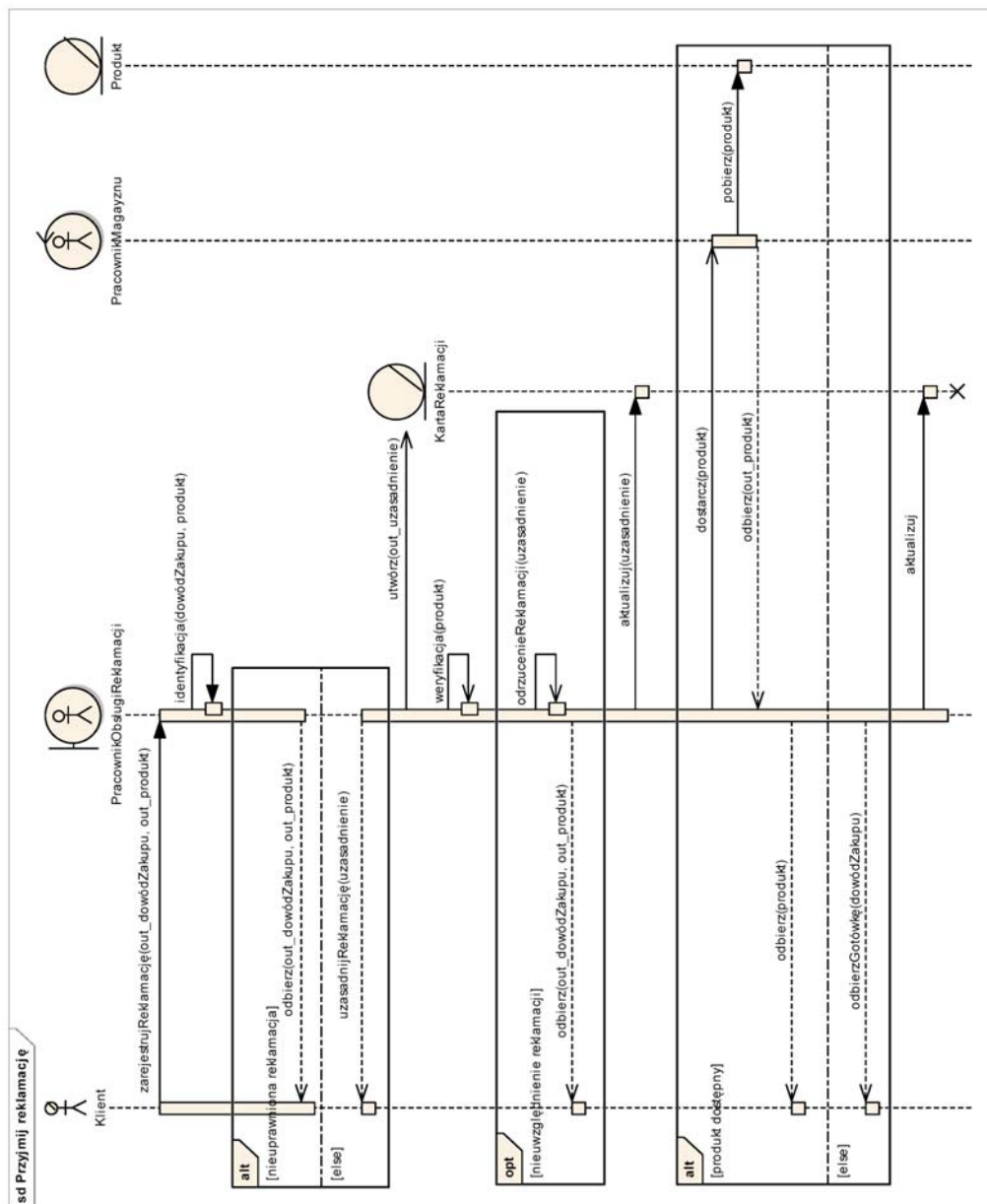


Z kolei statyczne aspekty funkcjonowania biznesu odwzorowywane są na podstawie kategorii pojęciowej biznesowej klasy przechowującej. I tak aspekty te modelowane są przede wszystkim przy wykorzystaniu biznesowych diagramów klas (rysunek 15.7).

Jak zaprezentowano na rysunku 15.7, w funkcjonowaniu księgarni kluczowe znaczenie ma *Dokument finansowy*. Jest to kategoria abstrakcyjna, uogólniająca następujące dokumenty:

- ♦ *Rozliczenie utargu*,
- ♦ *Raport strat*,
- ♦ *Dowód zakupu*,

przy czym ten ostatni dokument również jest kategorią abstrakcyjną — uogólniającą *Paragon* oraz *Fakturę*. *Paragon* jest wystawiany dla każdego sprzedanego *Produktu*. Jeżeli klient wyrazi takie życzenie, wraz z *Produktem* otrzymuje także *Fakturę*. *Faktura* jest zawsze opracowywana w dwóch egzemplarzach — jej kopię przechowuje się

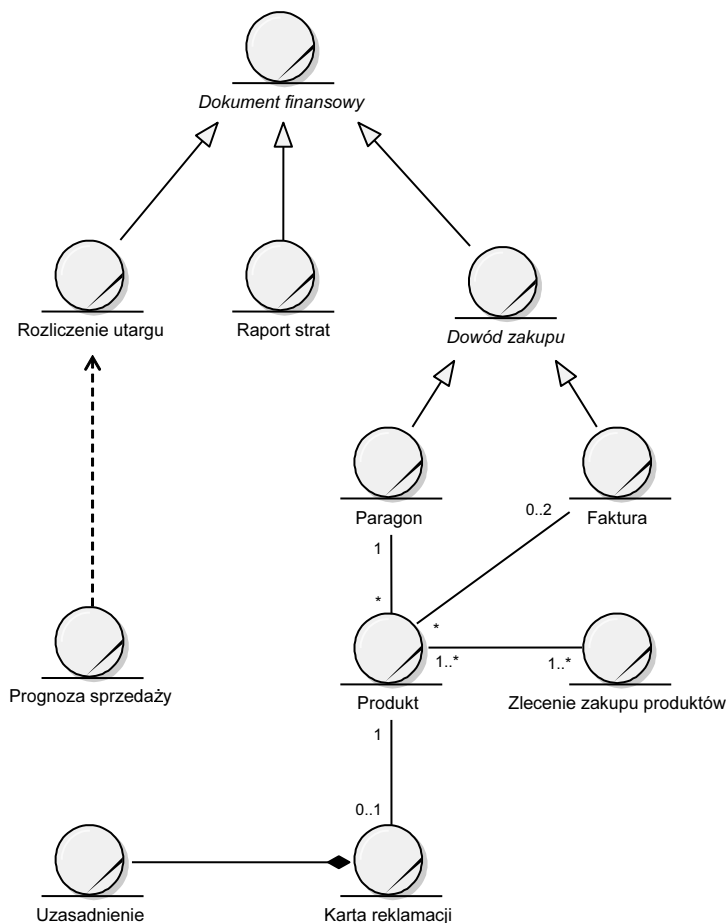


Rysunek 15.6. Biznesowy diagram sekwencji

w archiwum. W przypadku reklamowania *Produktu* sporządza się *Kartę reklamacji*, której częścią jest *Uzasadnienie* wniesienia reklamacji przez klienta. Na podstawie dokumentu *Rozliczenia utargu* opracowywana jest *Progniza sprzedaży*. Stosownie do niej sporządza się *Zlecenie zakupu produktów*, wysyłane do odpowiedniego wydawcy lub hurtowni książek.

Rysunek 15.7.

*Biznesowy
diagram klas
systemu księgarń*

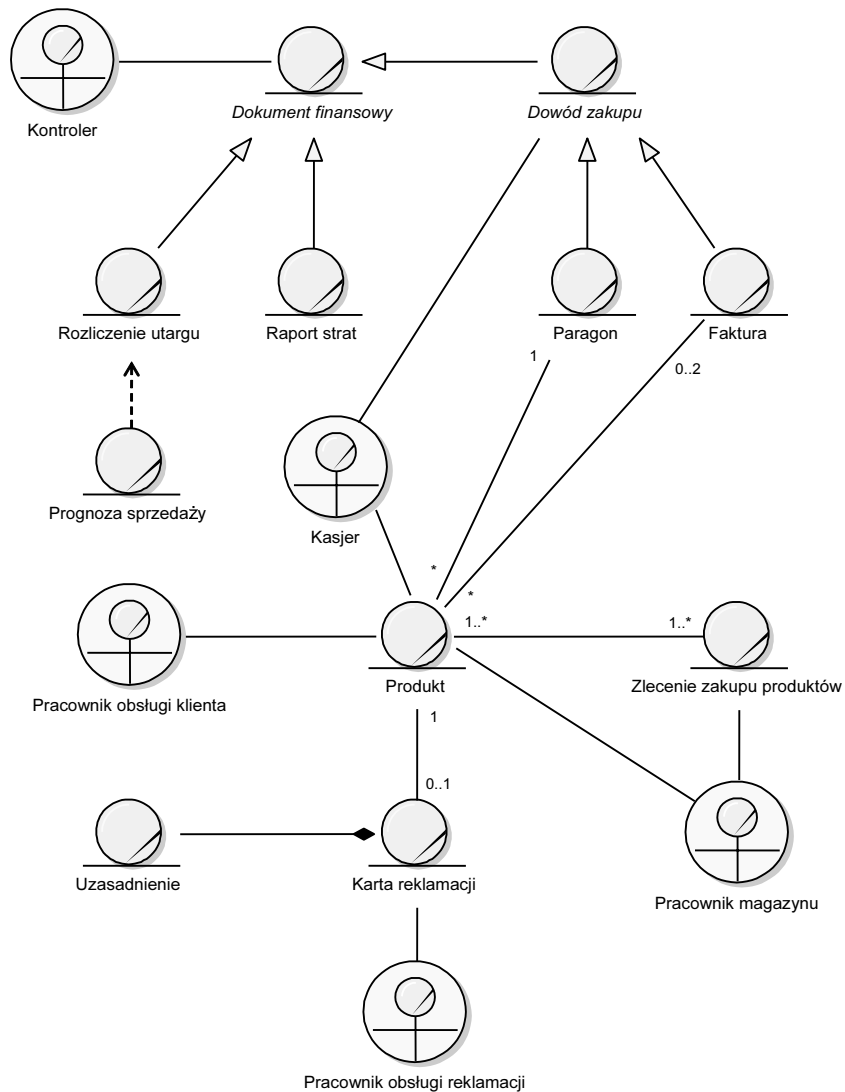


Biznesowe diagramy klas wykorzystywane są w tworzeniu ich systemowych odpowiedników, jak również w tworzeniu modelu analitycznego. Analogicznie do biznesowych diagramów przypadków użycia, na biznesowych diagramach klas zamieścić można pracowników biznesowych odpowiadających za poszczególne klasy przechowujące (rysunek 15.8). I tak:

- ♦ za zaopatrzenie (biznesowe klasy przechowujące *Zlecenie zakupu produktów* oraz *Produkt*) odpowiada *Pracownik magazynu*;
- ♦ za bieżącą obsługę klienta (biznesowa klasa przechowująca *Produkt*) odpowiada *Pracownik obsługi klienta*;
- ♦ za przyjmowanie wpłat i wydawanie dowodów zakupu (biznesowe klasy przechowujące *Dowód zakupu* oraz *Produkt*) odpowiada *Kasjer*;
- ♦ za weryfikowanie finansów firmy (*Dokumenty finansowe*) odpowiada *Kontroler*;
- ♦ za przyjmowanie reklamacji klientów (biznesowa klasa przechowująca *Karta reklamacji*) odpowiada *Pracownik obsługi reklamacji*.

Rysunek 15.8.

Biznesowy diagram klas z pracownikami biznesowymi



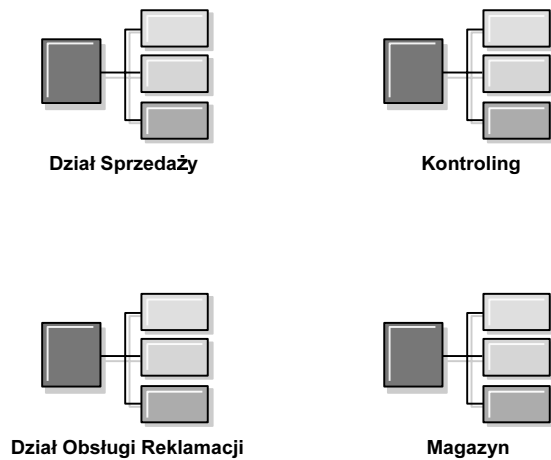
Zidentyfikowany przez analityka procesów biznesowych układ jednostek biznesowych, odpowiadający pod względem semantycznym systemowemu diagramowi pakietów i będący podstawą identyfikacji pakietów w systemie, przedstawia rysunek 15.9.

I tak ze względów praktycznych biznes księgarni podzielono na cztery różne działy:

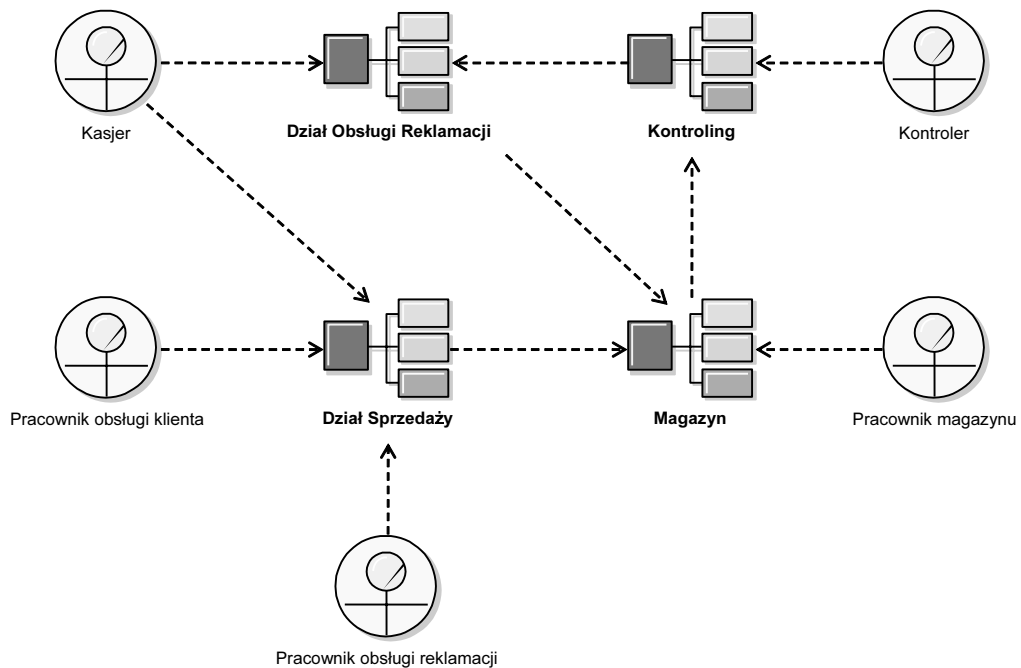
- ♦ *Dział Sprzedaży,*
- ♦ *Kontroling,*
- ♦ *Magazyn,*
- ♦ *Dział Obsługi Reklamacji.*

Rysunek 15.9.

Jednostki organizacyjne księgarni



Zależności pomiędzy zidentyfikowanymi pakietami oraz przydzielonymi do nich pracownikami, istotne z punktu widzenia przyszłego systemu informatycznego, przedstawia rysunek 15.10.

**Rysunek 15.10.** Zależności pomiędzy działami a pracownikami księgarni

Podstawowe pojęcia

Aktor biznesowy	Jednostka organizacyjna
Biznesowa klasa przechowująca	Model biznesowy
Biznesowy przypadek użycia	Pracownik biznesowy
Czynność biznesowa	Pracownik kontaktu
Diagram	Proces
Biznesowy	Biznesowy
Czynności	Reengineering
Klas	Systemowy
Pakietów	RUP
Przypadków użycia	Scenariusz
Sekwencji	Współdziałanie biznesowe
Systemowy	

Pytania i zadania

1. Jakie znaczenie ma w procesie tworzenia systemów informatycznych model biznesowy?
2. Z jakimi fazami oraz dyscyplinami procesu RUP należy kojarzyć model biznesowy?
3. Podaj nazwy diagramów biznesowych najczęściej wykorzystywanych w praktyce.
4. Przyporządkuj odpowiednie kategorie modelowania do poszczególnych diagramów biznesowych.
5. Wyjaśnij podobieństwa i różnice pomiędzy pracownikiem biznesowym a pracownikiem kontaktu.
6. Które z kategorii modelowania przedstawionych w tabeli 15.1 opisują elementy funkcjonujące poza procesem biznesowym?
7. Wskaż systemowe odpowiedniki:
 - ♦ jednostki organizacyjnej,
 - ♦ biznesowej klasy przechowującej.
8. Sporządź dokumentację scenariuszy biznesowych przypadków użycia księgarni.

9. Opierając się na sporządzonej dokumentacji, opracuj implementacyjne i wystąpieniowe diagramy sekwencji dokumentujące poszczególne biznesowe przypadki użycia księgarni.
10. Opisz funkcjonowanie niewielkiej firmy produkcyjnej w zakresie sterowania zapasami. Uwzględnij następujące diagramy:
 - ♦ biznesowy diagram przypadków użycia,
 - ♦ biznesowy diagram klas,
 - ♦ biznesowy diagram czynności,
 - ♦ biznesowe diagramy sekwencji.

Rozdział 16.

Modelowanie analityczne

Znaczenie modelowania analitycznego

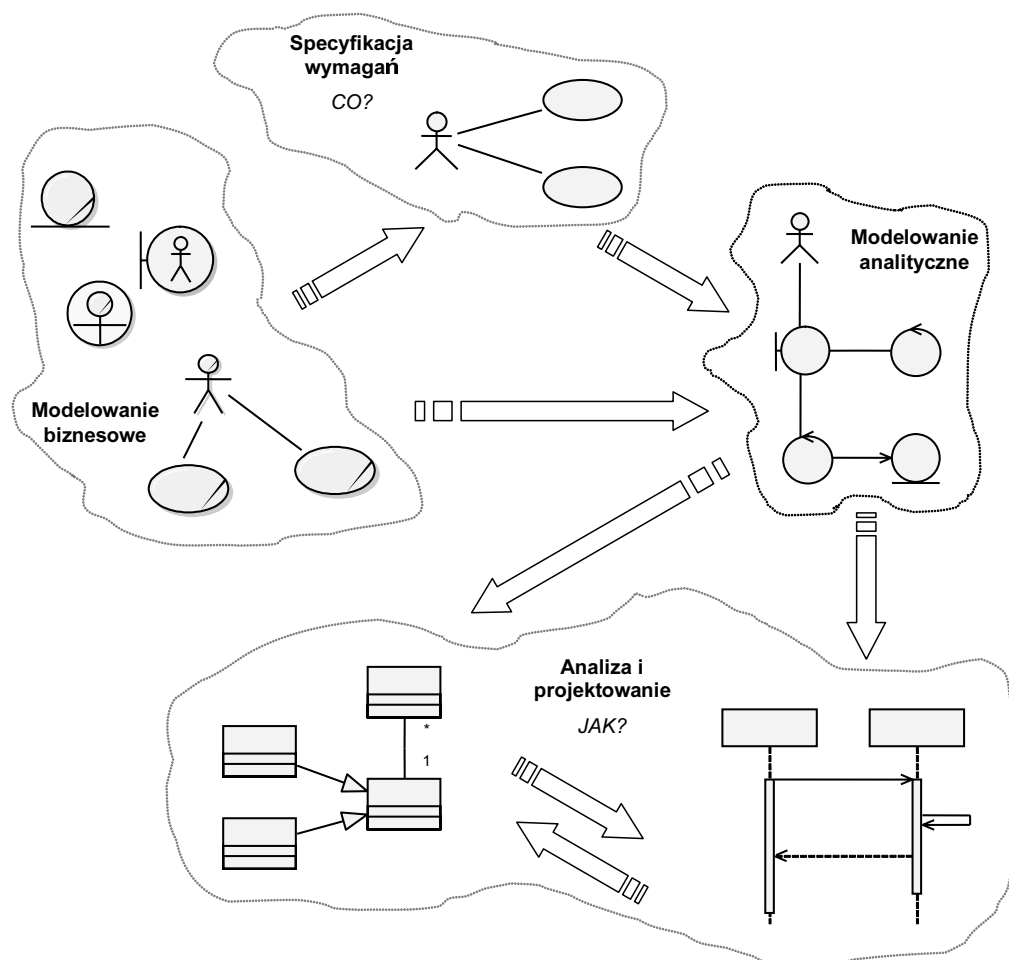
Zainteresowanie językiem UML w wielu ośrodkach naukowych i biznesowych przyczyniło się do powstania propozycji nowych diagramów lub rozszerzenia znaczenia już istniejących. Są one ściśle powiązane ze standardem języka UML i przyczyniają się do zwiększenia jakości, przejrzystości oraz szczegółowości procesu tworzenia systemów przy wykorzystaniu języka UML i na podstawie metodyki RUP. Do najbardziej znanych i użytkowanych w praktyce technik należą:

- ♦ diagramy **modelowania analitycznego**,
- ♦ diagramy modelowania biznesowego, omówione w rozdziale 15.



Modelowanie analityczne to technika wspomagająca język UML, która służy do dokumentowania wyników prac analitycznych i wczesnych prac projektowych.

Diagramy modelowania analitycznego wspomagają dyscyplinę analizy i projektowania; zostały zaproponowane przez I. Jacobsona w metodyce OOSE [Jacobson-1992]. W praktyce diagramy te umożliwiają przejście od diagramów przypadków użycia oraz ich scenariuszy do diagramów klas oraz diagramów interakcji na poziomie koncepcyjnym i implementacyjnym. Diagramy modelowania analitycznego stanowią zatem element pośredni i opcjonalny w procesie tworzenia systemu. Pozwalają zminimalizować lukę semantyczną pomiędzy zrozumiałą dla większości odbiorców terminologią przypadków użycia a technicznymi aspektami projektu systemu informatycznego. Zależności te przedstawia rysunek 16.1.



Rysunek 16.1. Diagramy modelowania analitycznego w procesie tworzenia systemu

Podstawowe kategorie pojęciowe oraz notacja graficzna

Konsekwentne posługiwanie się diagramami modelowania analitycznego jako pomocą w specyfikowaniu wymagań opartą na przypadkach użycia zwiększa prawdopodobieństwo wiernego przeanalizowania i odzwierciedlenia przyszłej funkcjonalności tworzonego systemu. Osiąga się to poprzez analizę scenariuszy danego przypadku użycia.

Model analityczny (ang. *analysis/robustness model*), grupujący diagramy analityczne, można traktować jako rodzaj wstępnego projektu. Jego celem jest wspomaganie identyfikacji klas. Podstawowymi elementami diagramów modelowania analitycznego są:

- ♦ klasy analityczne,
- ♦ aktorzy,
- ♦ związki.

Diagramy analityczne modelowane są jako diagramy klas z zastosowaniem trzech **stereotypowanych** klas:

- ♦ granicznych (ang. *boundary*),
- ♦ sterujących (ang. *control*),
- ♦ przechowujących (ang. *entity*).

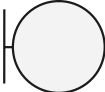
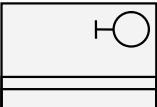
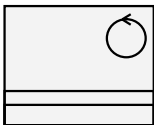
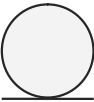
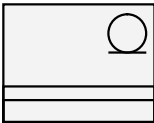
W praktyce podczas analizy scenariuszy przypadku użycia identyfikuje się obiekty analityczne, które podczas projektowania systemu ulegają przekształceniu w różne kategorie modelowania właściwe dla poziomu implementacyjnego, takie jak atrybuty klas, operacje lub same klasy. W związku z tym te kategorie modelowania analitycznego w literaturze przedmiotu określane są często mianem obiektów analitycznych. Notację oraz szczegółową charakterystykę klas analitycznych zawiera tabela 16.1.

W trakcie opracowywania diagramów modelowania analitycznego wykorzystywany jest również symbol **aktora** zaczerpnięty z diagramów przypadków użycia. Poszczególne typy klas są powiązane. Kluczowe znaczenie odgrywa w tym kontekście związek **asocjacji**, przy czym asocjacje te mogą być dwukierunkowe lub mieć przypisany kierunek nawigacji uszczegóławiający specyfikację sposobu przepływu informacji. Przykład diagramu analitycznego zamieszczono na rysunku 16.2.

Rysunek ten przedstawia rozpatrywanie podań o przyjęcie do firmy. Użytkownik systemu, *Kandydat*, wypełnia wszelkie pola zamieszczone na *Formatce rejestracji*, klikając następnie przycisk zatwierdzający złożenie podania. Po zatwierdzeniu sterowanie przejmuje obiekt klasy analitycznej *Rejestrowanie* danych. Udostępnione dane przechowywane są w formie podań. Za przygotowanie stosownych danych — zarówno danych indywidualnych, jak i zbiorczych statystyk — *Decydentowi* organizacji profesjonalnej, odpowiedzialne są odpowiednio obiekty sterujące: *Ocena podania* oraz *Przygotowanie zestawień syntetycznych*. *Decydent* komunikuje się z systemem poprzez obiekt graniczny *Interfejs decydenta*. Fakt przyjęcia *Kandydata* bądź odmowy jego przyjęcia odnotowywany jest w obiekcie przechowującym *Podanie*. Rezultaty wygenerowane przez obiekt sterujący *Przyjmowanie* dostępne są za pośrednictwem obiektu granicznego *Lista przyjętych*, udostępnionego *Kandydatowi*.

Istnieją określone **zasady**, zaproponowane przez D. Rosenberga [Rosenberg-1999], opisujące wzajemną korespondencję pomiędzy poszczególnymi rodzajami klas analitycznych i aktorami. Zasady te opracowano w odniesieniu do związku asocjacji. Przedstawia je tabela 16.2.

