# Wprowadzenie do inżynierii oprogramowania.

**Wykładowca**

**Dr inż. Zofia Kruczkiewicz**

# Struktura wykładu

I. Zasady zaliczenia kursu.
II. Wprowadzenie do inżynierii oprogramowania(IO)
III. Podstawowe pojęcia: specyfikacja, analiza, projekt, implementacja, test, wdrożenie, pielęgnacja.
IV. Rola IO w inżynierii biomedycznej.

# Literatura podstawowa (LPU) – UML


1. G. Booch, J. Rumbaugh, I. Jacobson, UML przewodnik użytkownika, Seria Inżynieria oprogramowania, WNT, 2001, 2002. .
2. M. Fowler, UML w kropelce, Wersja 2.0, LTP, Warszawa, 2005.

# Literatura uzupełniająca (LPW) – Wzorce projektowe


1. E. Gamma, R. Helm, R. Johnson, J. Vlissides, Wzorce projektowe, Elementy oprogramowania obiektowego wielokrotnego użytku, WNT, Warszawa, 2005.
2. Shalloway A.,Trott James R., Projektowanie zorientowane obiektowo. Wzorce projektowe. Gliwice, Helion, 2005

# Literatura uzupełniająca (LU) – Inżynieria oprogramowania

1. Roger S. Pressman, Praktyczne podejście do oprogramowania, WNT, 2004
2. Stephen H. Kan, Metryki i modele w inżynierii jakości oprogramowania, Mikom, 2006
3. Jacobson, Booch, Rumbaung, The Unified Software Development Process, Addison Wesley, 1999

# Strony internetowe

- **Karta przedmiotu INP002017:**

**Strona główna/Studenci/Karty przedmiotów**

**Inżynieria Biomedyczna - 1 stopień - Informatyka Medyczna**

- **Strona wykładowcy**

**http://zofia.kruczkiewicz.staff.iiar.pwr.wroc.pl/index.php?id=INP002017**

# I. Zasady zaliczenia kursu – na podstawie karty przedmiotu

# 1. Przedmiotowe efekty kształcenia

## Z zakresu wiedzy:

- **PEK_W01 Ma** uporządkowaną, podbudowaną teoretycznie wiedzę ogólną obejmującą kluczowe zagadnienia z zakresu inżynierii oprogramowania;

## Z zakresu umiejętności:

- **PEK_U01** Potrafi specyfikować wymagania w projekcie programistyczny;
- **PEK_U02** Potrafi zaprojektować system informatyczny w języku UML;
- **PEK_U03** Potrafi korzystać ze współczesnych technik tworzenie systemów informatycznych;
- **PEK_U04** Potrafi sprawdzić poprawność oprogramowania;

## Z zakresu kompetencji społecznych:

- **PEK_K01** potrafi współdziałać i współpracować w grupie przyjmując w niej różne role
- **PEK_K02** potrafi odpowiednio określić priorytety służące realizacji określonego przez siebie lub innych zadania

# 2. Ocena osiągnięcia przedmiotowych efektów kształcenia

| **Oceny** (F – formująca (w trakcie semestru), | Numer efektu kształcenia | Sposób oceny osiągnięcia efektu kształcenia |
|---|---|---|
| F1 | PEK_W01<br>PEK_U01, PEK_U02 | Egzamin |
| F2 | | Kolokwia sprawdzające |
| F3 | PEK_U01, PEK_U02, PEK_U03, PEK_U04<br>PEK_K01, PEK_K02 | Projekt programistyczny |
| F4 | PEK_U01, PEK_U02, PEK_U03, PEK_U04 | Listy zadań do samodzielnej realizacji |
| F5 | PEK_W01<br>PEK_U01, PEK_U02, PEK_U03, PEK_U04 | Krótkie testy sprawdzające |

**P** – podsumowująca (na koniec semestru)

**P – wykład = max(F1, F2)**

**P – projekt = F3**

**P – laboratorium = 0.5*F4 + 0.5*F5**

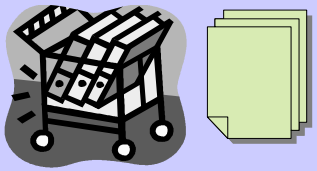
# II. Wprowadzenie do Inżynierii Oprogramowania(IO)

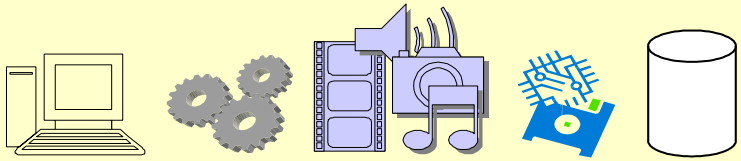
# 1. System informatyczny

(na podstawie Paul Beynon_Davies, Inżynieria systemów informacyjnych)

**Nieformalny system informacyjny:**
**zasoby osobowe - ludzie**

**Formalny system informacyjny:**
**procedury zarządzania, bazy wiedzy**





**Techniczny system informacyjny:**

- **Sprzęt**
- **Oprogramowanie**
- **Bazy danych, bazy wiedzy**

**System informatyczny**
**jest to zbiór powiązanych ze sobą elementów nieformalnych, formalnych i technicznych, którego funkcją jest przetwarzanie danych przy użyciu techniki komputerowej**

**Techniczny system informacyjny**

- **zorganizowany zespół środków technicznych (komputerów, oprogramowania, urządzeń teletransmisyjnych itp.)**
- **służący do gromadzenia, przetwarzania i przesyłania informacji.**

# 2. Definicje inżynierii oprogramowania

- [Fritz Bauer]
  - Opracowanie sprawdzonych zasad inżynierii oraz ich zastosowanie w celu wytworzenia niedrogiego i niezawodnego oprogramowania, działającego efektywnie na rzeczywistych maszynach
- [IEEE 1993]
  - (1)Zastosowanie systematycznego, zdyscyplinowanego, poddającego się ocenie ilościowej, podejścia do wytwarzania, stosowania i pielęgnacji oprogramowania, czyli wykorzystanie technik tradycyjnej inżynierii w informatyce.
  - (2) Dziedzina wiedzy zajmująca się badaniem metod jak w p. (1)

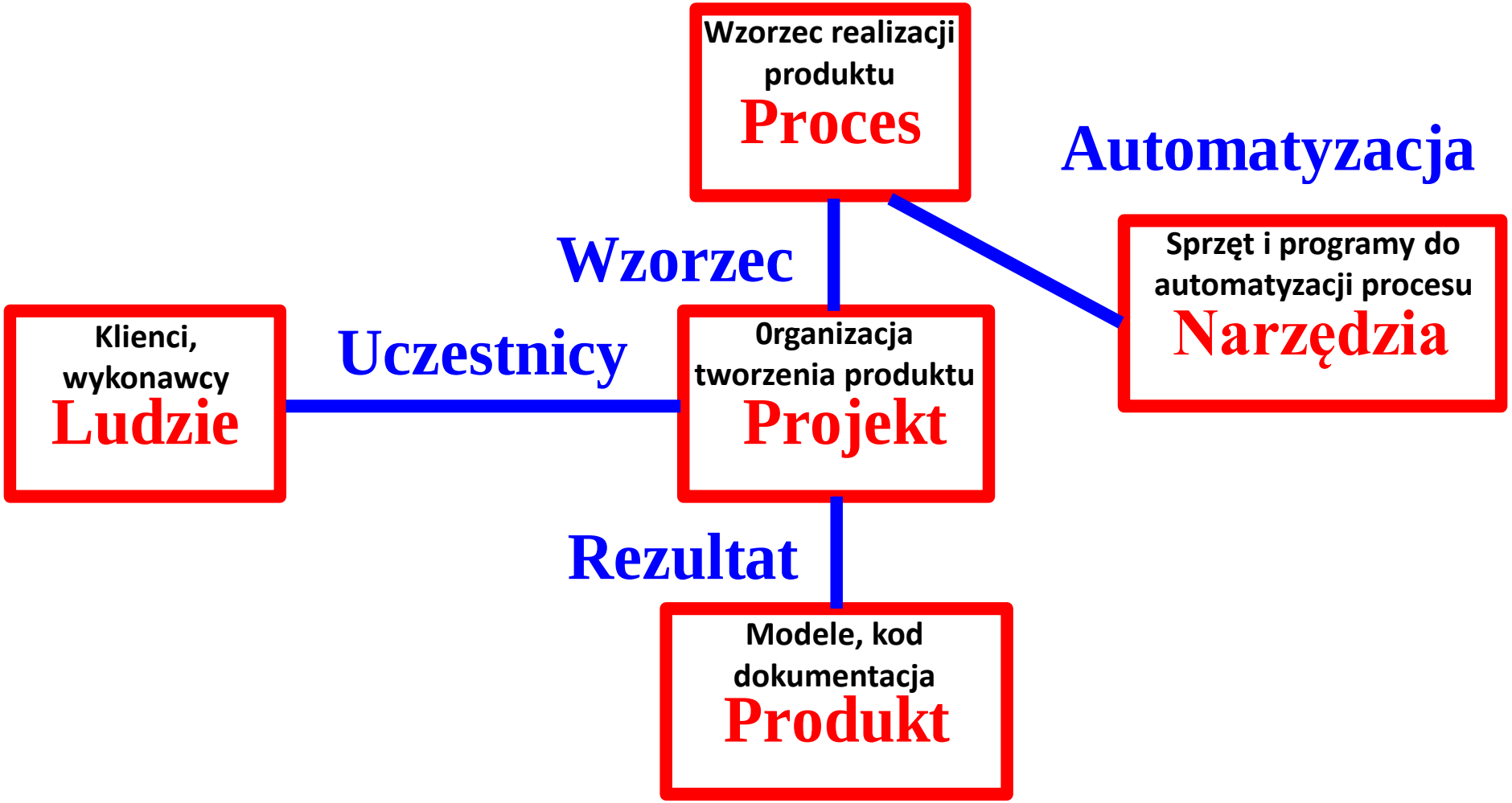
# 2.1. Warstwowe podejście w inżynierii oprogramowania [1LU]

