

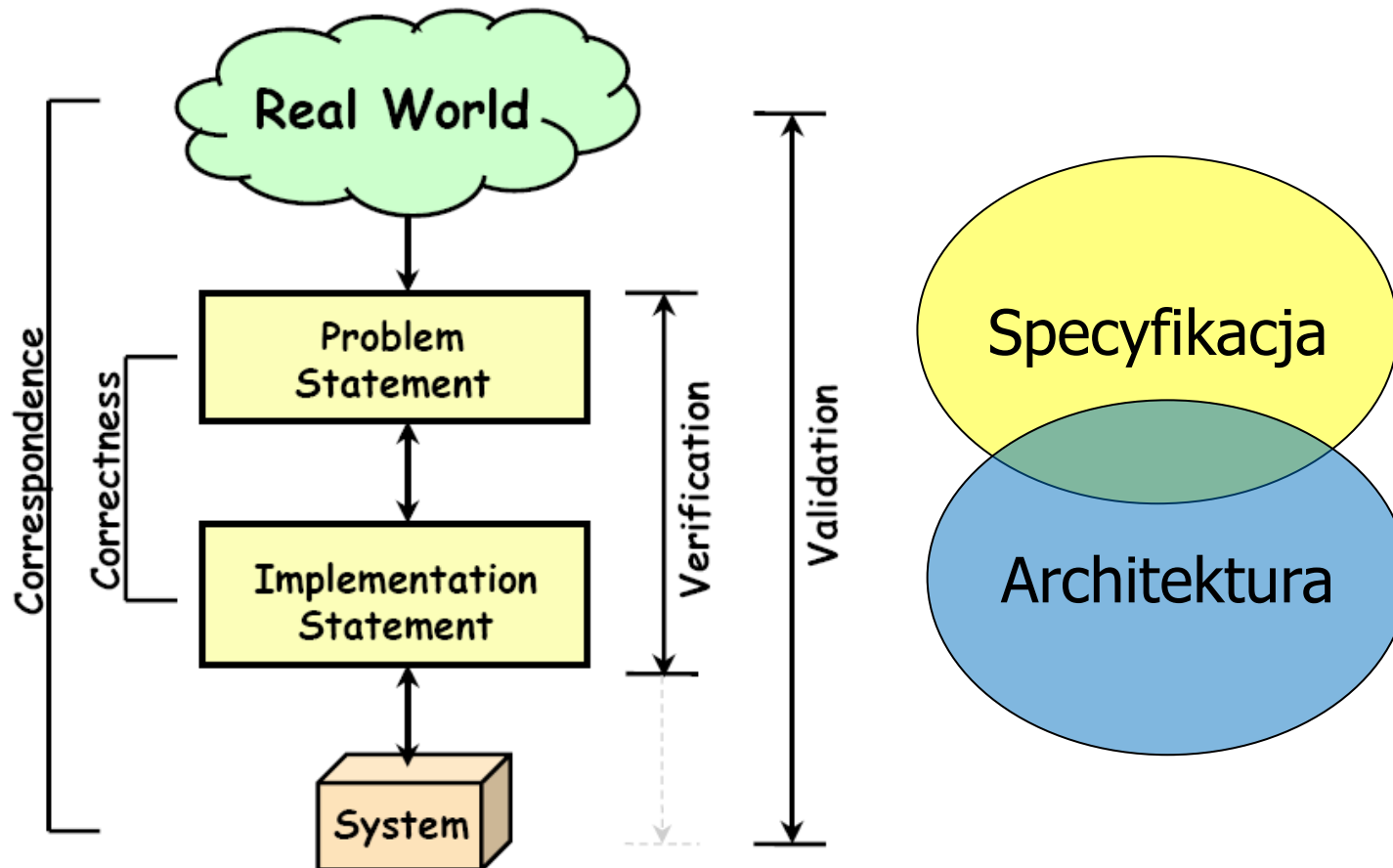
Inżynieria oprogramowania

Analizowanie i projektowanie

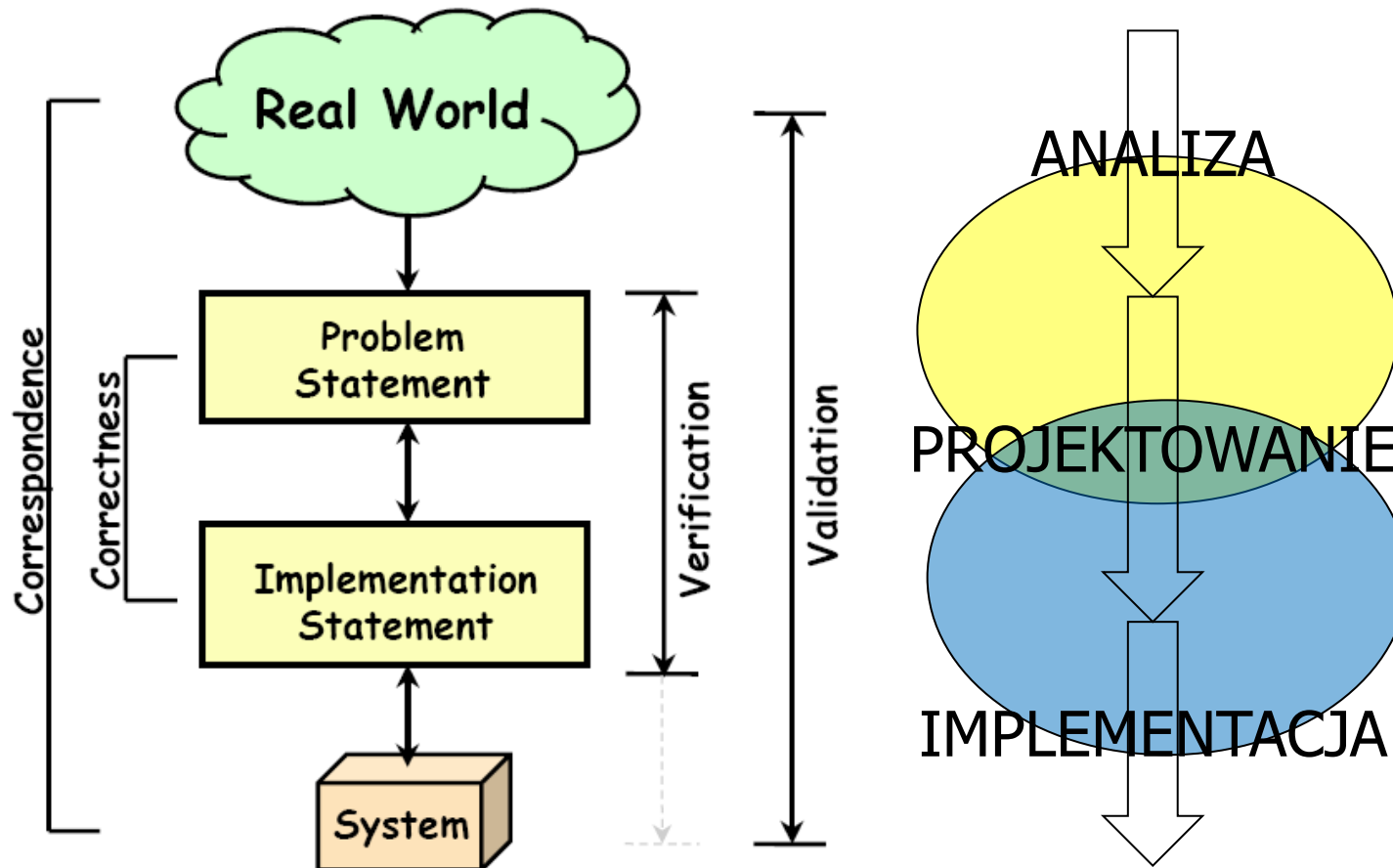


Wydział Matematyki, Informatyki i Mechaniki
Uniwersytet Warszawski
www.mimuw.edu.pl/~dabrowski

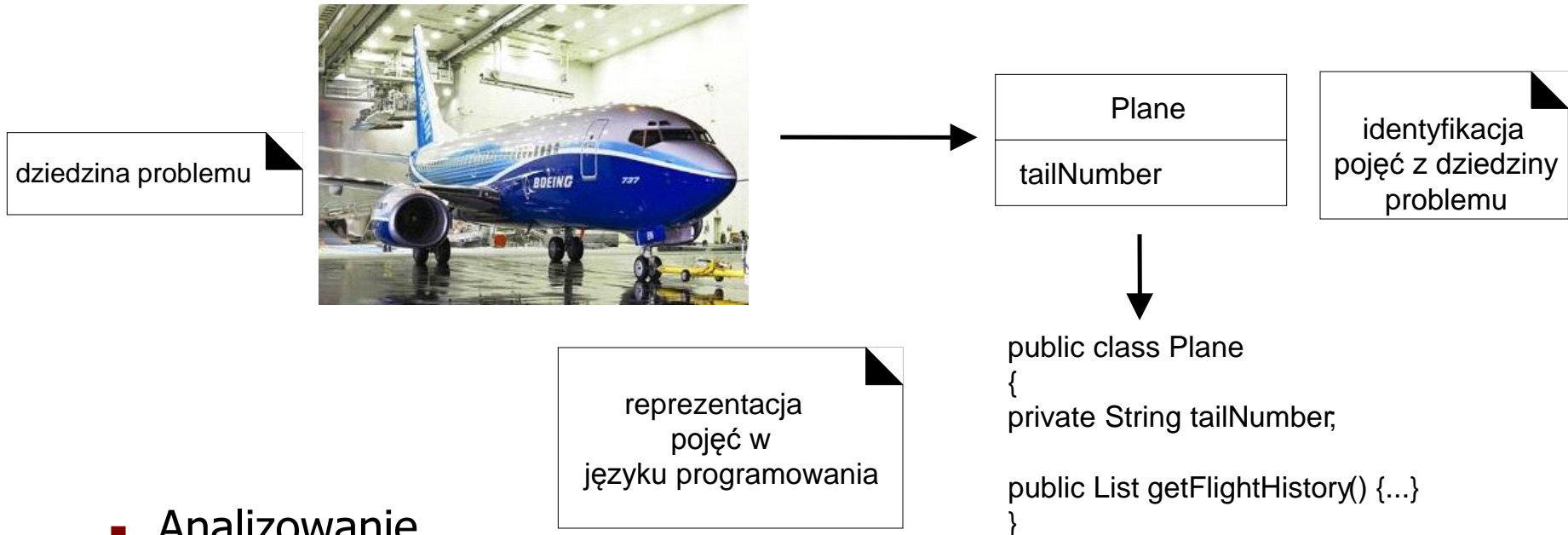
W poprzednim odcinku ...



W dzisiejszym odcinku ...



Analizowanie vs projektowanie



- Analizowanie
 - Znajdowanie i opisywanie bytów / pojęć z dziedziny problemu
 - Projektowanie
 - Definiowanie elementów oprogramowania i sposobów, w jaki będą współpracować w celu spełnienia wymagań
 - Cele analizowania i projektowania
 - Zrozumienie zagadnienia
 - Ustalenie założeń
- } to determinuje poziom szczegółowości

Przypomnienie – inżynieria oprogramowania

- Przypomnienie
 - „Managing complexity in software development”
- czyli
 - Jak radzić sobie ze skomplikowanym oprogramowaniem?
- Podejście
 - Abstrakcja
 - uprość i uogólnij
 - Dekompozycja
 - podziel i zwycięż



Abstrakcja

- Wnioskowanie na temat problemu
 - **upraszcza** problem
 - **nie rozwiązuje** problemu
- Dobra abstrakcja
 - **ukrywa/ignoruje** niepotrzebne **szczegóły**
 - **upraszcza** analizę
 - **znajduj analogie** pomiędzy różnymi bytami

Abstrakcja: przykłady

- Plik
 - pojęcie w systemie plików
 - sekwencja bitów na dysku

- Samolot
 - środek transportu
 - 3 miliony komponentów

Abstrakcja: główne techniki

- Abstrakcja przez parametryzację
 - opisywanie **parametrów** wejściowych i wynikowych
 - **wiadomo jak** przeprowadzać obliczenia
 - nie wiadomo w jakim celu
- Abstrakcja przez specyfikację
 - opisywanie **warunków/stanów** wstępnych i końcowych
- Obie formy są komplementarne

Dekompozycja

- Rozwiązywanie dużych problemów
 - metoda **dziel i zwyciężaj**
- Dobra dekompozycja
 - każdy podproblem jest podobnej wielkości
 - podproblemy można rozwiązywać niezależnie
 - z rozwiązań podproblemów można uzyskać rozwiązanie całości

Dekompozycja: przykłady

- Dobry przykład: **Projektowanie menu restauracji**
 - Wybierz styl graficzny
 - Zaprojektuj menu przystawek
 - Zaprojektuj menu dań głównych
 - Zaprojektuj menu deserów
 - Złóż całość

Dekompozycja: przykłady

- Zły przykład: **Pisanie sztuki teatralnej**
 - Wybierz bohaterów
 - Napisz rolę pierwszego bohatera
 - Napisz rolę drugiego bohatera
 - Napisz rolę trzeciego bohatera
 - Złóż całość

Dekompozycja: główne techniki

- Identyfikuj komponenty
 - minimalizacja powiązań między komponentami
 - komponenty ukrywają wewnętrzną reprezentację
- Modeluj komponenty
 - model wizualny na poziomie projektowania
 - deklaracje i definicje na poziomie implementacji



Chcemy pracować metodycznie

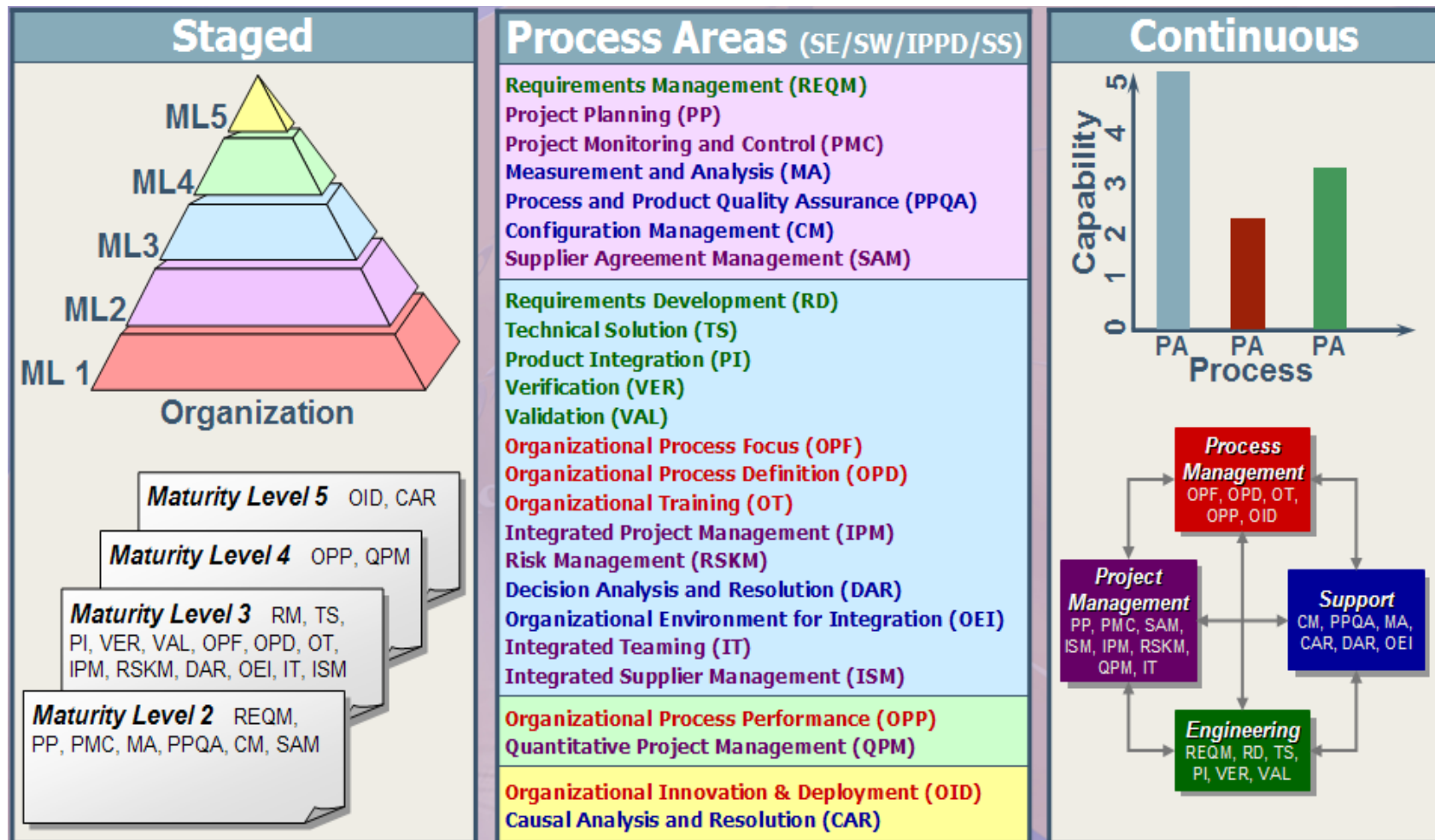
- Chcemy pracować metodycznie
 - Chcemy uniknąć:
 - *Optimism of youth: We can do it over the weekend*
 - *Marine Corps mentality: Real programmers don't need sleep*
- **Metodyki** o różnym poziomie formalizmu
 - co to jest metodyka?
 - Lekkie (*Small-scale projects*, 3-6 osób, 3-6 miesięcy)
 - np. Agile Software Development
 - Średnie (*Medium-size projects*, 20-30 osób, 1-2 lata)
 - np. Unified Process
 - Ciężkie (*Large-scale projects*, 100-300 osób, 3-5 lat)
 - np. Capability Maturity Model Integration
- Wybór metodyki
 - powinien być świadomy
 - dopasowany do specyfiki projektu