Tabela 16.1. Notacja stosowana na diagramach modelowania analitycznego

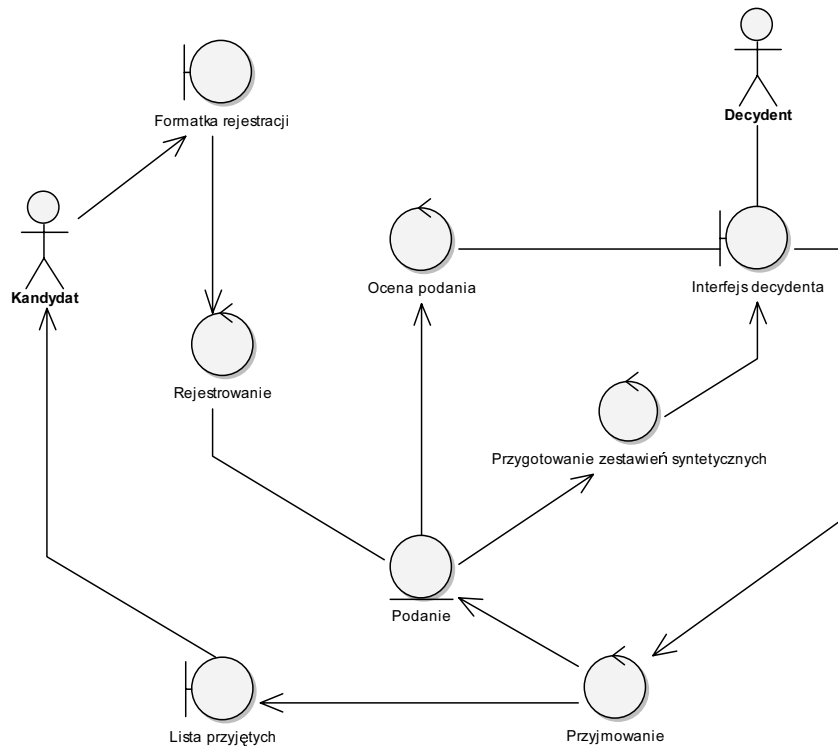
Nazwa	Stereotyp	Charakterystyka
Klasa graniczna	 	Wykorzystywana do modelowania interakcji aktora z systemem. Mimo że znajduje się na styku systemu z otoczeniem, jest jego integralną częścią. Reprezentuje takie elementy aplikacji jak formatki ekranowe, raporty, strony internetowe, protokoły komunikacyjne, terminale i różnorodne interfejsy wykorzystywane przez aktora. Stąd elementy te bywają również nazywane klasami interfejsowymi. Klasa graniczna w praktyce może przybrać następujące postaci: ♦ interfejsu użytkownika, zapewniającego kontakt z aktorami osobowymi; ♦ interfejsu systemu, zapewniającego kontakt z aktorami — systemami informatycznymi; ♦ interfejsu urządzenia, kontaktującego się z urządzeniami spoza systemu.
Klasa sterująca	 	Reprezentuje procesy, zawierając logikę aplikacji i określając przetwarzanie informacji. Klasy te pośredniczą pomiędzy wszystkimi rodzajami klas analitycznych, a więc granicznymi, przechowującymi oraz innymi klasami sterującymi. Klasa sterująca umożliwia koordynację, operowanie na innych klasach i sterowanie nimi. Obiekty klas sterujących często istnieją wyłącznie w trakcie wykonywania przypadku użycia. Mogą być implementowane na etapie projektowania zarówno w postaci samodzielnych klas, jak i operacji poszczególnych klas.
Klasa przechowująca	 	Reprezentuje dane, które muszą być przechowywane przez dłuższy czas; przykładem może być zawartość tablic baz danych oraz plików. Zazwyczaj związana jest z wieloma przypadkami użycia. Nie może samodzielnie inicjować interakcji. Klasa przechowująca może być implementowana na etapie projektowania zarówno w postaci samodzielnej klasy, jak i atrybutów poszczególnych klas.

Proces tworzenia modelu analitycznego

Tworzenie diagramów modelowania analitycznego poprzedzone jest w ramach iteracyjno-przyrostowego procesu RUP modelowaniem biznesu oraz specyfikacją wymagań tworzonego systemu w postaci systemowych diagramów przypadków użycia. Stąd proces ten można podzielić na następujące etapy:

Rysunek 16.2.

Diagram klas
analitycznych
systemu
rozpatrywania
podań

**Tabela 16.2.** Zasady obowiązujące w diagramach modelowania analitycznego

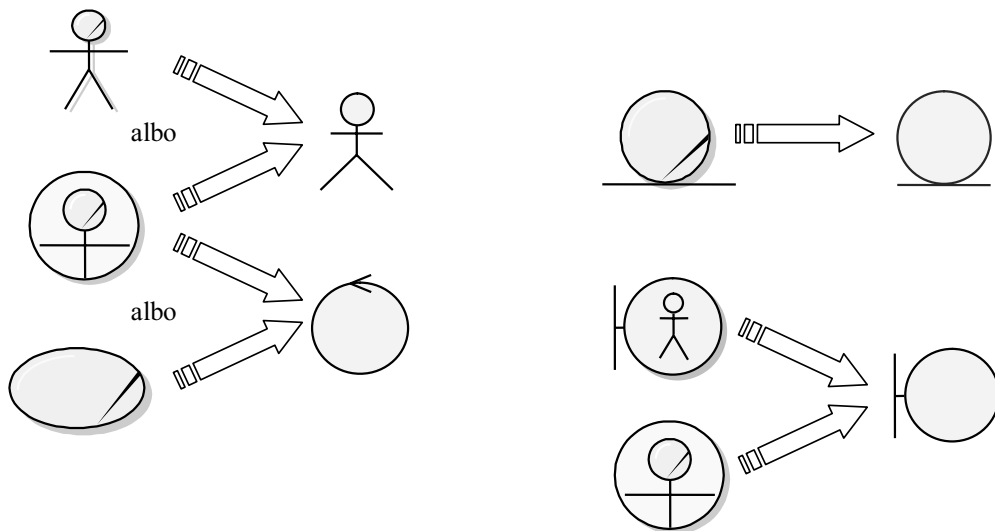
Może łączyć się z				
	–	+	–	–
	+	–	+	–
	–	+	+	+
	–	–	+	–

gdzie:

- + — elementy wyszczególnione w danym wierszu i kolumnie mogą się łączyć,
- — łączenie elementów wyszczególnionych w danym wierszu i kolumnie jest niepoprawne.

1. wyselekcjonowanie, stosownie do złożoności dziedziny przedmiotowej, odpowiednich:
 - ♦ diagramów modelu biznesowego,
 - ♦ diagramu (diagramów) przypadków użycia oraz jego (ich) scenariuszy,
2. przeniesienie aktorów z diagramów przypadków użycia na diagramy analityczne,
3. identyfikację klas analitycznych i określenie ich rodzajów,
4. integrację poszczególnych zidentyfikowanych elementów w formie diagramów analitycznych składających się na model analityczny.

W trakcie tego procesu obowiązują określone **zasady konwersji** z modeli biznesowych i systemowych diagramów przypadków użycia na kategorie diagramów modelowania analitycznego. I tak kluczowe zasady konwersji z modeli biznesowych zaprezentowano na rysunku 16.3.



Rysunek 16.3. Konwersja kategorii biznesowych na kategorie klas analitycznych

Analogicznie, należy zauważyć, że sporządzenie klas analitycznych na podstawie systemowych przypadków użycia nie polega na bezpośredniej zamianie notacji. Konwersja systemowego modelu przypadków użycia na modele analityczne obejmuje przekształcenie aktorów w klasy graniczne bądź ich bezpośrednie przeniesienie. Natomiast przypadki użycia są przekształcane w odpowiednie rodzaje klas analitycznych — graniczne, sterujące i przechowujące. W dalszych iteracjach RUP klasy analityczne przekształcane są w klasy systemu.

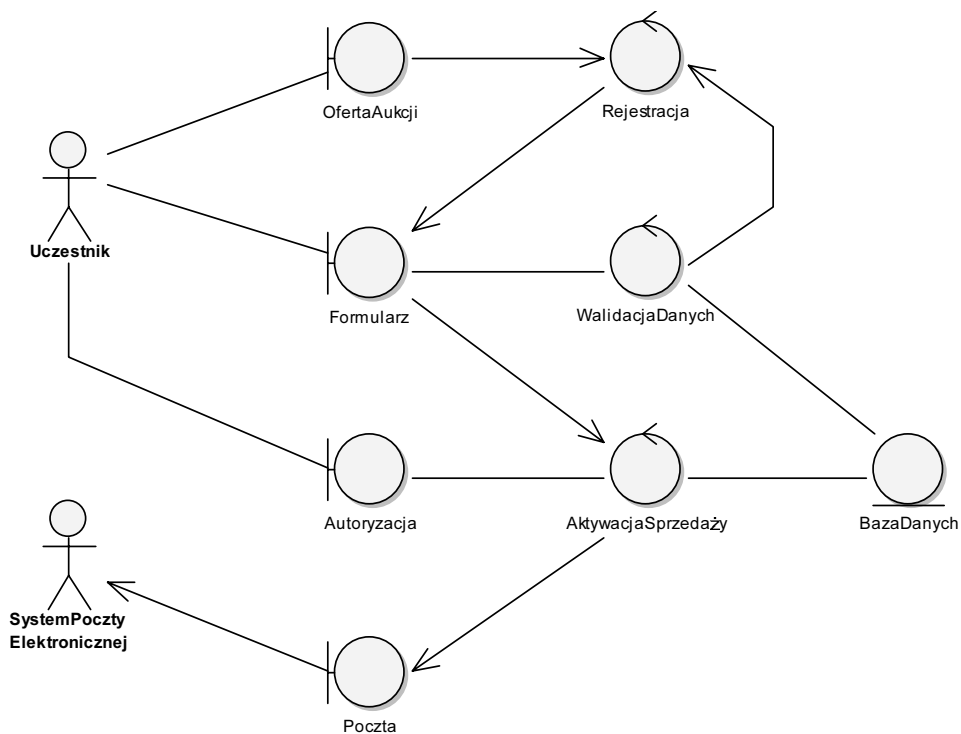
Studium modelu analitycznego

Diagram analityczny zamieszczony na rysunku 16.4 jest wysokopoziomową specyfikacją przypadku użycia rejestracji uczestnika na aukcji internetowej. Diagram ten zawiera dwóch aktorów:

- ♦ *Uczestnika aukcji,*
- ♦ *SystemPocztyElektronicznej,*

oraz osiem klas analitycznych:

- ♦ *OfertęAukcji,*
- ♦ *Rejestrację,*
- ♦ *Formularz,*
- ♦ *WalidacjęDanych,*
- ♦ *Autoryzację,*
- ♦ *AktywacjęSprzedaży,*
- ♦ *BazęDanych,*
- ♦ *Pocztę.*



Rysunek 16.4. Rejestrowanie uczestnika aukcji internetowej

Aktor *Uczestnik*, chcąc uzyskać prawo do aktywnego uczestnictwa w licytacjach oraz wystawiania artykułów na licytację, musi zarejestrować się w systemie aukcji internetowej. Wykorzystuje w tym celu obiekt klasy granicznej *OfertaAukcji*, który uruchamia *Formularz* rejestracyjny poprzez obiekt klasy sterującej *Rejestracja*. Po wprowadzeniu danych następuje ich walidacja. Za realizację tej operacji odpowiada obiekt sterujący o nazwie *WalidacjaDanych*. Poprawne dane zostają zapisane do bazy danych reprezentowanej przez obiekt przechowujący *BazaDanych*. Następnie *Uczestnik* zamierzający uzyskać prawa sprzedającego loguje się do systemu. Wykorzystuje w tym celu login i hasło uzyskane po wypełnieniu *Formularza*. Analityczny obiekt sterujący *AktywacjaSprzedaży* generuje kod sprzedaży, zapisuje go w obiekcie przechowującym *BazaDanych* i wysyła drogą elektroniczną wiadomość do rejestrującego się *Uczestnika*. Wiadomość jest wysyłana przez *SystemPocztyElektronicznej*, który nie jest integralną częścią systemu aukcji internetowej. Pośredniczącą instancją klasy granicznej jest w tym przypadku obiekt graniczny *Poczta*. Jest on wyspecjalizowanym modułem pozwalającym na obsługę elektronicznych wiadomości w systemie aukcji internetowej.

Podstawowe pojęcia

Aktor	Klasa analityczna
Analiza i projektowanie	Graniczna
Diagram	Przechowująca
Analityczny	Sterująca
Biznesowy	Model analityczny
UML	Specyfikacja wymagań
Interfejs	Związek
Systemu	Asocjacja
Urządzenia	Nawigacja
Użytkownika	

Pytania i zadania

1. Czy diagramy analityczne należy uznać za integralną część języka UML?
Odpowiedź uzasadnij.
2. Poniżej wymieniono kilka rodzajów diagramów. Korzystając z wiedzy zaprezentowanej w niniejszej książce, wskaż, w jakiej kolejności diagramy te są zazwyczaj tworzone:

- ♦ diagram komponentów,
- ♦ diagram analityczny,
- ♦ diagram sekwencji,
- ♦ diagram przypadków użycia,
- ♦ biznesowy diagram klas.

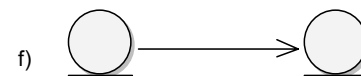
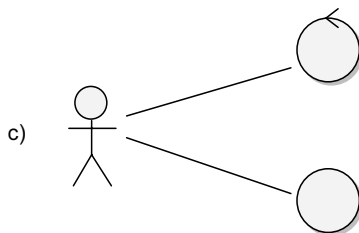
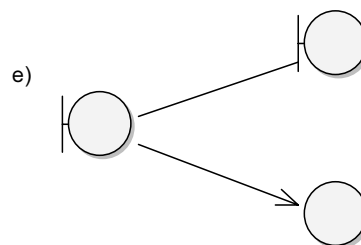
3. Jakie są podstawowe kategorie pojęciowe diagramów analitycznych?

4. Sklasyfikuj klasy analityczne.

5. Zinterpretuj i podaj przykłady:

- ♦ klasy granicznej,
- ♦ klasy sterującej,
- ♦ klasy przechowującej.

6. Które z poniższych przykładów należy uznać za poprawne?

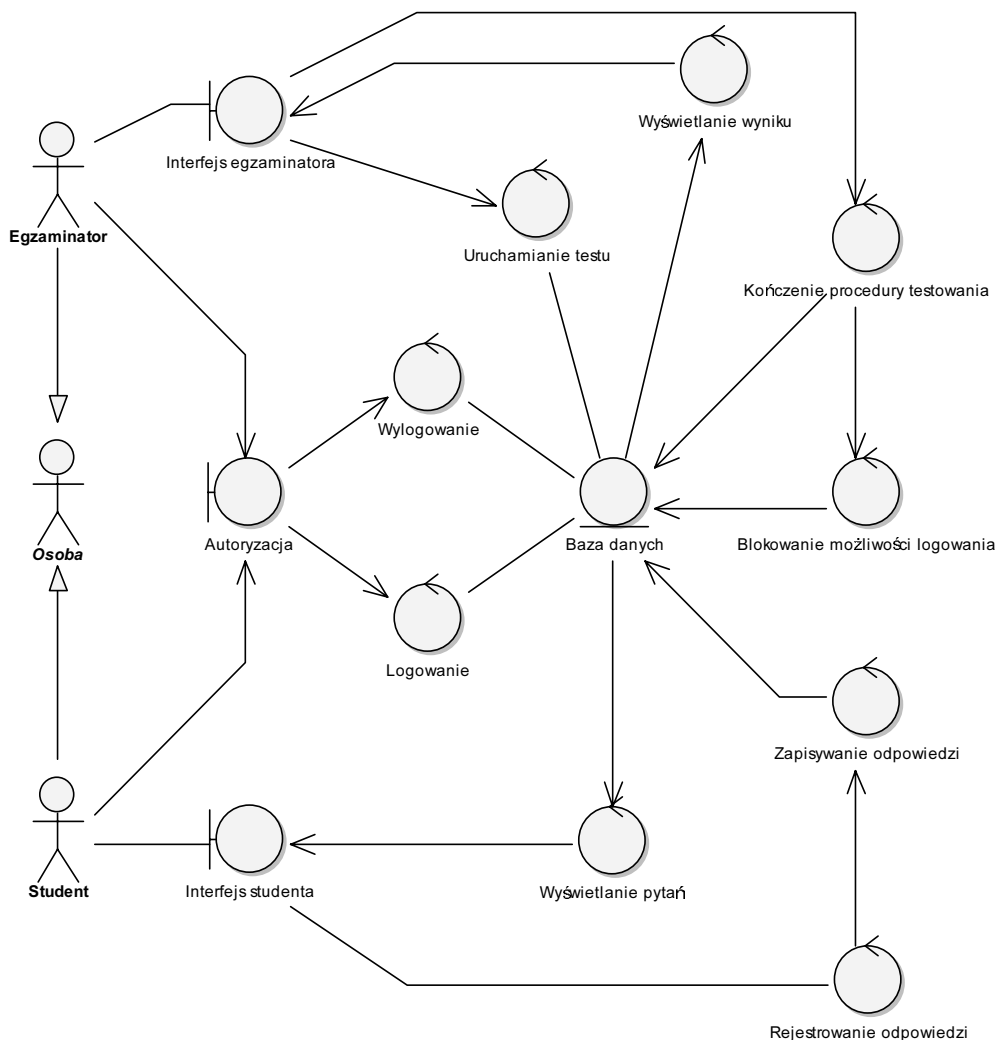


7. Do jakiego rodzaju obiektów klas analitycznych zaliczyłbyś okienko dialogowe?

8. Wymień wszystkie znane Ci rodzaje klas, których powiązanie z aktorem za pomocą związku asocjacji jest dozwolone.

9. Przedstaw i uzasadnij konwersję poszczególnych kategorii pojęciowych z modeli biznesowych na analityczne.

10. Sporządź diagram analityczny funkcjonalności bankomatu, opierając się na zidentyfikowanych przypadkach użycia.
11. Dokonaj twórczej interpretacji diagramu zamieszczonego na rysunku 16.5.



Rysunek 16.5. Proces egzaminowania w systemie elektronicznego egzaminowania

12. Jaki rodzaj diagramu — spośród diagramów będących częścią specyfikacji języka UML — uznałbyś za spełniający niektóre funkcje diagramów modelowania analitycznego w odniesieniu do opisu dynamiki systemu? Odpowiedź uzasadnij.
13. Zaproponuj reguły konwersji biznesowych diagramów przypadków użycia na kategorie systemowe.
14. Omów proces tworzenia diagramów modelowania analitycznego.

Rozdział 17.

Komputerowe wspomaganie modelowania systemu

Pakiety CASE wspomagające UML i RUP

Metodyki tworzenia systemów informatycznych są komputerowo **wspomagane** przez narzędzia CASE (*Computer Aided Software Engineering*). Firmy komputerowe oraz naukowe środowisko informatyczne opracowały wiele tego typu narzędzi modelowania systemów. Ich znaczenie rośnie wraz ze wzrostem zakresu projektu, co skutkuje rozbudowaną i niezwykle trudną w zarządzaniu dokumentacją. Zarówno metodyka RUP, jak i język UML są wspomagane przez narzędzia CASE. Ich zastosowanie pozwala na wykonanie następujących **zadań** w procesie tworzenia systemów:

- ◆ wspomaganie specyfikacji i dokumentowania projektów informatycznych,
- ◆ sprawdzanie semantycznej poprawności diagramów UML i modelu jako całości,
- ◆ wspomaganie iteracyjno-przyrostowego cyklu życia systemu,
- ◆ generowanie szkieletowego kodu źródłowego na podstawie modelu systemu,
- ◆ inżynierię zwrotną (ang. *reverse engineering*),
- ◆ synchronizację modelu systemu z kodem źródłowym,
- ◆ możliwość równoległego tworzenia oprogramowania przez wielu członków zespołu projektowego.

Dzięki narzędziom CASE, wspomagającym w sposób kompleksowy proces tworzenia systemu, osiąga się redukcję czasu i kosztów, a także zapewnia **wyższą jakość** tworzonego systemu. Celem niniejszego rozdziału jest analiza porównawcza najbardziej uznanych narzędzi CASE wspomagających język UML i metodykę RUP, jak również

zaznajomienie użytkownika z układem GUI, wykorzystywanym w tego typu narzędziach. Załączone zestawienia porównawcze mogą służyć menedżerom, analitykom i projektantom przy podejmowaniu decyzji o wyborze adekwatnego narzędzia. Z tych powodów, a także ze względu na zakres tematyki, zrezygnowano ze szczegółowego instruktażu instalacji i użytkowania tych narzędzi, co zapewniają ich producenci. Zawarte w tym rozdziale oceny i charakterystyki nie mają podtekstu marketingowego i zawierają obiektywną ocenę aktualnie dostępnych narzędzi.

Stopień akceptacji oprogramowania CASE, ukierunkowanego na język UML i metodykę RUP, jest w środowisku profesjonalnym zróżnicowany. W niektórych przypadkach jest ono traktowane jako nieodłączny element tworzenia oprogramowania, w innych wykorzystywane są jedynie poszczególne funkcje. Wybrane narzędzia CASE ze wskazaniem ich nazw oraz producentów wraz z adresami firmowych witryn internetowych zawiera tabela 17.1. Sporządzono ją w układzie alfabetycznym.

Tabela 17.1. Wybrane narzędzia CASE dla UML i RUP

Nazwa	Producent	Adres WWW
Ameos 9.1.5	Aonix	www.aonix.com
AnyLogic 5.1.1	XJ Technologies	www.xjtek.com
ArgoUML 0.16.1	Tigris	www.argouml.tigris.org
Enterprise Architect 4.5	Sparx Systems	www.sparxsystems.com.au
Innovator 8.1	MID	www.mid.de
Metamil 4.0	Metamil Software	www.metamill.com
Objectteering Enterprise Edition	Objectteering Software	www.objectteering.com
Poseidon for UML 3.1	Gentleware	www.gentleware.com
Rational Software Architect 6.0	IBM	www-306.ibm.com/software/rational
Teuta	Institute of Scientific Computing — Group for Software Science	www.par.univie.ac.at
Together 2005	Borland	www.borland.com

Należy podkreślić, że dostępna oferta rynkowa w zakresie narzędzi CASE wspierających język UML została istotnie wzbogacona i obejmuje bardzo wiele użytecznych produktów. Lista narzędzi zgodnych z wersją 2.0 tego standardu nieustannie jest wzbogacana o nowe pozycje. W celu wyboru oprogramowania najbardziej dostosowanego do indywidualnych potrzeb można posłużyć się szeregiem **kryteriów decyzyjnych**. W niniejszym opracowaniu zamieszczono porównanie wymienionych w tabeli 17.1 pakietów CASE względem następujących kryteriów:

- ♦ zakresu diagramów UML obsługiwanych przez narzędzie,
- ♦ języków programowania wspomaganych przez narzędzie w ramach generowania szkieletowego kodu źródłowego,
- ♦ języków programowania wspomaganych w ramach inżynierii zwrotnej,
- ♦ dostępności wersji narzędzia zorientowanych na różne systemy operacyjne oraz możliwości jego oceny przed dokonaniem zakupu.

Zakres wspomagania diagramów UML

Wprowadzenie wersji 2.0 standardu UML było poważnym wyzwaniem dla twórców narzędzi CASE. W tym kontekście najbardziej zauważalne dla użytkownika narzędzia jest to, że przez dłuższy okres bardzo niewiele produktów wspierało tworzenie **pełnego zestawu** diagramów aktualnej wersji UML. Także zakres semantyczny poszczególnych diagramów w wielu przypadkach był niepełny. Typowy projekt informatyczny wykorzystuje zaledwie część możliwości oferowanych przez standard UML, niemniej oddanie jego specyfiki wymaga od oprogramowania narzędziowego udostępnienia twórcy systemu jak najszerszego zakresu kategorii modelowania.

Porównanie wybranych narzędzi CASE pod kątem możliwości odwzorowywania poszczególnych diagramów UML 2.0 oraz deklarowanej przez producenta zgodności z określoną wersją standardu UML przedstawia tabela 17.2. Kolejne rodzaje diagramów reprezentowane są w tabeli przez ich wyróżniki, odpowiednio:

- ♦ *ud* — diagram przypadków użycia,
- ♦ *cld* — diagram klas,
- ♦ *sm* — diagram maszyny stanowej,
- ♦ *ad* — diagram czynności,
- ♦ *csd* — diagram struktur połączonych,
- ♦ *iod* — diagram sterowania interakcją,
- ♦ *sd* — diagram sekwencji,
- ♦ *cd* — diagram komunikacji,
- ♦ *cod* — diagram komponentów,
- ♦ *dd* — diagram rozlokowania,
- ♦ *td* — diagram harmonogramowania.

Tabela nie uwzględnia diagramów obiektów, pod względem technicznym zgodnych z diagramami klas i tym samym wspieranych przez narzędzia umożliwiające odwzorowywanie diagramów klas. Analogicznie, pakiety mogą być wykorzystywane w odniesieniu do pozostałych diagramów UML.

Jak wynika z tabeli 17.2, najsłabiej wspierane w narzędziach CASE są nowe diagramy języka UML, czyli diagram sterowania interakcją (ang. *interaction overview diagram*), diagram struktur połączonych (ang. *composite structure diagram*), a zwłaszcza diagram harmonogramowania (ang. *timing diagram*). Z kolei w niemal każdym narzędziu CASE edytować można tradycyjne diagramy UML 1.x. W największej liczbie narzędzi odwzorowywać można diagramy klas (ang. *class diagram*), a w konsekwencji diagramy obiektów i pakietów. Równie powszechne jest wspieranie diagramów maszyny stanowej (ang. *state-machine diagram*), diagramów czynności (ang. *activity diagram*) oraz diagramów komunikacji (ang. *communication diagram*).

Tabela 17.2. Zakres wspomagania diagramów UML — zestawienie porównawcze

Nazwa	Deklarowana zgodność z UML	Diagram UML											Σ
		ud	cld	sm	ad	csd	iod	sd	cd	cod	dd	td	
		+	+	+	+	–	–	+	+	+	+	–	8
AnyLogic	1.3	–	–	+	–	–	–	–	–	–	–	–	1
ArgoUML	1.3	+	+	+	+	–	–	+	+	+	+	–	8
Enterprise Architect	2.0	+	+	+	+	+	+	+	+	+	+	+	11
Innovator	1.4	+	+	+	+	–	–	–	+	+	–	–	6
Metamill	2.0	+	+	+	+	+	–	+	+	+	+	–	9
Objectteering Enterprise Edition	2.0	+	+	+	+	+	+	+	+	+	+	–	10
Poseidon for UML	1.5 z elementami 2.0	+	+	+	+	–	–	+	+	+	+	–	8
Rational Software Architect	2.0	+	+	+	+	+	–	+	+	+	+	–	9
Teuta	1.5	–	+	–	+	–	–	–	+	–	+	–	4
Together	2.0	+	+	+	+	+	–	+	+	+	+	–	9
Σ		9	10	10	10	5	2	8	10	9	9	1	—

Źródło: opracowanie własne na podstawie <http://www.oose.de/umltools.htm> oraz danych producentów narzędzi CASE

Producenci niektórych narzędzi dla wersji 1.x standardu UML deklarują wsparcie dla diagramów z wersji 2.0. W zestawieniu 17.2 zaznaczono możliwość modelowania tych nowych rodzajów diagramów jedynie w wersjach narzędzia CASE, zadeklarowanych przez producentów jako wspierające standard UML 2.0. Precyzyjna analiza porównawcza wymagałaby również uwzględnienia poszczególnych, licznych opcji w ramach diagramów wszystkich rodzajów, co ze względu na dużą skalę badań wykracza poza zakres niniejszego opracowania.

