Wprowadzenie, konsekwencje stosowania modelowania w projektach programistycznych Wykład1

Zofia Kruczkiewicz

Strony internetowe

- **Strona główna:**

http://zio.iiar.pwr.wroc.pl/io.html

- **Strona wykładowcy**

http://zofia.kruczkiewicz.staff.iiar.pwr.wroc.pl/index.php?id=INEK011

- **Karta przedmiotu o kodzie W04ITE-SI0011G, str. 84**

  (Studenci/Programy studiów/Studia I stopnia/Od roku akademickiego 2023/2024/Informatyka techniczna/Studia stacjonarne I stopnia – inżynierskie/Karty przedmiotów kierunkowe

  kpr_ite_st_ii_23.pdf (pwr.edu.pl)

WYDZIAŁ INFORMATYKI I TELEKOMUNIKACJI

KARTA PRZEDMIOTU

Nazwa przedmiotu w języku polskim: **Inżynieria oprogramowania**
Nazwa przedmiotu w języku angielskim: **Software engineering**
Kierunek studiów: **Informatyka techniczna**
Specjalność (jeśli dotyczy): **……..**
Poziom i forma studiów: **I stopień, stacjonarna**
Rodzaj przedmiotu: **obowiązkowy**
Kod przedmiotu: **W04ITE-SI0011G**
Grupa kursów: **TAK**

	Wykład	Ćwiczenia	Laboratorium	Projekt	Seminarium
Liczba godzin zajęć zorganizowanych w Uczelni (ZZU)	30		30		
Liczba godzin całkowitego nakładu pracy studenta (CNPS)	50		100		
Forma zaliczenia	Egzamin		Zaliczenie na ocenę		
Dla grupy kursów zaznaczyć kurs końcowy (X)	X				
Liczba punktów ECTS	**6**				
Liczba punktów odpowiadająca zajęciom o charakterze praktycznym (P)	-		4		
w tym liczba punktów ECTS odpowiadająca zajęciom wymagającym bezpośredniego udziału nauczycieli lub innych osób prowadzących zajęcia (BU)	1,5		1,5		

WYMAGANIA WSTĘPNE W ZAKRESIE WIEDZY, UMIEJĘTNOŚCI I KOMPETENCJI SPOŁECZNYCH

1.

CELE PRZEDMIOTU

C1. Nabycie umiejętności opracowania specyfikacji wymagań oprogramowania za pomocą diagramów przypadków użycia i diagramów czynności języka UML
C2. Nabycie umiejętności wyrażania struktury oprogramowania za pomocą diagramów klas i pakietów tego języka
C3. Zdobycie umiejętności opisywania dynamiki oprogramowania za pomocą diagramów czynności, sekwencji-i maszyn stanowych języka UML
C4. Opanowanie podstaw wiedzy z zakresu kierowania projektami programistycznymi
C5. Nabycie wiedzy z obszaru strukturalnych metod analizy i projektowania
C6. Zdobycie wiedzy z obszarów testowania, weryfikacji i walidacji oprogramowania
C7. Opanowanie umiejętności przygotowywania testów akceptacyjnych i funkcjonalnych przy pomocą narzędzi FitNesse oraz Selenium.
C8. Zdobycie umiejętności przygotowywania testów jednostkowych za pomocą narzędzia JUnit oraz poznanie metody programowania przez testy.

PRZEDMIOTOWE EFEKTY UCZENIA SIĘ

Z zakresu wiedzy:

PEU_W01 – Zna metody specyfikacji wymagań oprogramowania za pomocą diagramów przypadków użycia i diagramów czynności języka UML

PEU_W02 – Zna zasady wyrażania struktury oprogramowania za pomocą diagramów klas i pakietów tego języka; zna kontekst użycia projektowych wzorców strukturalnych i wytwórczych

PEU_W03 - Zna zasady opisywania dynamiki oprogramowania za pomocą diagramów sekwencji, czynności i maszyn stanowych języka UML; zna kontekst użycia projektowych wzorców zachowania

PEU_W04 - Opanowanie podstaw wiedzy z zakresu kierowania projektami programistycznymi

PEU W05 - Nabycie wiedzy z obszaru strukturalnych metod analizy i projektowania

PEU W06 - Zdobycie wiedzy z zakresów testowania, weryfikacji i walidacji oprogramowania

Z zakresu umiejętności:

PEU_U01 - Nabycie umiejętności opracowania specyfikacji wymagań za pomocą diagramów przypadków użycia i diagramów czynności języka UML

PEU_U02 - Nabycie umiejętności wyrażania struktury systemu za pomocą diagramów klas i pakietów tego języka; potrafi zastosować projektowe wzorce wytwórcze i strukturalne zgodnie z ich kontekstem użycia

PEU U03 - Zdobycie umiejętności opisywania dynamiki systemów za pomocą diagramów sekwencji, czynności i maszyn stanowych języka UML; potrafi zastosować projektowe wzorce zachowania zgodnie z ich kontekstem użycia

PEU U04 - Opanowanie umiejętności przygotowywania testów akceptacyjnych oraz funkcjonalnych za pomocą narzędzi FitNesse oraz Selenium

PEU_U05 - Nabycie umiejętności przygotowywania testów jednostkowych za pomocą narzędzia JUnit.

