Normalizacja i projektowanie uniwersalne

przygotowanie: dr inż. Grzegorz Rybak

kod przedmiotu: 02 09 5870 00

Wykład: nr 4
Normy związane z zarządzaniem wymaganiam

Na wykładzie poznamy:

- Standard IEEE 830
- Elementy specyfikacji wymagń
- Wariany sekcji wymagń funkcjonalnych
- Standard modelowania UML (Unified Modeliing Language) (zarys)

Standard IEEE 830

Dokument zawiera

opis zawartości i wskazówek jakościowych dotyczących specyfikacji wymagań oprogramowania (SRS).

Przedstawiono również kilka przykładów dobrych praktyk.

Prezenotwane podejście pozwala na wspieranie procesu wytwórczego oprogramowania zarówno w warunkach prywatnych jak i dla celów komercyjnych.

Wprowadzono również wskazówki zgodności do IEEE/EIA12207.1-1997.

Słowa kluczowe: umowa, klient, prototypowanie, specyfikacja wymagań oprogramowania, dostawca, system, specyfikacje wymagań.

Zastosowanie standardu IEEE jest całkowicie dobrowolne.

Standard IEEE 830

Klientom, dostawcom i innym osobom dobra SRS powinna zapewniać kilka konkretnych korzyści, takich jak jak:

- **Ustanowić podstawę porozumienia między klientami a dostawcami co do zakresu oprogramowania,**

Pełny opis funkcji do wykonania przez określone oprogramowanie
w SRS pomoże potencjalnym użytkownikom ustalić, czy określone oprogramowanie spełnia ich potrzeby lub w jaki sposób oprogramowanie musi zostać zmodyfikowane w celu zaspokojenia ich potrzeb.

- **Zmniejszyć wysiłek rozwojowy**,

Przygotowanie SRS zmusza różne zainteresowane grupy w organizacja klienta musi dokładnie rozważyć wszystkie wymagania przed rozpoczęciem projektowania i redukuje późniejsze przeprojektowanie, ponowne kodowanie i ponowne testowanie. Dokładny przegląd wymagań zawartych w SRS może wskazać pominięcia, nieporozumienia i niespójności na wczesnym etapie cyklu rozwoju, gdy są łatwiejsze do naprawienia.

Standard IEEE 830

Klientom, dostawcom i innym osobom dobra SRS powinna zapewniać kilka konkretnych korzyści, takich jak jak:

- **Zapewnienie podstawy do oszacowania kosztów i harmonogramów.**
Opis produktu, który ma zostać opracowany jako podana w SRS stanowi realistyczną podstawę do oszacowania kosztów projektu i może być wykorzystana do uzyskania zatwierdzenia oferty lub szacunkowe ceny.

- **Zapewnienie linii bazowej dla walidacji i weryfikacji.**
Organizacje mogą rozwinąć swoją walidację i plany weryfikacji znacznie bardziej produktywnie. W ramach umowy deweloperskiej SRS stanowi punkt odniesienia, na podstawie którego można zmierzyć zgodność.

- **Ułatwienie transferu.**
SRS ułatwia przeniesienie oprogramowania do nowych użytkowników lub nowych maszyn. Dzięki temu klienci mogą łatwiej przenosić oprogramowanie do innych części swojej organizacji, a dostawcy łatwiej przenoszą go do nowych klientów.

- **Służyć jako podstawa udoskonalenia.**
Ponieważ SRS omawia produkt, ale nie jego projekt, SRS służy jako podstawa do późniejszego ulepszenia gotowego produktu. SRS może wymagają zmiany, ale stanowi podstawę do dalszej oceny produkcji.

Standard IEEE 830

Plan omawianego dokumentu:

a) opis charakterystyki SRS;

b) opis środowiska SRS;

c) charakterystykę dobrego SRS;

d) wspólne przygotowanie SRS;

e) ewolucję i udoskonalenia SRS;

f) prototypowanie;

g) projekt osadzania w SRS;

h) Osadzanie wymagań projektowych w SRS

Standard IEEE 830

SRS to specyfikacja konkretnego oprogramowania, programu lub zestawu programów, które wykonuje niektóre funkcje w określonym środowisku.

SRS może być napisana przez jednego lub więcej przedstawicieli dostawcy, jednego lub więcej przedstawicieli klienta lub przez obu.