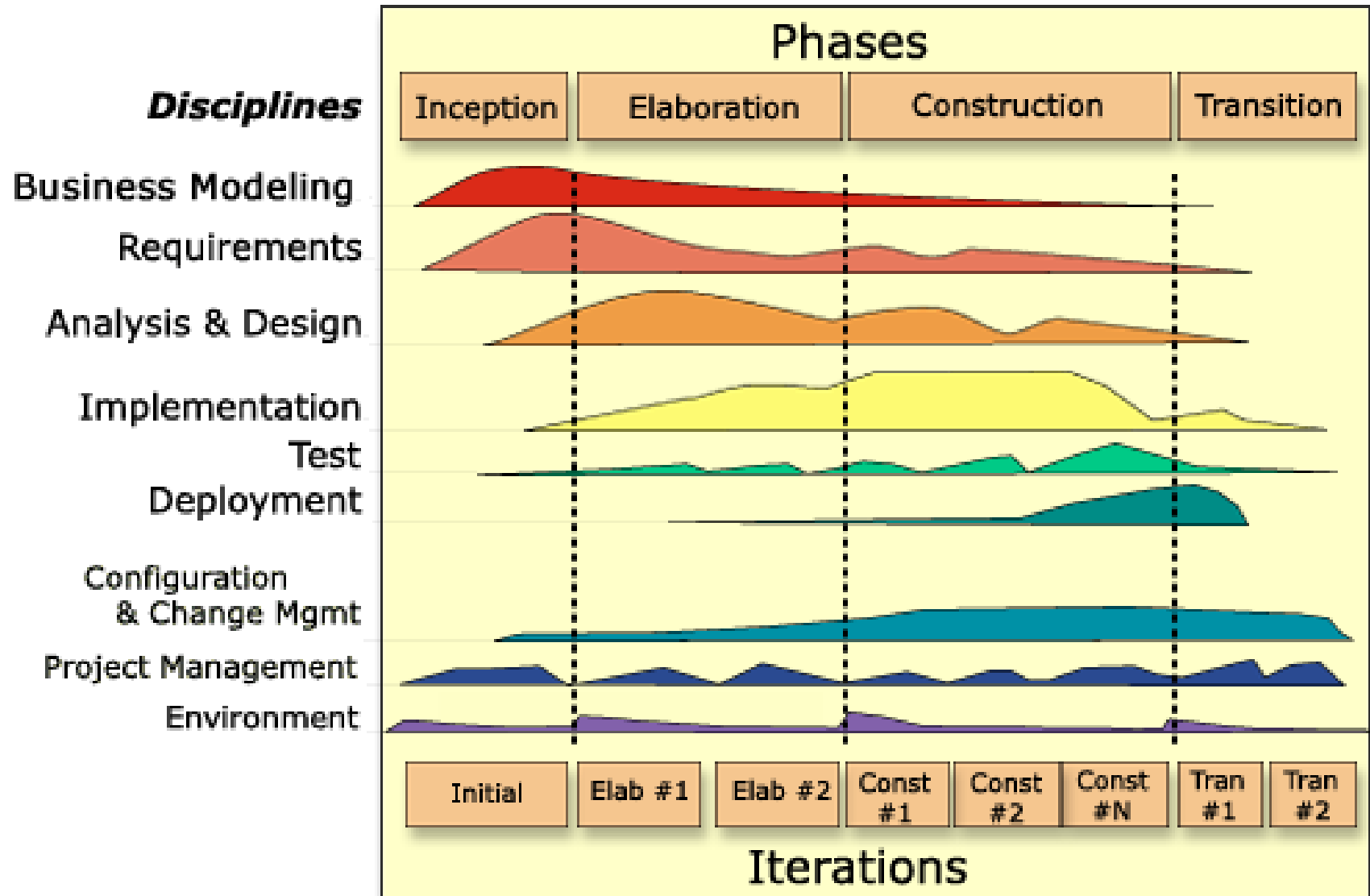
Agile Software Development (metodyka lekka)

- Manifesto for Agile Software Development
 - **Individuals and interactions** over processes and tools
 - **Working software** over comprehensive documentation
 - **Customer collaboration** over contract negotiation
 - **Responding to change** over following a plan
- Principles behind the Agile Manifesto
 - Our highest priority is to satisfy the customer through early and continuous **delivery of valuable software**.
 - Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
 - **Deliver working software frequently**, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
 - Business people and developers must work together daily throughout the project.
 - Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
 - The most efficient and effective method of conveying information to and within a development team is **face-to-face conversation**.
 - **Working software is the primary measure of progress**.
 - Agile processes promote sustainable development. The sponsors, developers, and users should be able to **maintain a constant pace** indefinitely.
 - Continuous attention to **technical excellence and good design enhances agility**.
 - **Simplicity** – the art of maximizing the amount of work not done – **is essential**.
 - The best architectures, requirements, and designs emerge from self-organizing teams.
 - At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

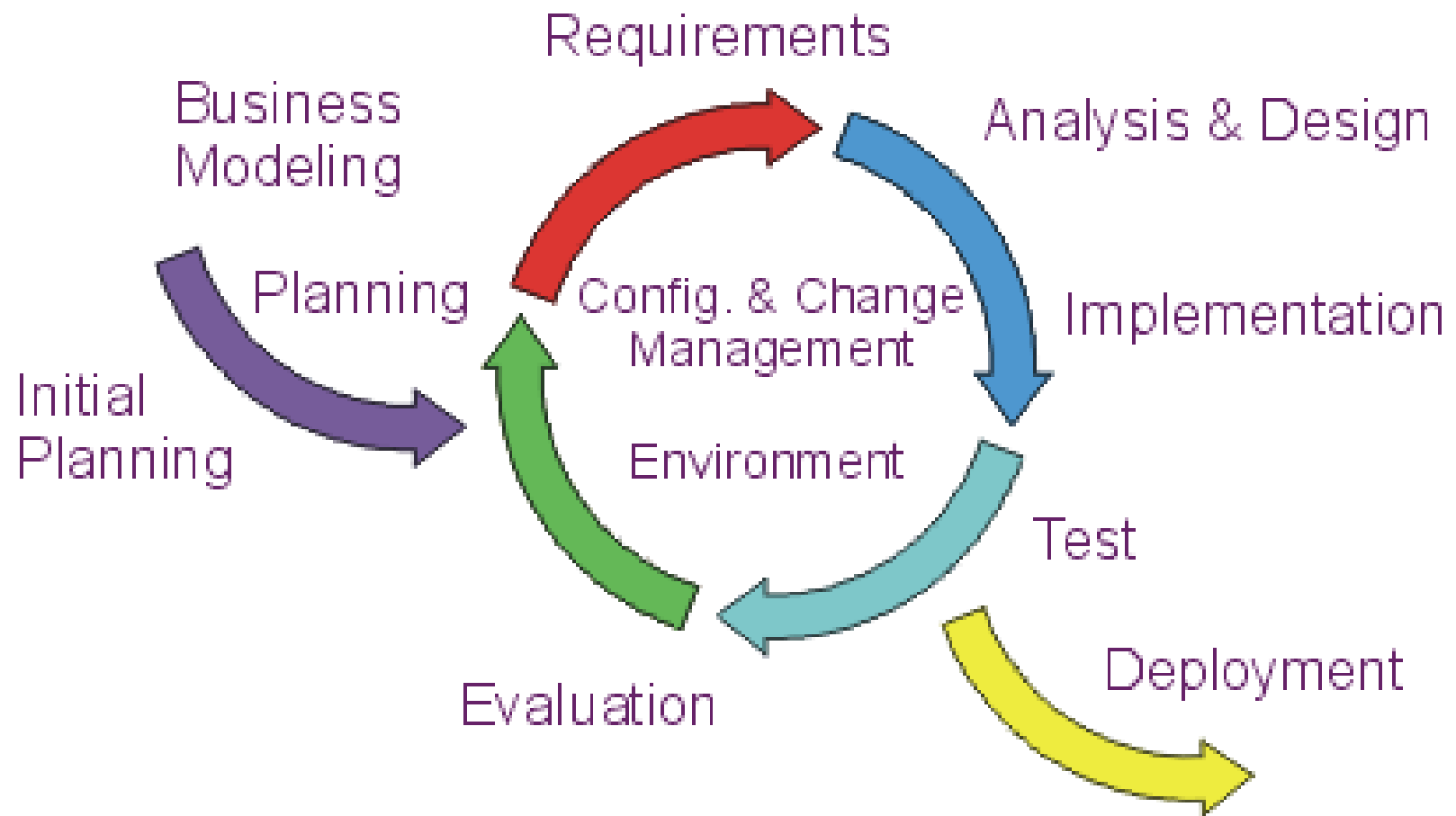
Capability Maturity Model Integration (metodyka ciężka)



Unified Process (coś pośrodku)



Dobra praktyka 1: *Develop Iteratively*

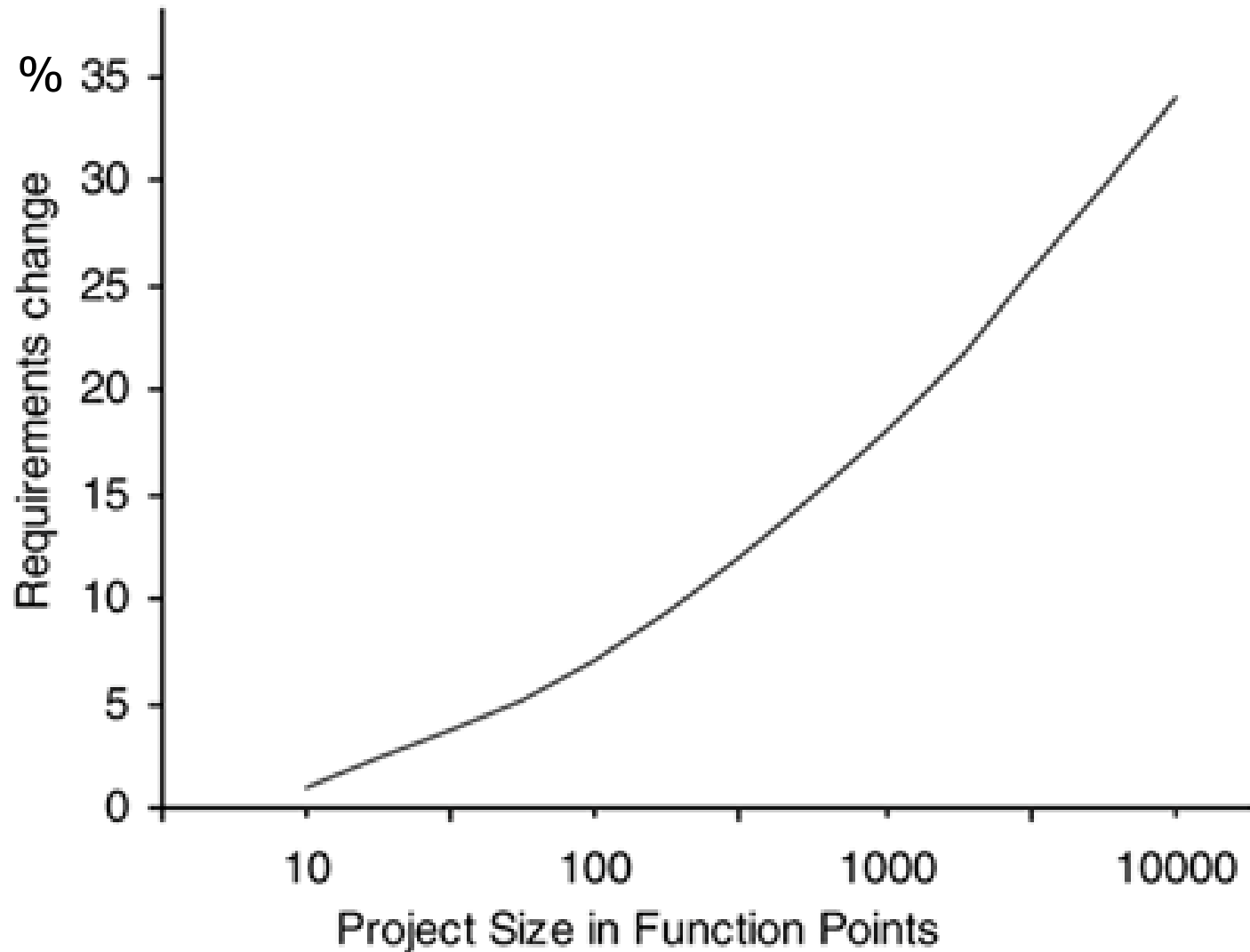




Dobra praktyka 1: *Develop Iteratively* – cd.

- Kluczowa praktyka dla większości współczesnych metodyk deweloperskich, np.:
 - Agile Software Development
 - Unified Process
 - CMMI} Różny poziom formalizmu
- Idea:
 - dewelopment podzielony na serię krótkich podprojektów / iteracji
 - każda iteracja zawiera wszystkie etapy
 - analizę
 - projektowanie
 - kodowanie
 - testowanie
 - wynikiem każdego podprojektu / każdej iteracji jest system
 - uruchamialny / zintegrowany
 - zgodny z projektem / wymaganiami
 - przetestowany

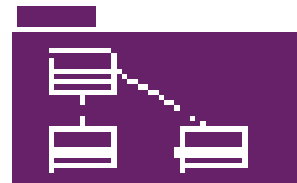
Dobra praktyka 1: *Develop Iteratively* – cd.