Z zakresu kompetencji społecznych:

PEU_K01 - Umiejętność pracy w dwuosobowym zespole przygotowującym specyfikacje wymagań, modele struktury i dynamiki oprogramowania oraz testów akceptacyjnych, funkcjonalnych i jednostkowych

TREŚCI PROGRAMOWE		
Forma zajęć - wykład		**Liczba godzin**
Wy1	Wprowadzenie, konsekwencje stosowania modelowania w projektach programistycznych	2
Wy2	Diagramy przypadków użycia UML	2
Wy3	Diagramy czynności i syntaktyka diagramów klas UML	2
Wy4	Diagramy klas i pakietów, diagramy sekwencji UML	2
Wy5	Diagramy maszyn stanowych UML, wzorce projektowe oprogramowania	2
Wy6	Koncepcja, projekt i implementacja wielowarstwowego systemu informatycznego	2
Wy7	Testowanie oprogramowania – rodzaje testów, testy akceptacyjne i funkcjonalne, techniki projektowania testów, FitNesse, Selenium	2
Wy8	Testowanie oprogramowania –testy jednostkowe, JUnit, obiekty imitacji, programowanie przez testy	2
Wy9	Wybrane zagadnienia zarządzania projektem	2
Wy10	Modele cyklu życia systemu	2
Wy11	Analiza strukturalna – diagramy ERD	2
Wy12	Analiza strukturalna – diagramy DFD, diagramy stanów	2
Wy13	Zapewnienie jakości w projekcie	2
Wy14	Metody weryfikacji i walidacji	2
Wy15	Bezpieczeństwo i konserwacja oprogramowania	2
	Suma godzin	**30**

Forma zajęć - laboratorium		Liczba godzin
La1	Szkolenie stanowiskowe BHP. Sprawy organizacyjne. Zapoznanie się z wybranym narzędziem UML	2
La2-La4	Wykonanie opisu biznesowego „świata rzeczywistego" projektowanego oprogramowania , definicja wymagań funkcjonalnych i niefunkcjonalnych projektowanego oprogramowania , specyfikacja tych wymagań za pomocą diagramów przypadków użycia	6
La5-La6	Budowa diagramu czynności reprezentującego model biznesowy „świata rzeczywistego" na podstawie wykonanego opisu procesów biznesowych; budowa diagramów czynności reprezentujących scenariusze wybranych przypadków użycia	4
La7	Identyfikacja klas reprezentujących logikę biznesową projektowanego oprogramowania, definicja atrybutów i operacji klas oraz związków między klasami - na podstawie analizy scenariuszy przypadków użycia. Opracowanie diagramów klas i pakietów. Zastosowanie projektowych wzorców strukturalnych i wytwórczych	2
La8-La10	Opracowanie diagramów sekwencji dla wybranych przypadków użycia reprezentujących usługi oprogramowania wynikających również z wykonanych diagramów czynności; definicja operacji klas na podstawie diagramów sekwencji w języku Java. Zastosowanie projektowych wzorców zachowania.	6
La11	Opracowanie diagramu stanów dla wybranej klasy, reprezentującego wpływ różnych przypadków użycia na zmiany stanów tej klasy, modelowanych za pomocą diagramów sekwencji	2
La12	Wprowadzenie do testowania, testy funkcjonalne	2
La13	Testy akceptacyjne z wykorzystaniem narzędzia FitNess	2
La14-La15	Testy jednostkowe z użyciem narzędzia JUnit	4
	Suma godzin	**30**

STOSOWANE NARZĘDZIA DYDAKTYCZNE
N1. Wykład tradycyjny z wykorzystaniem wideoprojektora N2. Ćwiczenia laboratoryjne N3. Konsultacje N4. Praca własna – przygotowanie do ćwiczeń laboratoryjnych N5. Praca własna – samodzielne studia

OCENA OSIĄGNIĘCIA PRZEDMIOTOWYCH EFEKTÓW UCZENIA SIĘ

Oceny (F – formująca (w trakcie semestru), P – podsumowująca (na koniec semestru))	Numer efektu uczenia się	Sposób oceny osiągnięcia efektu uczenia się
F1	PEU_W01 ÷ PEU_W03, PEU_U01 ÷ PEU_U05, PEU_K01	Obserwacja przygotowania do zajęć laboratoryjnych i ich wykonywania
F2	PEU_W01 ÷ PEU_W03	½ egzaminu pisemnego
F3	PEU_W04 ÷ PEU_W06	½ egzaminu pisemnego
P= 0,5F1 + 0,5F3 jeśli F1 >= 4.5 lub P = 0,5F2 + 0,5 F3 jeśli 2.0< F1 < 4.5		