Podstawowe kwestie, które autor lub autorzy SRS powinni rozwiązać, są następujące:

a) **Funkcjonalność**. Co powinno robić oprogramowanie?

b) **Interfejsy zewnętrzne**. W jaki sposób oprogramowanie wchodzi w interakcje z ludźmi, sprzętem systemu, innym sprzętem i innym oprogramowaniem?

c) **Wydajność**. Jaka jest prędkość, dostępność, czas reakcji, czas odzyskiwania różnych funkcji oprogramowania itp.?

d) **Atrybuty**. Jakie są kwestie związane z przenośnością, poprawnością, konserwacją, bezpieczeństwem itp.?

e) **Ograniczenia projektowe nałożone na implementację**. Czy obowiązują jakieś wymagane standardy, język implementacji, zasady integralności bazy danych, ograniczenia zasobów, środowisko (-a) operacyjne itp.?

Standard IEEE 830

Charakterystyka dobrej specyfikacji wymagań oprogramowania (SRS)

Specyfikacja powinna być:

- Poprawna,
- Jednoznaczna,
- Kompletna,
- Spójna,
- Wyposażona w ocenę ważności,
- Weryfikowalna,
- Modyfikowalna,
- Możliwa do śledzenia zmian/ zależności.

Standard IEEE 830

Sposób wykonania dokumentu

Specyfikacja powinna być:

Proces tworzenia oprogramowania powinien rozpoczynać się od umowy z dostawcą i klientem dotyczącej tego, co gotowe oprogramowanie musi zrobić. Umowa ta, w formie SRS, powinna zostać wspólnie przygotowana. To jest ważne, ponieważ zwykle ani klient, ani dostawca nie jest uprawniony do napisania dobrego SRS samodzielnie.

a) Klienci zazwyczaj nie rozumieją wystarczająco dobrze procesu projektowania i tworzenia oprogramowania.

b) Dostawcy zwykle nie rozumieją problemu klienta i zakresu jego starań wystarczająco dobrze aby określić wymagania dotyczące zadowalającego systemu.

Dlatego klient i dostawca powinni współpracować, aby utworzyć dobrze napisany, kompletny oraz zrozumiał SRS.

Standard IEEE 830

Ewolucja dokument SRS

W trakcie rozwoju produktu, SRS może wymagać modyfikacji. Czasami jest niemożliwe wyspecyfikowanie pewnych szczegółów w momencie gdy projekt sie rozpoczyna. (np. zdefiniowanie podczas fazy wymagań, wszystkich formatów ekranu dla programu interaktywnego może być niemożliwe). Dodatkowe zmiany mogą się również pojawić, gdy zidentyfikowane zostaną braki czy niedokłaności.

Dwa główne zagadnienia w tym procesie są następujące:

a) Wymagania powinny być określone tak kompletnie i dokładnie, jak jest to znane w danym momencie, nawet jeśli zmiany ewolucyjne można przewidzieć jako nieuniknione. Fakt, że są niekompletne, powinien być odnotowany.

b) Należy zainicjować formalny proces zmian w celu identyfikacji, kontroli, śledzenia i zgłaszania prognozowanych zmian.

Zatwierdzone zmiany wymagań powinny zostać włączone do SRS w taki sposób, aby
1) Zapewnić dokładną i pełną ścieżkę audytu zmian;
2) Zezwolić na przegląd bieżących i zastąpionych części SRS.

Standard IEEE 830

Prototypowanie

Prototypowanie używane jest często podczas fazy projektu dotyczącej wymagań. Istnieje wiele narzędzi, które pozwalają na wykoanie prototypu, wykazującego pewne cechy systemu, które można bardzo szybko i łatwo utworzyć.

Prototypy są przydatne z następujących powodów:

a) Klient może częściej oglądać prototyp i reagować na niego niż czytać SRS i reagować. W ten sposób prototyp zapewnia szybką informację zwrotną.

b) Prototyp wyświetla nieprzewidziane aspekty zachowania systemów. Tak więc produkuje nie tylko odpowiedzi, ale także nowe pytania. Pomaga to osiągnąć zamknięcie SRS.

c) SRS oparty na prototypie ma tendencję do mniejszych zmian podczas rozwoju, a tym samym skraca się czas rozwoju.

Prototyp powinien być wykorzystywany jako sposób na uzyskanie wymagań dotyczących oprogramowania. Niektóre cechy, takie jak ekran lub formaty raportów można wyodrębnić bezpośrednio z prototypu. Inne wymagania można wywnioskować przez uruchomienie eksperymentu z prototypem.

Standard IEEE 830

Projektowanie rozwiązania w trakcie przygotowania specyfikacji wymagań

- Wymaganie określa zewnętrznie widoczną funkcję lub atrybut systemu.