Generowanie szkieletowego kodu źródłowego

Generowanie kodu jest jedną fundamentalnych opcji w cyklu życia systemu i ważną cechą narzędzia CASE. Naturalnie w efekcie jej wykorzystania nie powstaje pełna, możliwa do natychmiastowego uruchomienia aplikacja. Generowany kod źródłowy jest określany jako **szkieletowy** — generowana jest struktura oprogramowania, w szczególności klasy, atrybuty wraz z ich właściwościami, sygnatury operacji, związki pomiędzy elementami modelu i komponenty. Zawartość poszczególnych operacji musi

być następnie uzupełniona przez programistów. Zakres kodowanych elementów różni się w zależności od języka programowania, w którym model jest implementowany, oraz charakterystyki danego narzędzia.

Wykorzystanie funkcji generowania kodu pomaga w utrzymaniu przejrzystości jego struktury i eliminuje wiele pomyłek technicznych, które programiści mogą popełnić na wczesnym etapie kodowania. Zasadniczo skraca też czas tworzenia gotowego systemu. Nie wyręcza jednak twórcy systemu w czynnościach innych niż rutynowe, takich jak projektowanie interfejsu użytkownika (GUI). Wymaga także konsekwentnego przestrzegania pewnych **reguł modelowania**. Przykładowo, aby w momencie zamieszczania komunikatu pomiędzy dwoma obiektami zaprezentowanymi na diagramie sekwencji nastąpiło automatyczne zadeklarowanie wywoływanej operacji w stosownym obiekcie (a w istocie w klasie tego obiektu), należy uprzednio w sposób jawny wskazać, instancją której klasy jest ten obiekt. Wskazane jest ponadto wykorzystywanie funkcji sprawdzania poprawności semantycznej modelu przed przejściem do generowania kodu, jeśli opcja taka oferowana jest przez narzędzie CASE.

Zestawienie możliwości generowania kodu oferowanych przez wyselekcjonowane narzędzia CASE przedstawia tabela 17.3. Uwzględnia ona najpopularniejsze języki programowania i standardy. W osobnej kolumnie tabeli wymieniono wspierane przez dane narzędzie technologie spoza standardowego zestawu. Ponadto tabela wymienia w ostatniej kolumnie format dokumentacji generowanej przez dane narzędzie.

Z zestawienia wynika, że największy zakres generowania kodu źródłowego oferują narzędzia *Enterprise Architect* oraz *Poseidon for UML*. Zdecydowana większość wyselekcjonowanych narzędzi oferuje ponadto generowanie dokumentacji.

Inżynieria zwrotna

Kolejną weryfikowaną cechą oprogramowania narzędziowego jest zdolność przeprowadzenia inżynierii zwrotnej, czyli zdolność do utworzenia bądź modyfikacji modelu systemu na podstawie kodu źródłowego aplikacji. Inżynieria zwrotna, wykorzystywana w połączeniu z funkcją generowania kodu źródłowego, umożliwia zachowanie spójności kodu systemu względem jego modelu. Zależność ta jest dwukierunkowa i jest określana mianem inżynierii wahadłowej (ang. *round-trip engineering*). Dzięki inżynierii zwrotnej można **zsynchronizować** oba te artefakty, nawet jeśli w kodzie dokonano zmian, nie nanosząc przy tym poprawek w dokumentacji. Praktyka tworzenia systemów informatycznych dowodzi, że posiadanie aktualnego modelu systemu jest nie do przecenienia.

Możliwości poszczególnych narzędzi CASE w zakresie inżynierii zwrotnej przedstawia tabela 17.4. Zakres wspieranych języków programowania oraz standardów zazwyczaj jest dla danego narzędzia nieco węższy, niż miało to miejsce w przypadku generowania kodu.

Tabela 17.3. Generowanie szkieletowego kodu źródłowego — zestawienie porównawcze

Nazwa	Język programowania / standard									Dokumen- tacja
	CORBA IDL	JAVA	C++	VB.NET	C#	DELPHI	PHP	SQL	inne	
Ameos	+	+	+	–	–	–	–	–	Ada95, Mistra, C	RTF, HTML
AnyLogic	–	+	–	–	–	–	–	–	–	–
ArgoUML	–	+	+	–	+	–	+	–	PHP4 jak i PHP5	–
Enterprise Architect	–	+	+	+	+	+	+	+	MS SQL, MySQL, Oracle, DB2, Interbase, MS Access, Postgres, Sybase	RTF, HTML
Innovator	+	+	+	+	–	–	–	+	–	XML, MS Word, ASCII, PS
Metamil	–	+	+	–	+	–	–	–	ANSI C	HTML
Objecteering Enterprise Edition	–	+	+	+	–	–	–	–	–	HTML, RTF
Poseidon for UML	+	+	+	+	+	+	+	+	–	HTML
Rational Software Architect	–	+	+	–	–	–	–	–	–	XML, PDF, HTML
Teuta	–	–	+	–	–	–	–	–	XML	–
Together	+	+	+	+	+	–	–	–	–	HTML

Źródło: opracowanie własne na podstawie <http://www.oose.de/umltools.htm>

Obsługiwane platformy

Ostatnim kryterium decyzyjnym uwzględnionym w niniejszym opracowaniu jest dostępność wersji narzędzia zorientowanych na różne systemy operacyjne oraz możliwości jego przetestowania przed dokonaniem zakupu. Wysiłki producentów oprogramowania narzędziowego zmierzające do wydłużenia listy obsługiwanych platform zwiększają **elastyczność** zespołu projektowego.

Wspieranie systemów z rodziny Windows można określić jako powszechną praktykę. Jest tak w przypadku wszystkich analizowanych narzędzi. Popularną platformą jest również Linux. Jedynie narzędzia AnyLogic oraz Enterprise Architect nie występują w wersji przeznaczonej dla tego systemu operacyjnego. ArgoUML, Poseidon for UML oraz Together występują ponadto w wersji przeznaczonej dla systemu operacyjnego MacOS.

Tabela 17.4. Inżynieria zwrotna — zestawienie porównawcze

Nazwa	Język programowania/standard								
	CORBA IDL	JAVA	C++	VB.NET	C#	DELPHI	PHP	SQL	inne
Ameos	+	+	+	–	–	–	–	–	Ada95
AnyLogic	–	–	–	–	–	–	–	–	–
ArgoUML	+	+	–	–	–	–	–	–	betakod języka JAVA (pliki .class)
Enterprise Architect	–	+	+	+	+	+	+	+	wsparcie MS SQL, MySQL, Oracle, DB2, Interbase, MS Access, Postgres, Sybase
Innovator	+	+	+	+	–	–	–	+	–
Metamil	–	+	+	–	+	–	–	–	ANSI C
Objectteering Enterprise Edition	+	+	+	+	–	–	–	+	–
Poseidon for UML	–	+	–	–	–	–	–	–	–
Rational Software Architect	–	+	+	–	–	–	–	–	–
Teuta	–	–	–	–	–	–	–	–	–
Together	+	+	+	+	+	–	–	–	–

Źródło: <http://www.oose.de/umltools.htm>

Ponadto dla użytkownika indywidualnego szczególnie istotne jest to, że wszyscy producenci porównywanych narzędzi zdecydowali się zaproponować możliwość nieodpłatnej oceny. Wersje te charakteryzują się najczęściej ograniczoną funkcjonalnością i (lub) są limitowe czasowo. Tylko jedno z wyselekcjonowanych narzędzi przynależy do kategorii Open Source (ArgoUML), czego konsekwencją jest m.in. całkowity brak opłat licencyjnych.

Sparx Systems Enterprise Architect

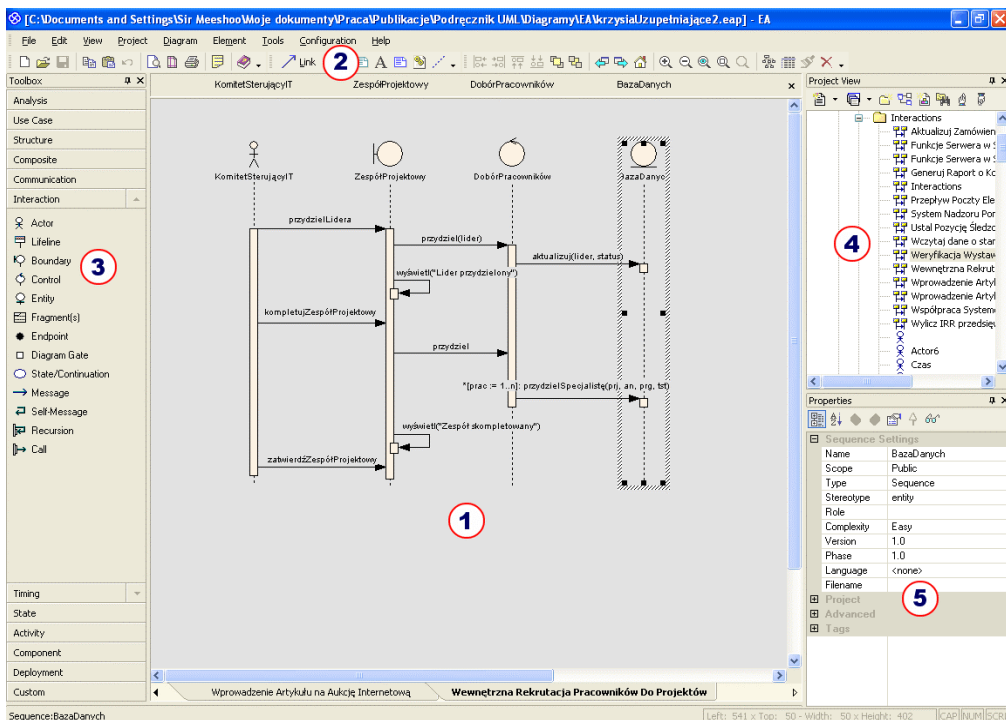
Dla ilustracji najważniejszych cech oprogramowania typu CASE wybrano trzy produkty popularne zarówno w środowisku akademickim, jak i profesjonalnym:

- ♦ Enterprise Architect firmy Sparx Systems;
- ♦ Rational Software Modeler firmy IBM;
- ♦ Poseidon for UML firmy Gentleware.

Firma Sparx Systems zaprezentowała bardzo wszechstronne i łatwe w użytkowaniu narzędzie CASE pod nazwą Enterprise Architect. Jako jeden z niewielu dostępnych na rynku produktów oferuje ono bardzo wysoki stopień zgodności z UML 2.0, w tym możliwość modelowania **wszystkich diagramów** przewidzianych w tym standardzie. W nawiązaniu do pierwszej części niniejszej książki strukturę interfejsu narzędzia Enterprise Architect zilustrowano na przykładzie diagramu sekwencji (rysunek 17.1), diagramu harmonogramowania (rysunek 17.2) oraz diagramu czynności (rysunek 17.3). Opisy tych diagramów znajdują się odpowiednio w rozdziałach 7., 9. i 4.

Układ interfejsu Enterprise Architect (rysunek 17.1) określić należy jako typowy dla narzędzi typu CASE. Jest on czytelny, a zarazem nieskomplikowany, co ułatwia użytkowanie narzędzia. Można go podzielić na pięć podstawowych obszarów:

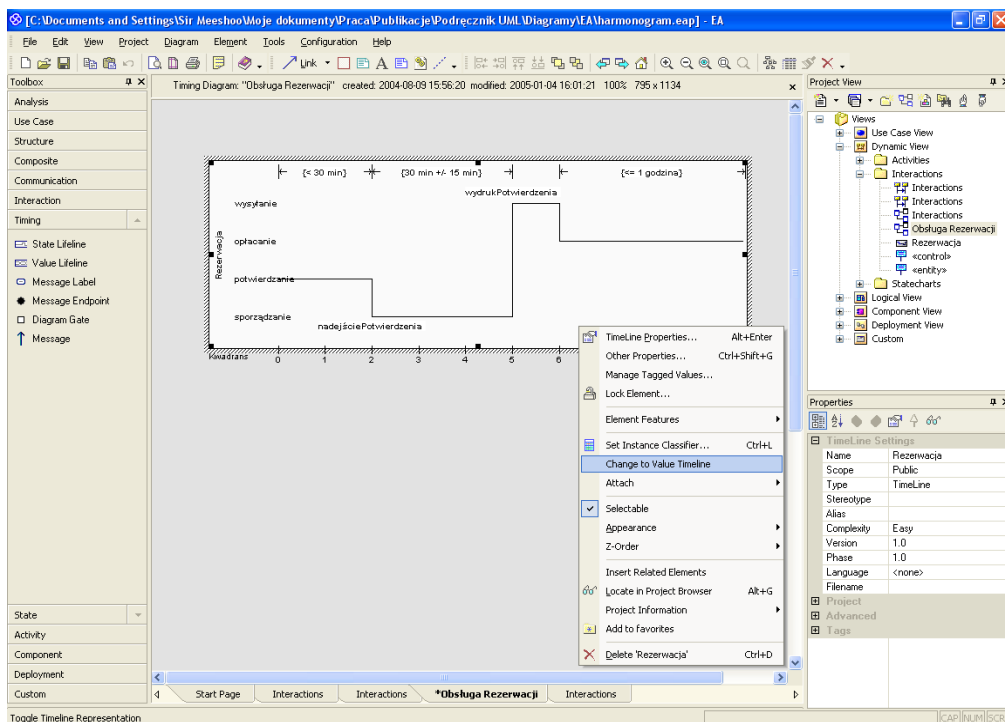
1. okno główne,
2. pasek narzędziowy,
3. zasobnik narzędziowy,
4. przeglądarkę,
5. okienko właściwości.



Rysunek 17.1. Interfejs narzędzia Enterprise Architect

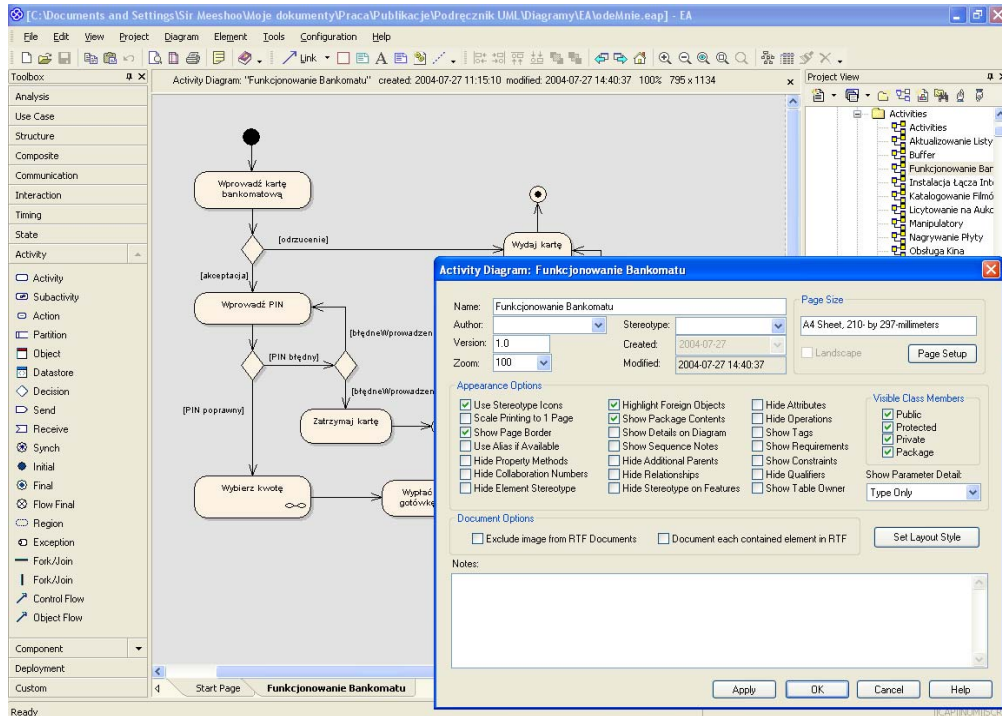
W oknie głównym program umieszcza tworzone diagramy. Przełączanie się pomiędzy nimi możliwe jest przy wykorzystaniu zakładek w dolnej części ekranu. Pasek narzędziowy zawiera opcje programu niezależne od aktualnie edytowanego diagramu. Zasobnik narzędziowy udostępnia możliwe do wykorzystania kategorie modelowania języka UML posegregowane pomiędzy poszczególne diagramy. Zarządzanie poszczególnymi elementami modelu ułatwia przeglądarka oparta na technice przeciągnij-i-upuść oraz mechanizmie menu kontekstowych. Z kolei wykorzystanie okienka właściwości to jedna z możliwości ustalania szczegółowych cech obecnie zaznaczonej kategorii modelowania.

Opisywane narzędzie wykazuje znaczną elastyczność w dostosowywaniu wykorzystywanej notacji do potrzeb użytkownika. Objawia się to chociażby możliwością automatycznej zmiany konwencji prezentowania informacji w ramach diagramu harmonogramowania (rysunek 17.2), stosowania rozgałęzień i ograniczeń czasowych na diagramach sekwencji oraz wprowadzania znacznej liczby dostępnych stereotypów.



Rysunek 17.2. Zmiana konwencji diagramu harmonogramowania

Enterprise Architect wykracza ponadto poza typową dla narzędzi CASE praktykę udostępniania opcji konfiguracyjnych dotyczących projektu jako całości. Narzędzie umożliwia ustalanie właściwości (takich jak stosowana składnia atrybutów czy operacji) każdego tworzonego diagramu z osobna. Ustalanie właściwości na przykładzie diagramu czynności prezentuje rysunek 17.3.



Rysunek 17.3. Ustalanie właściwości diagramu

Narzędzia CASE firmy IBM pod marką Rational

Jednym z liderów na rynku narzędzi wspomagających UML jest koncern IBM, który przejął w 2003 roku korporację Rational. Opracowano całą rodzinę adekwatnych narzędzi pod marką Rational, wspomagających produkcję oprogramowania we wszystkich dyscyplinach metodyki RUP. Narzędzia te składają się na **zintegrowany**, modułarny pakiet narzędziowy przeznaczony dla zespołów tworzących aplikacje o zastosowaniach biznesowych, oprogramowanie wbudowane oraz produkty informatyczne ogólnego zastosowania. Integracja ta sprawia, że opisywana linia narzędzi CASE, która nosi nazwę IBM Software Development Platform, stanowi kompleksową oraz efektywną podstawę realizacji cyklu życia systemu.

IBM Software Development Platform

Różnorodność narzędzi marki Rational umożliwia ich konfigurowanie według specyficznych potrzeb użytkowników oraz członków zespołu projektowego [IBM-2005]. W istocie — przyjmując jako podstawę klasyfikacji **obszar zastosowań** — można je podzielić na 5 grup. Grupy te odpowiadają:

- ♦ projektowaniu i budowie,
- ♦ zarządzaniu procesem i zasobami,
- ♦ specyfikacji wymagań i analizie,
- ♦ zarządzaniu konfiguracją oprogramowania,
- ♦ jakości oprogramowania.

W kontekście niniejszego opracowania na szczególną uwagę zasługuje grupa narzędzi nastawionych na realizację pierwszego z wymienionych obszarów zastosowań. Grupę tę tworzy szereg narzędzi:

- ♦ Rational Software Architect,
- ♦ Rational Software Modeler,
- ♦ Rational Web Developer for WebSphere Software,
- ♦ Rational Application Developer for WebSphere Software,
- ♦ Rational Rose Technical Developer,
- ♦ Rational Rose XDE Developer,
- ♦ Rational Rose Data Modeler,
- ♦ Rational PurifyPlus,
- ♦ Rational Test RealTime,
- ♦ Rational Ada Developer.

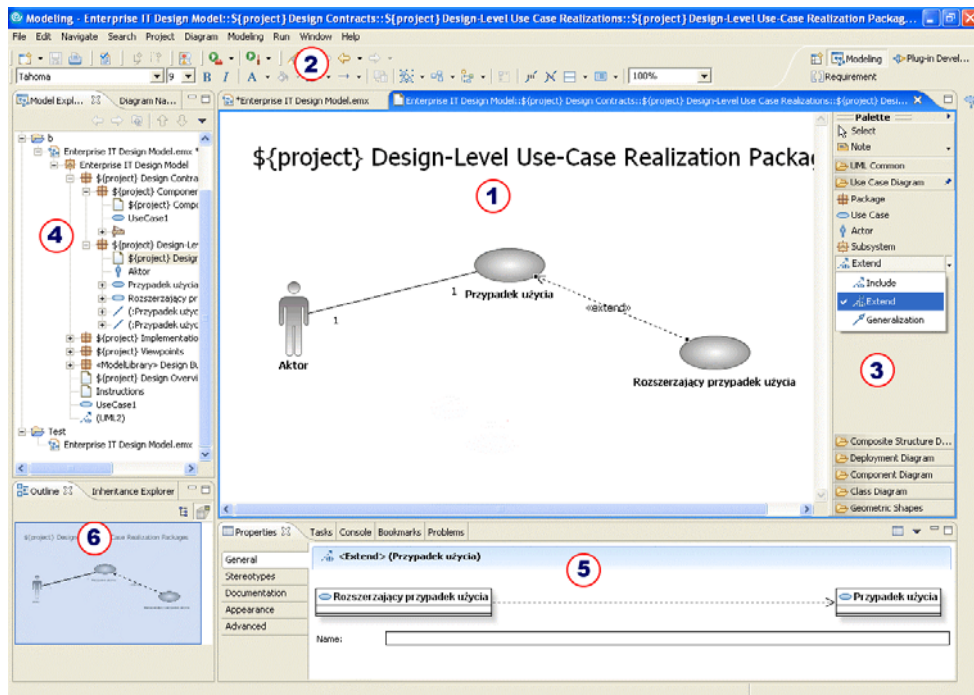
Szersza charakterystyka wymienionych narzędzi dostępna jest w opracowaniu [IBM-2005].

Zastosowanie IBM Rational Software Modeler

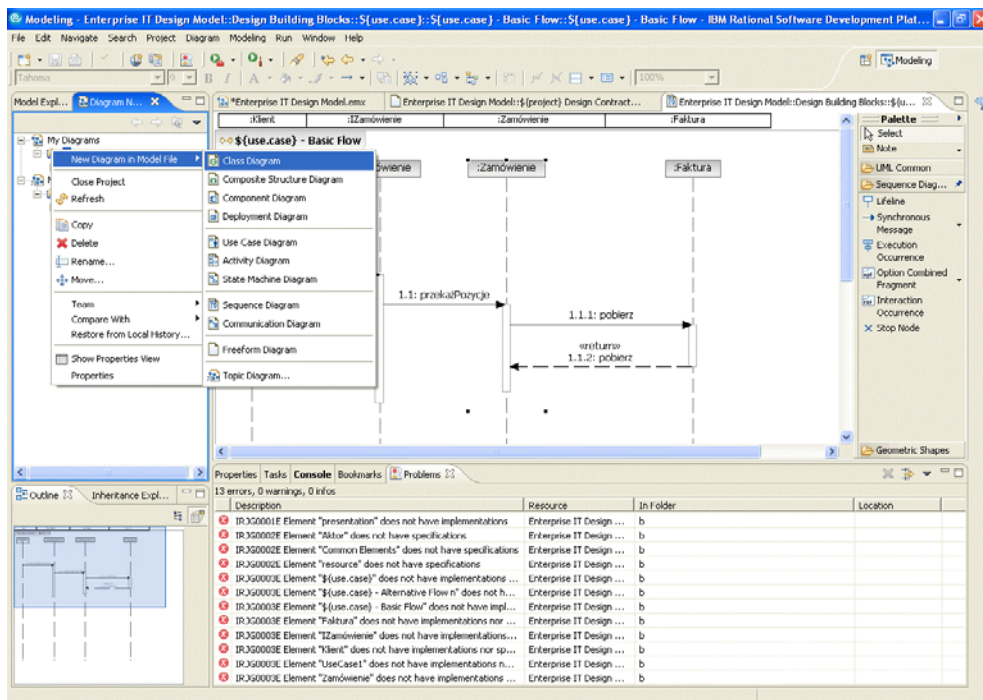
Funkcjonalność IBM Rational Software Modeler jest **rozbudowana**. Jej charakterystykę zamieszczono na stronie [IBM-2005a]. W niniejszej książce zaprezentowano zatem:

- ♦ interfejs narzędzia (rysunek 17.4),
- ♦ wprowadzanie nowego diagramu do modelu (rysunek 17.5),
- ♦ szczegółową specyfikację klasy (rysunek 17.6).