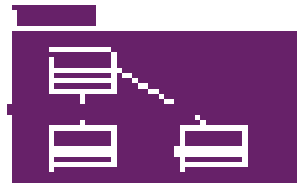
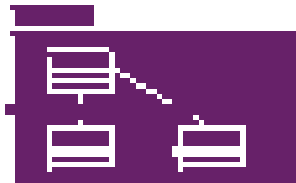
Dobra praktyka 2: *Use Component Architectures*



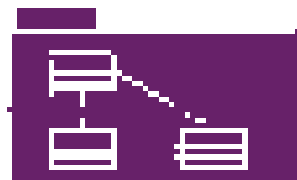
Application-specific



Business-specific

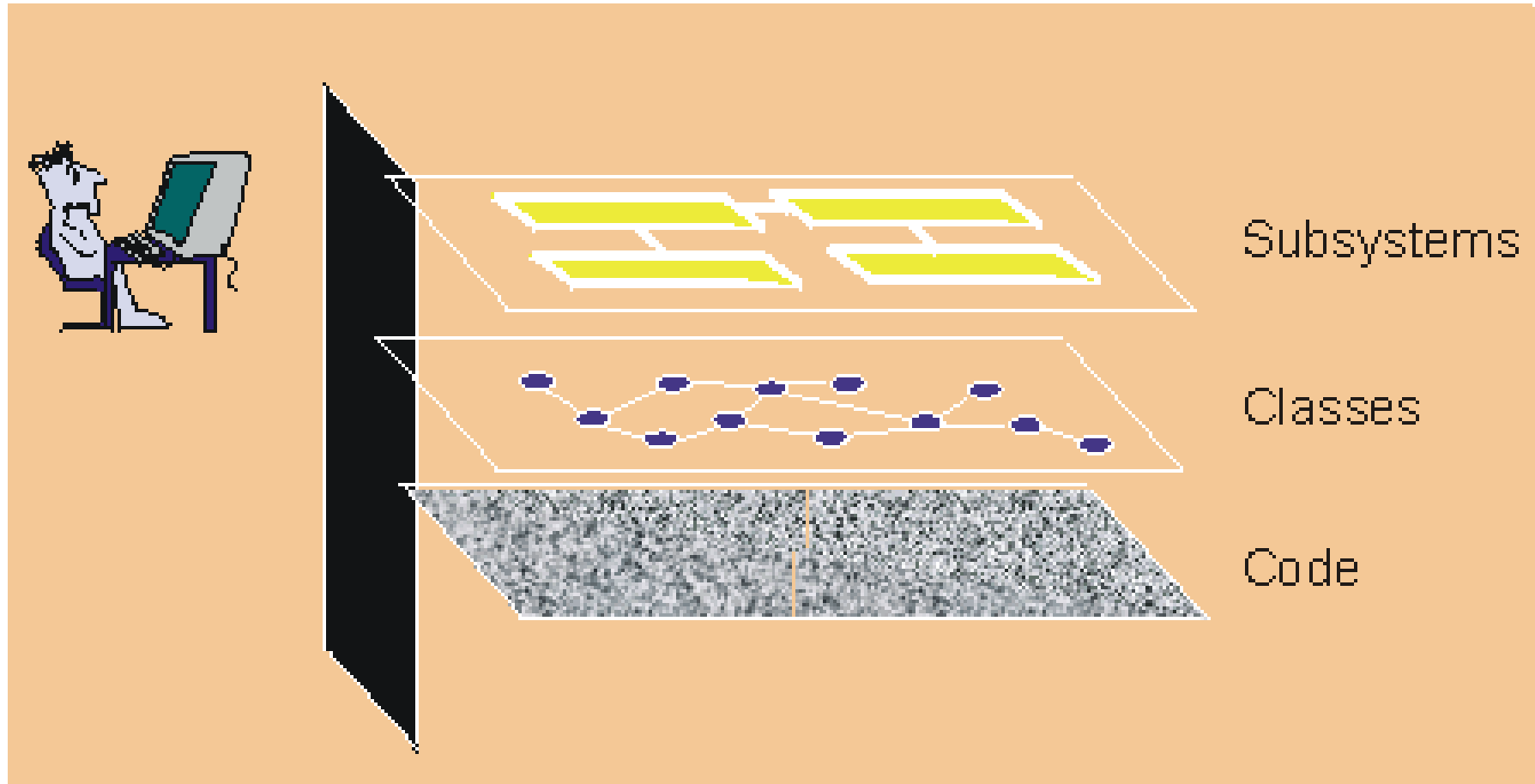


Middleware

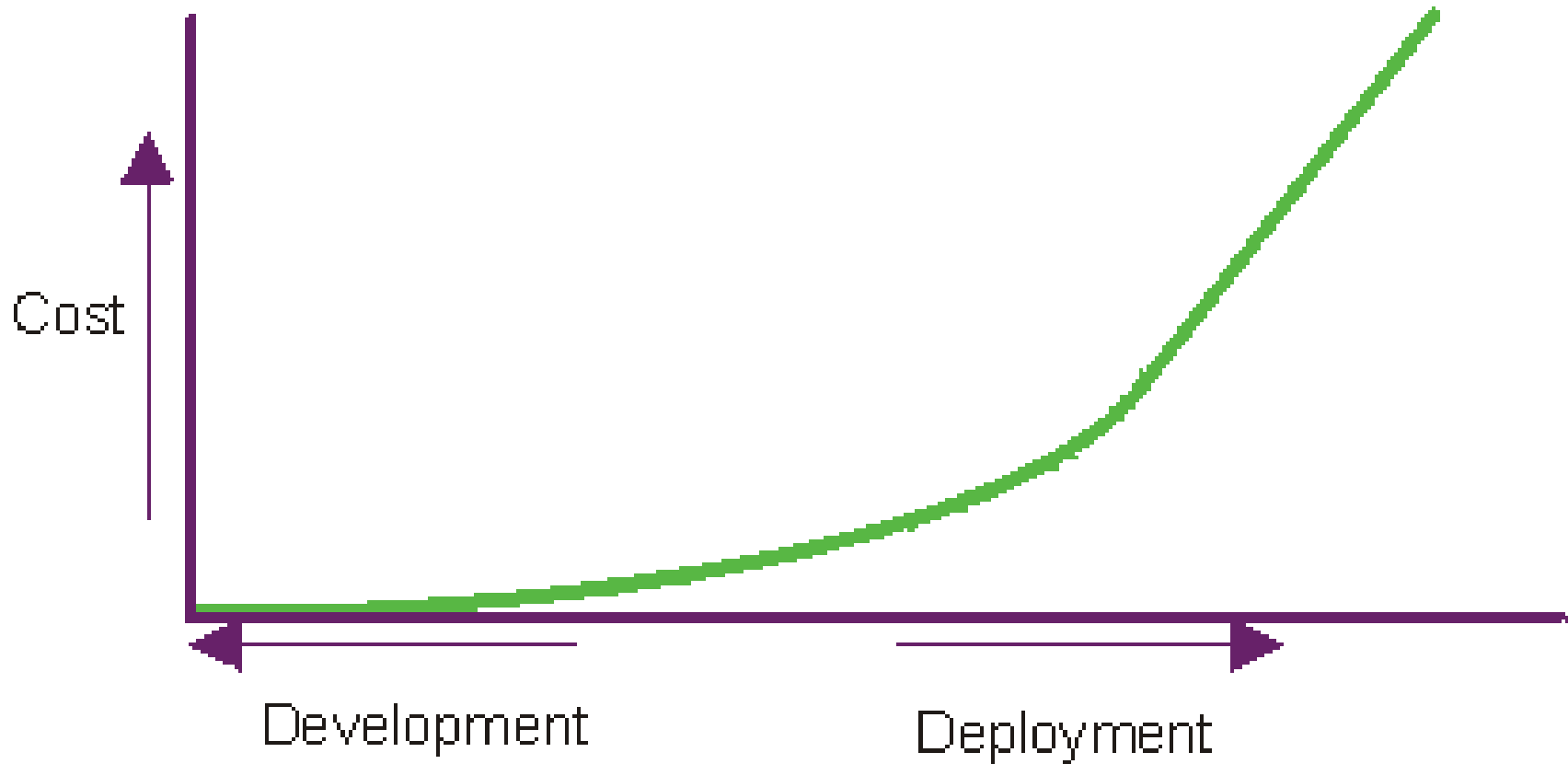


System-software

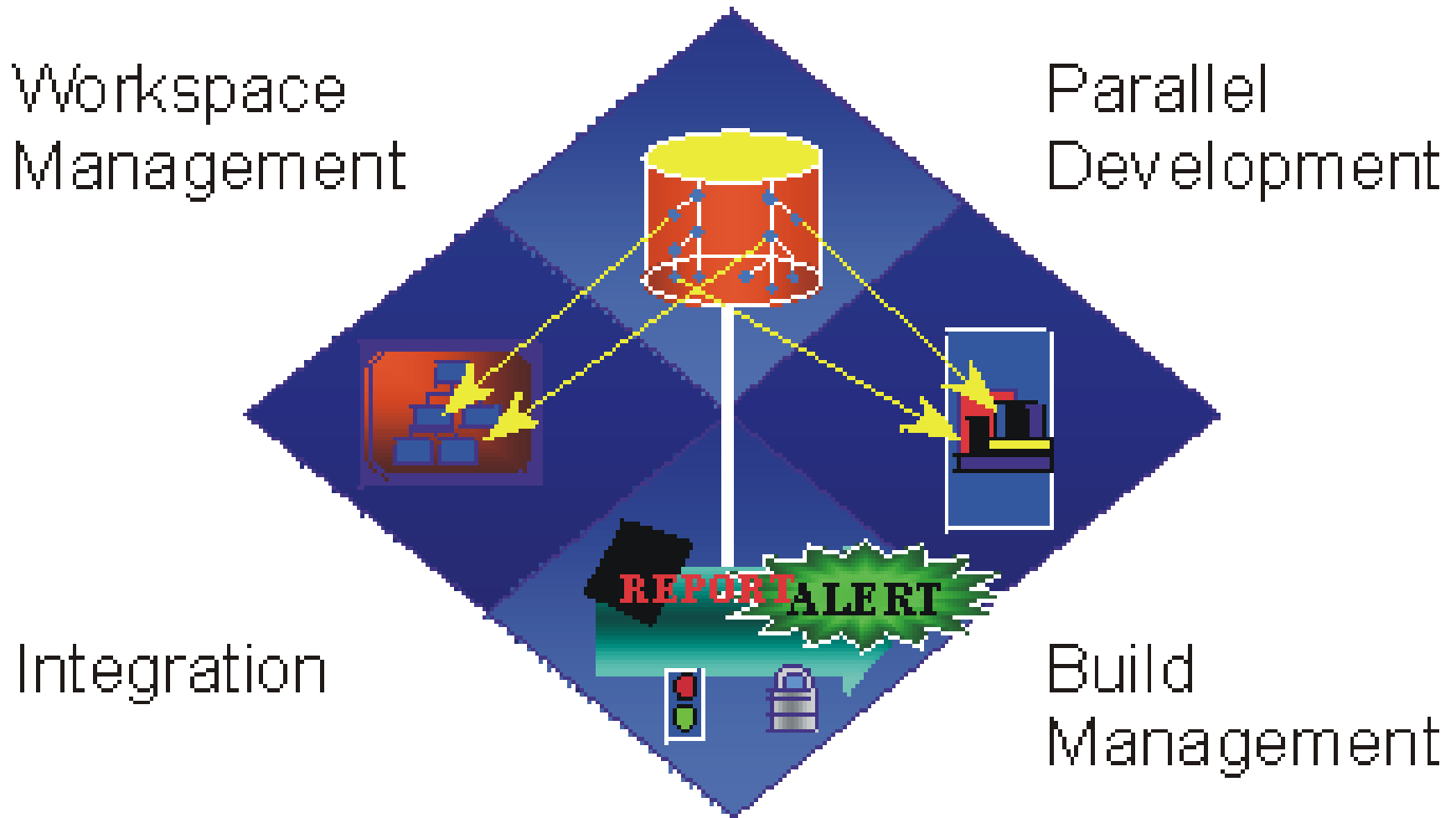
Dobra praktyka 3: *Model Visually*



Dobra praktyka 4: *Continuously Verify Quality*



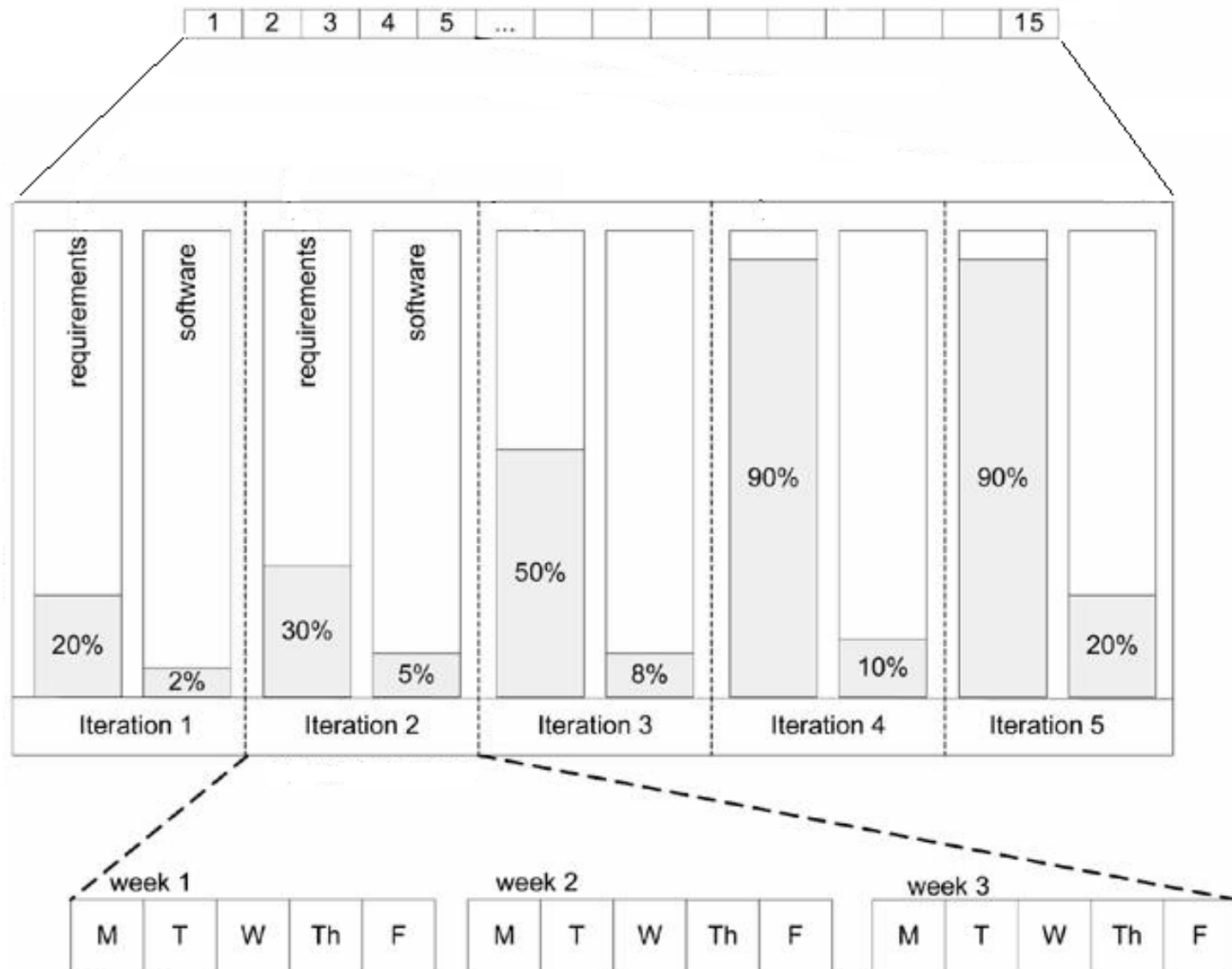
Dobra praktyka 5: *Manage Change*



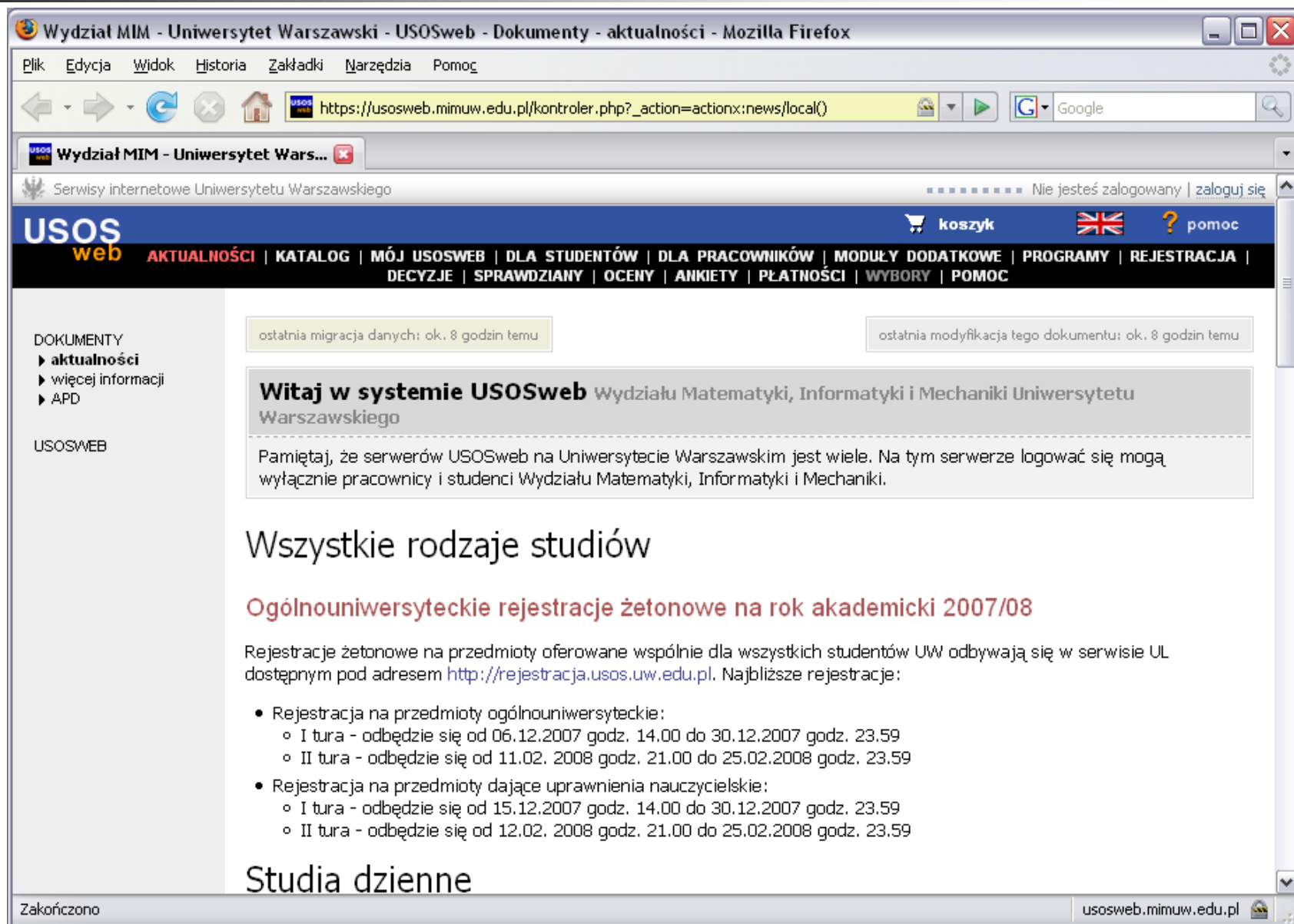
Pytanie

- Jaka powinna być metodyka dla IO+ZPP?
 - 4 osoby, 18 miesięcy

„Agile Unified Process”