- Projekt opisuje konkretny podskładnik systemu i / lub jego interfejsy z innymi podskładnikami.

Autor (autorzy) SRS powinien wyraźne rozróżnienie między identyfikacją wymaganych ograniczeń projektowych, a rzutowaniem konkretnego projektu.

Uwaga: każde wymaganie w SRS ogranicza możliwości projektowania.
Nie oznacza to jednak, że każde wymaganiem to projekt.

Standard IEEE 830

Dokument specyfikacji wymagań:

Spis treści

1. Wprowadzenie

1.1. Cel

a) Określ cel SRS;
b) Podaj docelową grupę odbiorców SRS

1.2. Zakres

a) Zidentyfikuj oprogramowanie, które ma zostać wyprodukowane według nazwy (np. Host DBMS, Generator raportów itp.);
b) Wyjaśnij, co będzie robić oprogramowanie, a jeśli to konieczne, nie będzie;
c) Opisz zastosowanie określonego oprogramowania, w tym odpowiednie korzyści, cele;
d) Zachowaj spójność z podobnymi stwierdzeniami w specyfikacjach wyższego poziomu (np. wymaganiach systemowych) specyfikacja), jeśli istnieją;

1.3. Definicje, akronimy, skróty

Standard IEEE 830

Dokument specyfikacji wymagań:

cd..

1.4. Referencje

a) dostarcz pełną listę wszystkich dokumentów wymienionych w innym miejscu w SRS;
b) Zidentyfikuj każdy dokument według tytułu, numeru raportu (jeśli dotyczy), daty i organizacji wydawniczej;
c) Podaj źródła, z których można uzyskać odniesienia.

1.5. Zarys ogólny

a) Opisz, co zawiera reszta SRS;
b) Wyjaśnij, jak zorganizowana jest SRS.

2. Ogólny opis

2.1. Persperktywa produktu

Pomocny może być schemat blokowy przedstawiający główne komponenty większego systemu, połączenia i interfejsy zewnętrzne.

Standard IEEE 830

Dokument specyfikacji wymagań:

cd..

W tej podsekcji należy również opisać, w jaki sposób oprogramowanie działa w ramach różnych ograniczeń. Na przykład, ograniczenia te mogą obejmować

a) interfejsy systemowe;
b) Interfejsy użytkownika;
c) interfejsy sprzętowe; opis może zawierać np.:
- nazwę;
- mnemoniki;
- numer specyfikacji;
- numer wersji;
- źródło.

d) interfejsy oprogramowania;
e) interfejsy komunikacyjne;
f) pamięć;
g) operacje;
h) Wymagania dotyczące dostosowania strony.

Standard IEEE 830

Dokument specyfikacji wymagań:

cd..

2.2. Funkcje produktu

a) Funkcje powinny być zorganizowane w taki sposób, aby lista funkcji była zrozumiała dla klienta lub kogokolwiek czytającego dokument po raz pierwszy.
b) W celu pokazania różnych funkcji i ich zależności można zastosować metody tekstowe lub graficzne.Taki schemat nie ma na celu przedstawienia projektu produktu, ale po prostu pokazuje logiczne relacje między zmiennymi.

2.3. Charakterystyka użytkowników

W tej podsekcji SRS należy opisać ogólne cechy charakterystyczne użytkowników produktu w tym poziom wykształcenia, doświadczenie i wiedzę techniczną. Nie należy jej używać do określania konkretnych stanów wymagania, ale raczej powinny podać powody, dla których niektóre szczegółowe wymagania są później określone w sekcji 3 SRS.

Standard IEEE 830

Dokument specyfikacji wymagań:

cd..

2.4. Ograniczenia

Ta podsekcja SRS powinna zawierać ogólny opis wszelkich innych elementów, które ograniczą opcje programisty. Obejmują one:

a) Polityka regulacyjna;
b) Ograniczenia sprzętowe (np. wymagania dotyczące taktowania sygnału);
c) Interfejsy do innych aplikacji;
d) Praca równoległa;
e) Funkcje audytu;
f) Funkcje kontrolne;
g) Wymagania językowe wyższego rzędu;
h) Protokoły uzgadniania sygnałów (np. XON-XOFF, ACK-NACK);
i) Wymagania dotyczące niezawodności;
j) Krytyczność wniosku;
k) Względy bezpieczeństwa i ochrony.

Standard IEEE 830

Dokument specyfikacji wymagań:

cd..