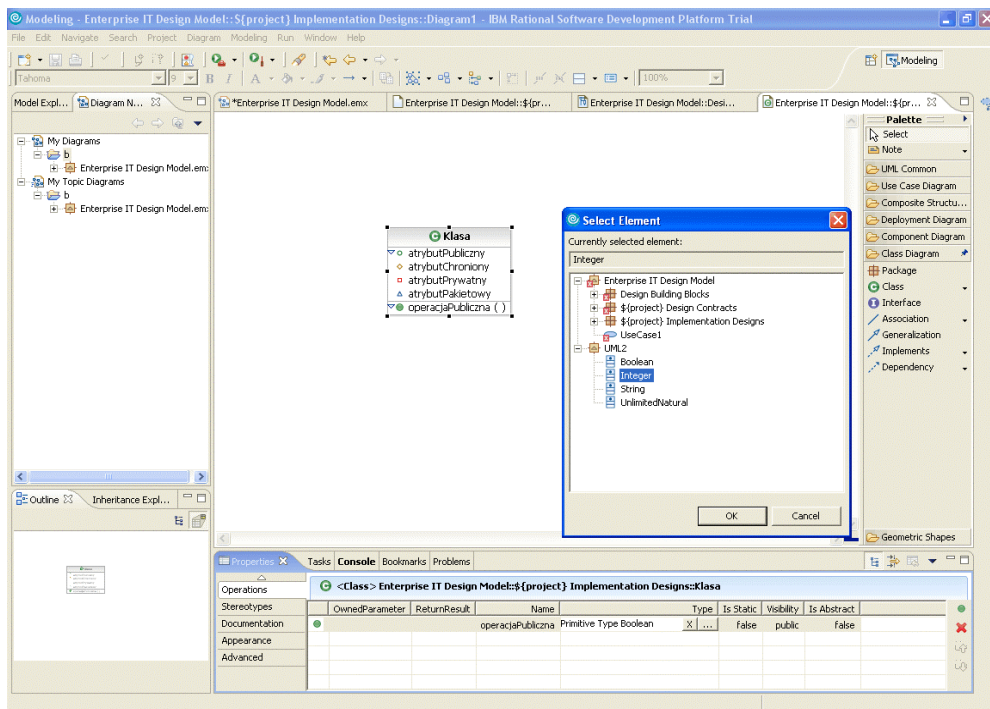
W ramach właściwego interfejsu użytkownika można wyodrębnić sześć podstawowych obszarów, wskazanych na rysunku 17.4. Interfejs w pewnym stopniu różni się od rozwiązań GUI przedstawionych w punkcie *Sparx Systems Enterprise Architect*. Obszary interfejsu użytkownika na rysunku 17.4 ponumerowano analogicznie jak na rysunku 17.1. Zasygnalizowane różnice obejmują przede wszystkim uwzględnienie okienka pomocniczego (obszar 6), zawierającego takie elementy jak ogólny zarys aktualnego diagramu oraz informacje związane z dziedziczeniem elementów modelu. Producent



Rysunek 17.4. Interfejs narzędzia IBM Rational Software Modeler



Rysunek 17.5. Wprowadzanie nowego diagramu do modelu



Rysunek 17.6. Szczegółowa specyfikacja klasy

rozwinął ponadto koncepcję przeglądarki (obszar 4) oraz okienka zawierającego szczegółowe informacje związane z aktualnym projektem (obszar 5). Przeglądarka przyjmuje kilka różnych układów — przeglądarki modelu jako całości, przeglądarki poszczególnych diagramów oraz przeglądarki wzorców projektowych. Z kolei okienko informacji szczegółowych uwzględnia m.in. właściwości wskazanego elementu modelu, zaplanowane zadania oraz semantyczne niezgodności w ramach modelu.

Kluczowe znaczenie w tworzeniu diagramu zamieszczonego na rysunku 17.4 miał zasobnik narzędziowy. W tej konkretnej sytuacji w zasobniku wyświetlone zostały wyłącznie kategorie modelowania typowe dla diagramów przypadków użycia, chociaż możliwe jest jednoczesne wyświetlenie kategorii modelowania kilku różnych diagramów. Poszczególne kategorie modelowania znajdujące się w zasobniku uporządkowano w hierarchiczny sposób. Warto zwrócić uwagę na różnice pomiędzy notacją stosowaną w narzędziu a notacją wyspecyfikowaną w standardzie UML (część pierwsza książki). Producent przyjął swobodną konwencję, zorientowaną na zwiększenie atrakcyjności przekazu przy jednoczesnym zachowaniu czytelności. Różnice te należy interpretować w kategoriach stosowania stereotypów graficznych.

Rysunek 17.5 ilustruje wprowadzanie nowego diagramu do modelu systemu. W tym celu wykorzystywany jest mechanizm menu kontekstowych oraz przeglądarka diagramów. IBM Rational Software Modeler umożliwia tworzenie większości diagramów UML 2.0 (por. tabela 17.2). Producent zdecydował się udostępnić opcję tworzenia diagramów o swobodnej, niezdefiniowanej formie (ang. *freeform diagrams*), wykorzystujących stereotypowane kategorie modelowania.

Narzędzie umożliwia sprawdzanie semantycznej spójności tworzonego modelu. I tak na rysunku 17.5 przedstawiono listę niespójności, uzyskaną w efekcie uruchomienia funkcji sprawdzania semantycznej poprawności modelu.

Przykład zaprezentowany na rysunku 17.6 ilustruje właściwości klasy. Właściwości podzielono na szereg kategorii, uwzględniających m.in. atrybuty, operacje oraz stereotypy przypisane do omawianej klasy. Zakładka właściwości umożliwia ponadto modyfikację ustawień związanych ze sposobem graficznego prezentowania danego elementu. Jak przedstawiono na rysunku, szczegółowa specyfikacja właściwości elementów jest wspomagana dodatkowymi okienkami dialogowymi.

Poseidon for UML

Poseidon jest narzędziem CASE zaproponowanym przez organizację Gentleware. Wersja Poseidon 3.1 dostępna jest w pięciu wydaniach:

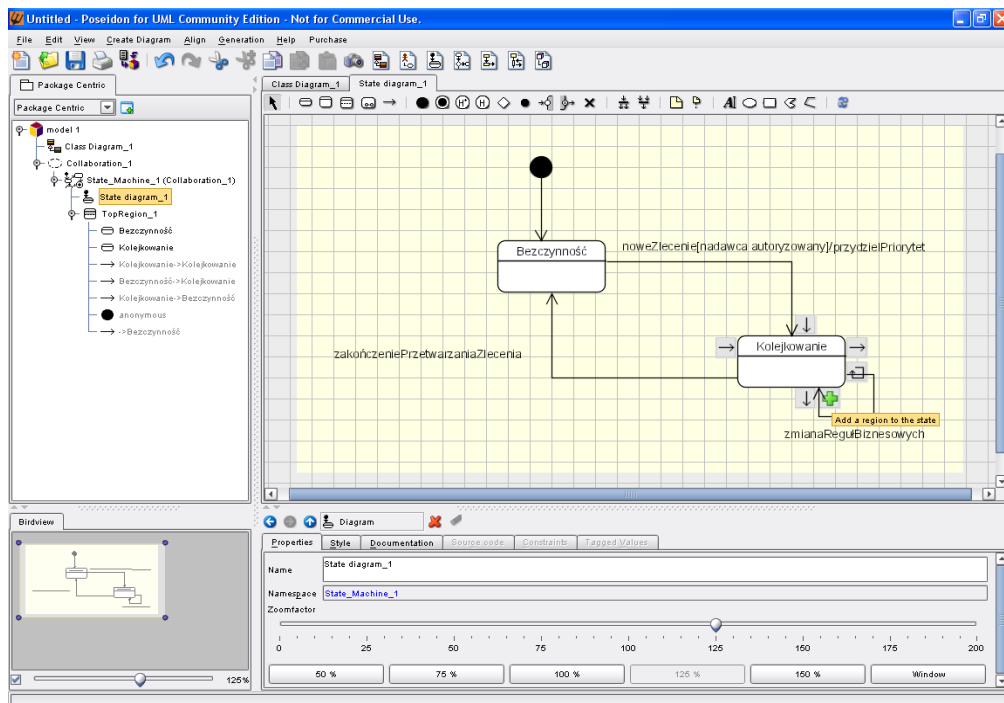
1. Community Edition — dla studentów (niekomercyjne użytkowanie);
2. Standard Edition — dla analityków;
3. Professional Edition — dla programistów;
4. Enterprise Edition — dla zespołów programistycznych;
5. Embedded Enterprise Edition — dla programistów systemów wbudowanych (ang. *embedded systems*).

Wyczerpujące porównanie cech poszczególnych wydań narzędzia Poseidon 3.1 dostępne jest na stronie producenta w formie tabelarycznej [Gentleware-2005].

Narzędzie umożliwia modelowanie wszystkich diagramów UML 1.5. Należy podkreślić, że producent deklaruje częściową zgodność narzędzia z wersją 2.0 języka UML. W wersji 3.1 zgodność ta ogranicza się do składni diagramów maszyn stanowych (rysunek 17.7). Pełne wsparcie modelowania zgodnie ze standardem UML 2.0 przewidziane jest w wersjach narzędzia powyżej 3.2.

Platforma Eclipse

Istotną rolę w warsztacie narzędziowym twórców systemów informatycznych odgrywa platforma Eclipse, która powstała w ramach projektu realizowanego przez konsorcjum założone przez firmę IBM (www.eclipse.org). Platforma Eclipse jest darmowym, zintegrowanym środowiskiem programistycznym (ang. *Integrated Development Environment*) napisanym w Javie. Jest podstawą tworzenia rozbudowanych środowisk programistycznych. Kompleksowa realizacja cyklu życia projektu informatycznego we wszystkich fazach jest możliwa dzięki zastosowaniu modułów dodatkowych — wtyczek (ang. *plug-ins*), które po instalacji stają się modułami platformy Eclipse. Na rynku dostępne są moduły dodatkowe:



Rysunek 17.7. Poseidon for UML

- ♦ komercyjne,
- ♦ otwarte (ang. *Open Source*).

Platformę Eclipse należy zatem zakwalifikować jako otwarte środowisko narzędziowe, umożliwiające pracę nad projektami, w których wykorzystuje się wiele języków programowania: Java, C/C++, JavaScript, XML, HTML, PHP, Python, Cobol i wiele innych. Zastosowanie konstrukcji opartej na modułach dodatkowych umożliwia daleko idące dostosowanie środowiska do potrzeb twórców oprogramowania, w tym analityków, projektantów, programistów i osób odpowiedzialnych za testy. Należy podkreślić, że funkcjonalność modułów dodatkowych nie ogranicza się wyłącznie do programowania; wspomagane są także inne etapy tworzenia systemów informatycznych, w tym:

- ♦ modelowanie systemów informatycznych,
- ♦ zarządzanie wymaganiami,
- ♦ kontrola wersji,
- ♦ testowanie,
- ♦ tworzenie interfejsów,
- ♦ prototypowanie.

W kontekście platformy Eclipse indywidualne narzędzia CASE są modułami dodatkowymi.

Przykładowe komercyjne narzędzia modelowania z wykorzystaniem języka UML, będące modułami platformy Eclipse, to:

- ♦ Borland Together Edition for Eclipse 6.2 — <http://www.borland.com>;
- ♦ IBM Rational Software — Rational Software Architect — <http://www.ibm.com/software/rational>;
- ♦ innoQ — iQgen 2.0, Model Driven Software Generator — <http://www.innoq.com/iqgen>;
- ♦ Modelistic — Modelistic JME — <http://www.modelistic.com>;
- ♦ mvmssoft — Slime UML (Slim Modeling Environment) — <http://www.mvmssoft.de/content/plugins/slime/slime.htm>;
- ♦ No Magic, Inc — MagicDraw UML — <http://www.magicdraw.com>;
- ♦ Omondo — EclipseUML — <http://www.omondo.com>;
- ♦ Ontogenics — hyperModel — <http://xmlmodeling.com/products/hyperModel>;
- ♦ Visual Paradigm — SDE for Eclipse — <http://www.visual-paradigm.com>.

Modułami dodatkowymi platformy Eclipse o charakterze oprogramowania otwartego, wspierającymi standard UML, są przykładowo:

- ♦ MDE (MDE Studio: UML modeler) — <http://www.mlglobal.com>;
- ♦ Motion Modeling — <http://motionmodeling.sourceforge.net>;
- ♦ UniMod: Executable UML for Eclipse — <http://unimod.sf.net>;

Dzięki zastosowaniu niekomercyjnych modułów dodatkowych (lub zaprojektowaniu własnych) możliwe jest stworzenie zaawansowanego, nieodpłatnego warsztatu narzędziowego twórców systemów informatycznych.

Kompletna lista modułów dodatkowych, zarówno komercyjnych, jak i o charakterze oprogramowania otwartego, dostępna jest w portalu Eclipse [Eclipse-2005].

UML i RUP w zasobach Internetu

Zarówno język UML, jak i metodyka RUP są znacząco obecne w Internecie w postaci różnych serwisów i stron. Do najważniejszych należy zaliczyć:

- ♦ Object Management Group OMG — www.omg.org;
- ♦ UML Forum — www.uml-forum.com;
- ♦ UML 2.0 Partners — www.u2-partners.org;
- ♦ Community UML — community-ml.org;

- ♦ 2U Consortium — www.2uworks.org;
- ♦ pUML — www.cs.york.ac.uk;
- ♦ Popkin's UML Resources Center — www.popkin.com.

Podstawowe pojęcia

CASE	Przeglądarka
Generowanie	Modelu
Dokumentacji	Poszczególnych diagramów
Kodu źródłowego	Wzorców projektowych
Gentleware Poseidon for UML	Zasobnik narzędziowy
IBM Software Development Platform	Inżynieria
Software Modeler	Wahadłowa
Interfejs użytkownika	Zwrotna
Menu kontekstowe	Model
Okienko	RUP
Dialogowe	Sparx Systems Enterprise Architect
Główne	Specyfikacja
Pomocnicze	Stereotyp
Właściwości	UML
Pasek narzędziowy	

Pytania i zadania

1. Wyjaśnij skrót CASE.
2. Jakie są przyczyny i efekty stosowania narzędzi CASE?
3. Przy użyciu przeglądarki internetowej rozszerz zaprezentowaną listę oprogramowania narzędziowego wspomagającego język UML i metodykę RUP.
4. Wykorzystując adresy internetowe produktów, które zawiera tabela 17.1, dokonaj charakterystyki poszczególnych narzędzi.
5. Wskaż przykładowe kryteria decyzyjne przy wyborze oprogramowania narzędziowego.

6. Jakie języki programowania oraz standardy są najczęściej wspierane przez narzędzia CASE w zakresie generowania kodu i inżynierii zwrotnej?
7. Wymień podstawowe platformy, na których można zainstalować i użytkować produkty wspomagające proces tworzenia oprogramowania.
8. Wskaż języki programowania oraz standardy obsługiwane przez narzędzie Enterprise Architect firmy Sparx Systems w zakresie generowania szkieletowego kodu źródłowego oraz inżynierii zwrotnej.
9. Wskaż obszary zastosowania oprogramowania narzędziowego firmowanego przez IBM.
10. Wymień znane Ci narzędzia CASE tej firmy, zorientowane na zadania o charakterze projektowym oraz wytwórczym.
11. Pobierz ze strony producenta wersję demonstracyjną wybranego narzędzia CASE i zainstaluj ją.
12. Wykorzystując narzędzie IBM Rational Software Modeler, opracuj następujące diagramy zamieszczone w części studialnej poszczególnych rozdziałów części pierwszej niniejszego opracowania:
 - ♦ przypadków użycia,
 - ♦ klas,
 - ♦ czynności,
 - ♦ maszyny stanowej,
 - ♦ sekwencji,
 - ♦ rozlokowania.
13. Wymień grupy odbiorców, do których adresowane jest narzędzie Poseidon for UML w wersji 3.1.
14. Scharakteryzuj konstrukcję platformy Eclipse.
15. Wymień przykładowe moduły dodatkowe platformy Eclipse, wspomagające modelowanie systemów informatycznych.
16. Zaloguj się na wskazanych w punkcie *UML i RUP w zasobach Internetu* stronach i serwisach internetowych języka UML i RUP. Oceń ich praktyczną użyteczność.

Dodatki

Dodatek A

Definicje i pojęcia

Niniejszy dodatek podsumowuje definicje szczególnie istotnych elementów języka UML, jak również diagramów występujących w wersji 2.0 tego standardu. Zostały one przedstawione w układzie alfabetycznym.

Akcja

Elementarna jednostka specyfikacji zachowania, która reprezentuje transformację lub przetwarzanie w modelowanym systemie.

Aktor

Spójny zbiór ról odgrywanych przez użytkowników przypadku użycia w czasie interakcji z tym przypadkiem użycia.

Aktor biznesowy

Rola pełniona przez użytkownika działającego w otoczeniu organizacji.

Artefakt

Każdy sztucznie wytworzony produkt.

Asocjacja

Związek pomiędzy dwoma lub więcej klasyfikatorami, opisujący powiązania pomiędzy ich instancjami.

Biznesowa klasa przechowująca

Fizycznie istniejący byt, użytkowany i przetwarzany przez pracowników biznesowych oraz pracowników kontaktu.

Biznesowy przypadek użycia

Proces biznesowy, dostarczający wyników istotnych z punktu widzenia aktora biznesowego.

Brama

Punkt przejścia komunikatów z (do) diagramów sekwencji, przywoływanych wystąpień interakcji oraz fragmentów wyodrębnionych do (z) ich otoczenia.

Czynność

Czynność to określone zachowanie złożone z logicznie uporządkowanych ciągów podczynności, akcji oraz obiektów w celu wykonania pewnego procesu.

Czynność biznesowa

Określone zachowanie złożone z logicznie uporządkowanych ciągów podczynności, akcji oraz obiektów w celu wykonania pewnego procesu biznesowego.

Diagram analityczny

Technika wspomagająca język UML, która służy do dokumentowania wyników prac analitycznych i wczesnych prac projektowych.

Diagram czynności

Diagram czynności to diagram przedstawiający sekwencyjne i (lub) współbieżne przepływy sterowania i danych pomiędzy uporządkowanymi ciągami czynności, akcji i obiektów.

Diagram harmonogramowania

Diagram harmonogramowania jest rodzajem diagramu interakcji, reprezentującym na osi czasu zmiany dopuszczalnych stanów, jakie może przyjmować instancja klasyfikatora uczestnicząca w interakcji.

Diagram klas

Diagram klas to diagram stanowiący obraz statycznych, deklaratywnych elementów dziedziny przedmiotowej oraz związków między nimi.

Diagram komponentów

Diagram komponentów to rodzaj diagramu wdrożeniowego, który wskazuje organizację i zależności między komponentami.

Diagram komunikacji

Diagram komunikacji jest rodzajem diagramu interakcji, specyfikującym strukturalne związki pomiędzy biorącymi udział w interakcji instancjami klasyfikatorów oraz wymianę komunikatów pomiędzy tymi instancjami.

Diagram maszyny stanowej

Diagram maszyny stanowej to diagram odzwierciedlający dyskretne, skokowe zachowanie skończonych systemów stan-przejście.

Diagram obiektów

Diagram obiektów to instancja diagramu klas, odwzorowująca strukturę systemu w wybranym momencie jego działania.

Diagram pakietów

Diagram pakietów to diagram przedstawiający logiczną strukturę dokumentacji systemu w postaci zestawu pakietów połączonych zależnościami i zagnieżdżeniami.

Diagram przypadków użycia

Diagram przypadków użycia to diagram przedstawiający występujące w danej dziedzinie przedmiotowej przypadki użycia, aktorów oraz związki między nimi.

Diagram rozlokowania

Diagram rozlokowania to rodzaj diagramu wdrożeniowego, który przedstawia sieć połączonych ścieżkami komunikowania węzłów z ulokowanymi na nich artefaktami.

Diagram sekwencji

Diagram sekwencji to rodzaj diagramu interakcji, opisujący interakcje pomiędzy instancjami klasyfikatorów systemu w postaci sekwencji komunikatów wymienianych między nimi.

Diagram sterowania interakcją

Diagram sterowania interakcją jest rodzajem diagramu interakcji, dokumentującym przepływ sterowania pomiędzy logicznie powiązanymi diagramami a fragmentami interakcji z wykorzystaniem kategorii modelowania diagramów czynności.

Diagram struktur połączonych

Diagram struktur połączonych jest diagramem przedstawiającym wzajemnie współdziałające części w celu osiągnięcia pożądanej funkcjonalności współdziałania.

Dziedziczenie

Przyporządkowanie atrybutów i operacji klasom obiektów na podstawie hierarchicznej zależności między nimi.

Faza

Okres między kolejnymi punktami przeglądu cyklu iteracyjno-przyrostowego, w którym zrealizowano niezbędne czynności i opracowano adekwatne artefakty.

Fragment wyodrębniony

Logicznie spójny obszar interakcji, część diagramu sekwencji charakteryzująca się specyficznymi właściwościami określonymi przez operator interakcji.

Hermetyzacja

Różnicowanie dostępu do obiektu poprzez ujawnienie otoczeniu tylko tych informacji o jego atrybutach lub operacjach, które są niezbędne do efektywnego odwoływania się do tego obiektu w systemie za pośrednictwem komunikatów.

Interakcja

Wymiana bodźców, impulsów i komunikatów pomiędzy instancjami klasyfikatorów w systemie.

Interfejs

Zestaw operacji, które wyznaczają usługi oferowane przez klasę lub komponent.

Iteracja

Wielokrotne, policzalne powtórzenie jednostki zachowania w systemie.

Jednostka organizacyjna

Grupa pracowników biznesowych, biznesowych jednostek przechowujących, związków między nimi, współdziałań biznesowych i innych składników organizacji.

Karta CRC

Klasa, której podstawową zawartość stanowią zobowiązania oraz lista współpracujących klas.

Klasa

Uogólnienie zbioru obiektów, które mają takie same atrybuty, operacje, związki i znaczenie.

Klasa aktywna

Klasa, której instancje stanowią aktywne obiekty, mogące autonomicznie inicjować własne wątki i sterować wykonaniem tych wątków.

Klasyfikator

Abstrakcyjny byt, który stanowi generalizację kategorii modelowania języka UML służących do opisanie cech statycznych i dynamicznych systemu.

Komponent

Hermetyczny, wymienny moduł oprogramowania systemu, realizujący określone jego usługi za pośrednictwem interfejsów.

Komunikat

Specyfikacja wymiany informacji między obiektami, zawierająca zlecenia wykonania określonej operacji.

Kwalifikator

Atrybut lub lista atrybutów, których wartości porządkują zbiór obiektów klasy dostępnych w asocjacji.

Liczebność

Specyfikacja zakresu dopuszczalnej liczby obiektów danej klasy biorących udział w danym związku.

Linia życia

Powiązana z konkretną instancją klasyfikatora linia na diagramie sekwencji, wskazująca okres istnienia tej instancji.

Maszyna stanowa

Zachowanie określające sekwencje stanów, przez które przechodzi obiekt bądź interakcja w odpowiedzi na zdarzenia zachodzące w czasie swojego cyklu życia wraz z ich reakcjami na te zdarzenia.

Metka

Jawne zdefiniowanie właściwości w postaci przyporządkowania nazwa-wartość.

Model

Odwzorowanie, uogólnienie rzeczywistości.

Modelowanie biznesowe

Sposób odwzorowywania i dokumentowania procesów biznesowych.

Obiekt

Każdy byt — pojęcie lub rzecz — mający znaczenie w kontekście rozwiązywania problemu w danej dziedzinie przedmiotowej.

Obszar przerwania

Grupa czynności, w obrębie której w wyniku działania przepływu przerwania realizacja wszystkich czynności jest bezzwłocznie przerywana.

Obszar rozszerzenia

Ściśle zdefiniowany fragment diagramu czynności z jednoznacznie wyspecyfikowanymi wejściami i wyjściami, wykonywany wielokrotnie, stosownie do liczby elementów na wejściu.

Ograniczenie

Wyrażenie semantyczne określające warunek bądź zastrzeżenie związane z ograniczonym elementem modelowania bądź grupą elementów.

Operand interakcji

Oddzielna część fragmentu wyodrębnionego; subfragment.

Operator interakcji

Sprecyzowanie funkcjonalności realizowanej przez fragment wyodrębniony.

Ośrodek sterowania

Specyfikacja wykonywania czynności, operacji lub innej jednostki zachowania w ramach interakcji.

Pakiet

Mechanizm ogólnego zastosowania, służący do organizowania elementów w grupy.

Partycja diagramu czynności

Mechanizm grupowania elementów diagramu czynności powiązanych przepływami sterowania i przepływami danych, pełniących określoną, wspólną rolę na diagramie.

Polimorfizm

Możliwość nadawania tej samej nazwy różnym atrybutom i operacjom oraz wykonywania różnych procedur i akcji przez operacje o tych samych nazwach.

Pracownik biznesowy

Funkcjonujący w ramach organizacji pracownik lub system, pełniący określoną rolę lub zestaw ról wewnątrz procesu biznesowego.

Pracownik kontaktu

Pracownik biznesowy obsługujący proces biznesowy i jednocześnie będący w bezpośredniej interakcji z aktorami biznesowymi.

Przejście

Relacja między dwoma stanami, wskazująca, że obiekt znajdujący się w pierwszym stanie wykona pewne akcje i przejdzie do drugiego stanu, ilekroć zajdzie określone zdarzenie i będą spełnione określone warunki.

Przepływ sterowania

Relacja między dwoma czynnościami bądź akcjami, wskazująca, że po wykonaniu źródłowej czynności albo akcji sterowanie zostanie przekazane do docelowej czynności albo akcji.

Przypadek użycia

Specyfikacja ciągu akcji i ich wariantów, które system (lub inna jednostka) może wykonać poprzez interakcje z aktorami tego systemu.

Realizacja

Związek znaczeniowy między klasyfikatorami, z których jeden określa kontrakt, a drugi zapewnia wywiązanie się z niego.

Rola

Powinność pełniona przez jedną klasę obiektu wobec drugiej klasy.

Rozlokowanie

Alokacja artefaktów lub instancji artefaktów w określonym węźle.

Rozgałęzienie

Sposób przekazania sterowania na diagramie sekwencji z linii życia (ośrodka sterowania) klasyfikatora-nadawcy do klasyfikatorów-odbiorców w zależności od spełnienia warunku przypisanego do przesyłanego komunikatu.

Scenariusz

Określony ciąg akcji dokumentujący zachowanie.

Stan

Okoliczność lub sytuacja, w jakiej obiekt znajduje się w cyklu swojego życia, kiedy spełnia warunek, wykonuje czynność lub czeka na zdarzenie.

Stereotyp

Mechanizm rozszerzalności, który wzbogaca zbiorowość standardowych kategorii modelowania języka UML.

Uogólnienie

Związek o charakterze taksonomicznym pomiędzy klasyfikatorem ogólnym a specjalizowanym.

Węzeł

Fizyczny lub logiczny zasób przetwarzający, na którym są osadzone artefakty użytkowanego systemu.

Współbieżność

Wystąpienie dwóch lub więcej czynności w tym samym czasie, co pozwala na przeplatanie lub równoczesne wykonywanie wątków.

Współdziałanie

Specyfikacja pożądanej funkcjonalności zestawu powiązanych i współpracujących klasyfikatorów, z których każdy pełni dedykowaną rolę.

Współdziałanie biznesowe

Zestaw diagramów przedstawiających wspomagające proces biznesowy elementy organizacji, takie jak pracownicy biznesowi czy biznesowe obiekty przechowujące.

Zależność

Zależność to taki związek pomiędzy dwoma elementami modelowania, w którym zmiana jednego z nich, niezależnego, wpływa na drugi, zależny.

Zobowiązanie

Kontrakt lub odpowiedzialność klasy.