# 2.2. Ogólne spojrzenie na Inżynierię Oprogramowania [1LU]

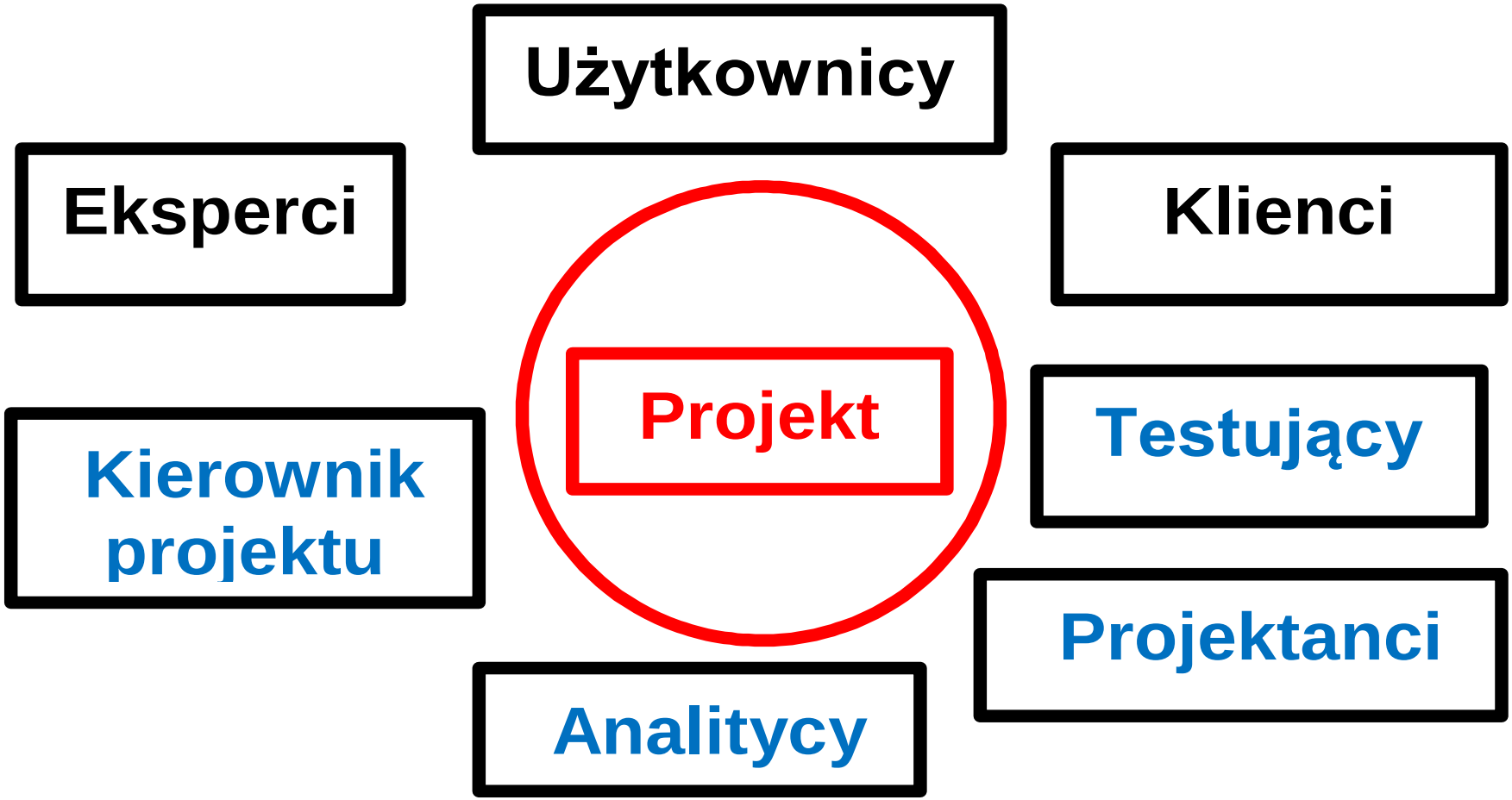
1) Jaki problem należy rozwiązać
2) Jakie cechy produktu umożliwiają rozwiązanie problemu
3) Jak ma wyglądać produkt (rozwiązanie problemu)
4) Jak skonstruować taki produkt
5) Jak wykrywać błędy w projekcie lub podczas konstrukcji produktu
6) Jak obsługiwać i pielęgnować gotowy produkt i jak uwzględniać uwagi, reklamacje i żądania jego użytkowników: **poprawianie, adaptowanie, rozszerzanie, zapobieganie** (aby łatwo poprawić, adaptować i rozszerzać)

# III. Podstawowe pojęcia: specyfikacja, analiza, projekt, implementacja, test, wdrożenie, pielęgnacja

# Elementy tworzenia oprogramowania – struktura [3LU]



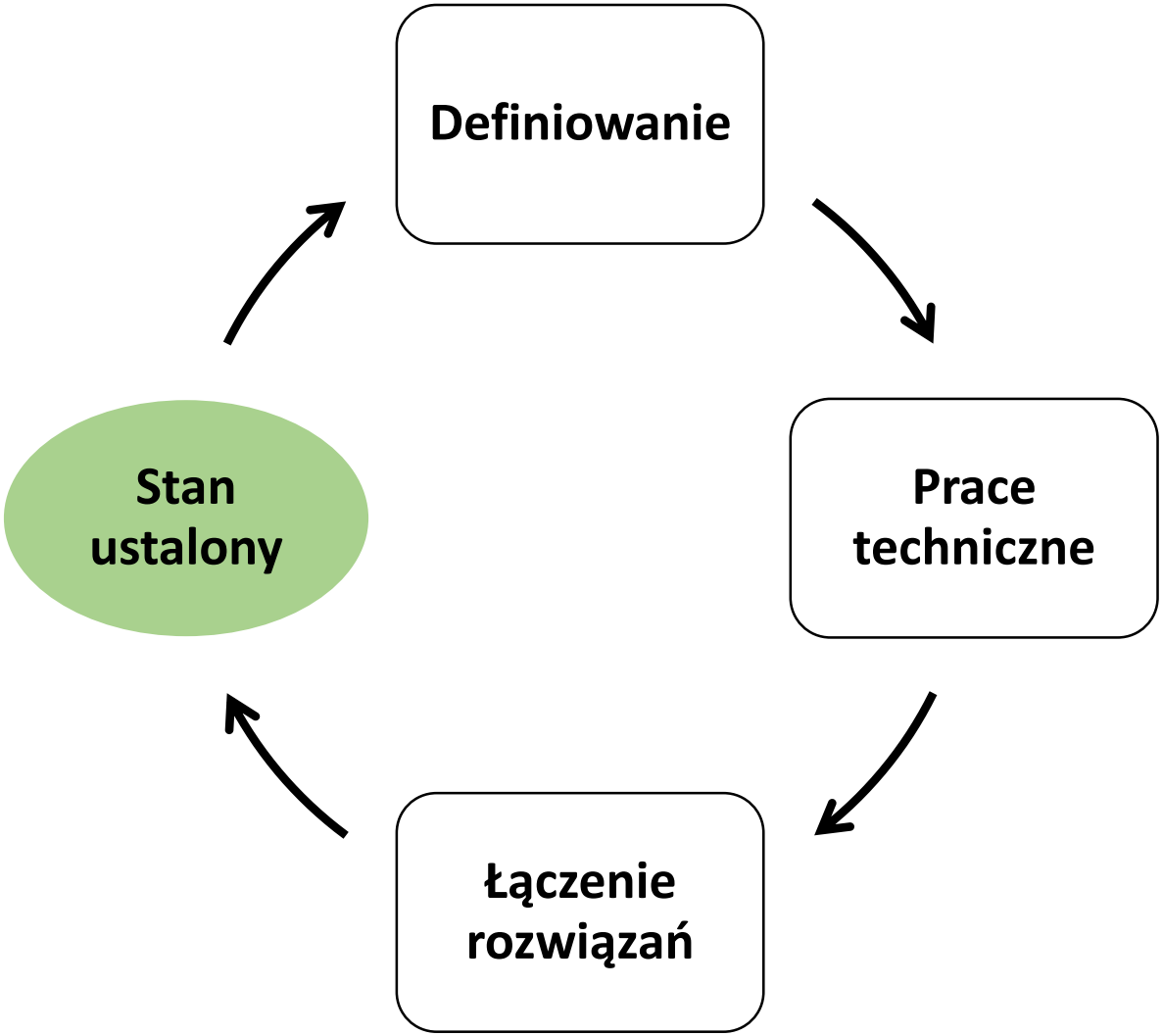
# 1. Ludzie [3LU]


Użytkownicy
Eksperci
Klienci
Projekt
Kierownik projektu
Testujący
Projektanci
Analitycy

# 2. Proces tworzenia oprogramowania [3LU]

1) **Proces tworzenia oprogramowania** jest definicją kompletnego zbioru aktywności potrzebnych do **odwzorowania wymagań użytkownika w oprogramowanie**
2) **Obejmuje czynniki:**
   - Czynniki organizacyjne
   - Czynniki dziedzinowe
   - Czynniki **cyklu życia oprogramowania**
   - Czynniki techniczne

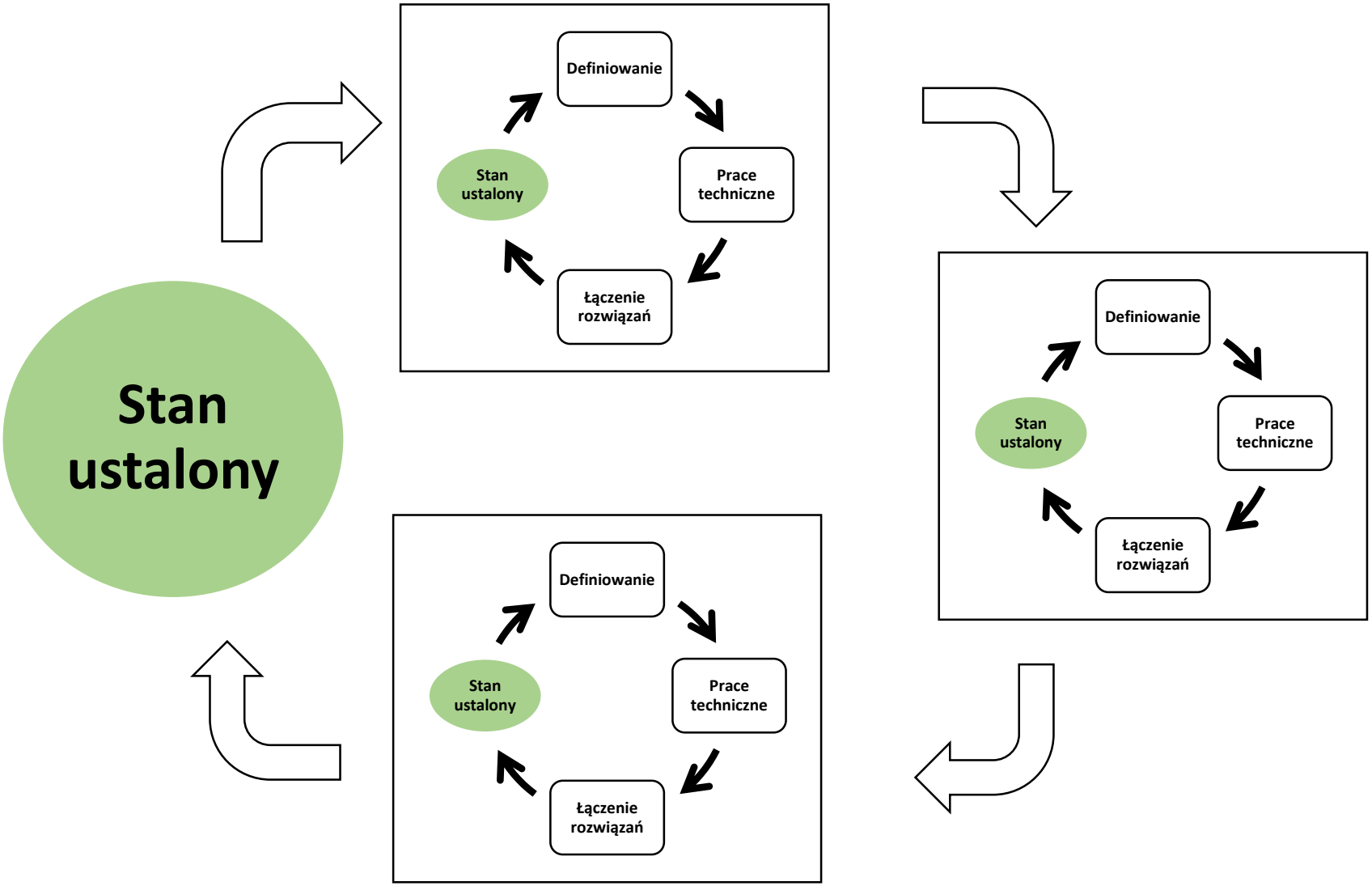
# 2.1. Uniwersalny schemat procesu wytwórczego - model cyklu życia tworzenia oprogramowania [1LU]