Spróbujmy coś zaprojektować...



Wydział MIM - Uniwersytet Warszawski - USOSweb - Dokumenty - aktualności - Mozilla Firefox

Plik Edycja Widok Historia Zakładki Narzędzia Pomoc

https://usosweb.mimuw.edu.pl/kontroler.php?action=actionx:news/local()

Wydział MIM - Uniwersytet Wars...

Serwisy internetowe Uniwersytetu Warszawskiego

USOS web

koszyk

Nie jesteś zalogowany | zaloguj się

AKTUALNOŚCI | KATALOG | MÓJ USOSWEB | DLA STUDENTÓW | DLA PRACOWNIKÓW | MODUŁY DODATKOWE | PROGRAMY | REJESTRACJA | DECYZJE | SPRAWDZIANY | OCENY | ANKIETY | PŁATNOŚCI | WYBORY | POMOC

DOKUMENTY

- ▶ aktualności
- ▶ więcej informacji
- ▶ APD

USOSWEB

ostatnia migracja danych: ok. 8 godzin temu

ostatnia modyfikacja tego dokumentu: ok. 8 godzin temu

Witaj w systemie USOSweb Wydziału Matematyki, Informatyki i Mechaniki Uniwersytetu Warszawskiego

Pamiętaj, że serwerów USOSweb na Uniwersytecie Warszawskim jest wiele. Na tym serwerze logować się mogą wyłącznie pracownicy i studenci Wydziału Matematyki, Informatyki i Mechaniki.

Wszystkie rodzaje studiów

Ogólnouniwersyteckie rejestracje żetonowe na rok akademicki 2007/08

Rejestracje żetonowe na przedmioty oferowane wspólnie dla wszystkich studentów UW odbywają się w serwisie UL dostępnym pod adresem <http://rejestracja.usos.uw.edu.pl>. Najbliższe rejestracje:

- Rejestracja na przedmioty ogólnouniwersyteckie:
 - I tura - odbędzie się od 06.12.2007 godz. 14.00 do 30.12.2007 godz. 23.59
 - II tura - odbędzie się od 11.02. 2008 godz. 21.00 do 25.02.2008 godz. 23.59
- Rejestracja na przedmioty dające uprawnienia nauczycielskie:
 - I tura - odbędzie się od 15.12.2007 godz. 14.00 do 30.12.2007 godz. 23.59
 - II tura - odbędzie się od 12.02. 2008 godz. 21.00 do 25.02.2008 godz. 23.59

Studia dzienne

Zakończono

usosweb.mimuw.edu.pl



- Cel
 - jednolity Uniwersytecki System Obsługi Studiów (USOS)
- Zainteresowani
 - 17 polskich uniwersytetów
- Przyjęte podejście
 - Wykonanie samodzielne, Wydział MIM UW
- Rozważane alternatywy
 - Rozwiązanie gotowe
 - brak na rynku adekwatnego rozwiązania (są częściowe)
 - Wykonanie przez firmę komercyjną
 - zbyt wysoki koszt

- Główne wymagania funkcjonalne
 - *students' and teachers' personal data*
 - *study programs and requirements of degree certificates*
 - *course catalog*
 - *course registration*
 - *class schedules*
 - *dormitories*
 - *tuition and financial aid (what we pay students)*
 - *paid courses (what students pay us)*
 - *issuing documents, gathering statistics, producing reports, etc.*



- Główne wymagania niefunkcjonalne
 - *university-wide*
 - *flexible (configurable, adaptable to programs and procedures of participating universities, supporting standard and non-standard solutions)*
 - *open (easily adaptable to changes)*
 - *compatible with ECTS standards*
 - *user-friendly (also for not IT professionals)*
 - *support easy access for students, faculty members, and administration officers*
 - *ensure security of data and lower administration expenses*



■ Architektura

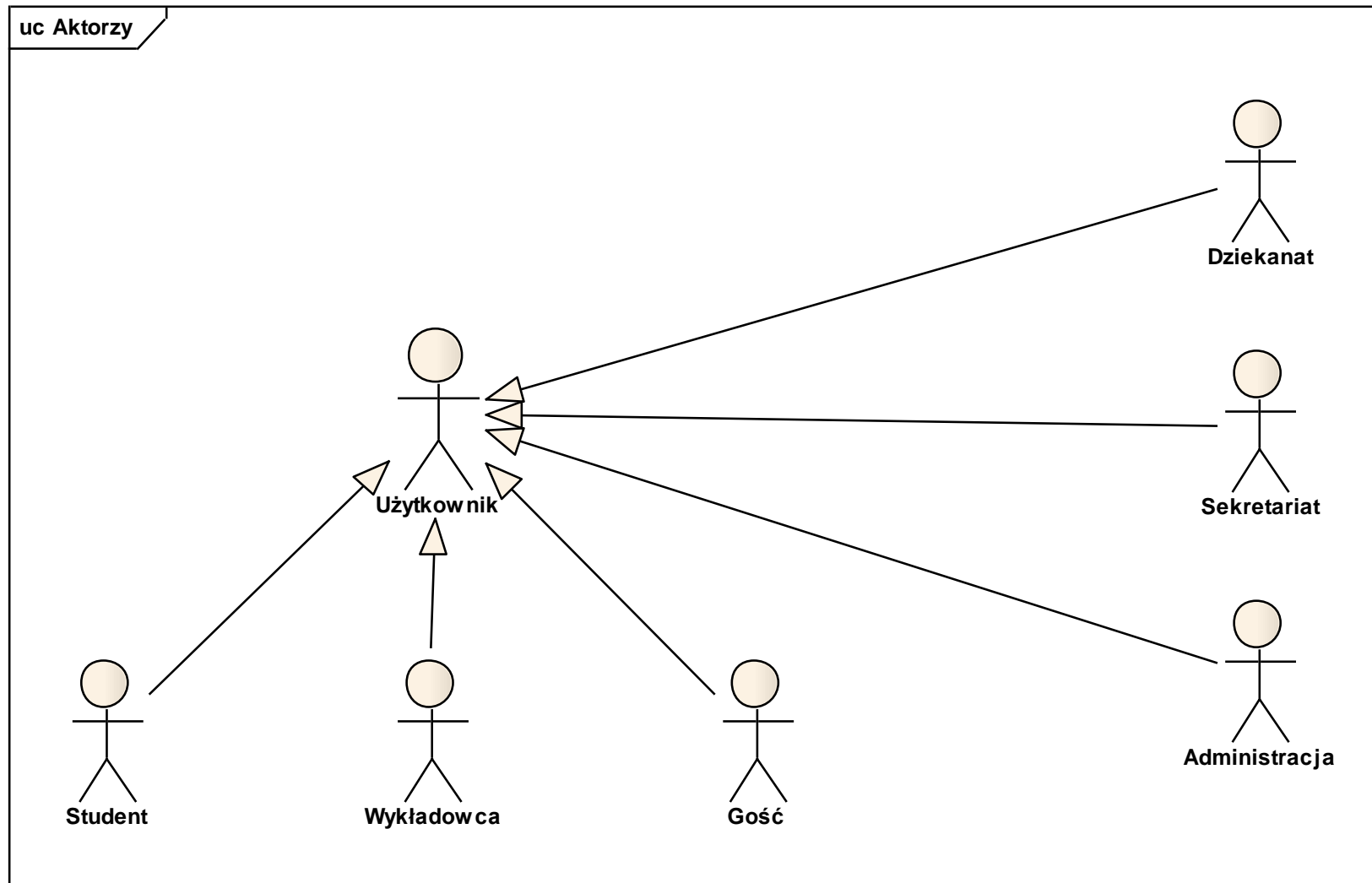
- *main database accessed on a strictly limited bases by administration staff — in Oracle technology (Oracle 8.0.6, Oracle Designer/2000, Oracle Developer, Oracle Forms, Oracle Reports)*
- *few Internet databases accessed through web browsers by academic teachers, students — in open-source technology (MySQL, PHP, Apache)*

Mniej więcej tak system USOS był prezentowany na konferencji SAIAC'2002, Tartu, November 18–20

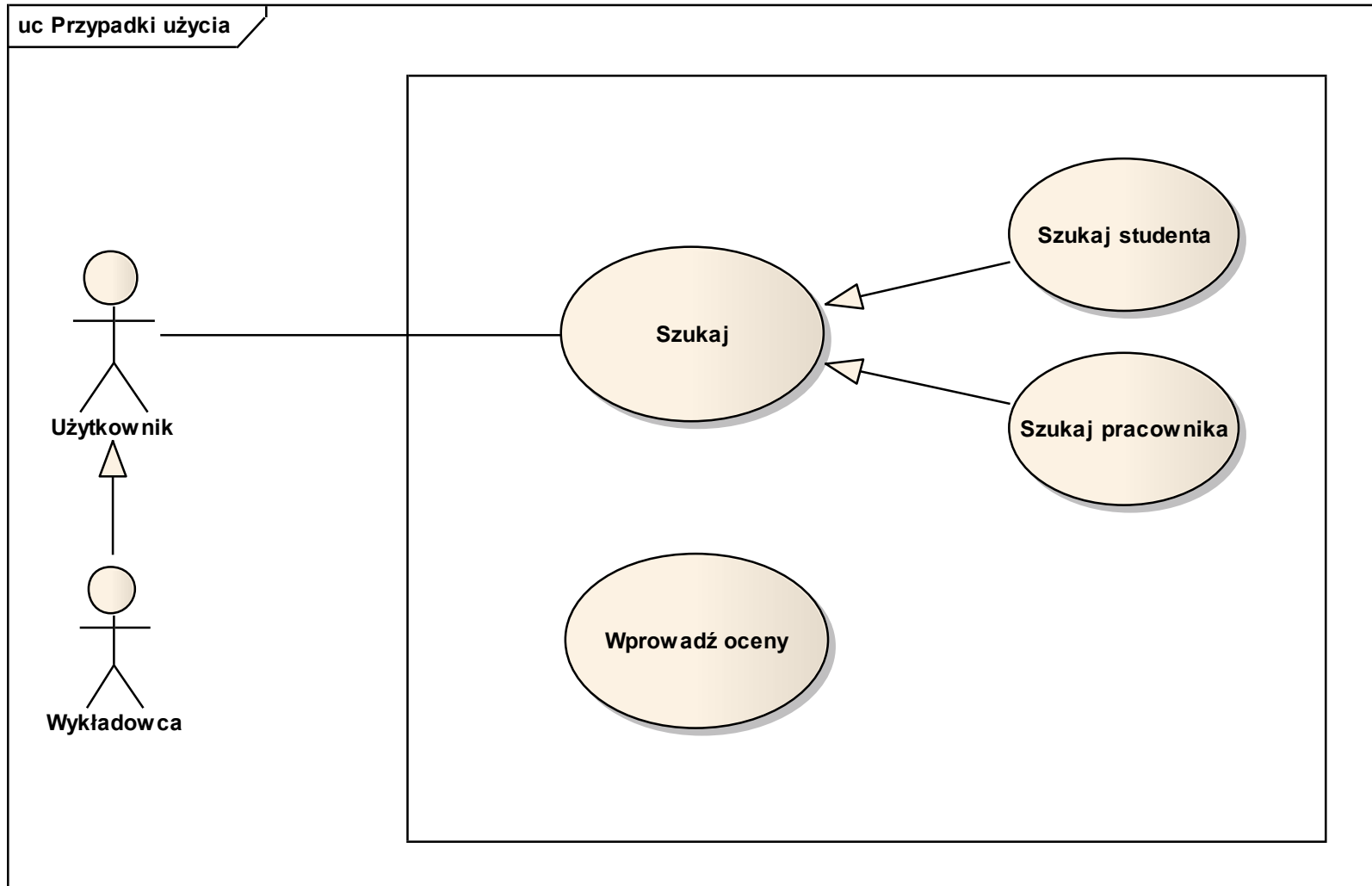


Wymagania funkcjonalne

Wymagania funkcjonalne – aktorzy

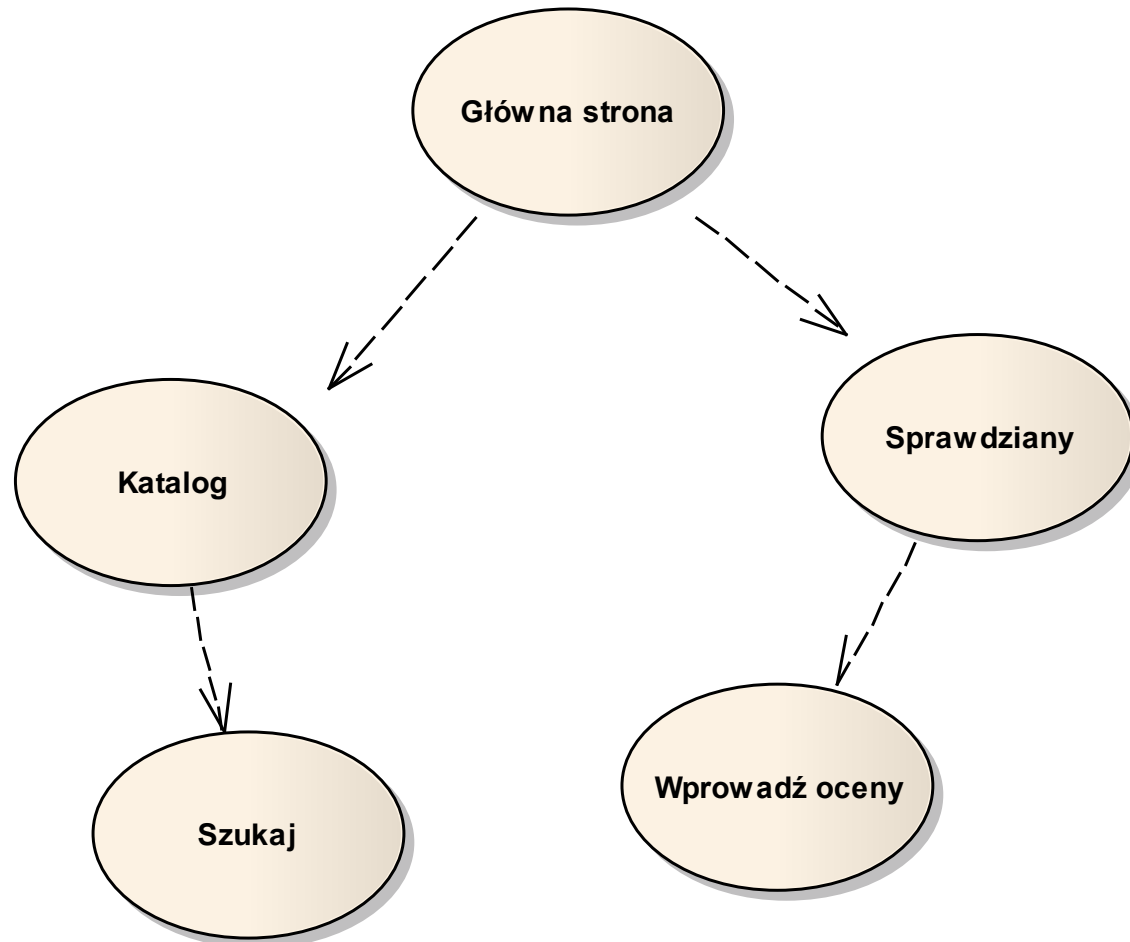


Wymagania funkcjonalne – przypadki użycia



Wymagania funkcjonalne – interfejs użytkownika

uc Interfejs użytkownika





Wymagania niefunkcjonalne

Wymagania niefunkcjonalne

custom Wymagania nief...