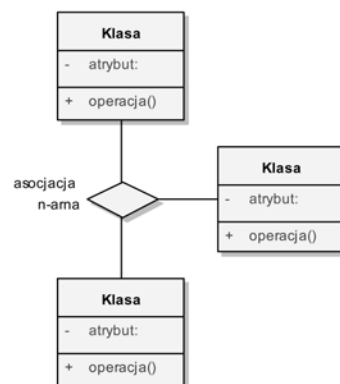
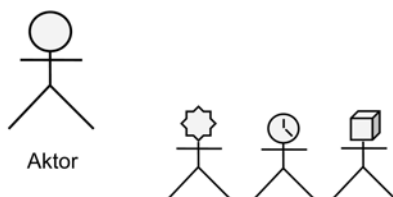
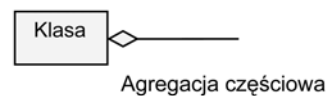
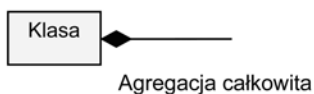
Związek

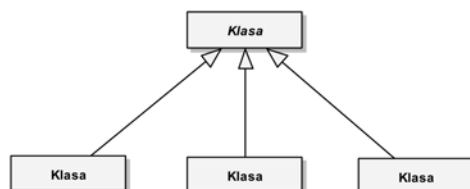
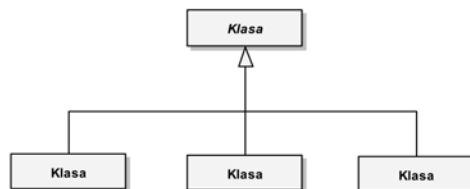
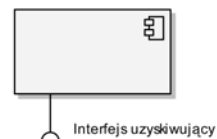
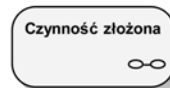
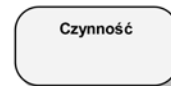
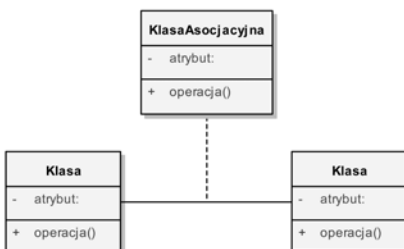
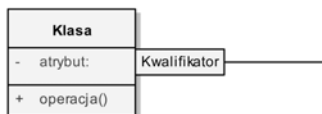
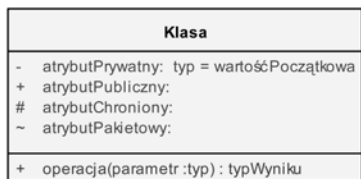
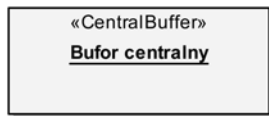
Semantyczne powiązanie pomiędzy elementami modelu.

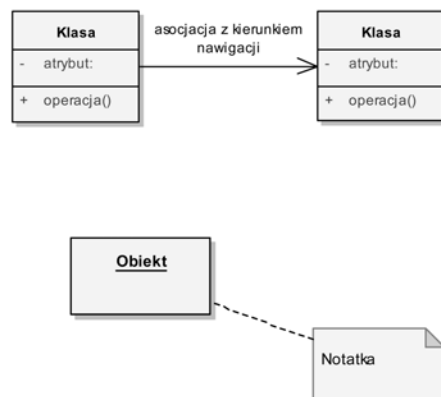
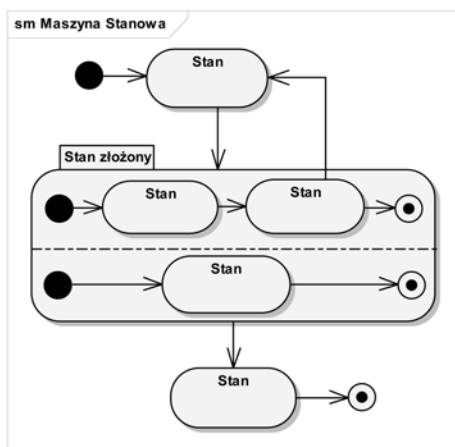
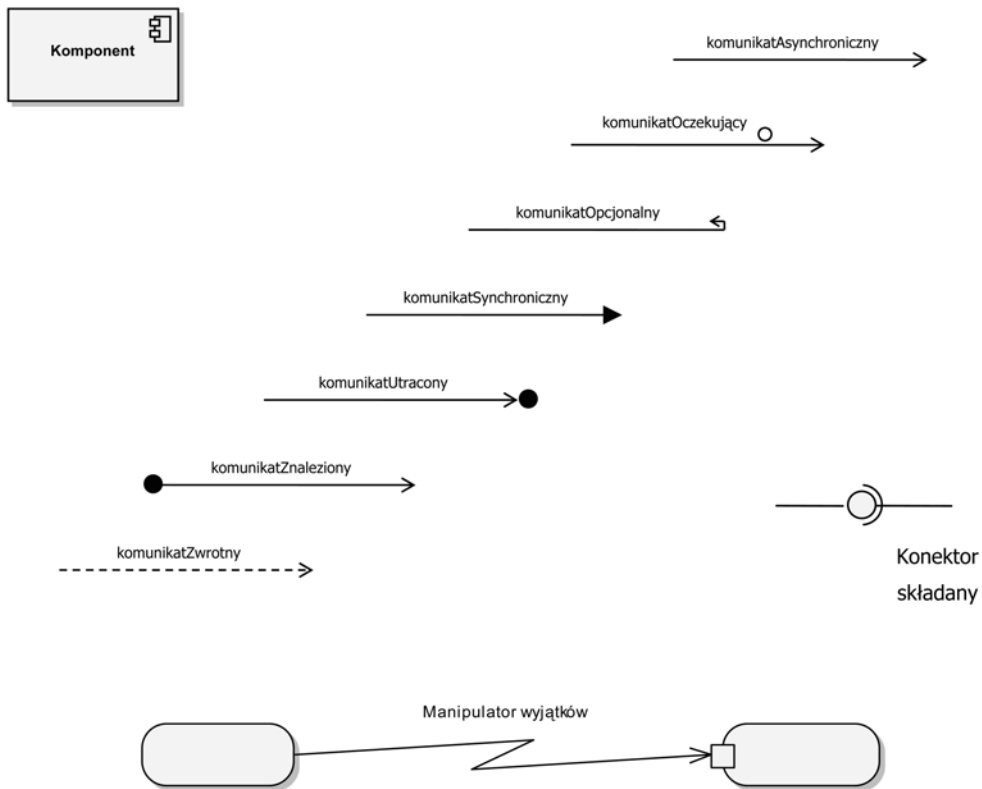
Dodatek B

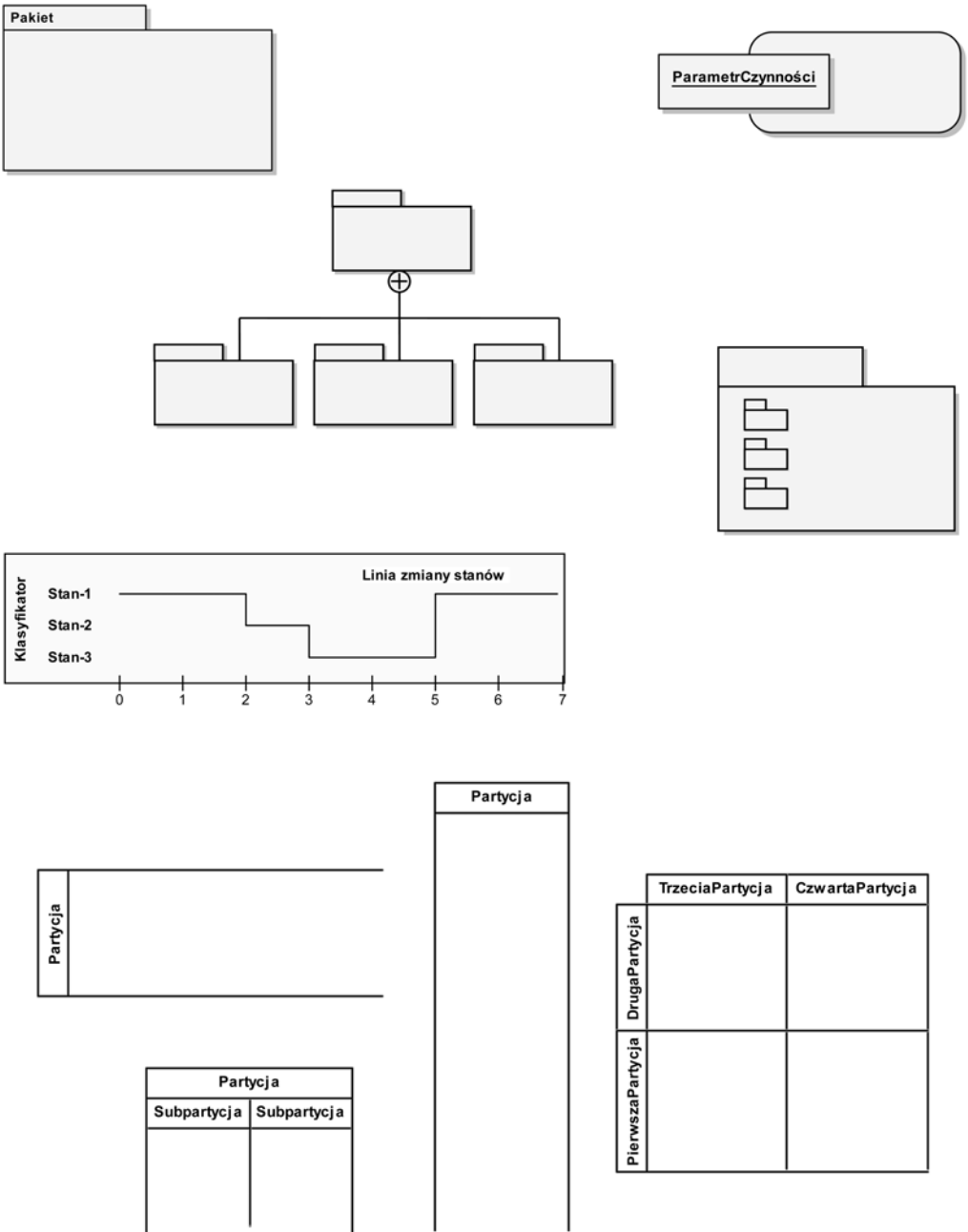
Notacja graficzna

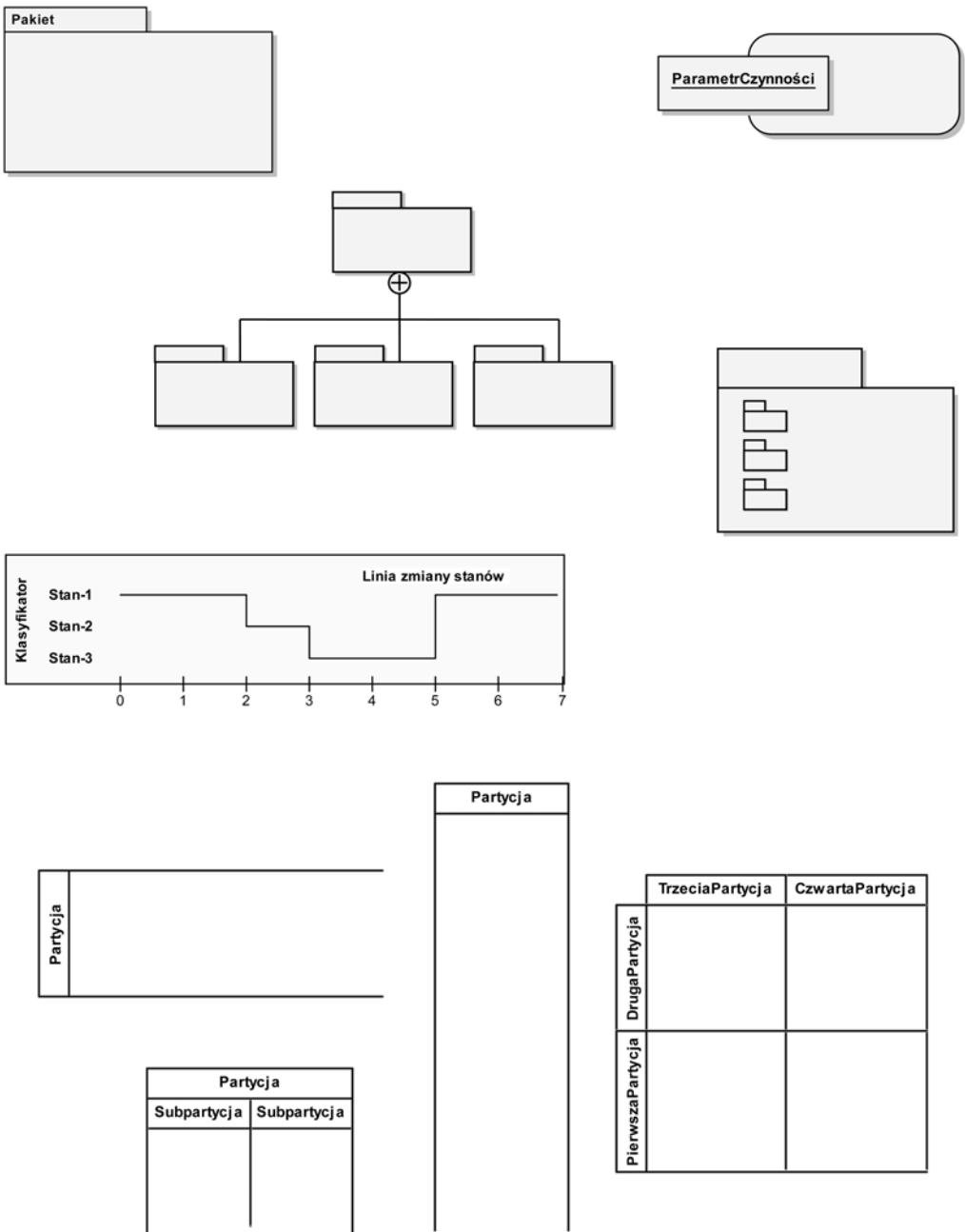
Dodatek B podsumowuje notację kategorii modelowania stosowanych do konstruowania poszczególnych rodzajów diagramów. Kompletne diagramy zostały zilustrowane w dodatku C.