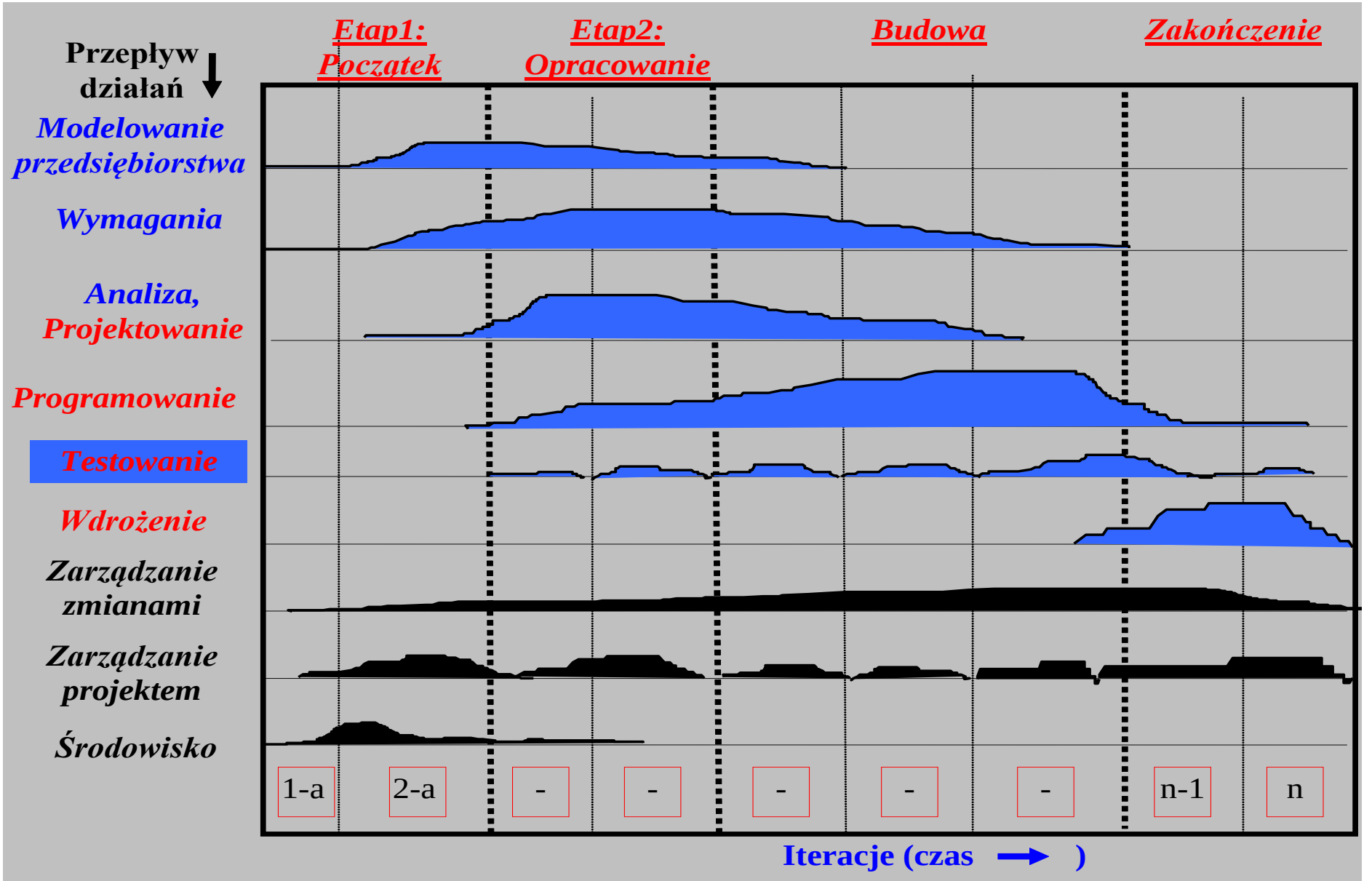
# 2.1 (cd). Proces - *model* procesu wytwarzania oprogramowania - czyli *model* cyklu życia oprogramowania [3LU, 2LPW]

<table>
<tr><th colspan="2">Definiowanie<br>Modelowanie struktury i dynamiki systemu</th><th>Prace techniczne, łączenie rozwiązań<br>Implementacja systemu, struktury i dynamiki generowanie kodu</th></tr>
<tr><td>Perspektywa koncepcji<br>co należy wykonać?</td><td>Perspektywa specyfikacji<br>jak należy używać?</td><td>Perspektywa implementacji<br>jak należy wykonać?</td></tr>
<tr><td>• model problemu np. przedsiębiorstwa<br>• wymagania<br>• analiza (model konceptualny )<br>• testy modelu</td><td>• projektowanie (model projektowy: architektura sprzętu i oprogramowania; dostęp użytkownika; przechowywanie danych)<br>• testy projektu</td><td>• programowanie (specyfikacja programu : deklaracje, definicje; dodatkowe struktury danych: struktury „pojemnikowe”, pliki, bazy danych)<br>• testy oprogramowania<br>• wdrażanie<br>• testy wdrażania</td></tr>
</table>

# 2.2. Model iteracyjnego cyklu życia procesu tworzenia oprogramowania [1LU]



# 2.2 (cd). Proces - zunifikowany iteracyjno- przyrostowy proces tworzenia oprogramowania – kiedy należy wykonać? [3LU]



# 2.2 (cd). Proces - przepływy czynności [3LU]

- Modelowanie przedsiębiorstwa – opis dynamiki i struktury przedsiębiorstwa
- Wymagania – zapisanie wymagań metodą opartą na przypadkach użycia
- Analiza i projektowanie – zapisanie różnych **perspektyw** architektonicznych
- Implementacja – tworzenie oprogramowania, testowanie modułów, scalanie systemu
- Testowanie – opisanie danych testowych, procedur i metryk poprawności
- Wdrożenie – ustalenie konfiguracji gotowego systemu
- Zarządzanie zmianami – panowanie nad zmianami i dbanie o spójność elementów systemu
- Zarządzanie projektem - opisane różnych strategii prowadzenia procesu iteracyjnego
- Określenie środowiska – opisanie struktury środowiska niezbędnej do opracowania systemu

# 2.3. Proces - co i jak wykonać? [2LPW]

## Perspektywy projektowania obiektowych systemów informacyjnych

- **koncepcji** (*model* analizy)

  ( co obiekty powinny robić?)

- **specyfikacji interfejsów** (*model* projektowy)

  ( jak używać obiektów? )

- **implementacji** (implementacja)

  ( w jaki sposób zaimplementować interfejs ?)

- **tworzenia i zarządzania obiektami** (implementacja)

  (*obiekt A w roli fabryki obiektów tworzy obiekt B lub zarządza obiektem)*

- **używania obiektów** (implementacja)

  ( *obiekt A tylko używa obiektu B – nie może go jednocześnie tworzyć;*

  *opiera się na hermetyzacji i polimorfizmie obiektu B )*

# 2.3 (cd). Proces - perspektywy rozumienia obiektów – identyfikacji obiektów [2LPW]

- **Perspektywa koncepcji** (*modelu* konceptualnego)

  - obiekt jest zbiorem różnego rodzaju odpowiedzialności

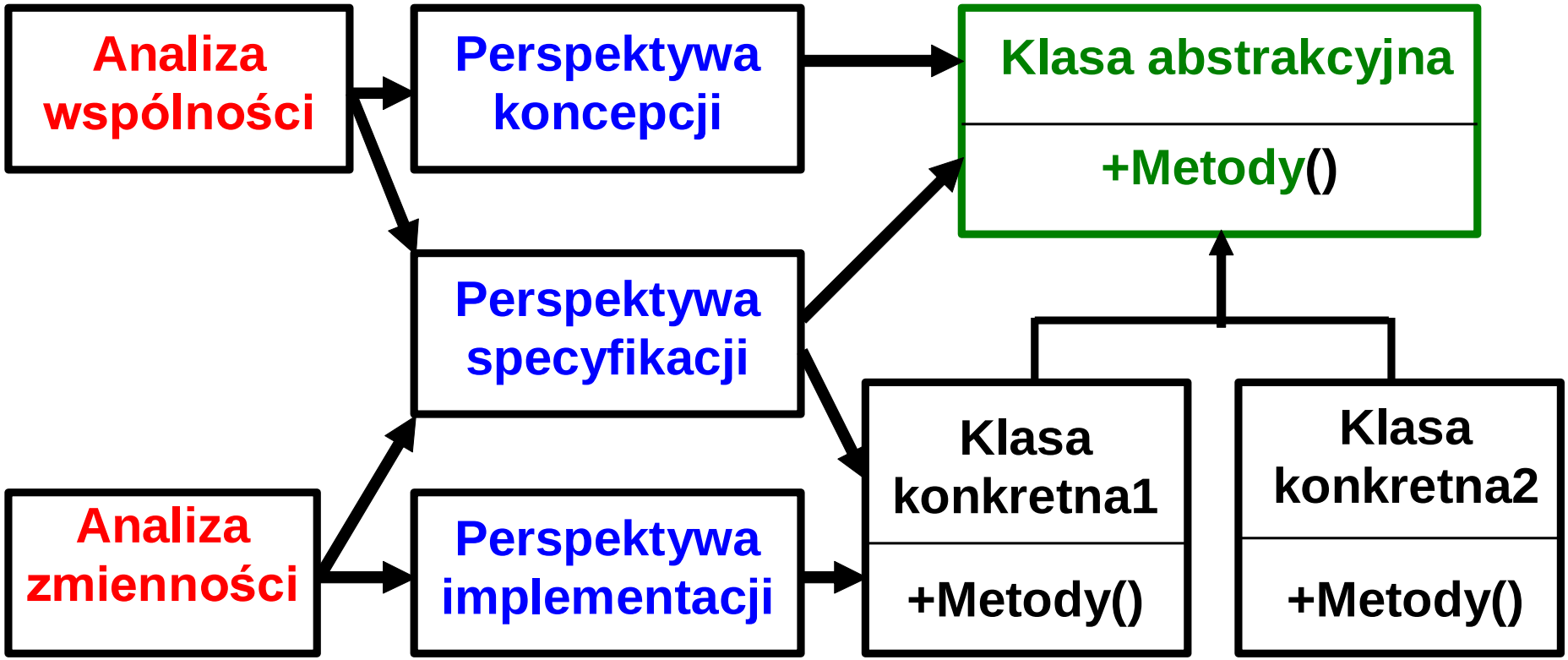
- **Perspektywa specyfikacji** (*modelu* projektowego)

  - obiekt jest zbiorem metod (zachowań), które mogą być wywoływane przez metody tego obiektu lub innych obiektów

- **Perspektywa implementacji** (kodu źródłowego)