NFR01
Pojemność - (...)

NFR02
Wydajność - (...)

NFR03
Skalowalność -
(...)

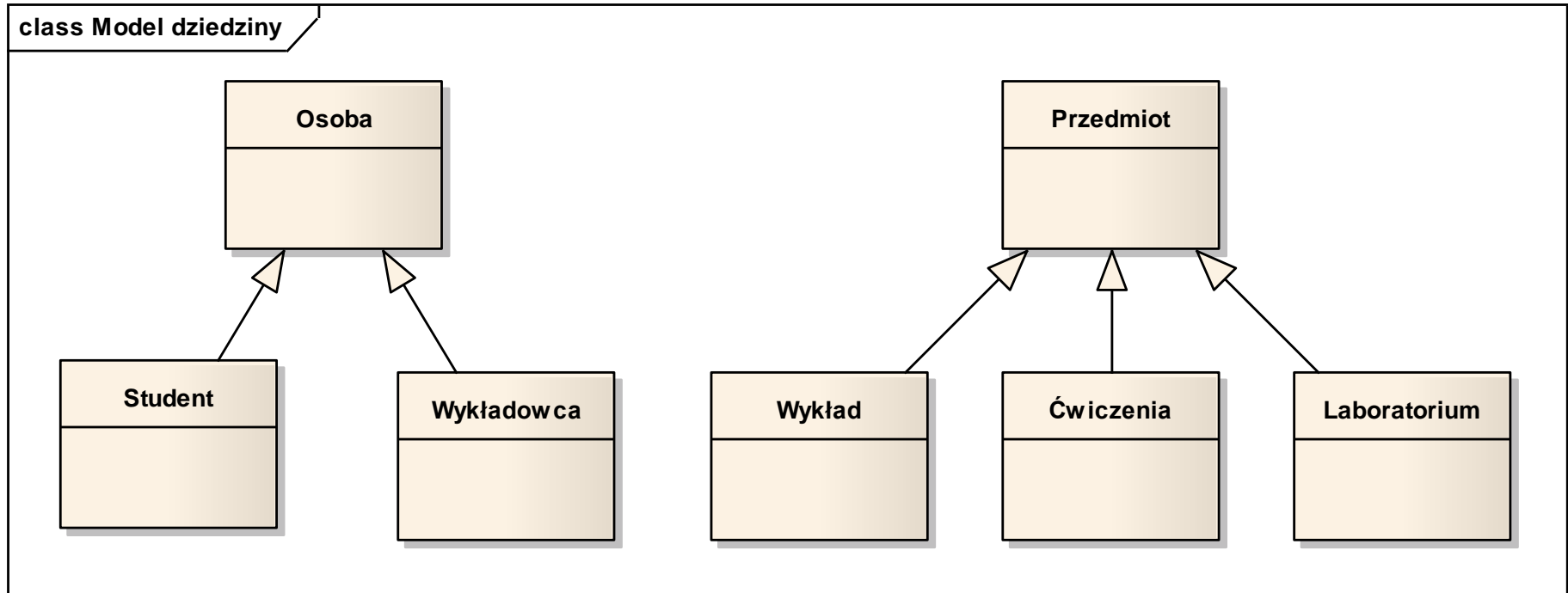
NFR04
Bezpieczeństwo -
(...)

NFR05
Ergonomia - (...)



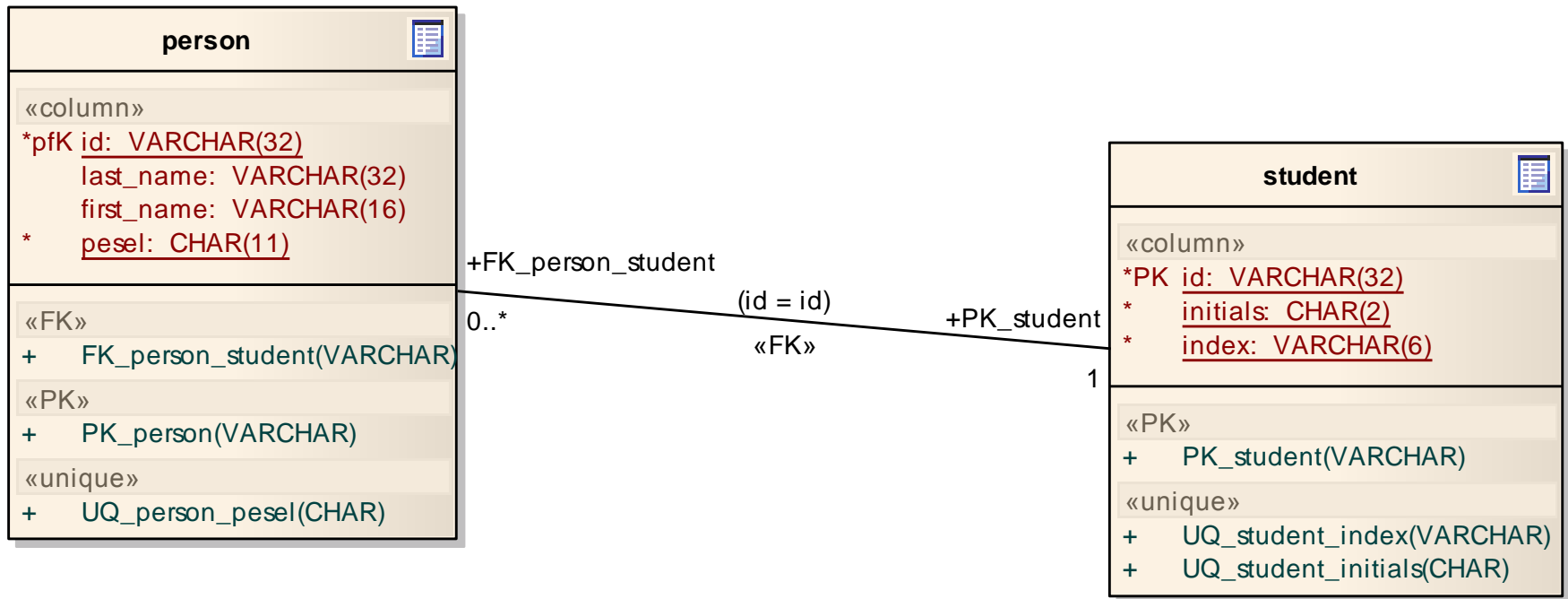
Model dziedziny / klas / danych / ...