2.5. Założenia i zależności

W tej podsekcji SRS należy wymienić każdy z czynników wpływających na wymagania określone w SRS. Czynniki te nie są ograniczeniami projektowymi w oprogramowaniu, ale są raczej zmianami, które mogą mieć na nie wpływ wymagania SRS. Na przykład można założyć, że będzie to określony system operacyjny dostępne na sprzęcie przeznaczonym dla oprogramowania. Jeśli w rzeczywistości system operacyjny nie jest dostępny, SRS musiałby odpowiednio się zmienić.

3. specyfikacja wymagań funkcjonalnych

Są różne warianty w zależności od przeznaczenia dokumentu

Załączniki
Indeksy

Standard IEEE 830

Dokument specyfikacji wymagań:

cd..

3. specyfikacja wymagań funkcjonalnych

3.1 External interface requirements
- 3.1.1 User interfaces
- 3.1.2 Hardware interfaces
- 3.1.3 Software interfaces
- 3.1.4 Communications interfaces

3.2 Functional requirements
- 3.2.1 Information flows
  - 3.2.1.1 Data flow diagram 1
    - 3.2.1.1.1 Data entities
    - 3.2.1.1.2 Pertinent processes
    - 3.2.1.1.3 Topology
  - 3.2.1.2 Data flow diagram 2
    - 3.2.1.2.1 Data entities
    - 3.2.1.2.2 Pertinent processes
    - 3.2.1.2.3 Topology
  - .
  - .
  - .
  - 3.2.1.*n* Data flow diagram *n*

Standard IEEE 830

Dokument specyfikacji wymagań:

cd..

3.1. Zewnetrzne interfejsy

a) nazwa inferfejsu;
b) opis celu interfejsu;
c) źródło wejścia lub miejsce docelowe produkcji;
d) obowiązujący zakres, dokładność i / lub tolerancja;
e) jednostki miary;
f) czasy wykonania;
g) relacje z innymi wejściami / wyjściami;
h) formaty / organizacja ekranu;
i) formaty / organizacja okien;
j) formaty danych;
k) formaty poleceń;
l) podsumowanie opisu interfejsu.

Standard IEEE 830

Dokument specyfikacji wymagań:

cd..

3.2.Wymagania funkcjonalne

Wymagania funkcjonalne powinny określać podstawowe działania, które muszą mieć miejsce w oprogramowaniu przyjmowanie i przetwarzanie danych wejściowych oraz przetwarzanie i generowanie danych wyjściowych. Są one ogólnie wymienione jako oświadczenia „powinien” zaczynające się od „**System powinien…**”

Obejmują one:

a) Walidację danych wejściowych
b) Dokładna sekwencja operacji
c) Reakcje na sytuacje nienormalne, w tym
    1) Przepełnienie (ang. overflow)
    2) Urządzenia komunikacyjne
    3) Obsługa błędów i odzyskiwanie
d) Wpływ parametrów
e) Relacja wyników do nakładów, w tym
    1) Sekwencje wejściowe / wyjściowe
    2) Wzory do konwersji danych wejściowych na wyjściowe

Może być właściwe podzielenie wymagań funkcjonalnych na podfunkcje lub podprocesy.

UML

Unified Modeling Language
(zunifikowany język modelowania)

Jest to język pół-formalny wykorzystywany do modelowania różnego rodzaju systemów, stworzony przez Grady Boocha, Jamesa Rumbaugha oraz Ivara Jacobsona, obecnie rozwijany przez Object Management Group

Służy do modelowania dziedziny problemu (opisywania-modelowania fragmentu istniejącej rzeczywistości – na przykład modelowanie tego, czym zajmuje się jakiś dział w firmie) – w przypadku stosowania go do analizy oraz do modelowania rzeczywistości, która ma dopiero powstać – tworzy się w nim głównie modele systemów informatycznych. UML jest przeważnie używany wraz ze swoją reprezentacją graficzną – jego elementom przypisane są odpowiednie symbole wiązane ze sobą na diagramach [2].