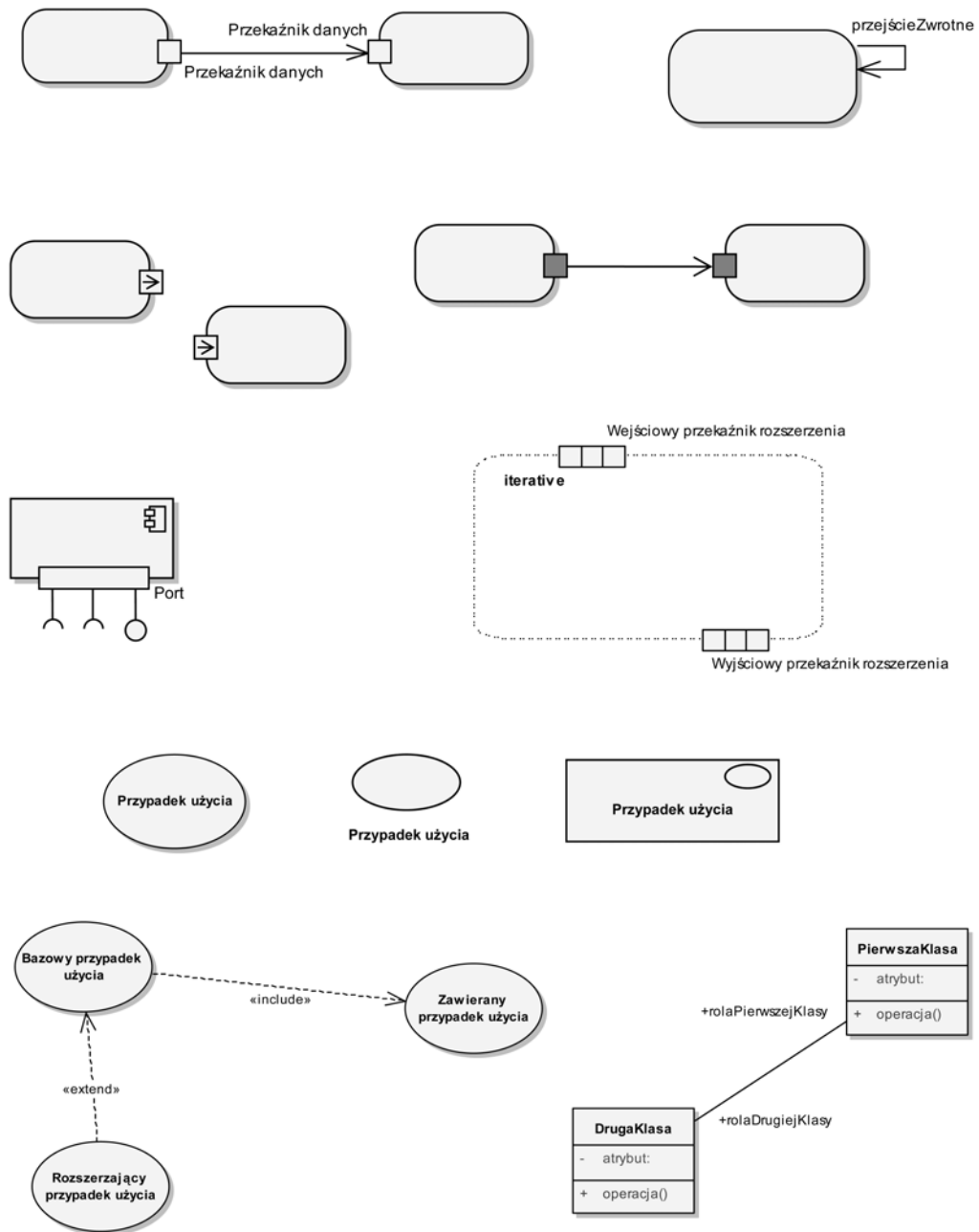


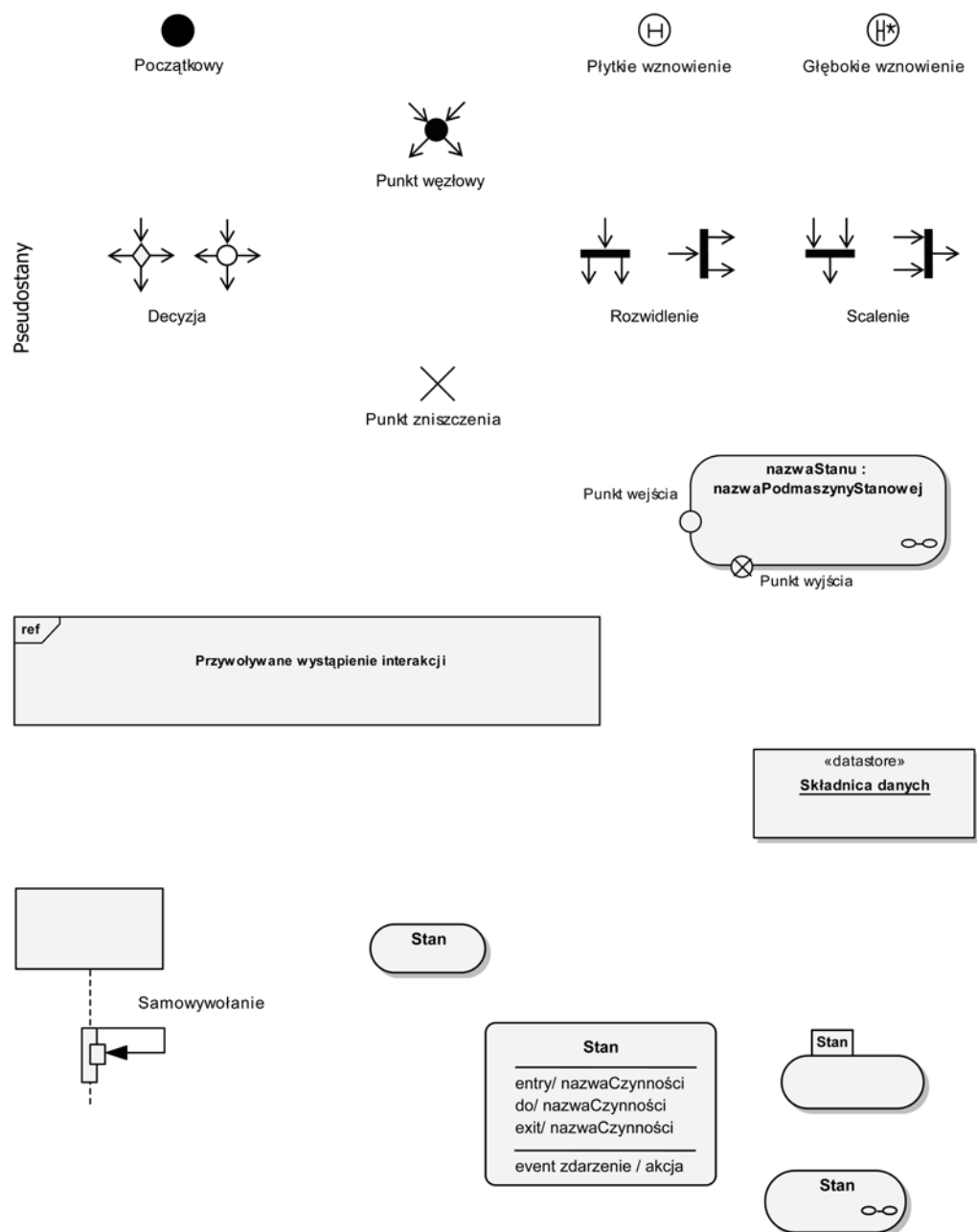


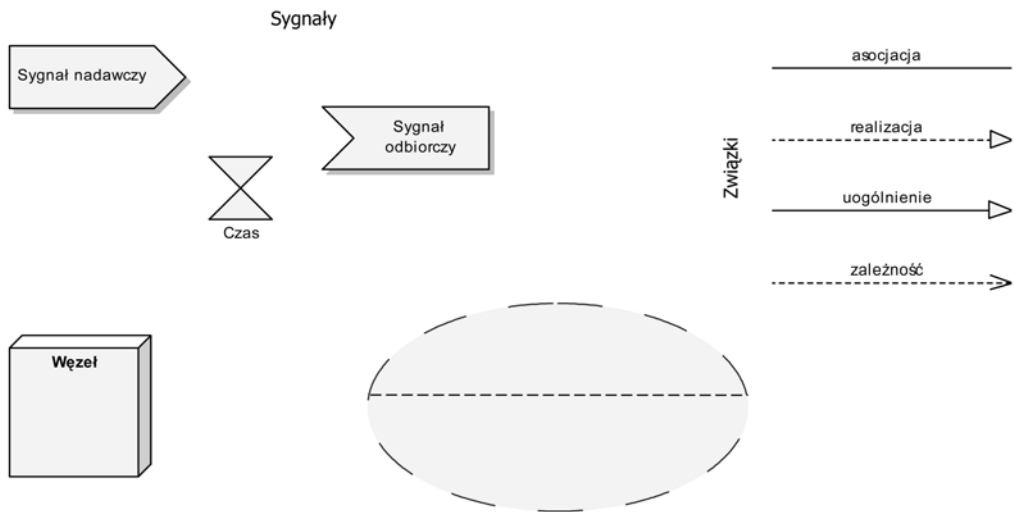








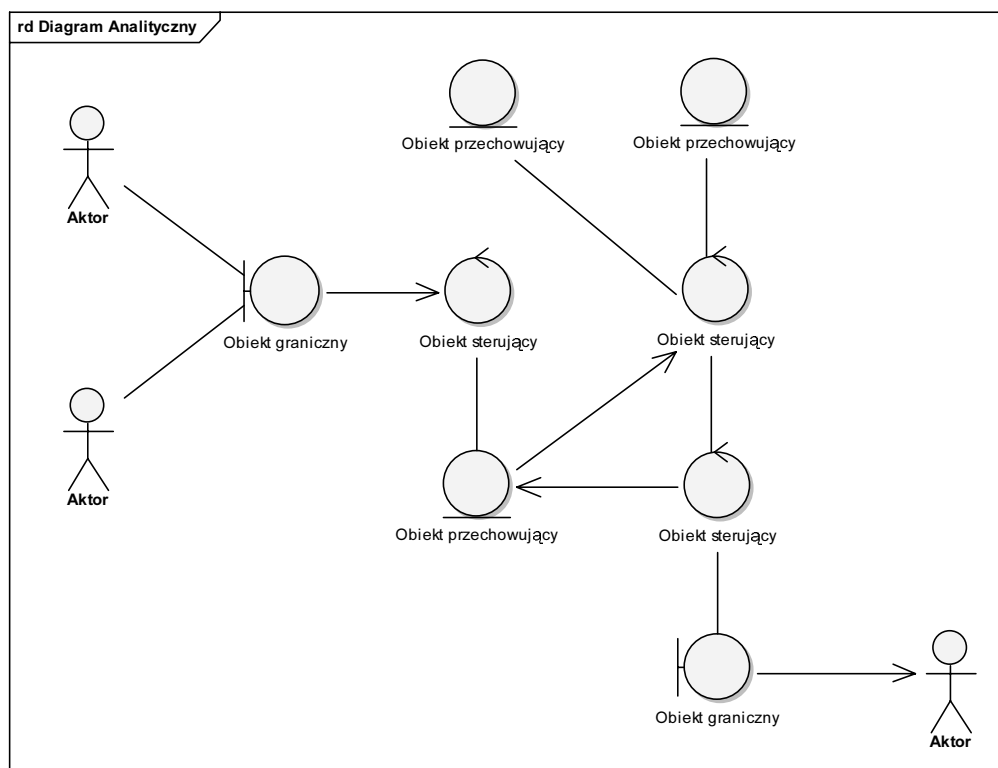


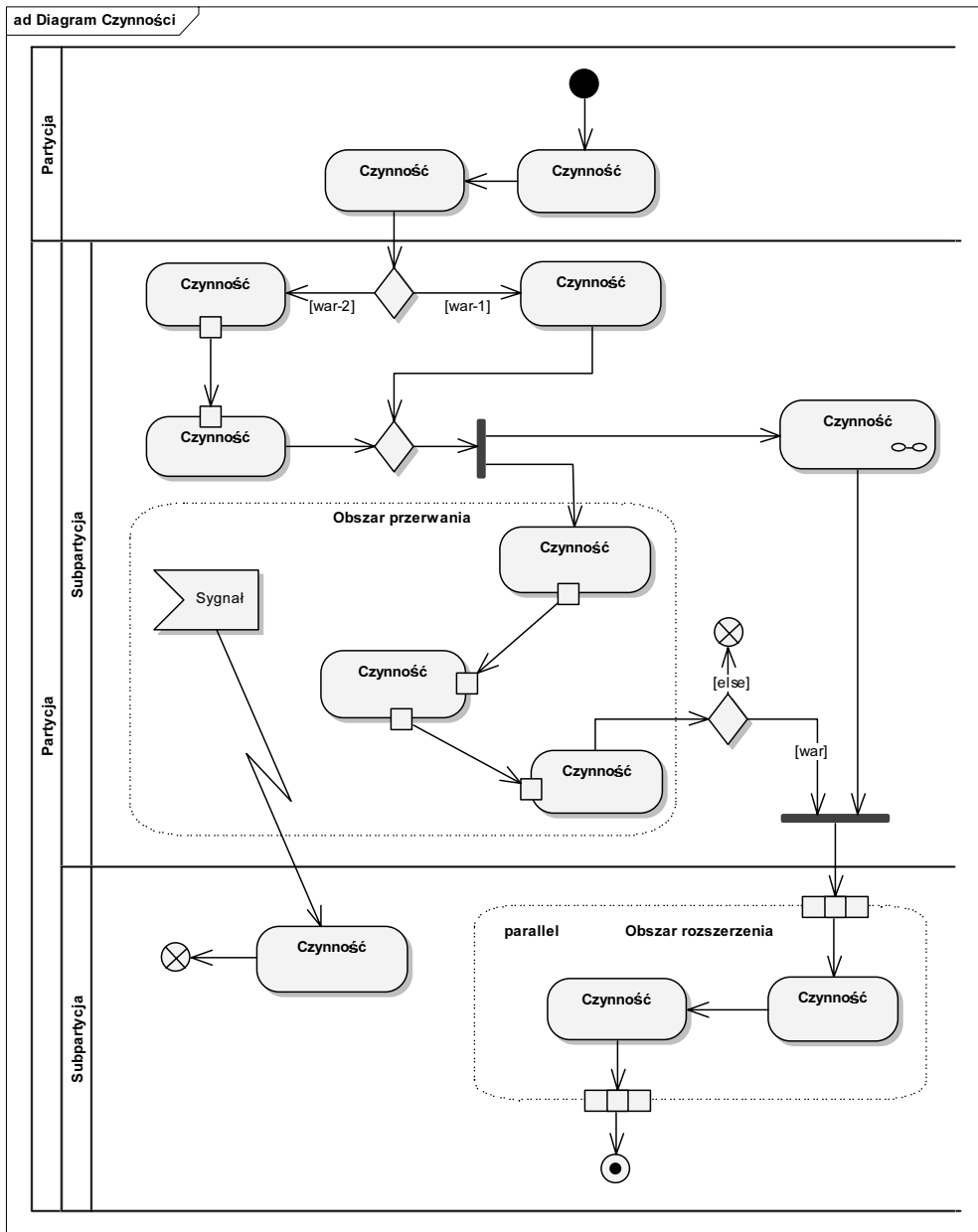


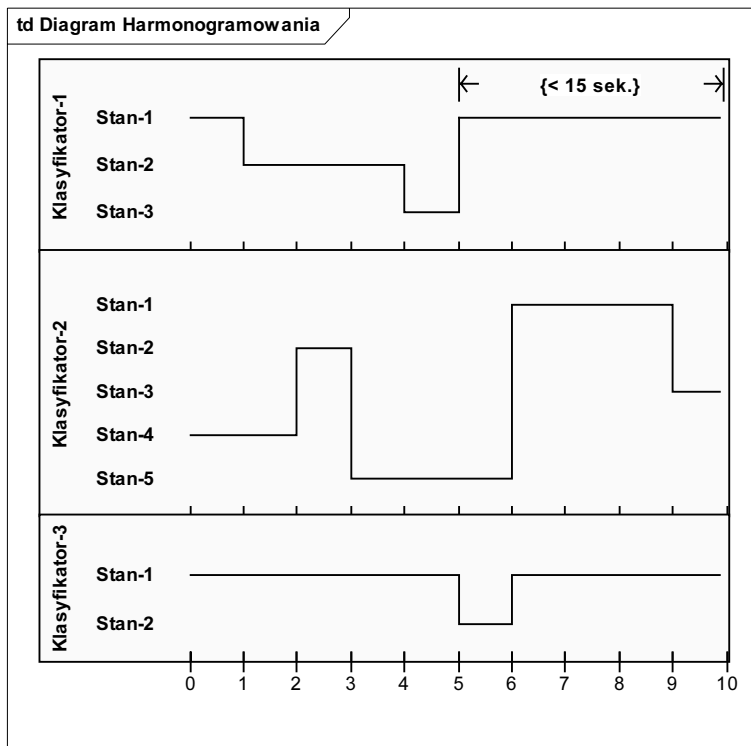
Dodatek C

Diagramy

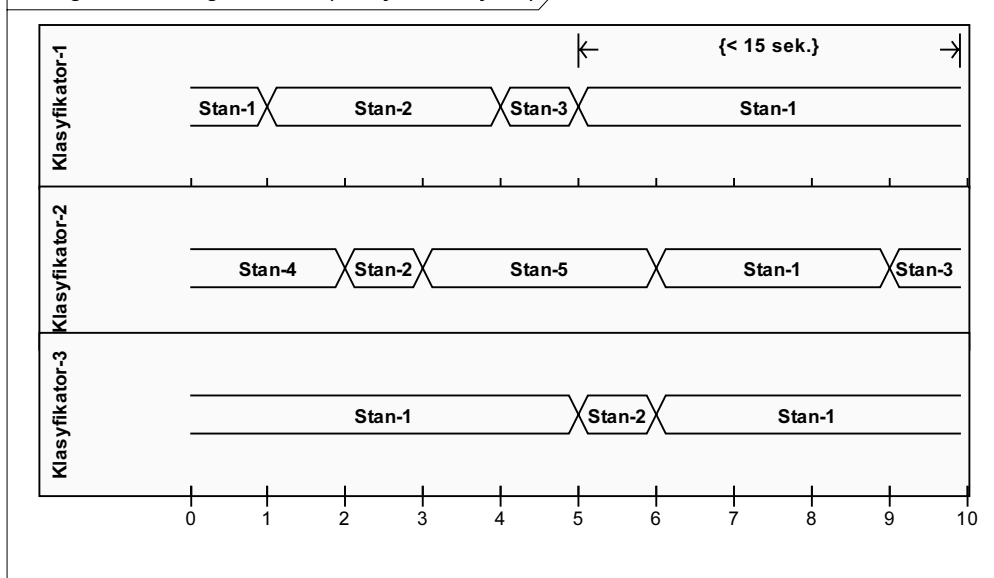
Dodatek C przedstawia podstawowe reguły łączenia poszczególnych kategorii modelowania wszystkich trzynastu diagramów UML 2, jak również diagramów analitycznych uzupełniających standard UML. Indywidualne kategorie modelowania zilustrowano w dodatku B.

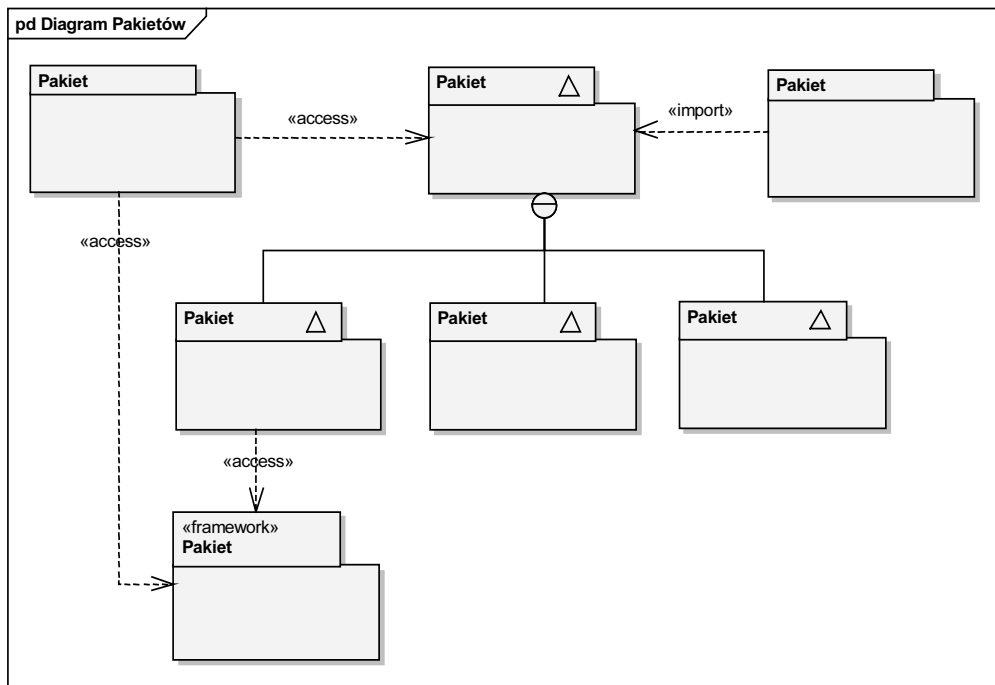
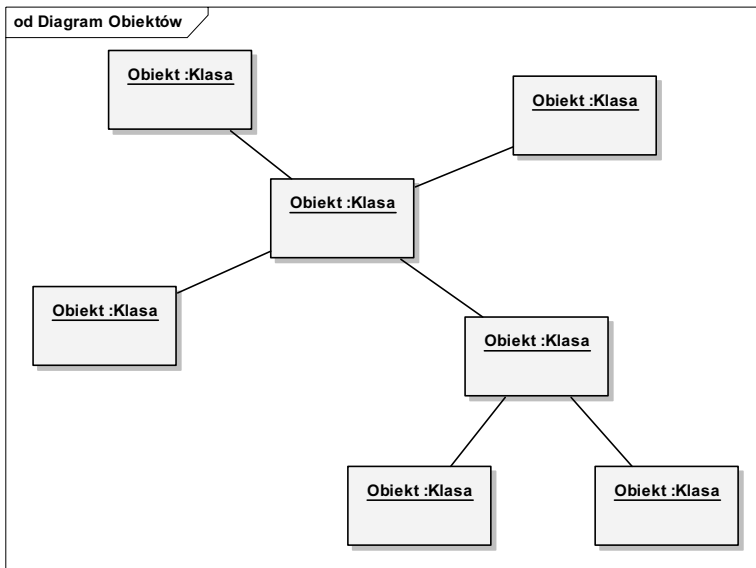


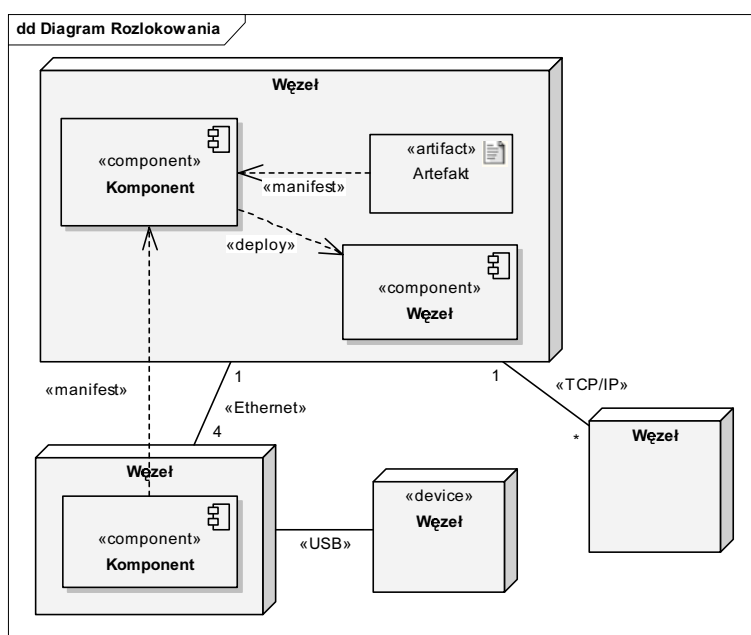
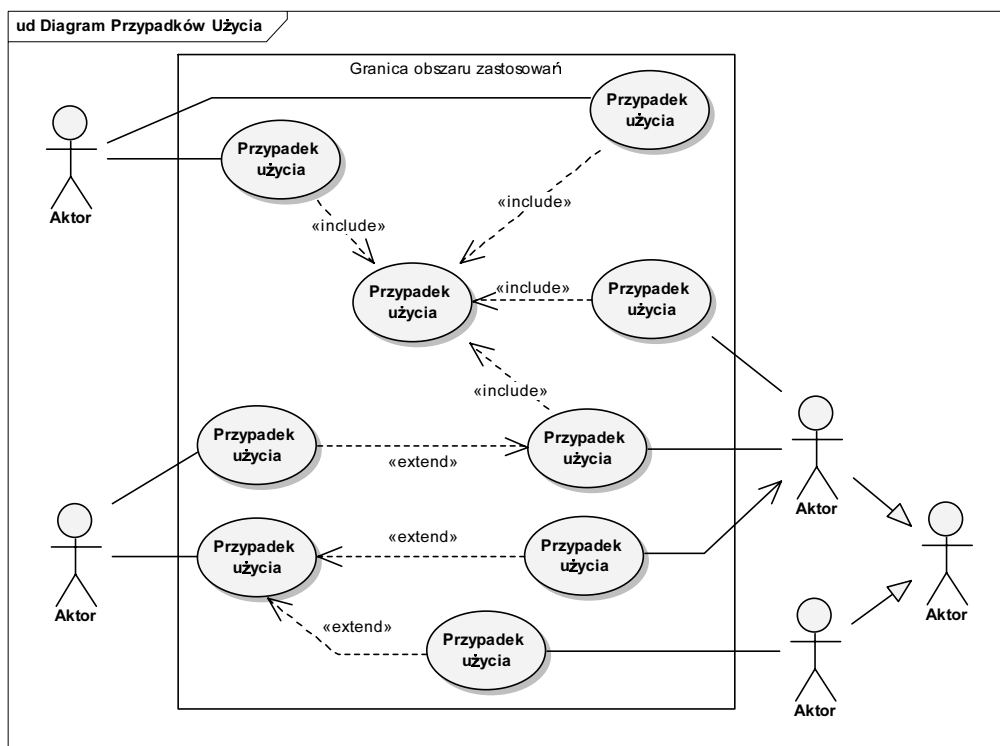


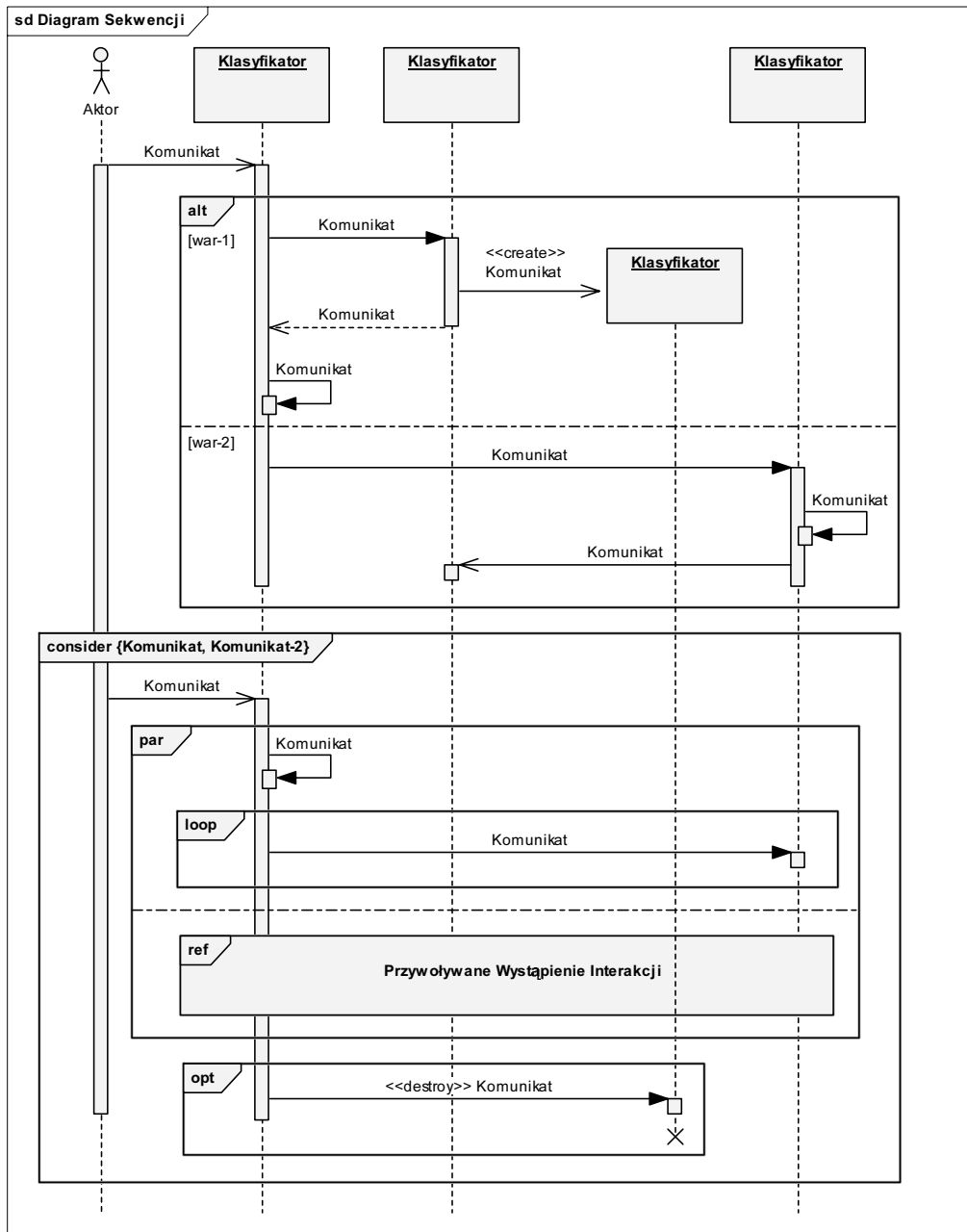


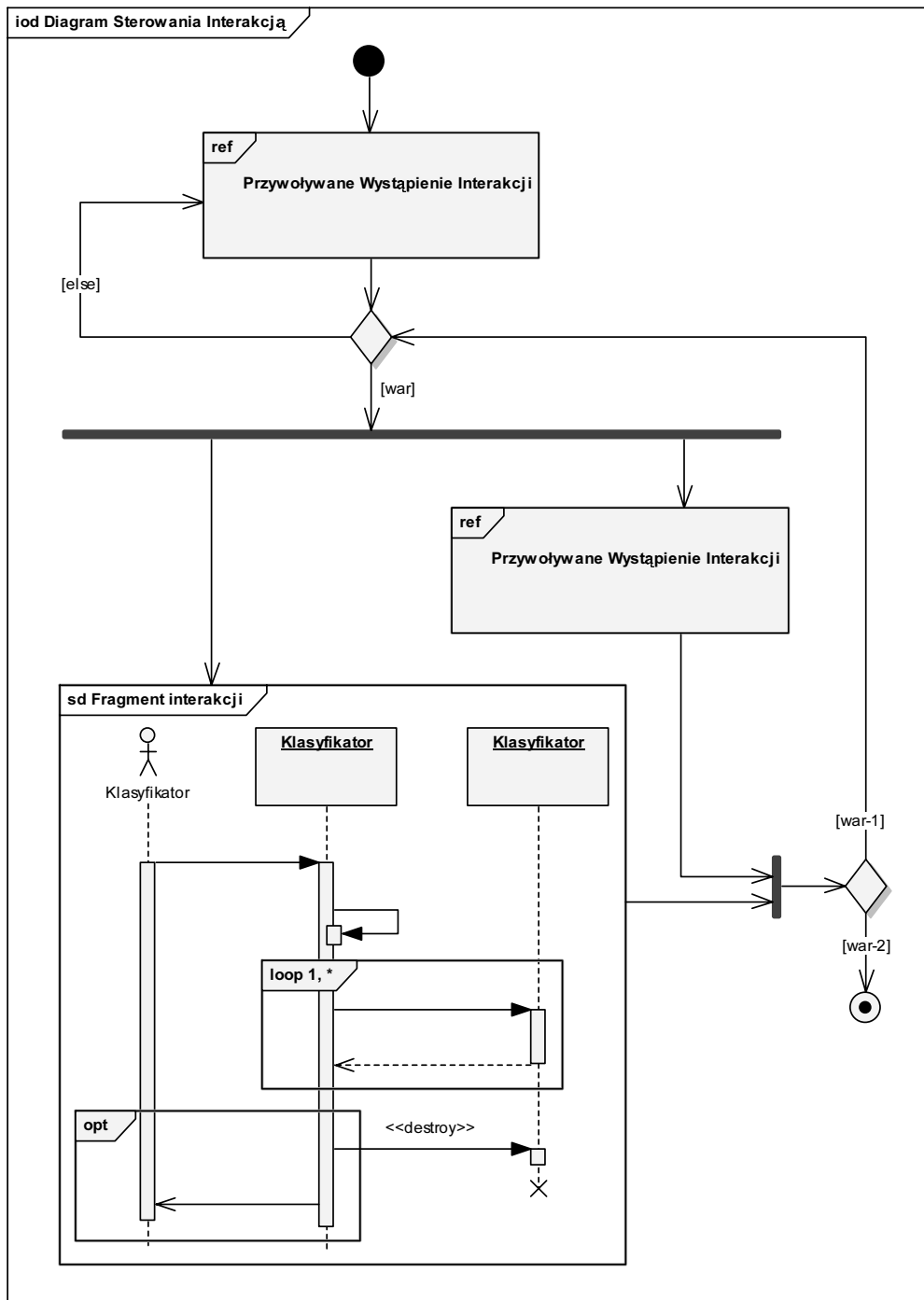
td Diagram Harmonogramowania (notacja alternatywna)

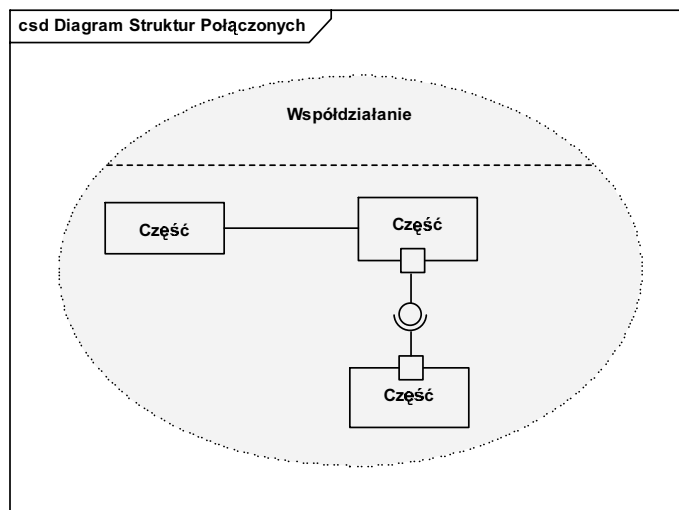
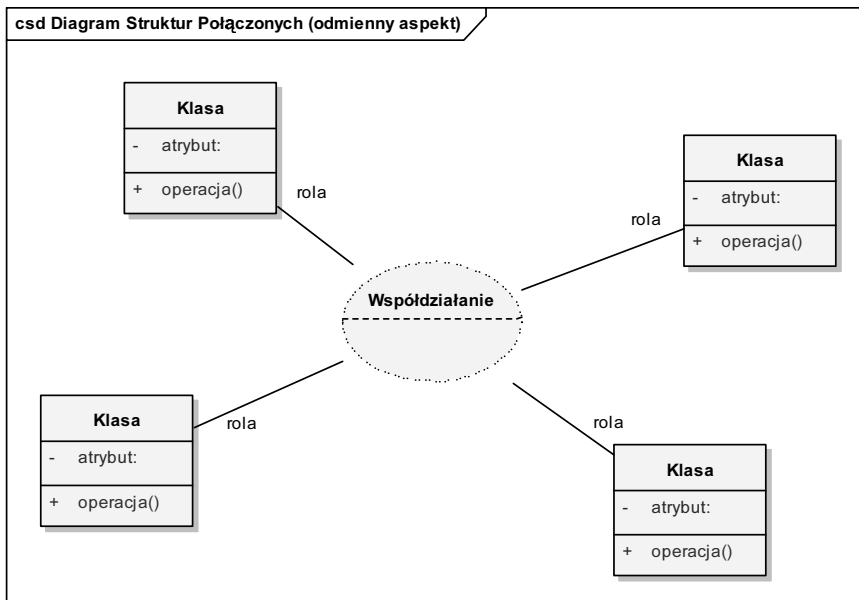








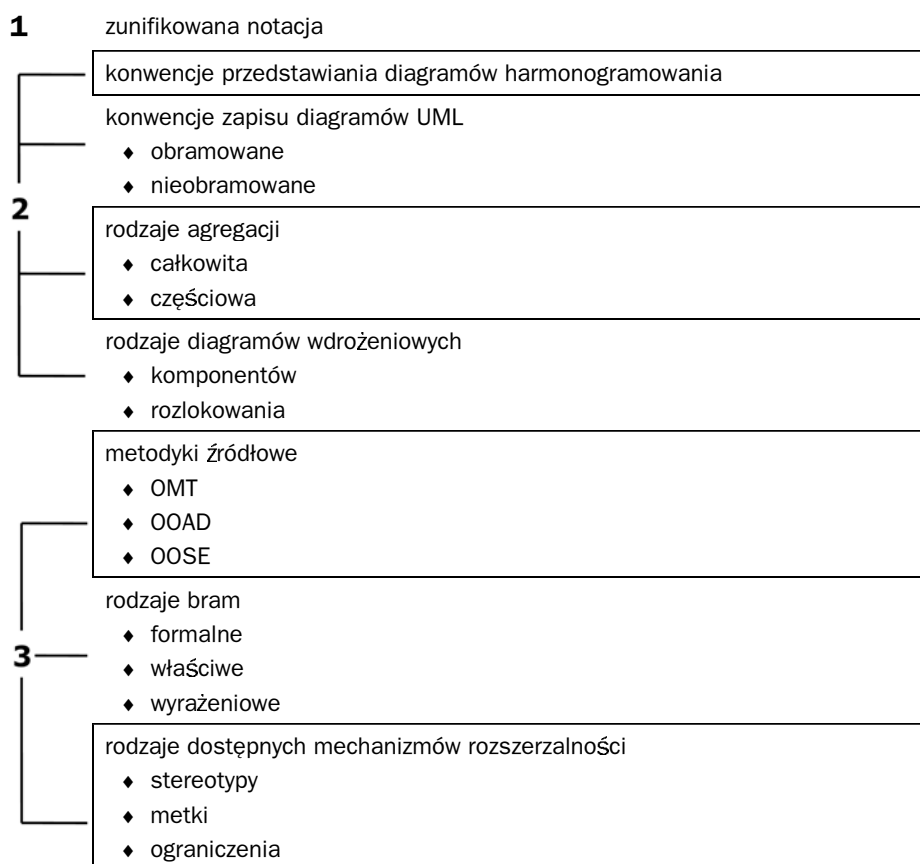




Dodatek D

UML 2.0 w liczbach

Aktualna wersja języka UML charakteryzuje się znaczną liczbą kategorii pojęciowych i rozmaitych koncepcji. Zależności liczbowe pomiędzy nimi są niezliczone. Najciekawsze z nich, stanowiące ilościowe podsumowanie języka UML, przedstawiono w niniejszym dodatku.



	odrębne specyfikacje stanowiące pełny standard
	♦ infrastruktura UML ♦ superstruktura UML ♦ OCL ♦ wymiana diagramów
4	poziomy widoczności ♦ publiczny ♦ prywatny ♦ chroniony ♦ pakietowy rodzaje diagramów interakcji ♦ sekwencji ♦ komunikacji ♦ harmonogramowania ♦ sterowania interakcją rodzaje związków ♦ asocjacja ♦ uogólnienie ♦ zależność ♦ realizacja
5	perspektyw w opisie architektury systemu ♦ przypadków użycia ♦ logiczna ♦ dynamiczna ♦ komponentów ♦ rozlokowania
7	typów komunikatów na diagramach interakcji
8	rodzajów przejść na diagramach maszyny stanowej
9	istotniejszych rewizji języka UML
10	wyspecyfikowanych pseudostanów
12	różnych rodzajów operatorów interakcji
13	diagramów, począwszy od wersji UML 2.0
34	różne stereotypy standardowe zdefiniowane w dokumentacji UML 2.0
624	strony składające się na pierwszą oficjalnie opublikowaną dokumentację superstruktury UML 2.0
1994	rok zapoczątkowania prac nad językiem UML
2003	rok wprowadzenia wersji 2.0 standardu

Dodatek E

Słownik polsko-angielski

W przypadku zjawisk i technologii podlegających nieustannemu rozwojowi, a zarazem będących przedmiotem nieustannego zainteresowania rozmaitych środowisk, zachowanie spójności terminologicznej pomiędzy poszczególnymi opracowaniami w różnych językach staje się problematyczne. Wiele pojęć związanych z językiem UML wraz z jego rozwojem straciło adekwatność; pojawiły się także nowe, autorskie propozycje dotyczące niektórych z nich. Poniżej zamieszczono uznane przez autorów za najtrafniejsze tłumaczenia polskich terminów na ich angielskie odpowiedniki.