  - obiekt składa się z kodu metod i danych oraz interakcji między nimi

# 2.3 (cd). Proces - metoda identyfikacji obiektów i klas [2LPW]




Związki między perspektywą specyfikacji, koncepcji i implementacji

# 2.4. Proces - rozłożenie pracy w czasie (1LU)

- Zasada **40-20-40 (przybliżona)**:
  - Początkowe analizowanie (10-25%) wymagań (co zrobić?), projektowanie (20-25%) wymagań (jak robić ?)
  - Pisanie kodu (15-20%) (jak robić ?)
  - Testowanie i usuwanie błędów (30-40%) (poprawa)

# 2.5. Proces - *modele* procesów wytwórczych (1LU)

1. Sekwencyjny model liniowy
2. Model oparty na prototypowaniu
3. Model szybkiej rozbudowy aplikacji
4. Modele ewolucyjne
   - **Model przyrostowy**
   - Model spiralny
   - Model spiralny WINWIN
   - Model równoległy
5. Model oparty na metodach formalnych
6. Model oparty na komponentach
7. Techniki czwartej generacji

# 3. Produkt (3LU)

1. *Modele*, kod źródłowy, kod wynikowy, dokumentacja
2. Podsystemy
3. Diagramy: klas, interakcji, kooperacji, stanów
4. Wymagania, testy, produkcja , instalacja
5. **Wytworzone oprogramowanie**
6. Jest to system związany **z procesem wytwórczym** (wymagania, analiza, projektowanie, programowanie, testy) – konieczne jest właściwe **dopasowanie procesu do produktu**

# 3.1. Produkt – dziedziny zastosowań

1. Oprogramowanie systemowe
2. Systemy czasu rzeczywistego
3. Systemy informacyjne dla przedsiębiorstw
4. Oprogramowanie inżynierskie i naukowe
5. Systemy wbudowane
6. Oprogramowanie komputerów osobistych
7. Oprogramowanie internetowe
8. Sztuczna inteligencja
9. **Systemy hybrydowe (2, 4, 5, 7, 8)**
   **np Zintegrowany laboratoryjny system informatyczny do zarządzania informacjami medycznymi – LIMS (Laboratory Information Management System) oraz Laboratoryjny system informatyczny -LIS (Laboratory Information System)**

# 3.1 (cd). Produkt – oprogramowanie na platformie Java EE

**Pięciowarstwowy model logicznego rozdzielania zadań aplikacji**
**(wg. D.Alur, J.Crupi, D. Malks, Core J2EE. Wzorce projektowe.)**

**Warstwa klienta**
Klienci aplikacji, aplety, aplikacje i inne elementy z graficznym interfejsem użytkownika

Interakcja z użytkownikiem, urządzenia i prezentacja interfejsu użytkownika

**Warstwa prezentacji**
Strony JSP, serwlety i inne elementy interfejsu użytkownika

Logowanie, zarządzanie sesją, tworzenie zawartości, formatowania i dostarczanie

**Warstwa biznesowa**
Komponenty EJB i inne obiekty biznesowe

Logika biznesowa, transakcje, dane i usługi

**Warstwa integracji**
JMS, JDBC, konektory i połączenia z systemami zewnetrznymi

Adaptery zasobów, systemy zewnętrzne, mechanizmy zasobów, przepływ sterowania

**Warstwa zasobów**
Bazy danych, systemy zewnętrzne i pozostałe zasoby

Zasoby, dane i usługi zewnętrzne

# 3.2. Produkt - oprogramowanie – byt logiczny

**Oprogramowanie to:**

- rozkazy , których wykonanie pozwala wypełnić określone funkcje w oczekiwany sposób
- struktury danych, które umożliwiają programom manipulowanie informacjami
- oraz dokumenty, które opisują działanie i sposób użytkowania programów.

# 3.3. Produkt - cechy charakterystyczne oprogramowania

- Oprogramowanie jest wytwarzane, ale nie jest fizycznie konstruowane (np. tak jak sprzęt)
- Oprogramowanie się nie zużywa, ale z czasem niszczeje
- Korzystanie z gotowych komponentów, ale większość jest tworzona od nowa

# 3.4. Produkt - mity wynikające z ignorowania zasad inżynierii oprogramowania [1LU]

- **Mity kierownictwa**
  - Standardy i procedury zapewniają tworzenie dobrego oprogramowania
  - Dobre oprogramowanie narzędziowe działające na dobrym sprzęcie gwarantuje wykonanie dobrych programów
  - Jeśli prace się opóźniają, wystarczy przydzielić do zadania więcej programistów
  - Jeżeli oprogramowanie wykonuje inna firma (outsourcing), to można pozbyć się problemów

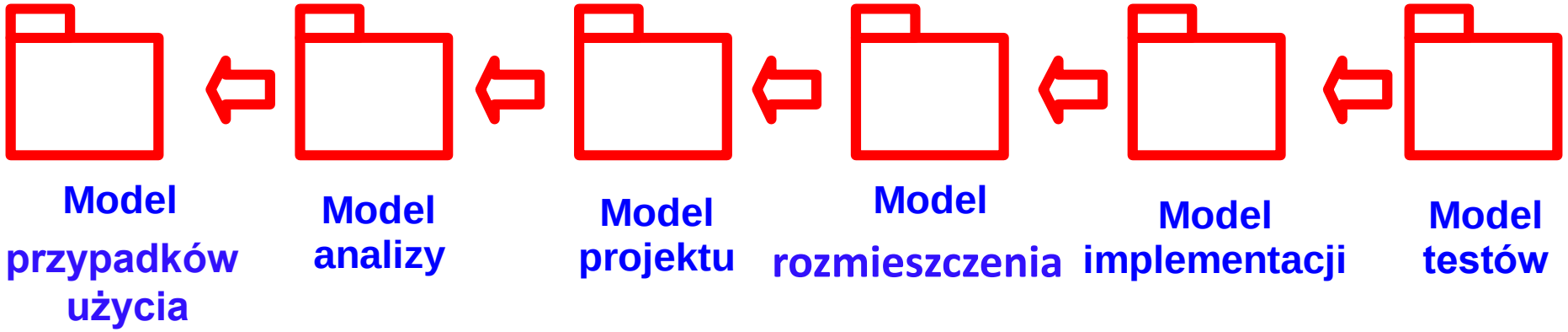
- **Mity klientów**
  - Ogólne określenie oczekiwań klienta wystarczy do rozpoczęcia prac, a szczegóły można dopracować później
  - Wymaganie wobec systemu wciąż się zmienia. Ale to nie problem, bo oprogramowanie jest elastyczne i łatwo je zmienić.

- **Mity informatyków**
  - Po napisaniu programu i uruchomieniu go praca jest wykonana
  - Dopóki program nie działa, nie da się ocenić jego jakości
  - Jedynym wynikiem pracy nad oprogramowaniem jest działający program komputerowy
  - Inżynieria oprogramowania zmusi nas do tworzenia przepastnych, zbędnych dokumentów i nieuchronnie spowolni pracę.

# 3.5. Produkt – modele [3LU]

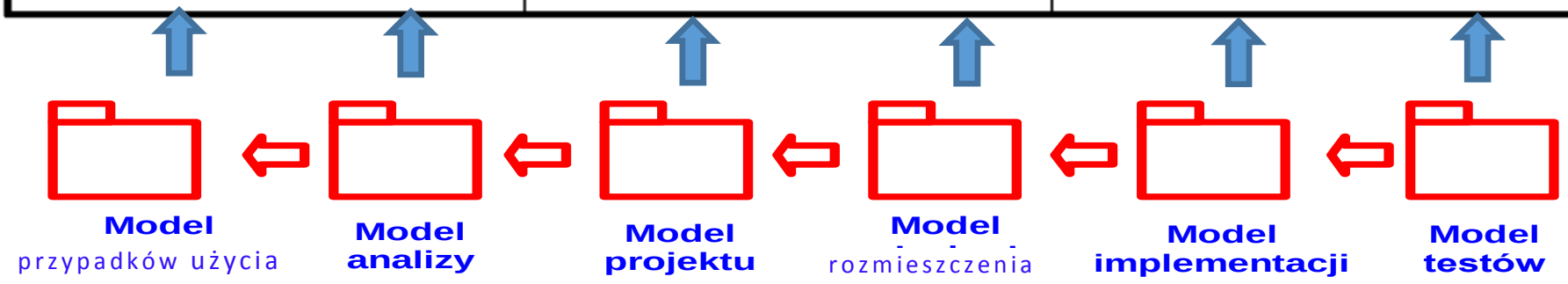
***Modele:***

- Abstrakcja systemu
- Przedstawianie różnych perspektyw systemu
- Związki między modelami



# 3.6. Produkt – modele i proces [3LU]

| Modelowanie struktury i dynamiki systemu | Implementacja systemu, | struktury i dynamiki generowanie kodu |
|---|---|---|
| Perspektywa koncepcji<br>*co należy wykonać?* | Perspektywa specyfikacji<br>*jak należy używać?* | Perspektywa implementacji<br>*jak należy wykonać?* |
| • model problemu np. przedsiębiorstwa<br>• wymagania<br>• analiza<br>(model konceptualny: diagram przypadków użycia, diagram klas, diagramy sekwencji, )<br>• testy modelu | • projektowanie<br>(model projektowy:<br>architektura sprzętu i oprogramowania;<br>dostęp użytkownika;<br>przechowywanie danych)<br>• testy projektu | • programowanie, wdrażanie<br>(specyfikacja programu :<br>deklaracje, definicje;<br>dodatkowe struktury danych:<br>struktury „pojemnikowe”,<br>pliki, bazy danych)<br>• testy oprogramowania<br>• wdrażanie<br>• testy wdrażania |



# 3.7. Produkt - języki modelowania

- **Języki modelowania poprzedzające UML jako odpowiedź na języki obiektowe programowania:**
  - Yourdon/Coad - **Object-Oriented Analysis and Design (OOA/OOD**)
  - Grady Booch (Rational Software) - **metoda Boocha 1994**
  - James Rumbaugh (General Electric) - **Object Modelling Technique (OMT) 1991**
  - Ivar Jacobson(Objectory AB) - **Object-oriented software engineering (OOSE) 1992**
- **Unified Modeling Language (UML,** zunifikowany język modelowania) – język pół-formalny wykorzystywany do modelowania nie tylko modeli struktury aplikacji, zachowania i architektury, ale również procesów biznesowych i struktury danych.
  - powstał w 1997,
  - autorzy: Grady Booch, James Rumbaugh, Ivar Jacobson
  - obecnie rozwijany przez OMG (Object Management Group)http://www.uml.org

# 3.7 (cd). Produkt - języki modelowania

- **Modelowanie usług sieciowych**

**BPEL** (Business Process Execution Language), czyli **WS-BPEL** (*Web Services Business Process Execution Language)*, - model procesu współpracy zbioru aplikacji w celu osiągnięcia celu biznesowego, który jednocześnie jest usługą sieciową. Oznacza to, że jest także opisywany jest przez **WSDL** (*Web Services Description Language*), który opisuje jednak pojedynczą usługę sieciową.

Komitet techniczny WS-BPEL: OASIS (*Organization for the Advancement of Structured Information Standards*)

https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel

- **Modelowanie procesów biznesowych**

**BPMN (***Business Process Model and Notation*), wcześniej skrót oznaczał *Business Process Modeling Notation* rozwijany przez OMG (http://www.bpmn.org).