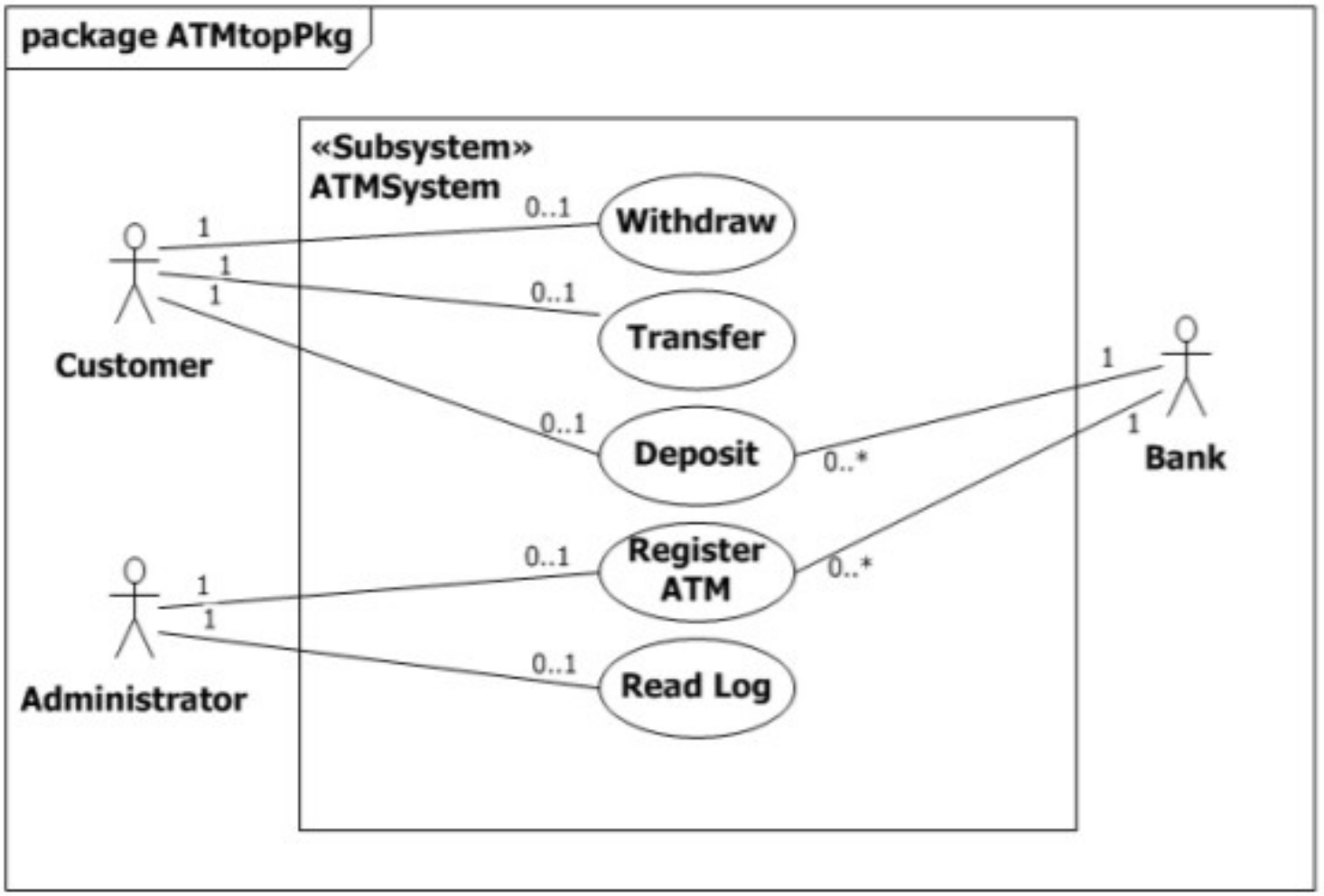
UML – narzędzia

CASE (Computer-Aided Software Engineering, Computer-Aided Systems Engineering) – oprogramowanie używane do komputerowego wspomagania projektowania oprogramowania.

Funkcje CASE-a to analiza, projektowanie i programowanie Narzędzia CASE automatyzują metody projektowania, dokumentacji oraz tworzenia struktury kodu programu w wybranym języku programowania, najczęściej w programowaniu obiektowym.

- Visual Paradigm for UML
- Lucid chart
- draw.io
- eclipse

UML – przykłady (use case)



UML – przykłady (extended use case)