Agregacja całkowita	Composition
Agregacja częściowa	Aggregation
Akcja	Action
Aktor	Actor
Aktor biznesowy	Business actor
Analiza i projektowanie	Analysis and design
Artefakt	Artifact
Asocjacja	Association
Asocjacja binarna	Binary association
Asocjacja n-arna	N-ary association
Atrybut	Attribute
Atrybut pochodny	Derived attribute
Biała skrzynka	White box
Biblioteka	Library
Biznesowy obiekt przechowujący	Business entity
Biznesowy przypadek użycia	Business use case
Brama	Gate

Brama formalna	Formal gate
Brama właściwa	Actual gate
Brama wyrażeniowa	Expression gate
Budowa	Construction
Bufor centralny	Central buffer node
Chroniony	Protected
Cykl życia systemu	System lifecycle
Czarna skrzynka	Black box
Część	Part
Czynność	Activity
Czynność biznesowa	Business activity
Decyzja	Decision node
Dekompozycja	Decomposition
Diagram analityczny	Robustness diagram
Diagram czynności	Activity diagram
Diagram harmonogramowania	Timing diagram
Diagram implementacyjny	Generic diagram
Diagram klas	Class diagram
Diagram komponentów	Component diagram
Diagram komunikacji	Communication diagram
Diagram koncepcyjny	Conceptual diagram
Diagram maszyny stanowej	State machine diagram
Diagram o niezdefiniowanej formie	Freeform diagram
Diagram obiektów	Object diagram
Diagram pakietów	Package diagram
Diagram przypadków użycia	Use case diagram
Diagram rozlokowania	Deployment diagram
Diagram sekwencji	Sequence diagram
Diagram sterowania interakcją	Interaction overview diagram
Diagram struktur połączonych	Composite structure diagram
Diagram wystąpieniowy	Instance diagram
Diagramy interakcji	Interaction diagrams
Diagramy wdrożeniowe	Implementation diagrams
Dyscyplina	Discipline

Dyscyplina podstawowa	Core engineering workflow
Dyscyplina wspomagająca	Core supporting workflow
Dyskryminator	Generalization set
Dziedziczenie	Inheritance
Faza	Phase
Fragment wyodrębniony	Combined fragment
Głębokie wznowienie	Deep history
Hermetyzacja	Encapsulation
Instancja	Instance
Interakcja	Interaction
Interfejs	Interface
Interfejs pozyskujący	Required interface
Interfejs udostępniający	Provided interface
Inżynieria wahadłowa	Round-trip engineering
Inżynieria zwrotna	Reverse engineering
Iteracja	Iteration
Jednostka organizacyjna	Organizational unit
Język opisu ograniczeń	Object Constraint Language
Karta CRC	Class-Responsibility-Collaboration (CRC) card
Kierunek nawigacji	Navigability
Kierunek wstępujący	Bottom-up direction
Kierunek zstępujący	Top-down direction
Klasa	Class
Klasa abstrakcyjna	Abstract class
Klasa aktywna	Active class
Klasa asocjacyjna	Association class
Klasa graniczna	Boundary class
Klasa przechowująca	Entity class
Klasa sterująca	Control class
Klasyfikator	Classifier
Klasyfikator ustrukturyzowany	Structured classifier
Komórka partycji	Cell
Komponent	Component

Komputerowo wspomagane tworzenie systemów informatycznych	Computer-Aided Software Engineering
Komunikat	Message
Komunikat asynchroniczny	Asynchronous message
Komunikat oczekujący	Timeout message
Komunikat opcjonalny	Balking message
Komunikat synchroniczny	Synchronous message
Komunikat utracony	Lost message
Komunikat znaleziony	Found message
Komunikat zwrotny	Return
Konektor	Connector
Konektor delegowany	Delegation connector
Konektor składany	Assembly connector
Koniec	Activity final
Kryterium partycjonowania	Dimension
Kwalifikator	Qualifier
Liczebność	Multiplicity
Linia zmiany stanów	Timeline
Linia życia	Object lifeline
Łącznik	Activity edge connector
Manipulator wyjątków	Exception handler
Maszyna stanowa	State machine
Maszyna stanowa zachowania	Behavioral state machine
Mechanizmy rozszerzalności	Extension mechanisms
Metka	Tagged value
Metodyka	Methodology
Metodyka rodzajowa	Generic methodology
Metodyka zwinna (żwawa)	Agile methodology
Miejsce rozszerzania	Extension point
Modelowanie biznesowe	Business modeling
Moduł dodatkowy	Plug-in
Nadawca	Sender
Numerowanie komunikatów	Message numbering
Obiekt	Object

Obiekt wielokrotny	Multiple object
Obszar rozszerzenia	Expansion region
Obszar współbieżny	Orthogonal region
Odbiorca	Receiver
Ograniczenie	Constraint
Operacja	Operation
Operacje wprowadzania, wyszukiwania, aktualizacji i skreślenia	CRUD operations
Operand interakcji	Interaction operand
Operator interakcji	Interaction operator
Opracowanie	Elaboration
Ośrodek sterowania	Execution specification
Pakiet	Package
Parametr czynności	Activity parameter node
Partycja diagramu czynności	Activity partition
Partycja zewnętrzna	External activity partition
Perspektywa	View
Perspektywa dynamiczna	Dynamic view
Perspektywa komponentów	Implementation view
Perspektywa logiczna	Logical view
Perspektywa przypadków użycia	Use case view
Perspektywa rozlokowania	Deployment view
Platforma użytkowania systemu	Execution environment
Płytkie wznowienie	Shallow history
Początek	Initial node
Podczynność	Subactivity
Podmaszyna stanowa	Submachine state
Podstan	Substate
Podsystem	Subsystem
Polimorfizm	Polymorphism
Połączenie	Connection
Poprzednik	Predecessor
Port	Port
Port złożony	Complex port

Pracownik biznesowy	Business worker
Pracownik kontaktu	Case worker
Proces systemowy	System process
Program wykonywalny	Executable
Programowanie	Implementation
Protokołowa maszyna stanowa	Protocol state machine
Prywatny	Private
Przejście	Transition
Przejście wewnętrzne	Internal transition
Przejście zwrotne	Self-transition
Przełącznik danych	Pin
Przełącznik rozszerzenia	Expansion node
Przepływ danych	Data flow
Przepływ przerwania	Interrupting edge
Przepływ sterowania	Control flow
Przygotowanie środowiska	Environment
Przypadek rozszerzany	Extended use case
Przypadek użycia	Use case
Przypadek zawierający	Including use case
Przywoływane wystąpienie interakcji	Interaction occurrence
Pseudostan	Pseudostate
Publiczny	Public
Punkt przeglądu	Milestone
Punkt wejścia	Entry point
Punkt węzłowy	Junction
Punkt wyjścia	Exit point
Rama	Frame
Realizacja	Realization
Reengineering procesów biznesowych	Business Process Reengineering
Rekurencja	Recurrence
Rola	Role
Rozgałęzienie	Branch
Rozłokowanie	Deployment

Rozpoczęcie	Inception
Rozwidlenie	Fork node
Samowywołanie	Message to self
Scalenie	Join node
Scenariusz	Use case description
Sekcja	Compartment
Sekcja czynności wewnętrznych	Internal activities compartment
Sekcja dekompozycji	Decomposition compartment
Sekcja nazwy	Name compartment
Sekcja przejść wewnętrznych	Internal transitions compartment
Składnica danych	Data store node
Specyfikacja rozlokowania	Deployment specification
Specyfikacja scalenia	Join specification
Specyfikacja wymagań	Requirements
Stan	State
Stan końcowy	Final state
Stan początkowy	Initial
Stan prosty	Simple state
Stan złożony	Composite state
Stereotyp	Stereotype
Subpartycja diagramu czynności	Activity subpartition
Sygnal	Signal
Sygnatura komunikatu	Message signature
Ścieżka komunikowania	Communication path
Testowanie	Test
Tor	Swimlane
Udziałowiec	Stakeholder
Ujednolicony język modelowania	Unified Modeling Language
Uogólnienie	Generalization
Waga	Weight
Warunek	Guard condition
Warunki wstępne	Preconditions
Wątek	Thread
Węzeł	Node

Węzeł sprzętowy	Device
Widoczność	Visibility
Współbieżność	Concurrency
Współdziałanie	Cooperation
Współdziałanie biznesowe	Business use case realization
Wyrażenie sekwencji	Sequence expression
Zagnieżdżenie	Nesting
Zakończenie przepływu	Flow final
Zależność	Dependency
Zależność rozszerzania	«extend» dependency
Zależność zawierania	«include» dependency
Zarządzanie konfiguracjami i zmianami	Configuration and change management
Zarządzanie projektem	Project management
Zdarzenie	Event
Zdarzenie czasowe	Time event
Zdarzenie niszczące	Destruction event
Zdarzenie odroczone	Deferred event
Zdarzenie wywołania	Call event
Zdarzenie zmiany stanu	Change event
Zestaw przekazników danych	Parameter set
Zintegrowane środowisko programistyczne	Integrated Development Environment
Złączenie	Merge node
Znacznik sterowania	Token
Zobowiązanie	Responsibility
Związek	Relationship

Dodatek F

Słownik angielsko-polski

Podczas korzystania ze wszelkich źródeł anglojęzycznych — w szczególności z oficjalnej dokumentacji języka UML — niezwykle przydatny staje się słownik w układzie angielsko-polskim, analogiczny do zamieszczonego w dodatku E. W przypadku istnienia kilku powszechnie stosowanych określeń dotyczących jednego terminu uwzględniono je obok sugerowanego tłumaczenia na język polski.

«extend» dependency	Zależność rozszerzania
«include» dependency	Zależność zawierania
Abstract class	Klasa abstrakcyjna
Action	Akcja
Active class	Klasa aktywna
Activity	Czynność
Activity diagram	Diagram czynności
Activity edge connector	Łącznik
Activity final	Koniec
Activity parameter node	Parametr czynności
Activity partition	Partycja diagramu czynności
Activity subpartition	Subpartycja diagramu czynności
Actor	Aktor
Actual gate	Brama właściwa
Aggregation	Agregacja częściowa
Agile methodology	Metodyka zwinna (żwawa)
Analysis and design	Analiza i projektowanie
Artifact	Artefakt
Assembly connector	Konektor składany

Association	Asocjacja
Association class	Klasa asocjacyjna
Asynchronous message	Komunikat asynchroniczny
Attribute	Atrybut
Balking message	Komunikat opcjonalny
Behavioral state machine	Maszyna stanowa zachowania
Binary association	Asocjacja binarna
Black box	Czarna skrzynka
Bottom-up direction	Kierunek wstępujący
Boundary class	Klasa graniczna
Branch	Rozgałęzienie
Business activity	Czynność biznesowa
Business actor	Aktor biznesowy
Business entity	Biznesowy obiekt przechowujący
Business modeling	Modelowanie biznesowe
Business Process Reengineering	Reengineering procesów biznesowych
Business use case	Biznesowy przypadek użycia
Business use case realization	Współdziałanie biznesowe
Business worker	Pracownik biznesowy
Call event	Zdarzenie wywołania
Case worker	Pracownik kontaktu
Cell	Komórka partycji
Central buffer node	Bufor centralny
Change event	Zdarzenie zmiany stanu
Class	Klasa
Class diagram	Diagram klas
Classifier	Klasyfikator
Class-Responsibility-Collaboration (CRC) card	Karta CRC
Combined fragment	Fragment wyodrębniony
Communication diagram	Diagram komunikacji
Communication path	Ścieżka komunikowania
Compartment	Sekcja
Complex port	Port złożony

Component	Komponent
Component diagram	Diagram komponentów
Composite state	Stan złożony
Composite structure diagram	Diagram struktur połączonych
Composition	Agregacja całkowita
Computer-Aided Software Engineering	Komputerowo wspomagane tworzenie systemów informatycznych
Conceptual diagram	Diagram konceptualny
Concurrency	Współbieżność
Configuration and change management	Zarządzanie konfiguracjami i zmianami
Connection	Połączenie
Connector	Konektor
Constraint	Ograniczenie
Construction	Budowa
Control class	Klasa sterująca
Control flow	Przepływ sterowania
Cooperation	Współdziałanie
Core engineering workflow	Dyscyplina podstawowa
Core supporting workflow	Dyscyplina wspomagająca
CRUD operations	Operacje wprowadzania, wyszukiwania, aktualizacji i skreślenia
Data flow	Przepływ danych
Data store node	Składnica danych
Decision node	Decyzja
Decomposition	Dekompozycja
Decomposition compartment	Sekcja dekompozycji
Deep history	Głębokie wznowienie
Deferred event	Zdarzenie odroczone
Delegation connector	Konektor delegowany
Dependency	Zależność
Deployment	Rozlokowanie
Deployment diagram	Diagram rozlokowania
Deployment specification	Specyfikacja rozlokowania
Deployment view	Perspektywa rozlokowania

Derived attribute	Atrybut pochodny
Destruction event	Zdarzenie niszczące
Device	Węzeł sprzętowy
Dimension	Kryterium partycjonowania
Discipline	Dyscyplina
Dynamic view	Perspektywa dynamiczna
Elaboration	Opracowanie
Encapsulation	Hermetyzacja
Entity class	Klasa przechowująca
Entry point	Punkt wejścia
Environment	Przygotowanie środowiska
Event	Zdarzenie
Exception handler	Manipulator wyjątków
Executable	Program wykonywalny
Execution environment	Platforma użytkownika systemu
Execution specification	Ośrodek sterowania
Exit point	Punkt wyjścia
Expansion node	Przełącznik rozszerzenia
Expansion region	Obszar rozszerzenia
Expression gate	Brama wyrażeniowa
Extended use case	Przypadek rozszerzany
Extension mechanisms	Mechanizmy rozszerzalności
Extension point	Miejsce rozszerzania
External activity partition	Partycja zewnętrzna
Final state	Stan końcowy
Flow final	Zakończenie przepływu
Fork node	Rozwidlenie
Formal gate	Brama formalna
Found message	Komunikat znaleziony
Frame	Rama
Freeform diagram	Diagram o niezdefiniowanej formie
Gate	Brama
Generalization	Uogólnienie
Generalization set	Dyskryminator

Generic diagram	Diagram implementacyjny
Generic methodology	Metodyka rodzajowa
Guard condition	Warunek
Implementation	Programowanie
Implementation diagrams	Diagramy wdrożeniowe
Implementation view	Perspektywa komponentów
Inception	Rozpoczęcie
Including use case	Przypadek zawierający
Inheritance	Dziedziczenie
Initial	Stan początkowy
Initial node	Początek
Instance	Instancja
Instance diagram	Diagram wystąpieniowy
Integrated Development Environment	Zintegrowane środowisko programistyczne
Interaction	Interakcja
Interaction diagrams	Diagramy interakcji
Interaction occurrence	Przywoływane wystąpienie interakcji
Interaction operand	Operand interakcji
Interaction operator	Operator interakcji
Interaction overview diagram	Diagram sterowania interakcją
Interface	Interfejs
Internal activities compartment	Sekcja czynności wewnętrznych
Internal transition	Przejsie wewnętrzne
Internal transitions compartment	Sekcja przejść wewnętrznych
Interrupting edge	Przepływ przerwania
Iteration	Iteracja
Join node	Scalenie
Join specification	Specyfikacja scalenia
Junction	Punkt węzłowy
Library	Biblioteka
Logical view	Perspektywa logiczna
Lost message	Komunikat utracony
Merge node	Złączenie

Message	Komunikat
Message numbering	Numerowanie komunikatów
Message signature	Sygnatura komunikatu
Message to self	Samowywoływanie
Methodology	Metodyka
Milestone	Punkt przeglądu
Multiple object	Obiekt wielokrotny
Multiplicity	Liczebność
Name compartment	Sekcja nazwy
N-ary association	Asocjacja n-arna
Navigability	Kierunek nawigacji
Nesting	Zagnieżdżenie
Node	Węzeł
Object	Obiekt
Object Constraint Language	Język opisu ograniczeń
Object diagram	Diagram obiektów
Object lifeline	Linia życia
Operation	Operacja
Organizational unit	Jednostka organizacyjna
Orthogonal region	Obszar współbieżny
Package	Pakiet
Package diagram	Diagram pakietów
Parameter set	Zestaw przekaźników danych
Part	Część
Phase	Faza
Pin	Przekaźnik danych
Plug-in	Moduł dodatkowy
Polymorphism	Polimorfizm
Port	Port
Preconditions	Warunki wstępne
Predecessor	Poprzednik
Private	Prywatny
Project management	Zarządzanie projektem
Protected	Chroniony

Protocol state machine	Protokołowa maszyna stanowa
Provided interface	Interfejs udostępniający
Pseudostate	Pseudostan
Public	Publiczny
Qualifier	Kwalifikator
Realization	Realizacja
Receiver	Odbiorca
Recurrence	Rekurencja
Relationship	Związek
Required interface	Interfejs pozyskujący
Requirements	Specyfikacja wymagań
Responsibility	Zobowiązanie
Return	Komunikat zwrotny
Reverse engineering	Inżynieria zwrotna
Robustness diagram	Diagram analityczny
Role	Rola
Round-trip engineering	Inżynieria wahadłowa
Self-transition	Przejście zwrotne
Sender	Nadawca
Sequence diagram	Diagram sekwencji
Sequence expression	Wyrażenie sekwencji
Shallow history	Płytkie wznowienie
Signal	Sygnał
Simple state	Stan prosty
Stakeholder	Udziałowiec
State	Stan
State machine	Maszyna stanowa
State machine diagram	Diagram maszyny stanowej
Stereotype	Stereotyp
Structured classifier	Klasyfikator ustrukturyzowany
Subactivity	Podczynność
Submachine state	Podmaszyna stanowa
Substate	Podstan
Subsystem	Podsystem

Swimlane	Tor
Synchronous message	Komunikat synchroniczny
System lifecycle	Cykl życia systemu
System process	Proces systemowy
Tagged value	Metka
Test	Testowanie
Thread	Wątek
Time event	Zdarzenie czasowe
Timeline	Linia zmiany stanów
Timeout message	Komunikat oczekujący
Timing diagram	Diagram harmonogramowania
Token	Znacznik sterowania
Top-down direction	Kierunek zstępujący
Transition	Przejście
Unified Modeling Language	Ujednolicony język modelowania
Use case	Przypadek użycia
Use case description	Scenariusz
Use case diagram	Diagram przypadków użycia
Use case view	Perspektywa przypadków użycia
View	Perspektywa
Visibility	Widoczność
Weight	Waga
White box	Biała skrzynka

Dodatek G

Spis rysunków

Dodatek G zawiera kompletne zestawienie rysunków zamieszczonych w niniejszej publikacji w kolejności ich pojawiania się. Ze względu na znaczną obszerność materiału spis rysunków zorganizowano względem poszczególnych rozdziałów książki.

Rozdział 1. Język UML — rozwój, struktura, pojęcia	15
Rysunek 1.1. Ewolucja języka UML	19
Rysunek 1.2. Diagramy w języku UML 2.0	22
Rysunek 1.3. <i>Format diagramu obramowanego</i>	25
Rysunek 1.4. Modelowanie perspektyw architektury systemu	26
Rozdział 2. Diagram przypadków użycia	33
Rysunek 2.1. Oznaczenia przypadków użycia	35
Rysunek 2.2. Alternatywne notacje przypadków użycia	35
Rysunek 2.3. Podstawowe rodzaje aktorów	36
Rysunek 2.4. Związki asocjacji w diagramach przypadków użycia	37
Rysunek 2.5. Diagram przypadków użycia aukcji internetowej	38
Rysunek 2.6. DPU systemu obsługi hurtowni	39
Rysunek 2.7. Zależność zawierania «include»	41
Rysunek 2.8. Zależność rozszerzania «extend»	41
Rysunek 2.9. Różne rodzaje zależności stereotypowanych	42
Rysunek 2.10. Miejsca rozszerzania w przypadku użycia	43
Rysunek 2.11. Miejsca rozszerzania — notacja alternatywna	43
Rysunek 2.12. Notatki w opisie miejsc rozszerzania	44
Rysunek 2.13. Przykłady uogólnień	44
Rysunek 2.14. Hierarchia dziedziczenia wyrażona poprzez związki uogólnienia	45
Rysunek 2.15. Stereotypy graficzne aktorów	46
Rysunek 2.16. Zastosowanie stereotypów graficznych aktorów	47
Rysunek 2.17. Stereotypy tekstowe aktorów	47
Rysunek 2.18. Liczebność na diagramach przypadków użycia	48
Rysunek 2.19. Nawigacja na diagramach przypadków użycia	48
Rysunek 2.20. Związek realizacji	49
Rysunek 2.21. Nazwa prosta a nazwa ścieżkowa	50
Rysunek 2.22. Diagram kontekstowy	51
Rysunek 2.23. Przypadek użycia i jego scenariusze	52
Rysunek 2.24. System aukcji internetowej	55
Rysunek 2.25. Rezerwacje w systemie hotelowym	58

Rozdział 3. Diagram klas	61
Rysunek 3.1. Przykłady opisu klas	63
Rysunek 3.2. Opcje specyfikacji klas	63
Rysunek 3.3. Asocjacje pomiędzy klasami	64
Rysunek 3.4. Asocjacja nazwana	65
Rysunek 3.5. Asocjacja z wyspecyfikowanymi rolami	65
Rysunek 3.6. Asocjacja nazwana z wyspecyfikowanymi rolami	65
Rysunek 3.7. Asocjacja z kierunkiem nawigacji	66
Rysunek 3.8. Zastosowanie liczebności asocjacji	68
Rysunek 3.9. Agregacja całkowita a agregacja częściowa	69
Rysunek 3.10. Diagram klas z uwzględnieniem agregacji	69
Rysunek 3.11. Sekcja zobowiązań w klasie	71
Rysunek 3.12. Oznaczenie widoczności w klasie	73
Rysunek 3.13. Złożona reprezentacja atrybutów i operacji	76
Rysunek 3.14. Asocjacyjna klasa obiektów	78
Rysunek 3.15. Związek n-arny i klasa asocjacyjna	78
Rysunek 3.16. Asocjacje wielokrotne między klasami	79
Rysunek 3.17. Asocjacja zwrotna	79
Rysunek 3.18. Asocjacja kwalifikowana	80
Rysunek 3.19. Dziedziczenie klas	80
Rysunek 3.20. Ograniczenie typu {complete}	81
Rysunek 3.21. Ograniczenie typu {incomplete}	81
Rysunek 3.22. Ograniczenia typu {disjoint} i {overlapping}	82
Rysunek 3.23. Diagram klas systemu płatności	82
Rysunek 3.24. Konwencje graficznej prezentacji uogólnień	83
Rysunek 3.25. Związek zależności	83
Rysunek 3.26. Związek realizacji i interfejs	84
Rysunek 3.27. Przykład diagramu obiektów	85
Rysunek 3.28. Przykład klas systemu księgarni elektronicznej	86
Rysunek 3.29. Diagram obiektów systemu księgarni elektronicznej	87
Rysunek 3.30. Diagram klas finansowego systemu transakcyjnego	88
Rysunek 3.31. Model konceptualny bazy danych systemu aukcji internetowej	90
Rozdział 4. Diagram czynności	95
Rysunek 4.1. Pojedyncza czynność uszczegółowiona na odrębnym diagramie	96
Rysunek 4.2. Dekompozycja czynności	98
Rysunek 4.3. Logowanie użytkownika do systemu operacyjnego	98
Rysunek 4.4. Funkcjonowanie bankomatu	101
Rysunek 4.5. Rejestracja użytkownika w księgarni internetowej	102
Rysunek 4.6. Specyfikacja czynności „Wprowadź dane osobowe”	103
Rysunek 4.7. Wykorzystanie złączeń na przykładzie wyboru kwoty	105
Rysunek 4.8. Zintegrowanie decyzji i złączenia	106
Rysunek 4.9. Rozliczenie konta odbiorcy energii cieplnej	108
Rysunek 4.10. Specyfikacja scalenia	108
Rysunek 4.11. Zintegrowanie funkcji rozwidlenia i scalenia	109
Rysunek 4.12. Specyfikacja algorytmu — obliczanie średniej	110
Rysunek 4.13. Specyfikacja algorytmu — losowanie pytań	111
Rysunek 4.14. Proces wypożyczania książki	113
Rysunek 4.15. Przeływ obiektu	114
Rysunek 4.16. Przeływ obiektu z wykorzystaniem przekazników danych	114
Rysunek 4.17. Przeływ obiektu — anonimowa notacja przekazników	114
Rysunek 4.18. Zestawy przekazników sterowania w segmentowaniu klientów	116
Rysunek 4.19. Notacja parametrów czynności	116
Rysunek 4.20. Przeprowadzanie licytacji	117