Model dziedziny



Model danych

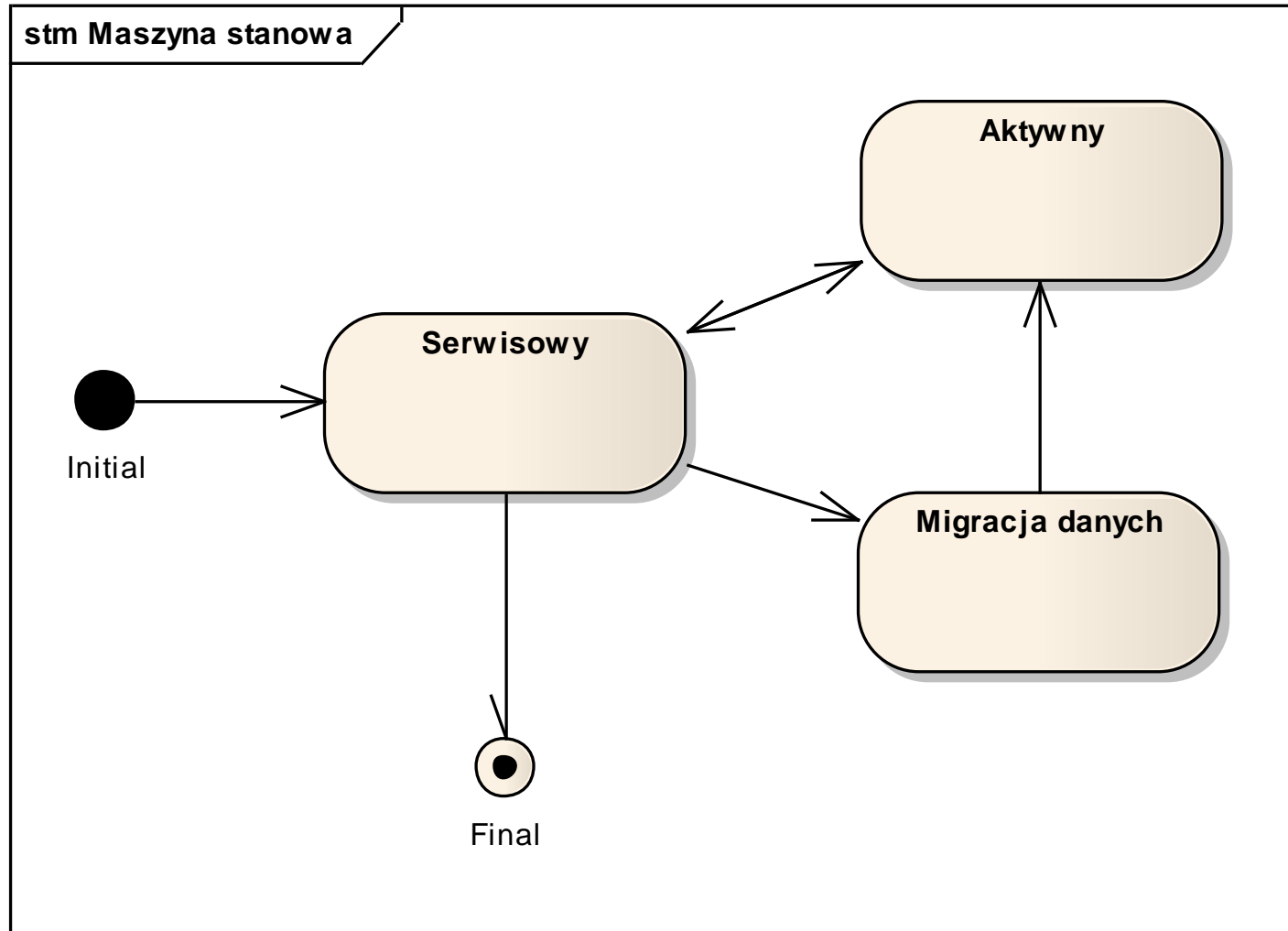
class Model danych



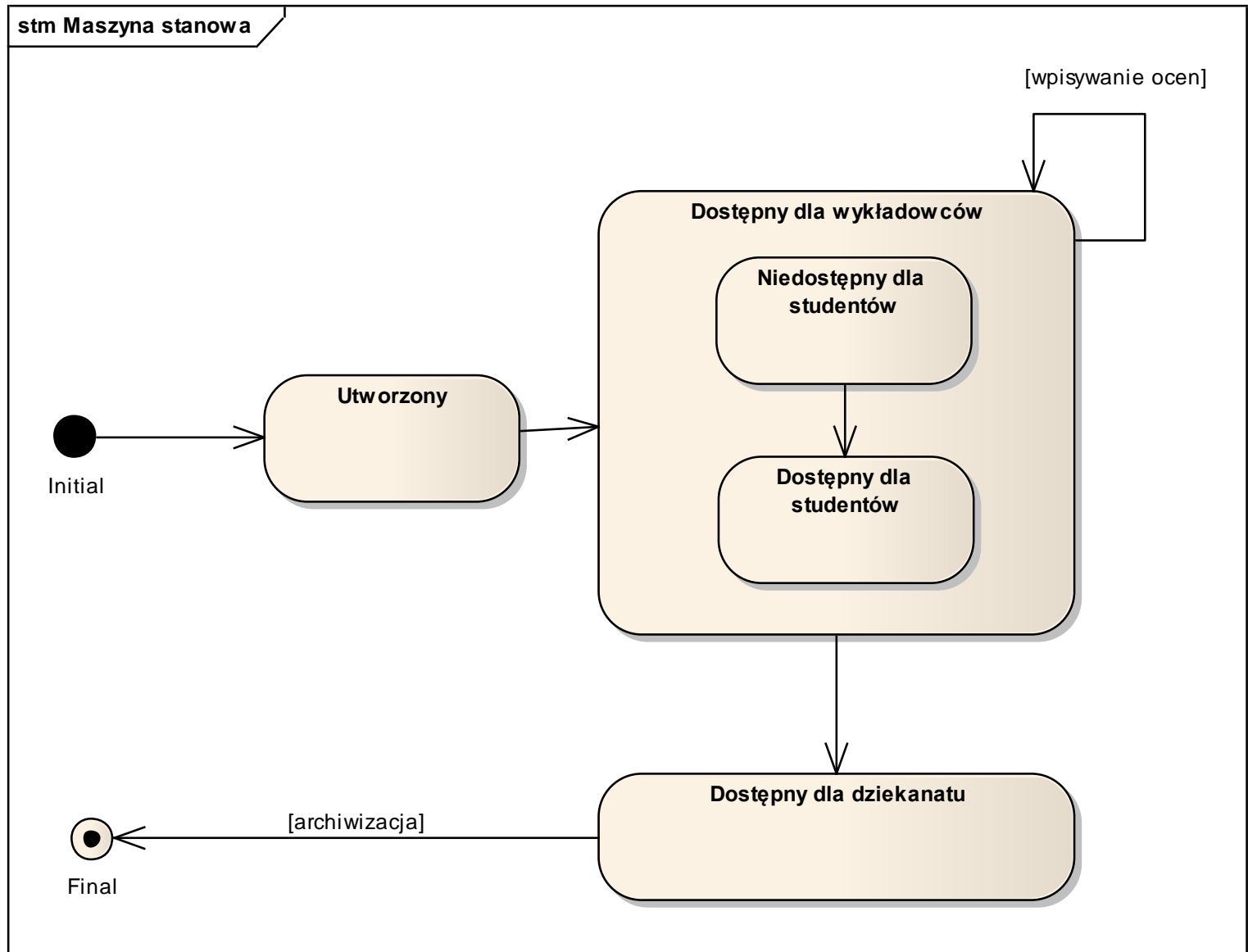


Maszyna stanowa

Maszyna stanowa – system

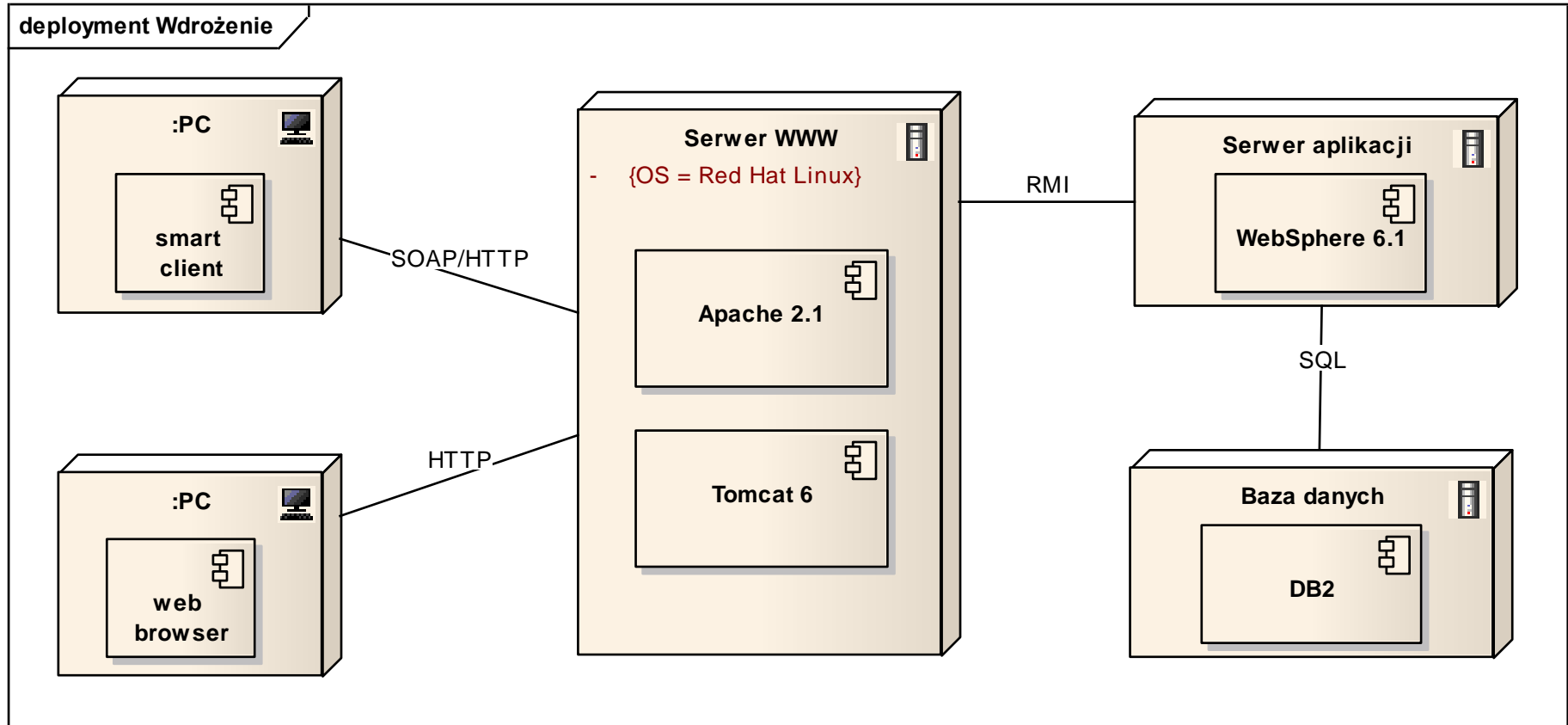


Maszyna stanowa – protokół egzaminacyjny

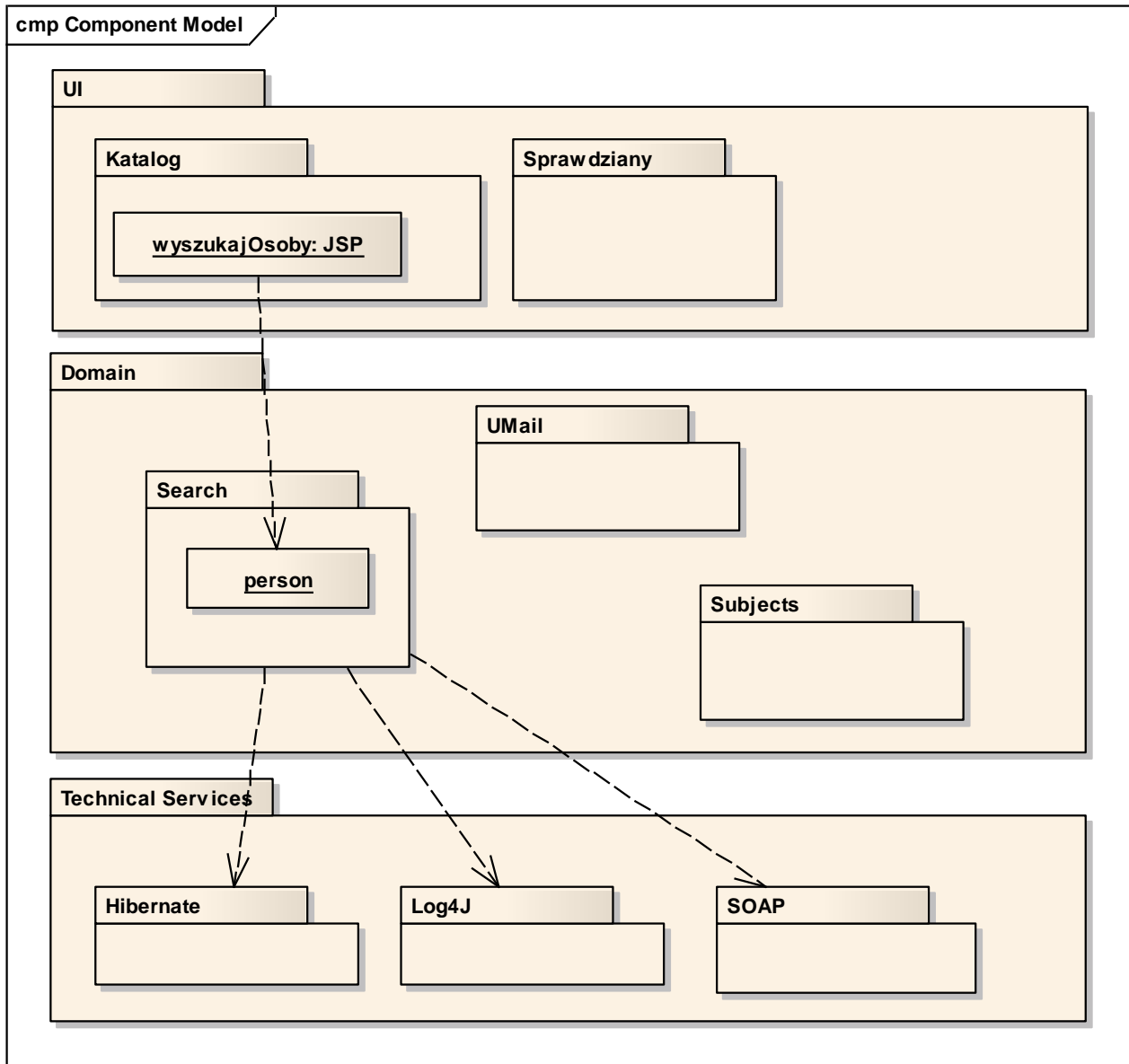




Architektura – wdrożenie, infrastruktura

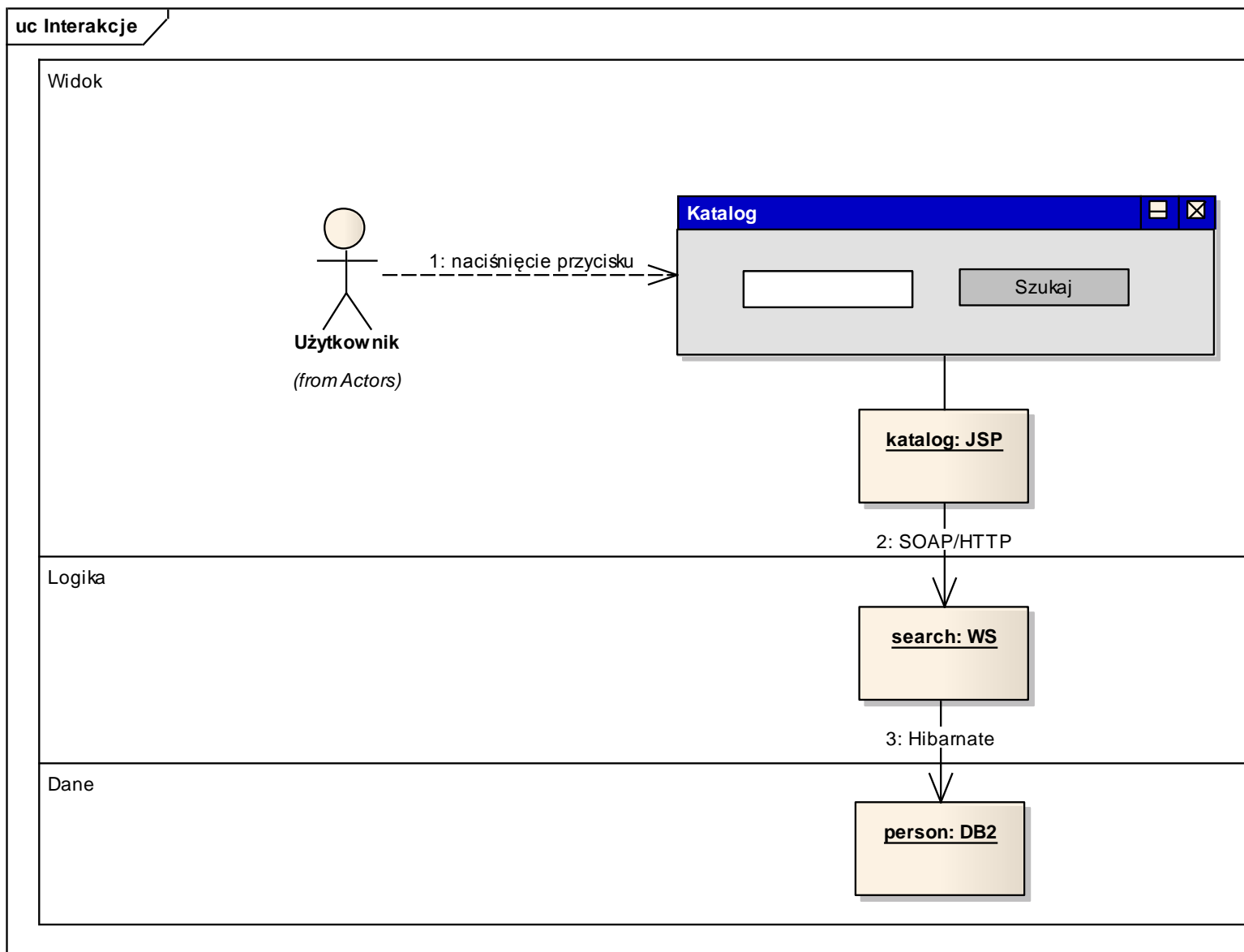


Architektura – warstwy, moduły / pakiety

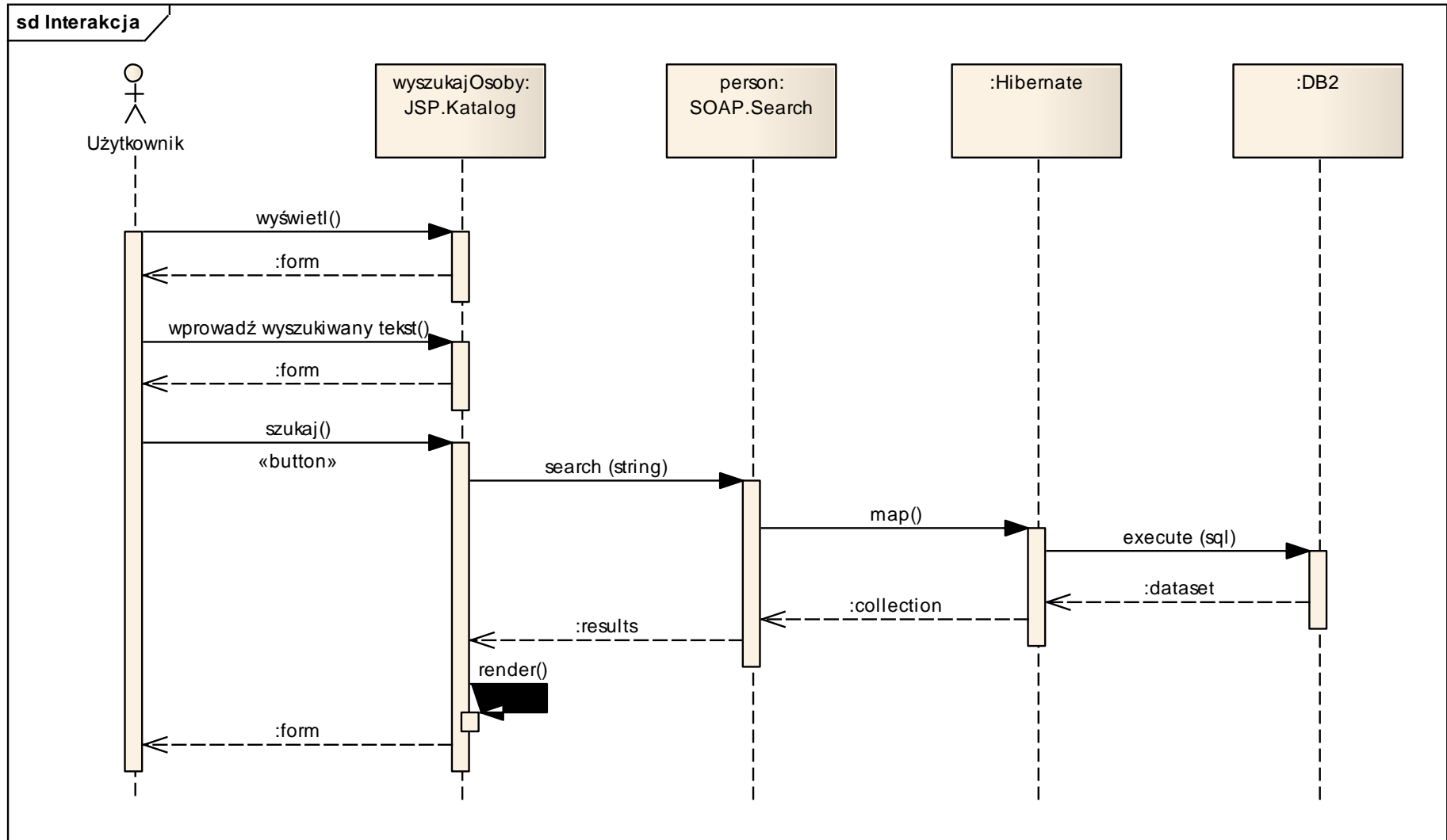




Interakcje



Interakcje



Podsumowanie

- Przeprowadzona iteracja
 - Określiliśmy wizję
 - Zrozumieliśmy główne wymagania
 - Podjęliśmy najważniejsze decyzje architektoniczne
- Pozostaje:
 - Zakodować
 - Przetestować
 - Uruchomić

Podsumowanie – Wizja

<Project Name> Vision

1. Introduction
2. Positioning

2.1 Problem Statement

[Provide a statement summarizing the problem being solved by this project. The following format may be used:]

The problem of	<i>[describe the problem]</i>
affects	<i>[the stakeholders affected by the problem]</i>
the impact of which is	<i>[what is the impact of the problem?]</i>
a successful solution would be	<i>[list some key benefits of a successful solution]</i>

2.2 Product Position Statement

[Provide an overall statement summarizing, at the highest level, the unique position the product intends to fill in the marketplace. The following format may be used:]

For	<i>[target customer]</i>
Who	<i>[statement of the need or opportunity]</i>
The (product name)	<i>is a [product category]</i>
That	<i>[statement of key benefit; that is, the compelling reason to buy]</i>
Unlike	<i>[primary competitive alternative]</i>
Our product	<i>[statement of primary differentiation]</i>

[A product position statement communicates the intent of the application and the importance of the project to all concerned personnel.]