Modelowanie procesów z użyciem diagramów przepływu **BPD** (*Business Process Diagram*) przypominające diagramy aktywności UML.

- **Modelowanie oprogramowania komputerowego na wysokim poziomie abstrakcji DSM** (*Domain Specific Modelling*) przy użyciu języków typu **DSL** (*Domain Specific Language*) np. HTML

# 3.7 (cd). Produkt - języki modelowania SysML

- **Modelowanie złożonych systemów zawierających sprzęt, oprogramowanie, informację, personel, procedury i ułatwienia**

**MBSE**(*Model-Based Systems Engineering*) – modelowanie za pomocą języka modelowania **SysML** (*Systems Modeling Language*), obecnie OMG SysML.

**SysML jest dialektem języka UML**.

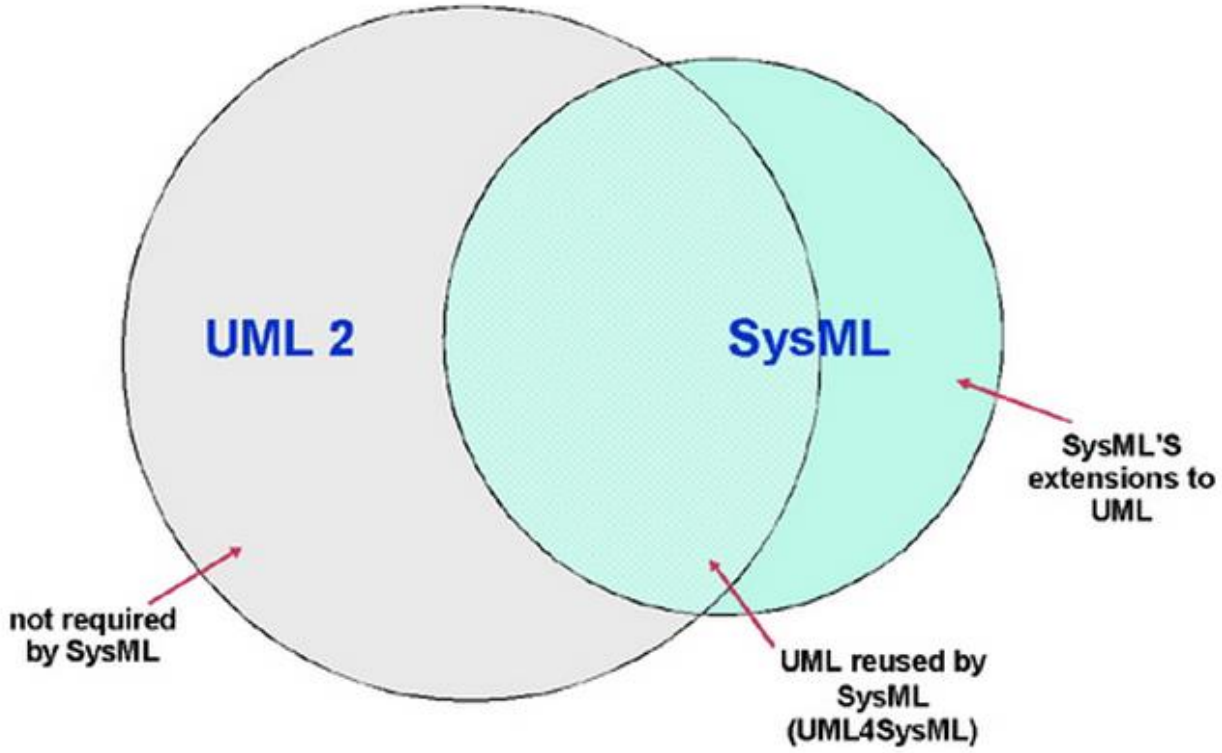
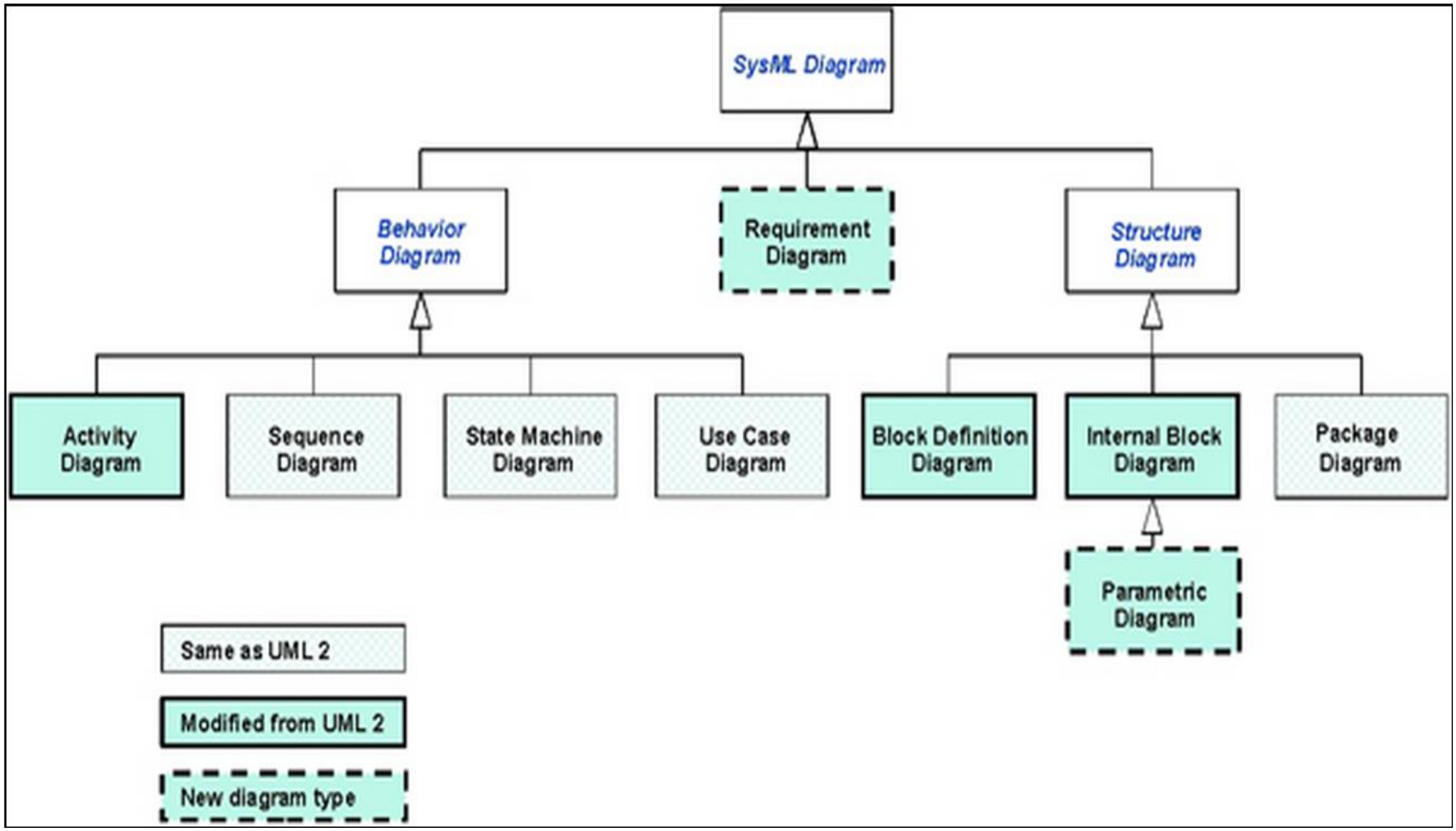



Figure 1. Relationship between SysML and UML

http://www.omgsysml.org

# 3.7 (cd). Produkt - diagramy SysML




http://www.omgsysml.org

# 3.8. Produkt - diagramy UML wspierające zunifikowany iteracyjno - przyrostowy proces tworzenia oprogramowania [1LPU]

## Diagramy UML modelowania struktury

1.1. Diagramy pakietów

1.2. Diagramy klas

1.3. Diagramy obiektów

1.4. Diagramy mieszane

1.5. Diagramy komponentów

1.6. Diagramy wdrożenia

# 3.8 (cd). Produkt - diagramy UML modelowania zachowania

2.1. Diagramy przypadków użycia

2.2. Diagramy aktywności

2.3. Diagramy stanów

2.4. Diagramy komunikacji

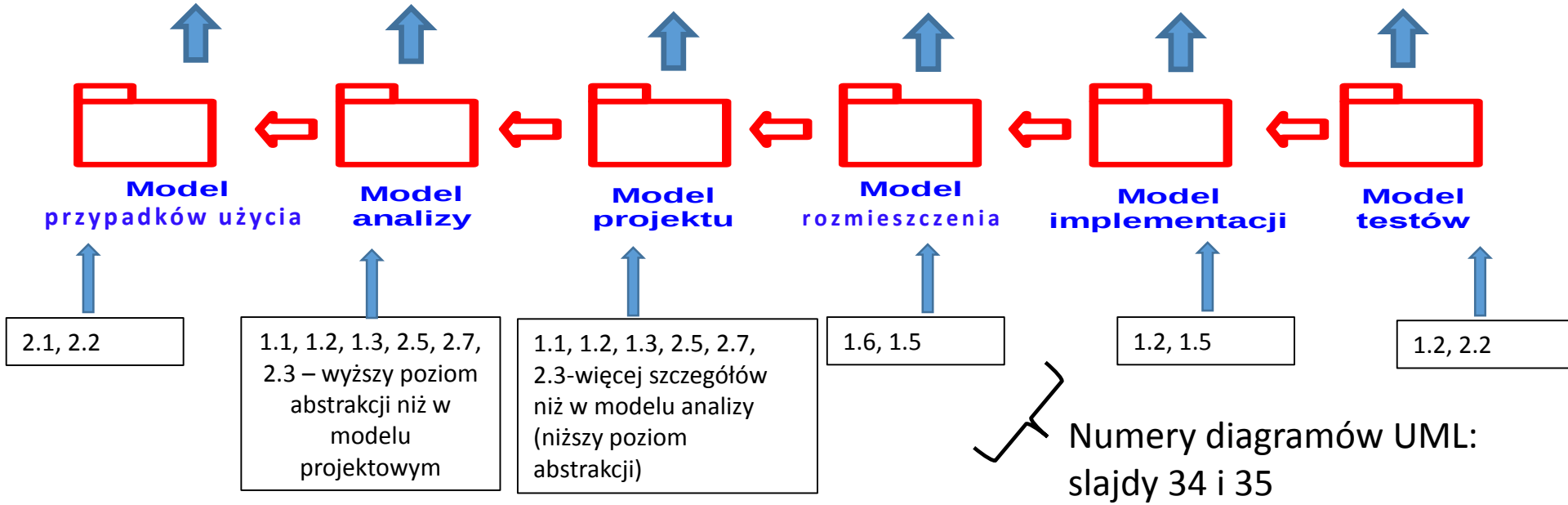
2.5. Diagramy sekwencji

2.6. Diagramy czasu

2.7. Diagramy interakcji

# 3.8(cd). Produkt - diagramy UML – modele, proces

| Modelowanie struktury i dynamiki systemu | Implementacja systemu, | struktury i dynamiki generowanie kodu |
|---|---|---|
| Perspektywa koncepcji<br>*co należy wykonać?* | Perspektywa specyfikacji<br>*jak należy używać?* | Perspektywa implementacji<br>*jak należy wykonać?* |
| • model problemu np. przedsiębiorstwa<br>• wymagania<br>• analiza<br>(model konceptualny: diagram przypadków użycia, diagram klas, diagramy sekwencji, )<br>• testy modelu | • projektowanie<br>(model projektowy:<br>architektura sprzętu i oprogramowania;<br>dostęp użytkownika;<br>przechowywanie danych)<br>• testy projektu | • programowanie, wdrażanie<br>(specyfikacja programu :<br>deklaracje, definicje;<br>dodatkowe struktury danych:<br>struktury „pojemnikowe”,<br>pliki, bazy danych)<br>• testy oprogramowania<br>• wdrażanie<br>• testy wdrażania |