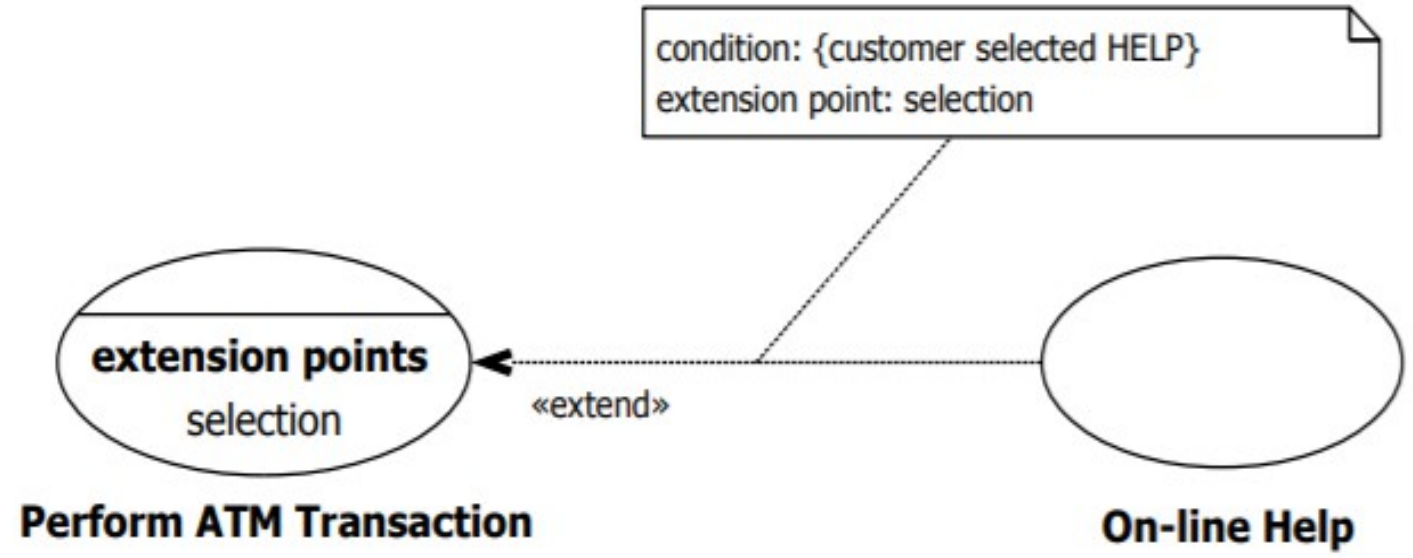


Figure 18.3 Example Extend

UML – przykłady (actor)

Figure 18.6, Figure 18.7 and Figure 18.8 exemplify the three different notations for Actors.

Customer

Customer

Figure 18.6 Actor notation using stick-man

Figure 18.7 Actor notation using Class rectangle

User

Figure 18.8 Actor notation using icon

UML – przykłady (use case w połączeniu z maszyną stanów)