3. Stakeholder Descriptions

3.1 Stakeholder Summary

Name	Description	Responsibilities
<i>[Name the stakeholder type.]</i>	<i>[Briefly describe the stakeholder.]</i>	<i>[Summarize the stakeholder's key responsibilities with regard to the system being developed; that is, their interest as a stakeholder. For example, this stakeholder: ensures that the system will be maintained.]</i>

Podsumowanie – Przypadki użycia

Buy a Product

Main Success Scenario:

1. Customer browses catalog and selects items to buy
2. Customer goes to check out
3. Customer fills in shipping information (address; next-day or 3-day delivery)
4. System presents full pricing information, including shipping
5. Customer fills in credit card information
6. System authorizes purchase
7. System confirms sale immediately
8. System sends confirming e-mail to customer

Extensions:

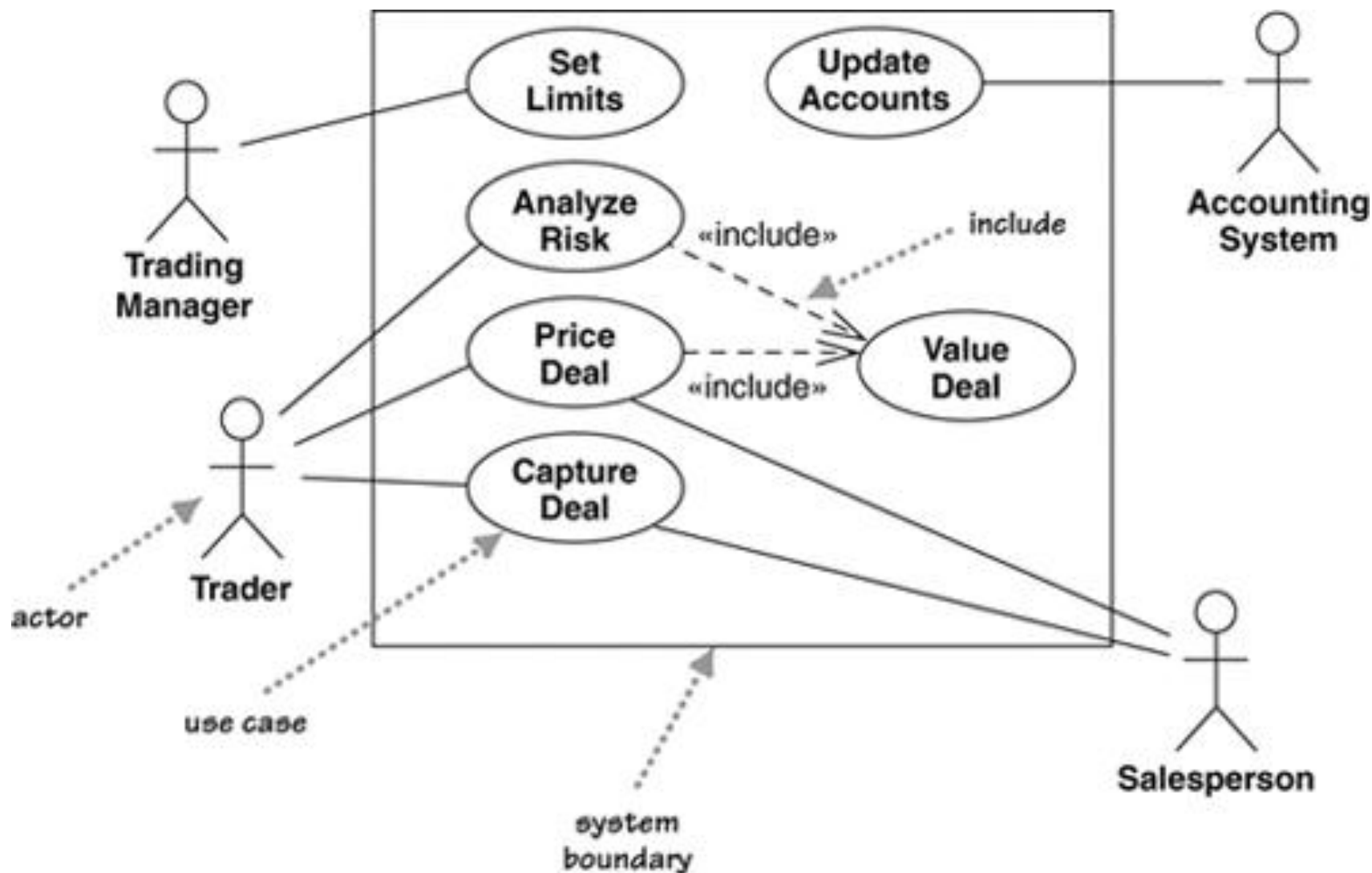
3a: Customer is regular customer

- .1: System displays current shipping, pricing, and billing information
- .2: Customer may accept or override these defaults, returns to MSS at step 6

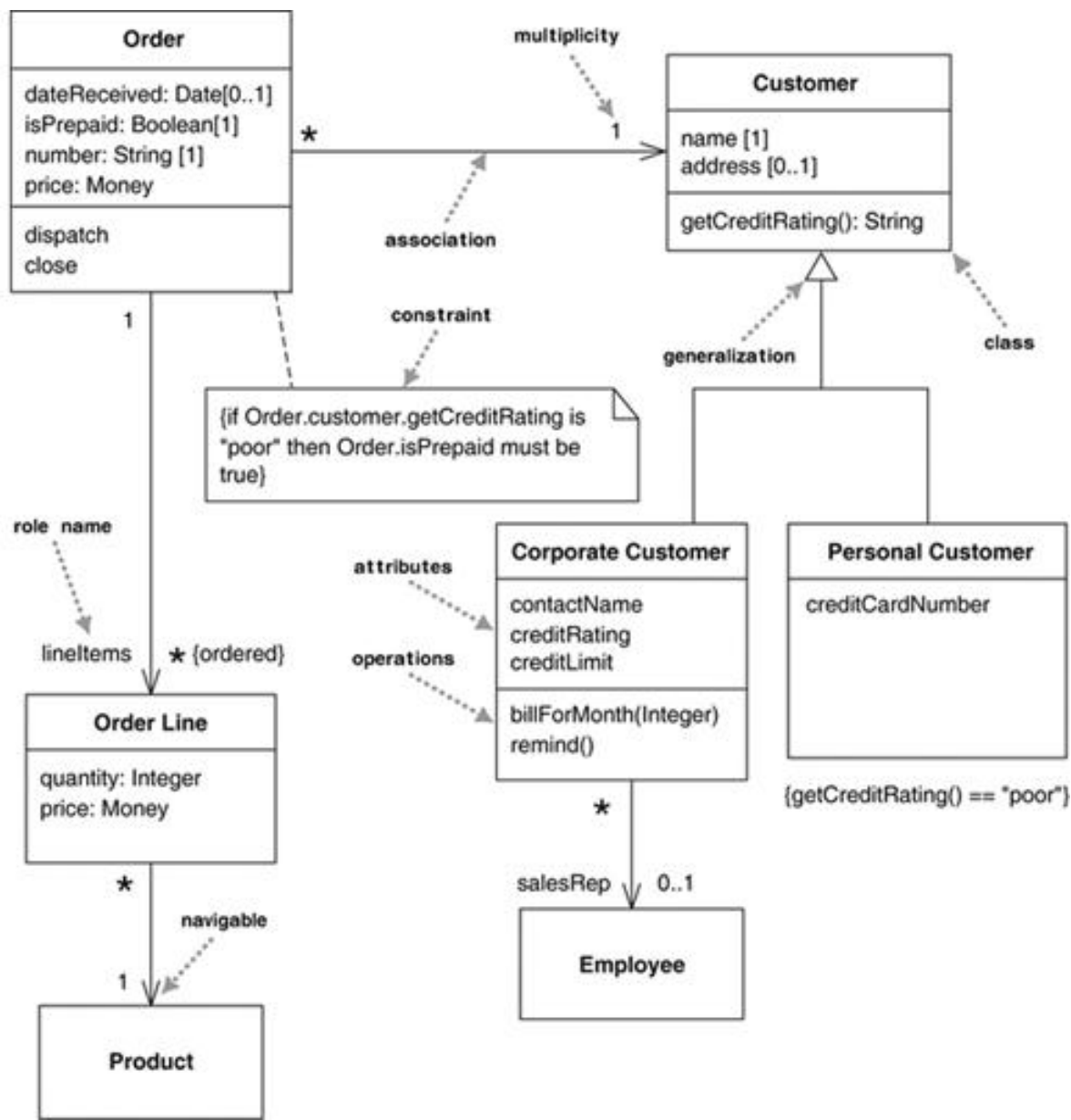
6a: System fails to authorize credit purchase

- .1: Customer may reenter credit card information or may cancel

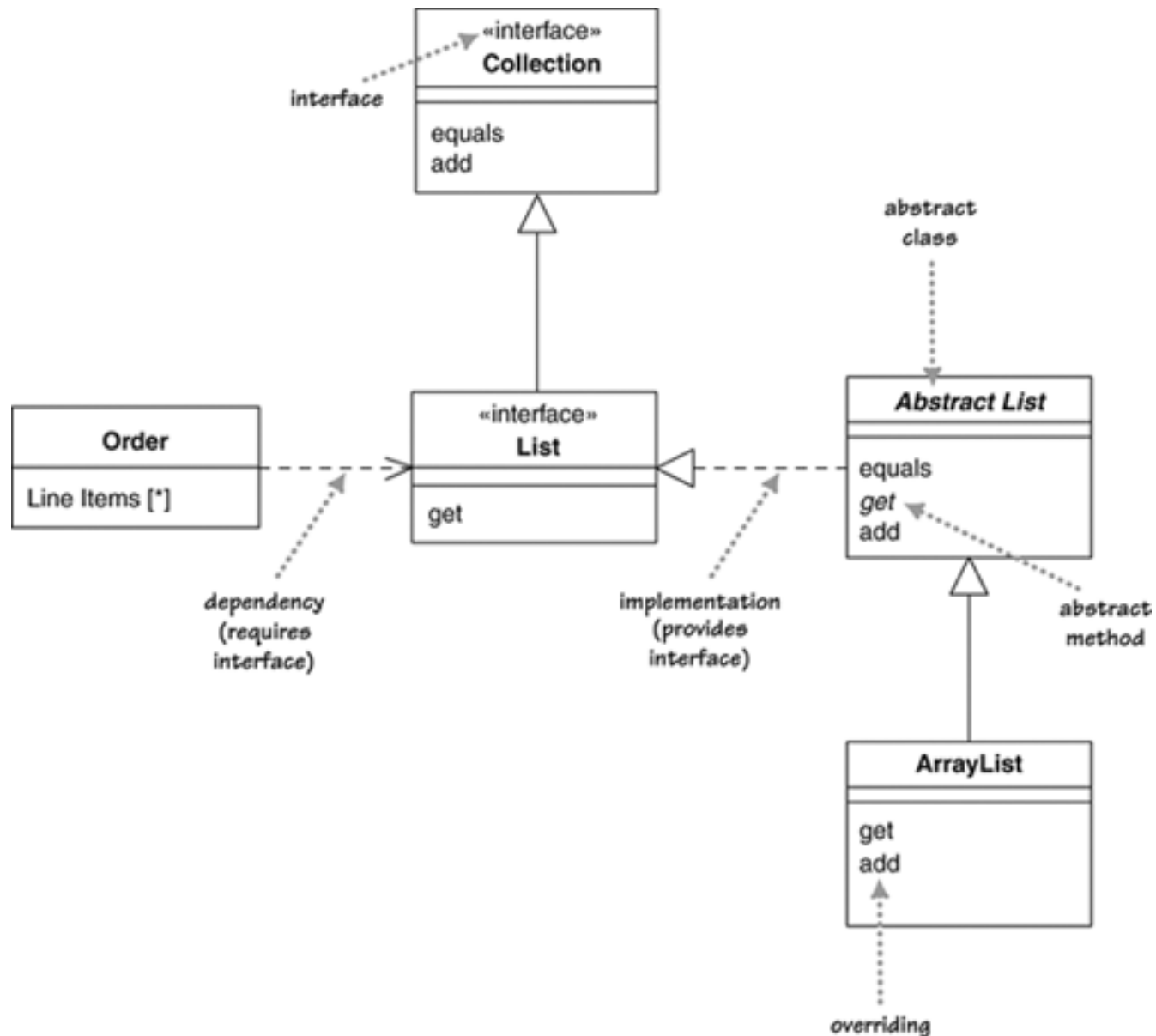
Podsumowanie – Przypadki użycia



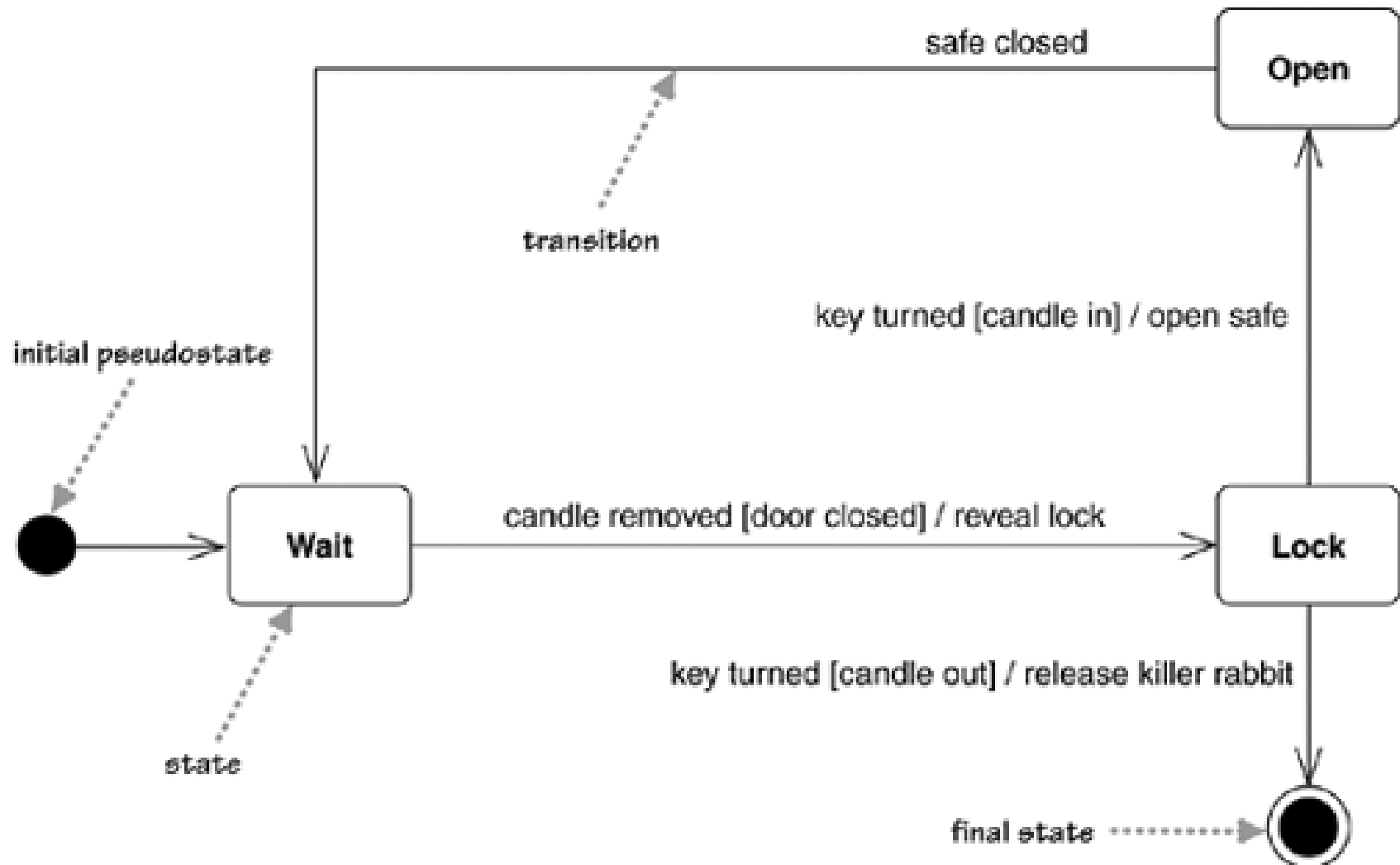
Podsumowanie – Dziedzina / klasy / interfejsy / dane