# 3.8 (cd). Produkt - rola diagramów UML

- praca zespołowa (potokowa realizacja oprogramowania)
- pokonanie złożoności produktu (wspieranie iteracyjno-rozwojowego tworzenia oprogramowania, język wielu perspektyw tworzenia oprogramowania)
- formalne, precyzyjne prezentowanie produktu, możliwość odwzorowania elementów produktu za pomocą kodu języków programowania
- tworzenie wzorca produktu
- możliwość testowania oprogramowania we wczesnym stadium jego tworzenia (podczas analizy i projektowania)

# 3.8 (cd). Produkt - opinie z ankiet o cechach UML

( Exploring Costs and Benefits of Using UML on Maintenance: Preliminary Findings of a Case Study in a Large IT Department , Ana M. Fernández-Sáez1, Michel R.V. Chaudron2, Marcela Genero)

| Zalety | Wady |
|---|---|
| Wysoki poziom abstrakcji | Brak uruchamiania modelu |
| Wysoka przydatność w modelowaniu systemów | Niejasna semantyka |
| Zorientowane obiektowo systemy | Brak standaryzacji modelowania - nazewnictwo, modelowanie warstw oprogramowania, poziom szczegółowości modelowania, niejasny związek między diagramami i kodem.. |
| Prezentowanie różnych punktów widzenia | Wysoki poziom abstrakcji |
| Standaryzacja | Brak punktu widzenia użytkownika |
| | Niska zdolność projektowania SOA (*Service-Oriented Architecture*) → BPMN |
| | Nie egzekwuje oddzielania tego, **co należy zrobić** od tego, **jak należy zrobić** |

# 3.8 (cd). Produkt - opinie z ankiet o stosowaniu UML

( Exploring Costs and Benefits of Using UML on Maintenance: Preliminary Findings of a Case Study in a Large IT Department , Ana M. Fernández-Sáez1, Michel R.V. Chaudron2, Marcela Genero)

| Zalety | Wady |
|---|---|
| Pomaga  precyzować procedury | Trudności w zrozumieniu notacji |
| Wspomaga sposób modelowania | Trudności w modelowaniu złożonych elementów |
| Poprawia dokumentację | Brak wystarczającej siły wyrażania |
| Wspólny, zaakceptowany na świecie język modelowania | |
| Jest jedynym językiem, którego można nauczyć się poprawnie | |
| Redukuje nieporozumienia i luki przy współpracy z firmami zagranicznymi (offshoring) | |

1. Pozytywny stosunek do modelowania mają architekci, programiści i inżynierowie serwisu
2. Negatywny stosunek mają osoby, które nie znały lub słabo znały UML
3. Część podanych wad pozwoliła zidentyfikować niedopracowane procedury użycia UML do budowy poszczególnych modeli oprogramowania (slajdy 37, 38 i 45 )

# 3.9. Produkt a proces wytwórczy [1LU]

1. Wadliwy proces może spowodować powstanie produktu o niskiej jakości
2. Dualna natura procesu i produktu – wspierająca wielożywalność produktu
3. Czerpanie satysfakcji z faktu tworzenia i jego rezultatu
4. Ścisły związek między procesem i produktem zachęca twórczych ludzi do pracy

# 4. Narzędzia [3LU]

1. Automatyzacja procesu
2. Funkcjonalne wspomaganie całego cyklu życia oprogramowania:
    1. wymagania,
    2. wizualne modelowanie i projektowanie (kontrola poprawności diagramów, nawigacja po elementach modeli),
    3. programowanie,
    4. testowanie
    5. inżynieria wprost
    6. inżynieria odwrotna
3. Standaryzacja procesu i produktów
4. Współpraca z innymi narzędziami (import, export, integracja modeli)
5. Zarządzanie jakością oprogramowania
6. Monitorowanie, kontrolowanie postępu prac
7. Repozytorium, wspieranie pracy zespołowej

# 5. Projekt - planowanie, organizowanie, monitorowanie i kontrolowanie [1LU]

**Główny, organizacyjny element** powiązany z:

- Ludzie, Produkt, Proces

**Pojęcia**:

1. Wykonalność projektu
2. Zarządzanie ryzykiem
3. Struktura grup projektowych
4. Szeregowanie zadań projektowych
5. Zrozumiałość projektu
6. Sensowność działań w projekcie

**Cechy projektu:**

1. Sekwencja zmian w projekcie
2. Seria iteracji
3. Wzorzec organizacyjny

# 6. Korzyści wynikające ze stosowania zasad inżynierii oprogramowania

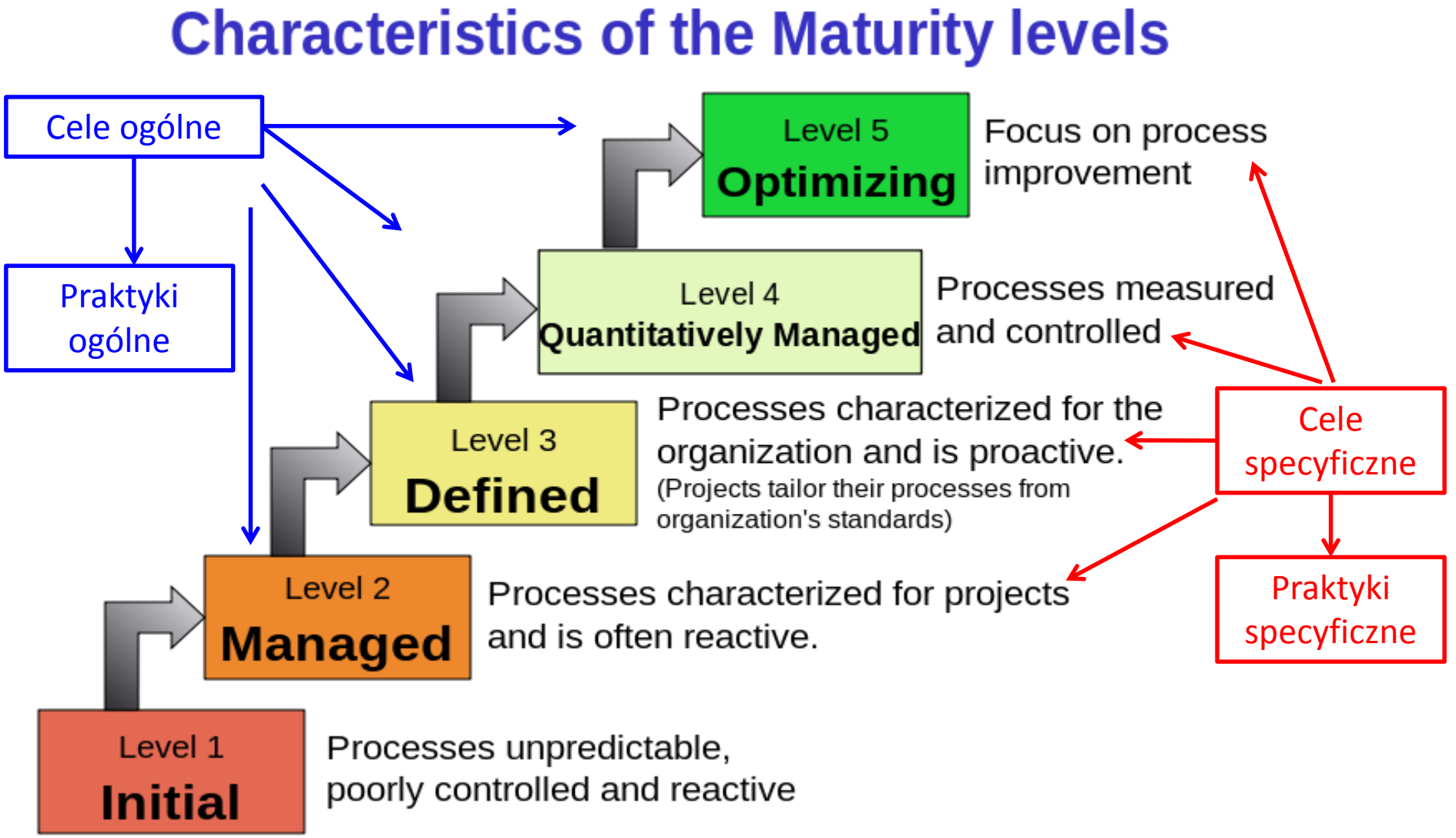
na przykładzie zastosowania
CMMI (Capability Maturity Model Integration ) - wytycznych dla poprawy jakości **produktu** i integracji **procesu**


1. http://whatis.cmmiinstitute.com, http://resources.sei.cmu.edu/library/results.cfm
2. http://www.tutorialspoint.com/cmmi/
3. http://en.wikipedia.org/wiki/Capability_Maturity_Model_Integration
4. http://msdn.microsoft.com/pl-pl/library/ee461556.aspx

# 6.1. CMMI - Capability Maturity Model Integration
## (Model dojrzałości procesu integrujący modele cząstkowe procesu)

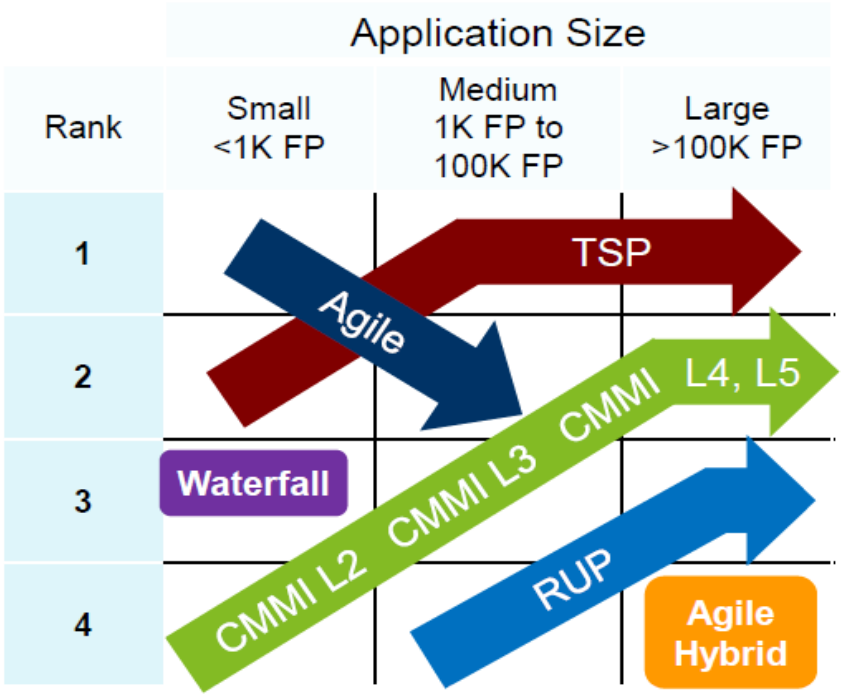
- **CMMI:** wytyczne dla **poprawy jakości produktu i integracji procesu**.
- **Pięć poziomów dojrzałości procesów wytwórczych CMM** (Capability Maturity Model, 1991) stanowią podstawę dla CMMI
- Celem CMMI jest zarządzanie ryzykiem i dostarczanie produktu wysokiej jakości
- Model CMMI pozwala zrozumieć elementy „świata rzeczywistego” i pomaga opracować koncepcje produktu i jego poprawę dzięki temu, że:
  - Dostarcza środowisko narzędziowe oraz języki komunikacji
  - Wykorzystuje lata doświadczeń
  - Ułatwia wykonawcom zapamiętanie dużego modelu pozwalając skupić się na poprawie jego jakości
  - Używany jest przez instruktorów i konsultantów
  - Dostarcza informacji wspierających rozwiązywanie sporów w oparciu o standardy