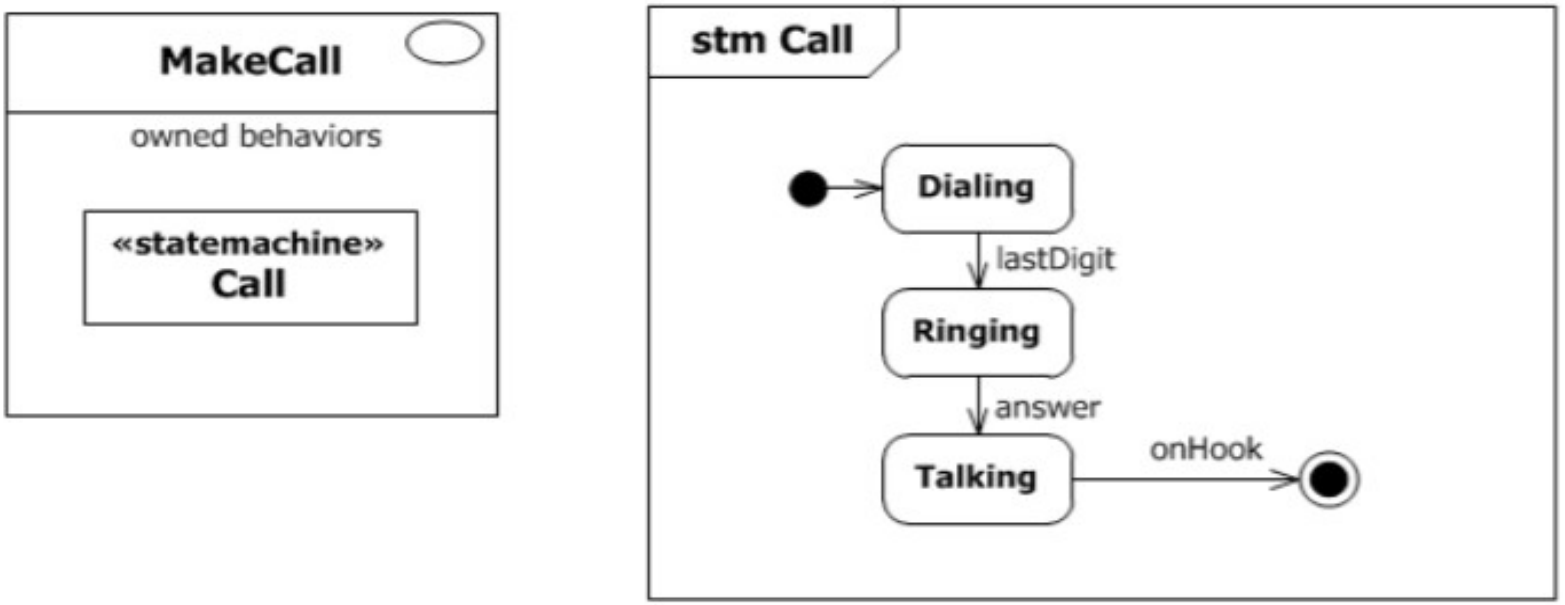
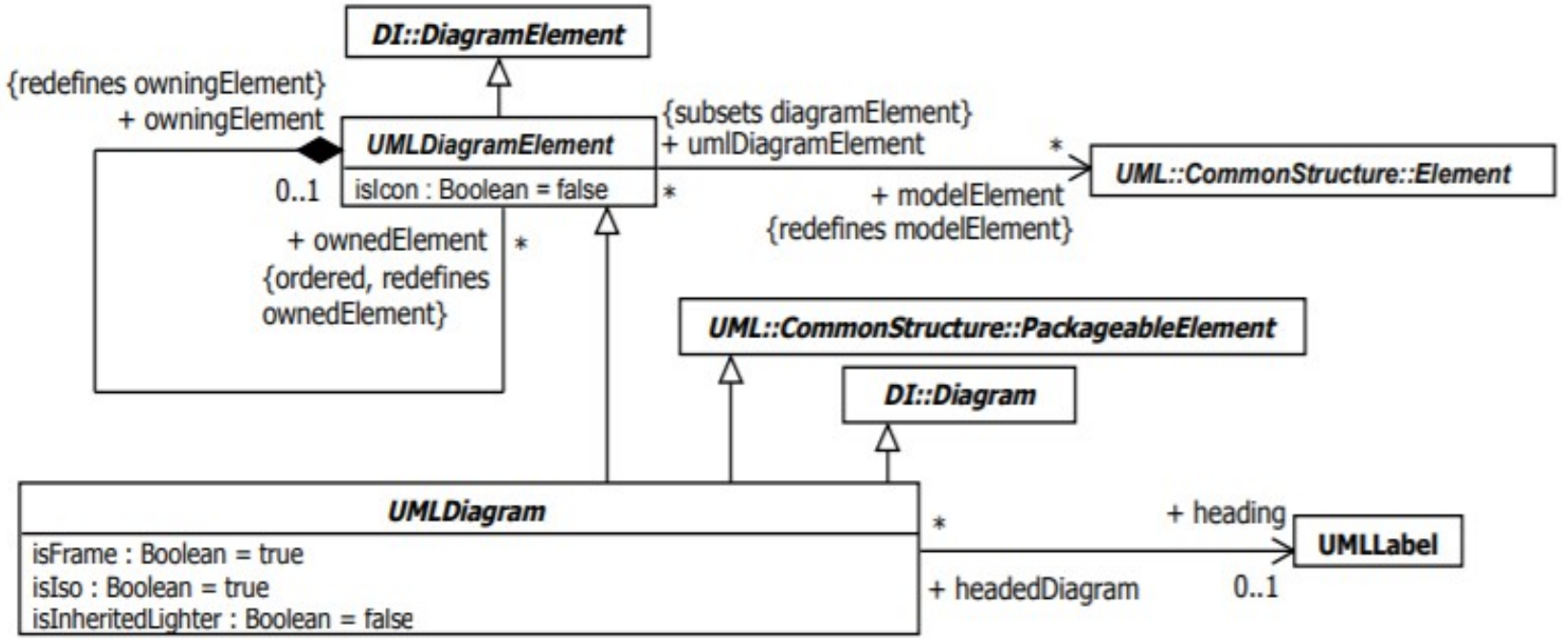