Rysunek 4.21. Zastosowanie wagi na diagramie	118
Rysunek 4.22. Zastosowanie sygnałów w procesie nagrywania płyt DVD	119
Rysunek 4.23. System harmonogramowania zadań	119
Rysunek 4.24. Przepływ spraw w systemie elektronicznego obiegu dokumentów WFM ..	120
Rysunek 4.25. Składnica danych systemu obsługi faksów	121
Rysunek 4.26. Wykorzystanie partycji w opisie obsługi kina	122
Rysunek 4.27. Połączenie poziomych i pionowych partycji	124
Rysunek 4.28. Partycje i subpartycje wewnętrzne oraz zewnętrzne	126
Rysunek 4.29. Alternatywna notacja partycji i subpartycji	127
Rysunek 4.30. Obszar rozszerzenia i przekaźniki rozszerzenia	129
Rysunek 4.31. Uproszczona notacja przekaźników rozszerzenia	129
Rysunek 4.32. Obszar przerwania w systemie aukcji internetowej	131
Rysunek 4.33. Indeks poza zakresem	132
Rysunek 4.34. Przekroczenie poziomu zamówieniowego	132
Rysunek 4.35. Przeprowadzanie rekrutacji na studia	134
Rysunek 4.36. Zamawianie wycieczki	139
Rysunek 4.37. Funkcjonowanie aukcji tradycyjnej	141
Rysunek 4.38. Anulowanie rezerwacji biletu kinowego	142
Rozdział 5. Diagram maszyny stanowej	143
Rysunek 5.1. System elektronicznego egzaminowania — obiekt klasy Student	145
Rysunek 5.2. Sekcje stanu	146
Rysunek 5.3. Czynności wewnętrzne stanu	147
Rysunek 5.4. Przejścia wewnętrzne i zwrotne dla stanu obiektu Projekt	148
Rysunek 5.5. Maszyny i podmaszyny stanowe	149
Rysunek 5.6. Podmaszyny stanowe ukryte i zaprezentowane w sposób jawny	150
Rysunek 5.7. Nazewnictwo podmaszyn stanowych	150
Rysunek 5.8. Obszary współbieżne w maszynie stanowej przetwarzania sprawy	151
Rysunek 5.9. Alternatywne konwencje graficzne pseudostanu decyzji w kwalifikowaniu rachunku bankowego	154
Rysunek 5.10. Wybrane pseudostany w maszynie stanowej systemu elektronicznego egzaminowania	155
Rysunek 5.11. Diagram maszyny stanowej cyklu życia systemu informatycznego	156
Rysunek 5.12. Pseudostany punktów wejścia i wyjścia — alternatywne notacje	159
Rysunek 5.13. Diagram protokołowej maszyny stanowej dla obiektu klasy Egzamin	160
Rysunek 5.14. Kolejowanie projektów jako przykład specyfikacji przejść	162
Rysunek 5.15. Tworzenie kopii zapasowej pliku	164
Rysunek 5.16. Maszyna stanowa systemu aukcji internetowej	166
Rysunek 5.17. Stan złożony obiektu klasy Wypożyczenie	170
Rysunek 5.18. Funkcjonowanie streamera	172
Rysunek 5.19. Stany i sygnały systemu monitorowania pomieszczeń	172
Rozdział 7. Diagram sekwencji	179
Rysunek 7.1. Instancje klasyfikatorów z liniami życia	182
Rysunek 7.2. Przesłanie komunikatu	183
Rysunek 7.3. Sprawdzanie stanu konta za pośrednictwem bankomatu	183
Rysunek 7.4. Kategorie pojęciowe diagramów sekwencji	185
Rysunek 7.5. Rezerwacja hotelowa	186
Rysunek 7.6. Okres aktywności instancji klasyfikatorów	186
Rysunek 7.7. Komunikat synchroniczny	188
Rysunek 7.8. Komunikat asynchroniczny	188
Rysunek 7.9. Komunikat zwrotny	189
Rysunek 7.10. Komunikat utracony	189
Rysunek 7.11. Komunikat znaleziony	189

Rysunek 7.12. Komunikat opcjonalny	190
Rysunek 7.13. Komunikat oczekujący	190
Rysunek 7.14. Tworzenie i niszczenie obiektów	191
Rysunek 7.15. Warunek realizacji komunikatu	191
Rysunek 7.16. Samowywołanie	192
Rysunek 7.17. Iteracja w wewnętrznej rekrutacji pracowników do projektów	193
Rysunek 7.18. Rozgałęzienie	194
Rysunek 7.19. Realizacja kontraktu terminowego	195
Rysunek 7.20. Fragment wyodrębniony i jego elementy	197
Rysunek 7.21. Aktualizacja zamówienia w systemie ERP	201
Rysunek 7.22. Przepływ poczty elektronicznej	202
Rysunek 7.23. Wypełnianie faktury	203
Rysunek 7.24. Wyszukiwanie ofert aukcji	204
Rysunek 7.25. Wczytywanie danych o saldach kont klientów	205
Rysunek 7.26. Ustalanie pozycji śledzonego obiektu	206
Rysunek 7.27. Ustalanie pozycji śledzonego obiektu — skrócona notacja współbieżności	207
Rysunek 7.28. Współpraca systemów CRM, WFM oraz archiwizatora zleceń	208
Rysunek 7.29. Szacowanie optymalnej partii dostaw	209
Rysunek 7.30. Rejestrowanie artykułu na aukcji internetowej — operator consider	210
Rysunek 7.31. Rejestrowanie artykułu na aukcji internetowej — operator ignore	211
Rysunek 7.32. Weryfikacja wystawionych czeków	212
Rysunek 7.33. Kolejność logowania na serwisach internetowych	213
Rysunek 7.34. Funkcjonowanie aukcji internetowej	215
Rysunek 7.35. Wybrane funkcje aukcji internetowej	216
Rysunek 7.36. Bramy w diagramach sekwencji	217
Rysunek 7.37. Funkcje serwera w sieci barów kawowych	217
Rysunek 7.38. Generowanie raportu o kosztach realizowanych projektów	220
Rysunek 7.39. Rejestracja sprzedającego na aukcji internetowej	221
Rysunek 7.40. Realizacja kontraktu terminowego	227
Rysunek 7.41. Oprogramowanie narzędziowe serwera	229
Rysunek 7.42. System nadzoru pomieszczeń kompleksu kinowego	230
Rozdział 8. Diagram komunikacji	231
Rysunek 8.1. Komunikaty w diagramach komunikacji	233
Rysunek 8.2. Proste numerowanie komunikatów	233
Rysunek 8.3. Zasady umieszczania komunikatów	234
Rysunek 8.4. Izomorfizm diagramów sekwencji i komunikacji	236
Rysunek 8.5. Zagnieżdżenie komunikatów w notacji Deweya	238
Rysunek 8.6. Współbieżność oraz poprzednicy	240
Rysunek 8.7. Obiekty wielokrotne na diagramach komunikacji	241
Rysunek 8.8. Obiekty klasy aktywnej w naliczaniu odsetek	242
Rysunek 8.9. Włączanie kupującego do licytacji	243
Rysunek 8.10. Egzaminowanie studenta w systemie elektronicznego egzaminowania ...	246
Rysunek 8.11. Interakcje pomiędzy instancjami klasyfikatorów	248
Rozdział 9. Diagram harmonogramowania	251
Rysunek 9.1. Diagram harmonogramowania dla obiektu klasy Rezerwacja	253
Rysunek 9.2. Diagram harmonogramowania ze zdarzeniami i ograniczeniami czasowymi	254
Rysunek 9.3. Alternatywna notacja diagramów harmonogramowania	254
Rysunek 9.4. Harmonogramowanie rejestrowania rezerwacji hotelowej	255
Rysunek 9.5. Rejestrowanie rezerwacji w bazie danych hotelu — notacja alternatywna	256
Rysunek 9.6. Komunikaty na diagramach harmonogramowania	257
Rysunek 9.7. Ograniczenia czasowe w systemie harmonogramowania spotkań	258

Rozdział 10. Diagram sterowania interakcją	263
Rysunek 10.1. Diagram sterowania interakcją	265
Rysunek 10.2. Diagram sterowania interakcją z procesami współbieżnymi i warunkowymi	267
Rysunek 10.3. Diagram sterowania interakcją — funkcjonowanie aukcji internetowej	269
Rozdział 11. Diagramy wdrożeniowe	273
Rysunek 11.1. Komponenty i ich stereotypy	275
Rysunek 11.2. Konceptualny diagram komponentów systemu rejestracji referatów	276
Rysunek 11.3. Zewnętrzna perspektywa komponentu Place	277
Rysunek 11.4. Specyfikacja interfejsów w postaci stereotypów tekstowych	278
Rysunek 11.5. Interfejsy w postaci instancji klasyfikatorów w systemie HRM	279
Rysunek 11.6. Specyfikacja komponentu w postaci białej skrzynki	280
Rysunek 11.7. Implementacyjny diagram komponentów aukcji internetowej	281
Rysunek 11.8. Diagram rozlokowania systemu obsługi konferencji	286
Rysunek 11.9. Alternatywne sposoby osadzania artefaktów na węzłach	287
Rysunek 11.10. Diagram rozlokowania systemu obsługi konferencji z komponentami	288
Rysunek 11.11. Manifestowanie w platformie e-learningowej	289
Rysunek 11.12. Specyfikacja rozlokowania portalu uniwersyteckiego	289
Rysunek 11.13. Diagram rozlokowania na poziomie fizycznym	290
Rysunek 11.14. Węzły aukcji internetowej	295
Rysunek 11.15. Definiowanie zasad kolejkowania w module systemu WFM	296
Rysunek 11.16. Diagram rozlokowania wydziału KRS oraz centrum CORS	297
Rozdział 12. Diagram struktur połączonych	299
Rysunek 12.1. Diagram struktur połączonych	301
Rysunek 12.2. Współdziałanie z przypisanymi klasyfikatorami systemu	302
Rysunek 12.3. Licytowanie jako wystąpienie współdziałania	302
Rozdział 13. Diagram pakietów	305
Rysunek 13.1. Nazwy pakietów	307
Rysunek 13.2. Pakiety i ich zawartość	307
Rysunek 13.3. Diagram pakietów systemu sterowania zapasami	308
Rysunek 13.4. Zagnieżdżenie pakietów systemu kompleksu kinowego	308
Rysunek 13.5. Alternatywne notacje dokumentowania zagnieżdżeń pakietów	309
Rysunek 13.6. Modele obsługi sieci restauracji jako pakiety stereotypowane	311
Rysunek 13.7. System zarządzania siecią kin	313
Rozdział 14. Metodyka RUP	319
Rysunek 14.1. Ewolucja metodyki RUP	321
Rysunek 14.2. Iteracyjno-przyrostowy cykl życia systemu w metodyce RUP	322
Rysunek 14.3. Układ dyscyplin cyklu życia RUP w iteracji	324
Rozdział 15. Modelowanie biznesowe	329
Rysunek 15.1. Modelowanie biznesowe w procesie tworzenia systemu	330
Rysunek 15.2. Biznesowy kontekst systemu księgarni	333
Rysunek 15.3. Biznesowy diagram przypadków użycia systemu księgarni	334
Rysunek 15.4. Mapa procesów biznesowych funkcjonowania księgarni	334
Rysunek 15.5. Czynności przypadku użycia „Dokonaj zakupu”	336
Rysunek 15.6. Biznesowy diagram sekwencji	337
Rysunek 15.7. Biznesowy diagram klas systemu księgarni	338
Rysunek 15.8. Biznesowy diagram klas z pracownikami biznesowymi	339
Rysunek 15.9. Jednostki organizacyjne księgarni	340
Rysunek 15.10. Zależności pomiędzy działami a pracownikami księgarni	340

Rozdział 16. Modelowanie analityczne	343
Rysunek 16.1. Diagramy modelowania analitycznego w procesie tworzenia systemu ...	344
Rysunek 16.2. Diagram klas analitycznych systemu rozpatrywania podań	347
Rysunek 16.3. Konwersja kategorii biznesowych na kategorie klas analitycznych	348
Rysunek 16.4. Rejestrowanie uczestnika aukcji internetowej	349
Rysunek 16.5. Proces egzaminowania w systemie elektronicznego egzaminowania ...	352
Rozdział 17. Komputerowe wspomaganie modelowania systemu	353
Rysunek 17.1. Interfejs narzędzia Enterprise Architect	360
Rysunek 17.2. Zmiana konwencji diagramu harmonogramowania	361
Rysunek 17.3. Ustalanie właściwości diagramu	362
Rysunek 17.4. Interfejs narzędzia IBM Rational Software Modeler	364
Rysunek 17.5. Wprowadzanie nowego diagramu do modelu	364
Rysunek 17.6. Szczegółowa specyfikacja klasy	365
Rysunek 17.7. Poseidon for UML	367

Dodatek H

Literatura

Źródła drukowane:

- [Ambler-2003] **Ambler S.W.** (2003); *The Elements of UML Style*; Cambridge: University Press.
- [Ambler-2005] **Ambler S.W.** (2005); *The Elements of UML 2.0 Style*; Cambridge: University Press.
- [Binder-2003] **Binder R.V.** (2003); *Testowanie systemów obiektowych*; Warszawa: WNT.
- [Boehm-2004] **Boehm B., Turner R.** (2004); *Balancing Agility and Discipline*; Boston: Addison-Wesley.
- [Boggs-2002] **Boggs W., Boggs M.** (2002); *Mastering UML with Rational Rose*; Alameda: Sybex.
- [Booch-1991] **Booch G.** (1991); *Object-Oriented Analysis and Design*; Redwood: Benjamin/Cummings.
- [Booch-1998] **Booch G., Jacobson I., Rumbaugh J.** (1998); *The Unified Software Development Process*; Boston: Addison-Wesley.
- [Booch-2001] **Booch G., Rumbaugh J., Jacobson I.** (2001); *UML* — Przewodnik użytkownika; Warszawa: WNT.
- [Booch-2004] **Booch G., Rumbaugh J., Jacobson I.** (2004); *The UML Reference Manual 2nd Edition*; Boston: Addison-Wesley.
- [Cadle-2004] **Cadle J., Yeates D.** (2004); *Zarządzanie procesem tworzenia systemów informatycznych*; Warszawa: WNT.
- [Chonoles-2003] **Chonoles M.J., Schardt J. A.** (2003); *UML 2*; New York: Wiley.

- [Cheesman-2004] **Cheesman J., Daniels J.** (2004); *Komponenty w UML*; Warszawa: WNT.
- [Dennis-2005] **Dennis A.** (2005); *Systems Analysis and Design with UML Version 2.0*; New York: Wiley.
- [Eriksson-2000] **Eriksson H., Penker M.** (2000); *Business Modeling with UML*; New York: Wiley.
- [Eriksson-2004] **Eriksson H., Penker M., Lyons B., Fado D.** (2004); *UML 2 Toolkit*; Indianapolis: OMG Press.
- [Fowler-2002] **Fowler M., Scott K.** (2002); *UML w kropelce*; Warszawa: Oficyna Wydawnicza LPT.
- [Fowler-2003] **Fowler M., Scott K.** (2003); *UML Distilled 3rd Edition*; Boston: Addison-Wesley.
- [Fowler-2005] **Fowler M., Rice D., Foemmel M., Hieatt E., Mee R., Stafford R.** (2005); *Architektura systemów zarządzania przedsiębiorstwem. Wzorce projektowe*; Gliwice: Helion.
- [Górski-2000] **Górski J.** (2000); *Inżynieria oprogramowania w projekcie informatycznym*; Warszawa: Mikom.
- [Hamlet-2003] **Hamlet D., Maybee J.** (2003); *Podstawy techniczne inżynierii oprogramowania*; Warszawa: WNT.
- [Jacobson-1992] **Jacobson I., Christerson M., Jonsson P., Overgaard G.** (1992); *Object-Oriented Software Engineering: A Use-Case Driven Approach*; Boston: Addison-Wesley.
- [Larman-2002] **Larman C.** (2002); *Applying UML and Patterns*; Boston: Prentice Hall PTR.
- [Larman-2004] **Larman C.** (2004); *Agile and Iterative Development*; Boston: Addison-Wesley.
- [Maciaszek-2001] **Maciaszek L.A.** (2001); *Requirements Analysis and System Design*; Boston: Addison-Wesley.
- [Maciaszek-2005] **Maciaszek L.A.** (2005); *Requirements Analysis and System Design 2nd Edition*; Harlow: Pearson.
- [Maksimchuk-2005] **Maksimchuk R.A., Naiburg E.J.** (2005); *UML for Mere Mortals*; Boston: Addison-Wesley.
- [Marcinkowski-2005] **Marcinkowski B.** (2005); *Isomorphism of Interaction Diagrams in UML 2* (in:) **Abramowicz W.** (ed.); *Proceedings of 8th International Conference on Business Information Systems*; Poznań: Poznań University of Economics Press.

- [Marcinkowski-2005a] **Marcinkowski B.** (2005); *Relevance of Use-Case Scenarios' Descriptions in System Requirements Specification* (in:) **Kubiak B.F., Korowicki A.** (eds.); *Information Management*; Gdańsk: Gdańsk University Press.
- [Marcinkowski-2005b] **Marcinkowski B., Wrycza S.** (2005); *Interaction Occurrences and Combined Fragments in System Dynamics Modeling with UML 2 Sequence Diagram* (in:) **Nilsson A.G., Gustas R., Wojtkowski W., Wojtkowski W.G., Wrycza S., Zupančič J.** (eds.); *Proceeding of the 14th International Conference on Information Systems Development. Advances in ISD: Bridging the Gap between Academia and Practice*; Karlstad: Karstad University Studies.
- [Martin-1997] **Martin J., Odell J.** (1997); *Podstawy metod obiektowych*; Warszawa: WNT.
- [Martin-2003] **Martin R.C.** (2003); *Agile Software Development: Principles, Patterns and Practices*; New Jersey: Prentice Hall.
- [Muller-2000] **Muller R.J.** (2000); *Bazy danych. Język UML w modelowaniu danych*; Warszawa: Mikom.
- [Pender-2002] **Pender T.A.** (2002); *Weekend Crash Course*; New York: Wiley.
- [Pilone-2005] **Pilone D., Pitman N.** (2005); *UML 2.0 in a Nutshell: A Desktop Quick Reference*; Beijing: O'Reilly.
- [Roff-2003] **Roff J.T.** (2003); *UML. A Beginner's Guide*; Berkeley: McGraw-Hill.
- [Rosenberg-1999] **Rosenberg D., Scott K.** (1999); *Use Case Driven Object Modeling with UML*; Boston: Addison-Wesley.
- [Rumbaugh-1991] **Rumbaugh J.** (1991); *Object-Oriented Modeling and Design*; New Jersey: Prentice Hall.
- [RUP-2002] Rational Software Corporation (2002); *Rational Unified Process, Process Made Practical* – plansza informacyjna.
- [Schmuller-2003] **Schmuller J.** (2003); *UML dla każdego*; Gliwice: Helion.
- [Schneider-2004] **Schneider G., Winters J.P.** (2004); *Stosowanie przypadków użycia*; Warszawa: WNT.
- [Scott-2004] **Scott K.** (2004); *Fast Track UML 2.0*; Berkeley: Apress.
- [Shalloway-2005] **Shalloway A., Trott J.R.** (2005); *Projektowanie zorientowane obiektowo. Wzorce projektowe*; Gliwice: Helion.

- [Subieta-1998] **Subieta K.** (1998); *Obiektowość w projektowaniu i bazach danych*; Warszawa: Oficyna Wydawnicza PLJ.
- [Śmiałek-2005] **Śmiałek M.** (2005); *Zrozumieć UML 2.0. Metody modelowania obiektowego*; Gliwice: Helion.
- [Warmer-2003] **Warmer J., Kleppe A.** (2003); *OCL. precyzyjne modelowanie w UML*; Warszawa: WNT.
- [Wrycza-1999] **Wrycza S.** (1999); *Analiza i projektowanie systemów informatycznych zarządzania*; Warszawa: PWN.
- [Wrycza-2000] **Wrycza S.** (2000); *Informatyka dla ekonomistów*; Gdańsk: Wydawnictwa UG.
- [Wrycza-2005] **Wrycza S., Marcinkowski B., Wyrzykowski K.** (2005); *Timing Diagram Functionalities in Information Systems Modeling with UML 2* (in:) **Kubiak B.F., Korowicki A.** (eds.); *Information Management*; Gdańsk: Gdańsk University Press.
- [Wrycza-2005a] **Wrycza S., Marcinkowski B., Wyrzykowski K.** (2005); *Rola fragmentów wyodrębnionych w języku UML 2* (w:) **Kozielski S., Małyśiak B., Kasprowski P., Mrozek D.** (red); *Bazy danych: Modele, Technologie, Narzędzia. Architektura, metody formalne, bezpieczeństwo*; Warszawa: Wydawnictwo Komunikacji i Łączności.
- [Wrycza-2005b] **Wrycza S., Marcinkowski B.** (2005); *UML 2 Teaching at Postgraduate Studies - Prerequisites & Practice; Proceedings of ISECON 2005, Volume 22, the 22nd Annual Conference for Information Systems Educators*; Chicago: AITP Foundation for IT Education.

Źródła elektroniczne:

- [Ameos-2005] <http://www.aonix.com>; stan na dzień 05.09.2005.
- [AnyLogic-2005] <http://www.xjtek.com>; stan na dzień 05.09.2005.
- [ArgoUML-2005] <http://www.argouml.tigris.org>; stan na dzień 05.09.2005.
- [Ariadne-2002] UML Applied 2nd Edition; <http://www.ariadnetraining.co.uk>; stan na dzień 05.09.2005.
- [EA-2005] <http://www.sparxsystems.com.au>; stan na dzień 05.09.2005.
- [Eclipse-2005] <http://www.eclipse.org/community>; stan na dzień 28.06.2005.
- [ESSModel-2005] <http://essmodel.sourceforge.net>; stan na dzień 05.09.2005.

- [Fujaba-2005] <http://www.fujaba.de>; stan na dzień 05.09.2005.
- [Gentleware-2005] <http://www.gentleware.com/index.php?id=34>; stan na dzień 29.06.2005.
- [IBM-2005] www-306.ibm.com/software/rational; stan na dzień 20.03.2005.
- [IBM-2005a] <http://www-306.ibm.com/software/awdtools/modeler/swmodeler/index.html>; stan na dzień 20.03.2005.
- [Innovator-2005] <http://www.mid.de>; stan na dzień 05.09.2005.
- [MDEStudio-2005] http://www.m1global.org/projects/mde_studio.html; stan na dzień 05.09.2005.
- [Metamil-2005] <http://www.metamil.com>; stan na dzień 05.09.2005.
- [Motion-2005] <http://motionmodeling.sourceforge.net>; stan na dzień 05.09.2005.
- [Objectteering-2005] <http://www.objectteering.com>; stan na dzień 05.09.2005.
- [OMG-2003] Object Management Group; *OMG Unified Modeling Language Specification. Version 1.5*, <http://www.omg.org/cgi-bin/doc?ptc/02-09-02.pdf>; stan na dzień 29.08.2003.
- [OMG-2003a] Object Management Group; *UML 2.0 Superstructure Specification*, <http://www.omg.org/cgi-bin/doc?ptc/03-08-02.pdf>; stan na dzień 28.11.2004.
- [OMG-2003b] Object Management Group; *UML 2.0 Infrastructure Specification*, <http://www.omg.org/cgi-bin/doc?ptc/2003-09-15.pdf>; stan na dzień 20.03.2005.
- [OMG-2003c] Object Management Group; *UML 2.0 Diagram Interchange Specification*, <http://www.omg.org/cgi-bin/doc?ptc/2003-09-01.pdf>; stan na dzień 20.03.2005.
- [OMG-2004] Object Management Group; *UML 2.0 Superstructure FTF Convenience Document*, <http://www.omg.org/docs/ptc/04-10-02.pdf>; stan na dzień 15.07.2005.
- [OMG-2005] Object Management Group; *Unified Modeling Language: Superstructure. Version 2.0 Formal*, <http://www.omg.org/docs/formal/05-07-04.pdf>; stan na dzień 05.09.2005.
- [Poseidon-2005] <http://www.gentleware.com>; stan na dzień 05.09.2005.

- [Rational-2001] Rational Software Corporation; *Business Modeling with UML: The Light at the End of the Tunnel*, <http://www-128.ibm.com/developerworks/rational/library/content/RationalEdge/dec01/BusinessModelingwithUMLDec01.pdf>; stan na dzień 05.09.2005.
- [Rosenberg-1999] <http://www.cs.hmc.edu/courses/2005/spring/cs121/slides/slides06.pdf>; stan na dzień 20.03.2005.
- [RSWP-1998] Rational Software White Paper; *Rational Unified Process. Best Practices for Software Development Teams*, ftp://ftp.software.ibm.com/software/rational/web/whitepapers/2003/rup_bestpractices.pdf; stan na dzień 05.09.2005.
- [RSWP-2001] Rational Software White Paper; *Business Modeling with the UML and Rational Suite AnalystStudio*, <http://www.fit.hcmuns.edu.vn/~lqv/docs/uml/BusinessModelingwithUMLandRSAS.pdf>; stan na dzień 05.09.2005.
- [RUP-2003] Rational Unified Process – hipertekstowa baza wiedzy; stan na dzień 10.01.2004.
- [Teuta-2005] <http://www.par.univie.ac.at>; stan na dzień 05.09.2005.
- [Together-2005] <http://www.borland.com>; stan na dzień 05.09.2005.
- [Umbrello-2005] <http://uml.sourceforge.net>; stan na dzień 05.09.2005.
- [UniMod-2005] <http://unimod.sourceforge.net/tools-packages.html>; stan na dzień 05.09.2005.
- [Xylophon-2005] <http://xylophon.sourceforge.net>; stan na dzień 05.09.2005.