# 6.2. Poziomy dojrzałości modelu CMMI



# 6.3. TSP – integracja wielu praktyk CMMI

## TSP: Software Engineering Best Practice



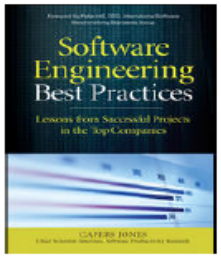

Development practices by size of application in function points (FP; 1FP ≈ 30 to 50 SLOC) [1] [2]

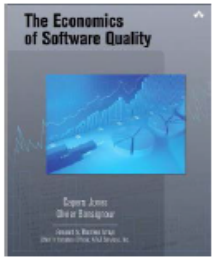
[1] Software Engineering Best Practices, by Capers Jones, 2010.

[2] The Economics of Software Quality, by Capers Jones, 2011.

### Demonstrated benefits

- scalable to application size
- situation tailorable
- predictable cost and schedule
- best quality (defect intolerant)
- continuous high throughput
- creates self-managed teams that own their processes and plans
- operationally defined for high-fidelity and clear end states, e.g. “done”



# 6.4. CMMI-Usługi (CMMI-SVC) – wsparcie organizacji zajmujących się dostarczaniem usług

## Impact for Organizations

| SIEMENS | Raytheon | GM | LOCKHEED MARTIN |
|---|---|---|---|
| Productivity improved by 25% using CMMI over a three-year period | 42% decrease in the costs of rework at CMMI Level 3 | Met milestones improved from 50% to 85% with focus on CMMI | 20% reduction in software costs by integrating its engineering processes using CMMI |

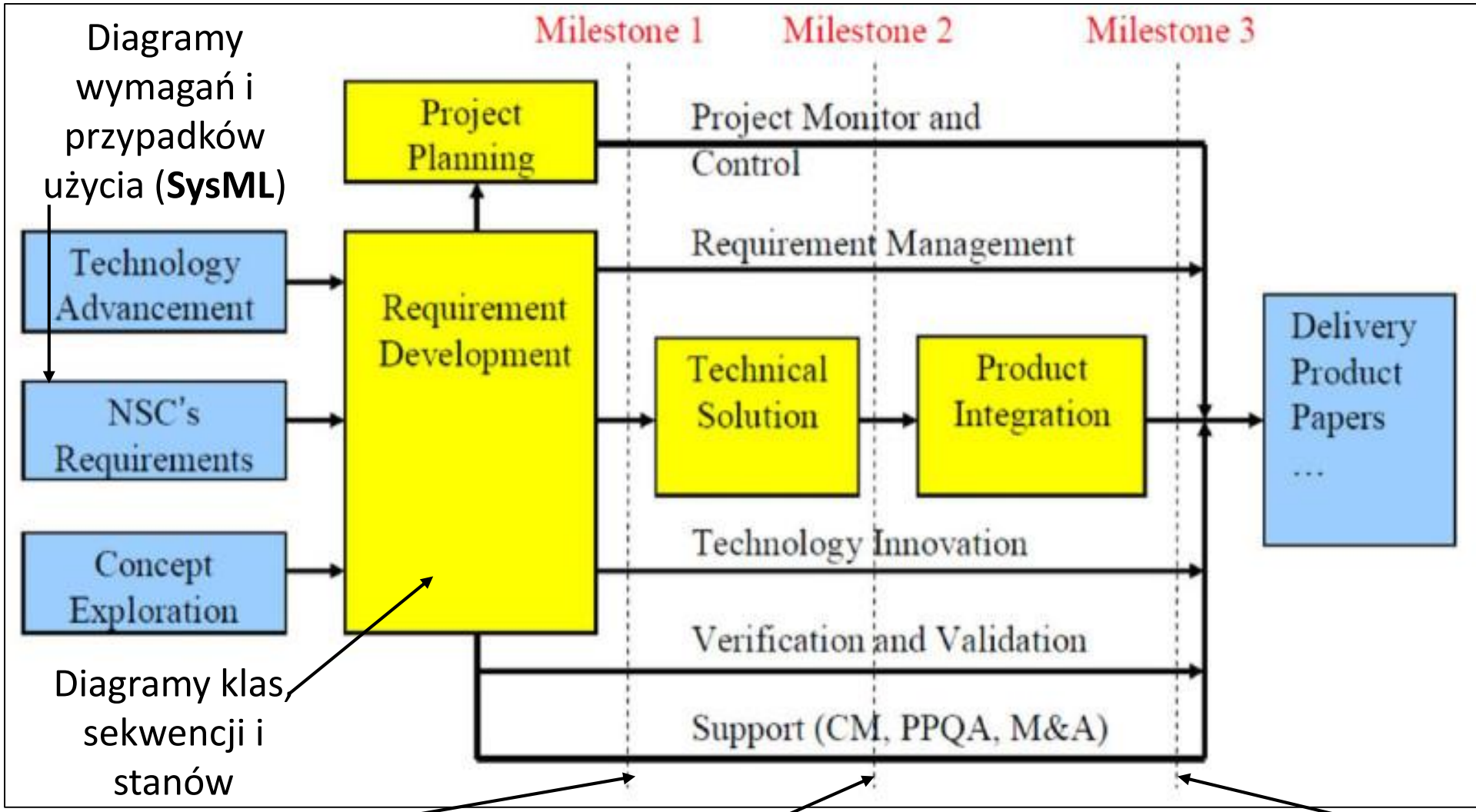
# IV. Rola IO w Inżynierii Biomedycznej

# 1. Podejście oparte na CMMI do cyklu życia tworzenia oprogramowania medycznego

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3699709/

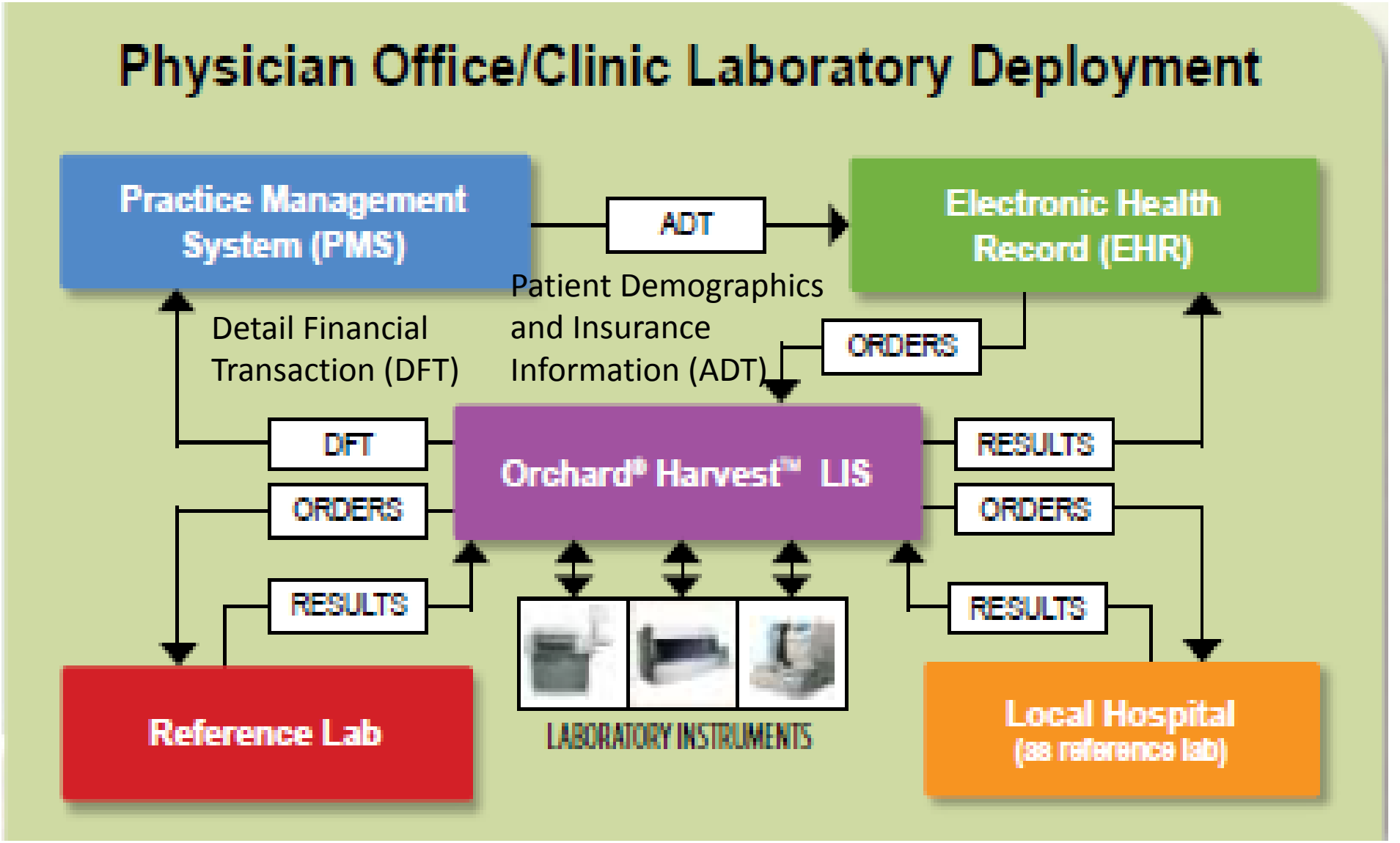
- W jaki sposób istniejące podejścia do inżynierii oprogramowania mogą zostać włączone / udoskonalone w kontekście zarządzania projektami medycznymi?
- W jaki sposób można efektywnie wykorzystać model koncepcyjny w celu ułatwienia zarządzania macierzą identyfikowalności w różnych fazach projektu za pomocą wielu perspektyw?