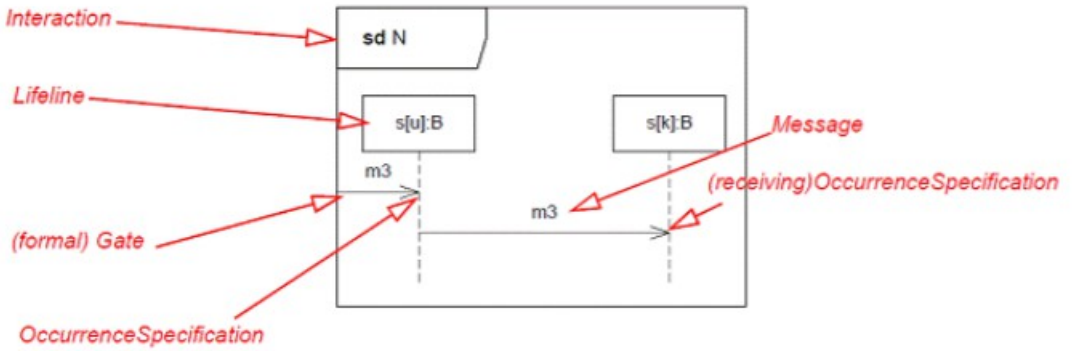
Figure 18.12 Example UseCase with associated StateMachine

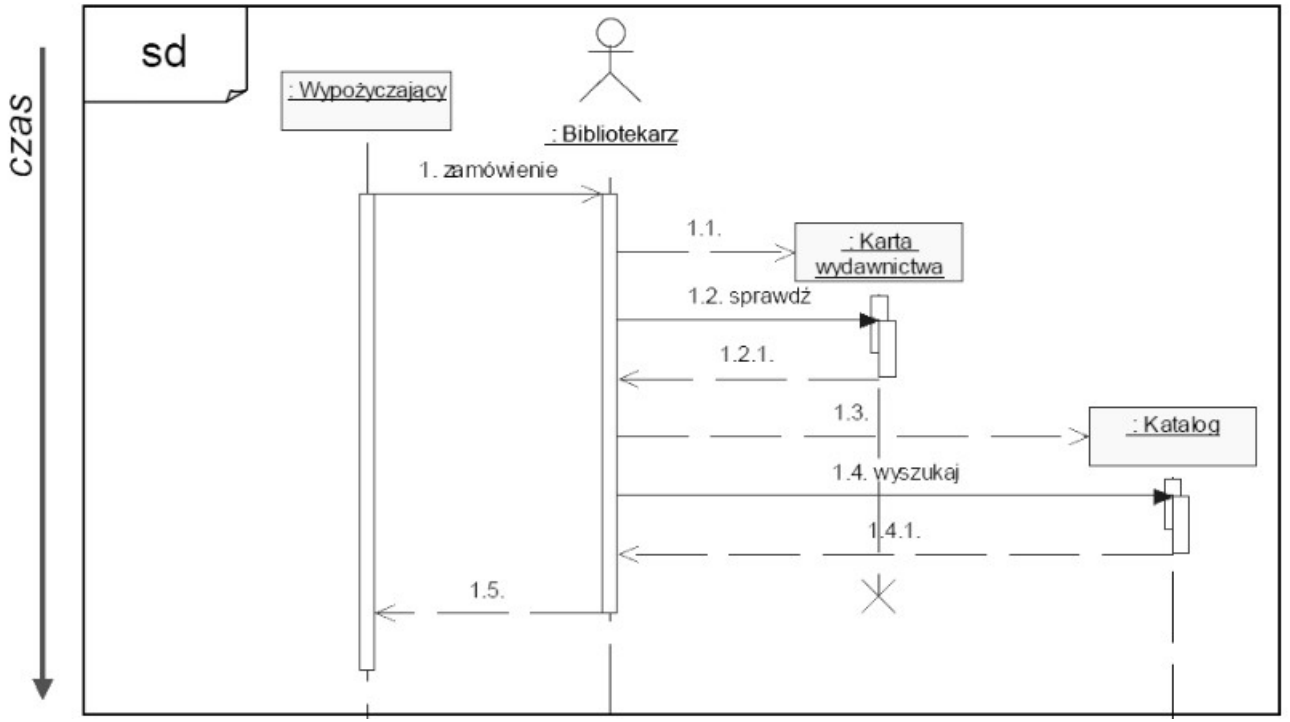
UML – przykłady (generalny model diagramów)



UML – przykłady (diagram sekwencji)

17.8.2 Example Sequence Diagram





Referencje

IEEE Recommended Practice for Software Requirements Specifications
IEEE Std 830-1998
(Revision of
IEEE Std 830-1993)

[2] https://pl.wikipedia.org/wiki/Unified_Modeling_Language

[3] https://www.uml.org/

[4] http://zasoby.open.agh.edu.pl/~09sbfraczek/diagram-sekwencji%2C1%2C13.html