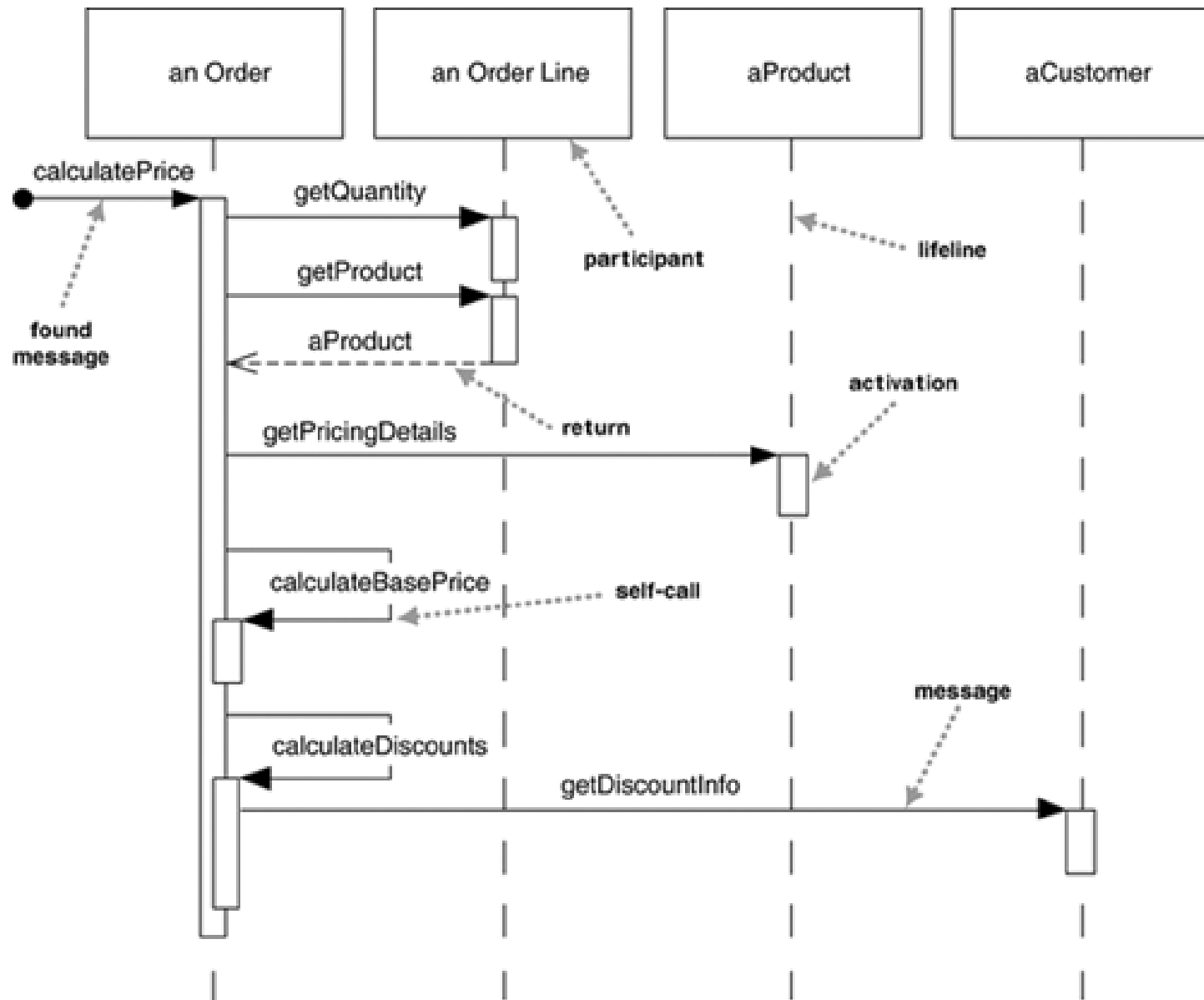
Podsumowanie – Dziedzina / klasy / interfejsy / dane



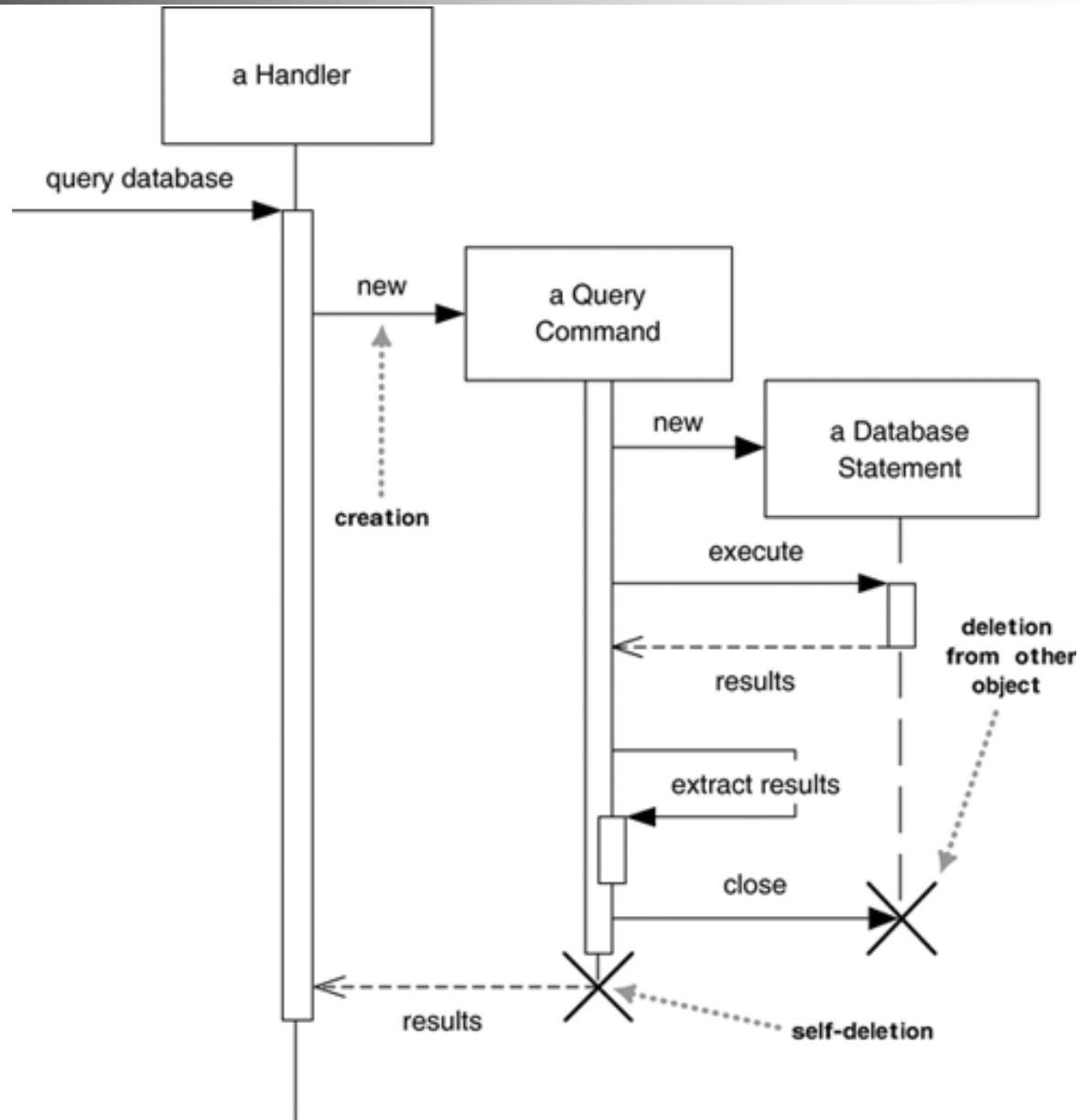
Podsumowanie – Maszyna stanowa



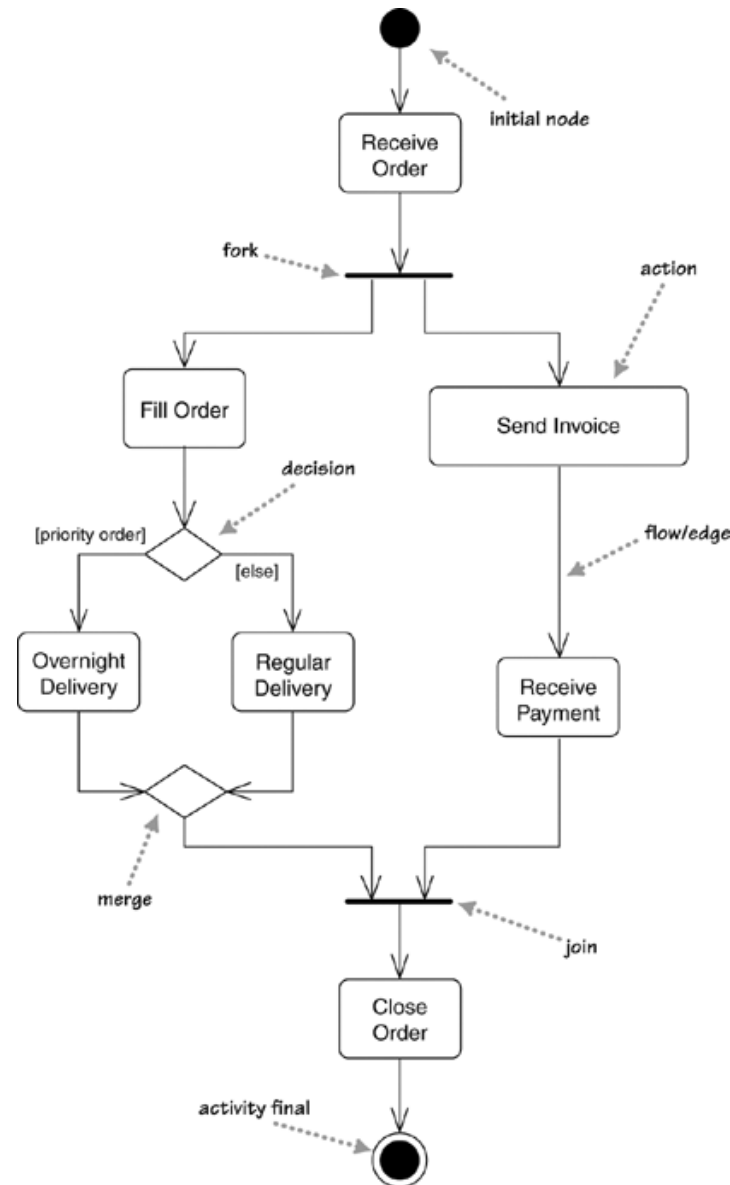
Podsumowanie – Interakcje



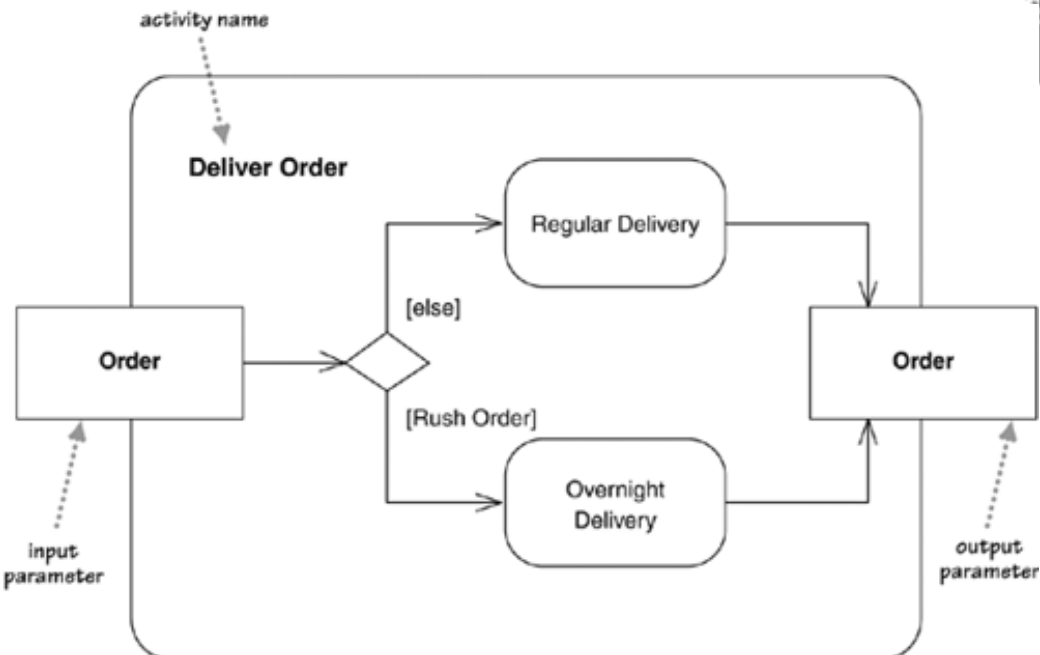
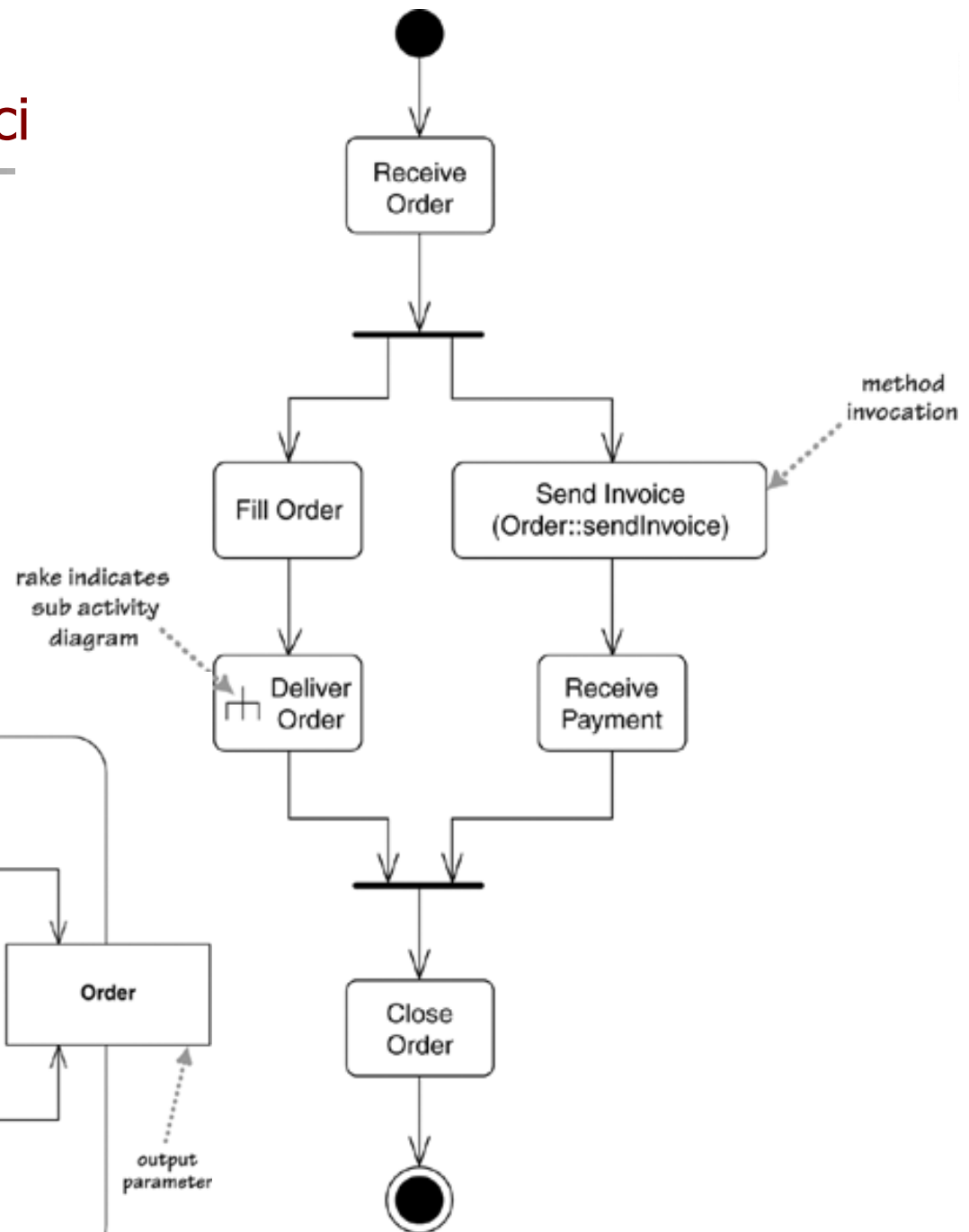
Podsumowanie – Interakcje



Podsumowanie – Aktywności



Podsumowanie – Aktywności



Podsumowanie – Komponenty, wdrożenie