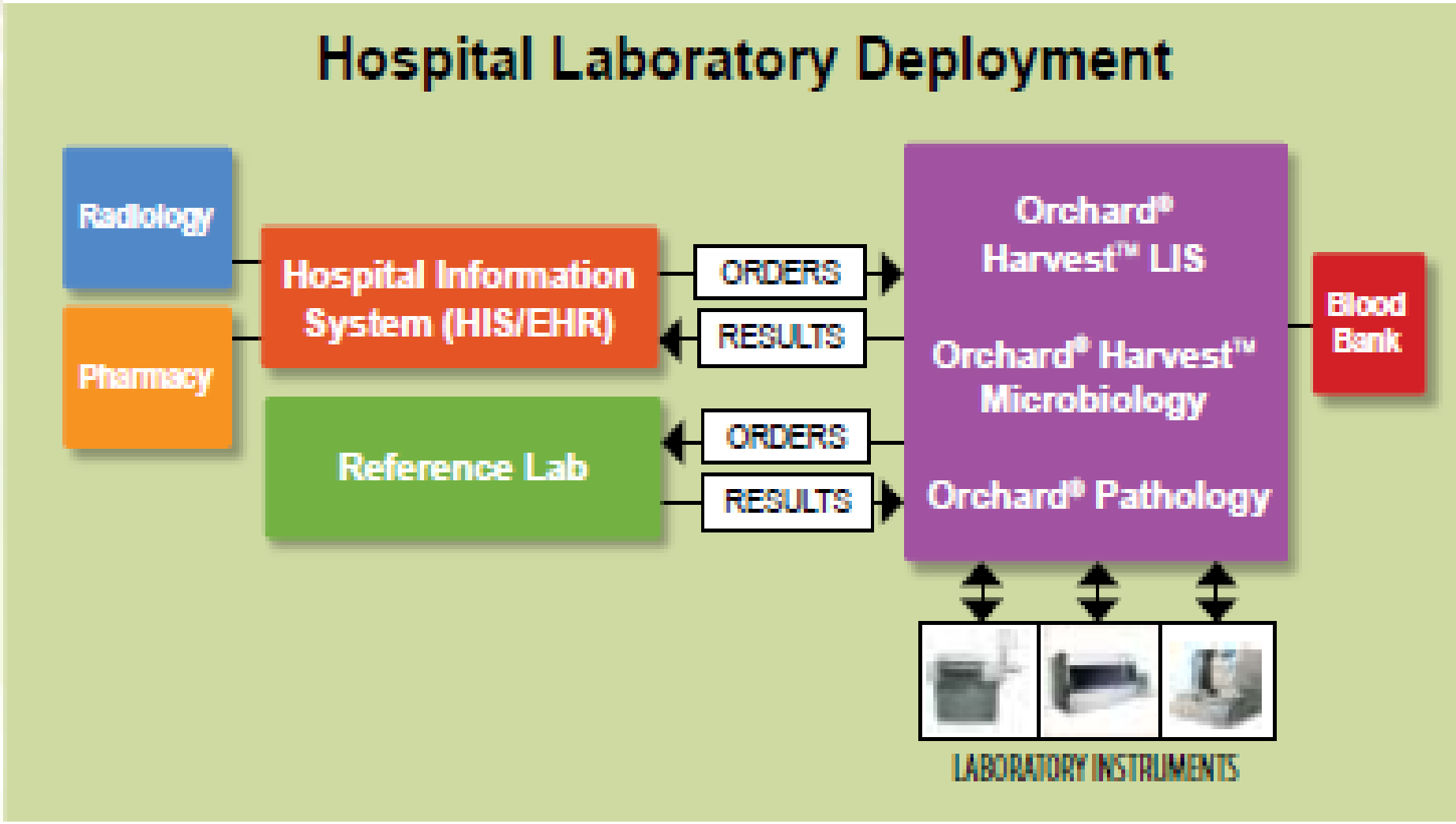
# 1. (cd) Cykl życia tworzenia oprogramowania w dziedzinie medycyny nuklearnej: Light-Weight Capability Maturity Model Integration (LW-CMMI)



# 2. Przykłady systemów LIS (Laboratory Information System) i LIMS (Laboratory Information Management System)

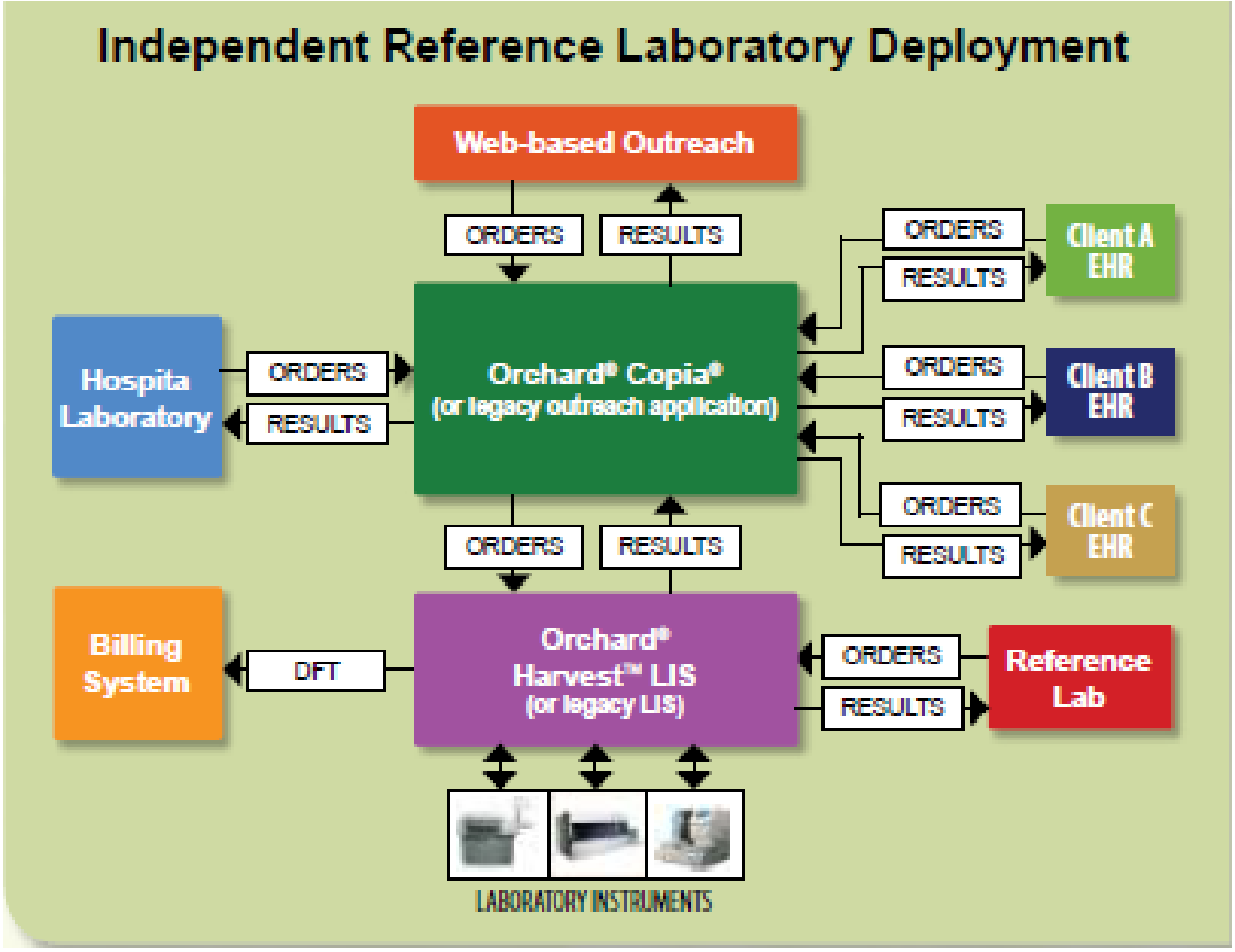
http://www.clpmag.com/2016/02/labs-information-age/






http://www.clpmag.com/2016/02/labs-information-age/

ORM (orders), ADT (demographics), ORU (results), DFT (charges). All messages are comprised of more than one segment.




http://www.clpmag.com/2016/02/labs-information-age/

# 3. Zintegrowany laboratoryjny system zarządzania informacjami (LIMS)

By, JYOTIRMOY ROY B.Phrm.6th sem BCDA COLLEGE OF PHARMACY AND TECHNOLOGY 78, Jessore Road(South), Hridaypur, Barasat, Kolkata – 700127


**5 faz procesu aboratoryjnego:**

1) przyjęcie i zarejestrowanie próbek wraz z danymi klienta
2) przypisanie, zaplanowanie i śledzenie prac analitycznych związanych z próbkami
3) przetwarzanie i zapewnianie jakości związane z próbkami oraz wyposażeniem laboratoryjnym
4) rejestrowanie danych związanych z analizami próbek
5) przeglądy, zatwierdzanie i podsumowanie danych dotyczących próbki do celów raportowych i dalszych analiz

https://www.labmaster.pl/czym-jest-lims/

**Zarządzanie próbkami**

Kluczową funkcją systemu LIMS jest zarządzanie próbkami. Funkcja jest zwykle inicjowana kiedy próbka zostaje dostarczona do laboratorium i zarejestrowana w systemie LIMS. Niektóre oprogramowania, takie jak LabMaster, pozwalają na wsparcie laboratorium we wcześniejszych fazach takich jak rejestrowanie zamówienia na pobranie próbki, wsparcie próbkobiorcy w zakresie metodyk pobierania, możliwych sposobów utrwalania, ilości próbki do pobrania, protokołów pobierania próbki oraz lokalizacji realizacji pobrania. Zarządzanie próbkami jest wspierane również przez identyfikowanie pobranych próbek za pomocą kodów kreskowych, które są generowane przez system, drukowane na etykietach samoprzylepnych oraz przyklejane na naczynia próbki. W momencie przyjęcia próbki do laboratorium, w systemie rejestrowane są wszelkie dane z pomiarów terenowych oraz opisy pozwalające na ocenę stanu próbki.

**Zarządzanie analizami**

Kolejną kluczową funkcją systemu LIMS jest harmonogramowanie oraz realizowanie analiz zarejestrowanych próbek. Oprogramowanie wspiera pracowników laboratorium w zakresie metodyk wykonania analiz oraz generowania sprawozdań.

**Zarządzanie konfiguracją**

Systemy typu LIMS zapewniają możliwość zarządzania procedurami i metodologiami obowiązującymi w laboratorium i przypisywania ich do procesu obsługi klienta, realizacji pobierania próbek i wykonywania analiz

**Zarządzanie obiegiem dokumentów**

LIMS zawiera mechanizmy wspierające poprawny obieg dokumentów na poszczególnych etapach procesów laboratoryjnych zapewniając poprawny format i treść produkowanych dokumentów oraz dostęp do dokumentów przez pracowników laboratorium.

**Obsługa klienta**

Nowoczesne systemy typu LIMS zawierają również elementy systemów typu CRM (z ang. Customer Relationship Management) oraz ERP (Enterprise Resource Planning), służące do zarządzania obsługą klientów oraz zasobami laboratorium. Dotyczy to m.in. funkcji związanych z obsługą zapytań ofertowych, ofert, zleceń i reklamacji, które wpisują się w proces obsługi klienta laboratorium badawczego, oraz procesów realizacji usług przez pracowników laboratorium.

**Zarządzanie wyposażeniem laboratoryjnym**

LIMS zawiera mechamizmy nadzoru nad wyposażeniem laboratoryjnym oferując funkcje rejestru wyposażenia, planowania przeglądów oraz walidacji sprzetów. Zapamiętuje również rejestr wykonanych czynności związanych z utrzymaniem wyposażenia.

# 4. Integracja modułów LIMS z modułami eksploracji danych

https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-7-430