OPIEKUN PRZEDMIOTU (IMIĘ, NAZWISKO, ADRES E-MAIL)
Prof. dr hab. inż. Olgierd Unold, olgierd.unold@pwr.edu.pl

LITERATURA PODSTAWOWA I UZUPEŁNIAJĄCA

LITERATURA PODSTAWOWA:

1. J. Górski, Inżynieria oprogramowania w projekcie informatycznym, Mikom, Warszawa, 1999.
2. S. Wrycza, B. Marcinkowski, K. Wyrzykowski, Język UML 2.0 w modelowaniu systemów informatycznych, Helion, Gliwice, 2005.
3. G. Booch, J. Rumbaugh, I. Jacobson,UML przewodnik użytkownika, Seria: Inżynieria oprogramowania, Warszawa : WNT, 2002.
4. M. Śmiałek, Zrozumieć UML 2.0, Metody modelowania obiektowego, Helion, Gliwice, 2005.
5. M. Fowler, UML w kropelce, Wersja 2.0, LTP, Warszawa, 2005.
6. E. Gamma, R. Helm, R. Johnson, J. Vlissides, Wzorce projektowe. Elementy oprogramowania obiektowego wielokrotnego użytku. Seria: Inżynieria oprogramowania, Warszawa, WNT, 2008,
7. E. Yourdon, Współczesna analiza strukturalna, WNT, Warszawa, 1996.
8. P. Coad, E. Yourdon, Analiza obiektowa, ReadMe, Warszawa, 1994.Jaszkiewicz, Inżynieria oprogramowania, Helion, Warszawa, 1997.
9. Jaszkiewicz, Inżynieria oprogramowania, Helion, Warszawa, 1997.
10. J. Roszkowski, Analiza i projektowanie strukturalne, Helion, Warszawa, 1998.
11. R. Barker, C. Longman, Case Method. Modelowanie funkcji i procesów, WNT, Warszawa, 1996.
12. R. Barker, Case Method. Modelowanie związków encji, WNT, Warszawa, 1996.

LITERATURA UZUPEŁNIAJĄCA:

1. M. Flasiński, Zarządzanie projektami informatycznymi, PWN, Warszawa, 2006
2. S. Snedaker, Zarządzanie projektami IT w małym palcu, Helion, Warszawa, 2007
3. A. Hunt, JUnit: Pragmatyczne testy jednostkowe w javie, Helion 2006.
4. R. Mugridge, W. Cunningham, Fit for Developing Software: Framework for integrated Tests, Prentice Hall, 2005.
5. K. Beck, TDD by example, Addison-Wesley 2002.
6. J. Myers: Sztuka testowania oprogramowania. Helion, Warszawa, 2005.
7. A. Roman, Testowanie i jakość oprogramowania. Modele, techniki, narzędzia, PWN, Warszawa 2015

Literatura podstawowa (LPU) – UML

1. G. Booch, J. Rumbaugh, I. Jacobson, UML przewodnik użytkownika, Seria Inżynieria oprogramowania, WNT, 2001, 2002.
2. S. Wrycza, B. Marcinkowski, K. Wyrzykowski, Język UML 2.0 w modelowaniu systemów informatycznych, Helion, Gliwice, 2005.
3. M. Śmiałek, Zrozumieć UML 2.0, Metody modelowania obiektowego, Helion, Gliwice, 2005.
4. M. Fowler, UML w kropelce, Wersja 2.0, LTP, Warszawa, 2005.

Literatura podstawowa (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 (dodatkowa)*

Literatura uzupełniająca (LU) – Inżynieria oprogramowania

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

Wprowadzenie, konsekwencje stosowania modelowania w projektach programistycznych

1. System informatyczny
2. Inżynieria oprogramowania
3. Tworzenie oprogramowania – podstawowe elementy
4. Korzyści wynikające ze stosowania zasad inżynierii oprogramowania
5. Dodatek do wykładu

Wprowadzenie, konsekwencje stosowania modelowania w projektach programistycznych

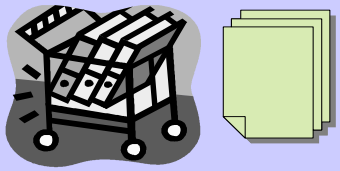
1. **System informatyczny**

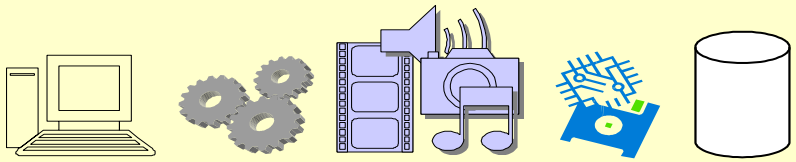
1. System informatyczny

(na podstawie Paul Beynon Davies, Inżynieria systemów informacyjnych)

Nieformalny system informacyjny
zasoby osobowe – ludzie,
zasoby sprzętowe

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.

Wprowadzenie, konsekwencje stosowania modelowania w projektach programistycznych

1. System informatyczny
2. Inżynieria oprogramowania

2. Definicje inżynierii oprogramowania

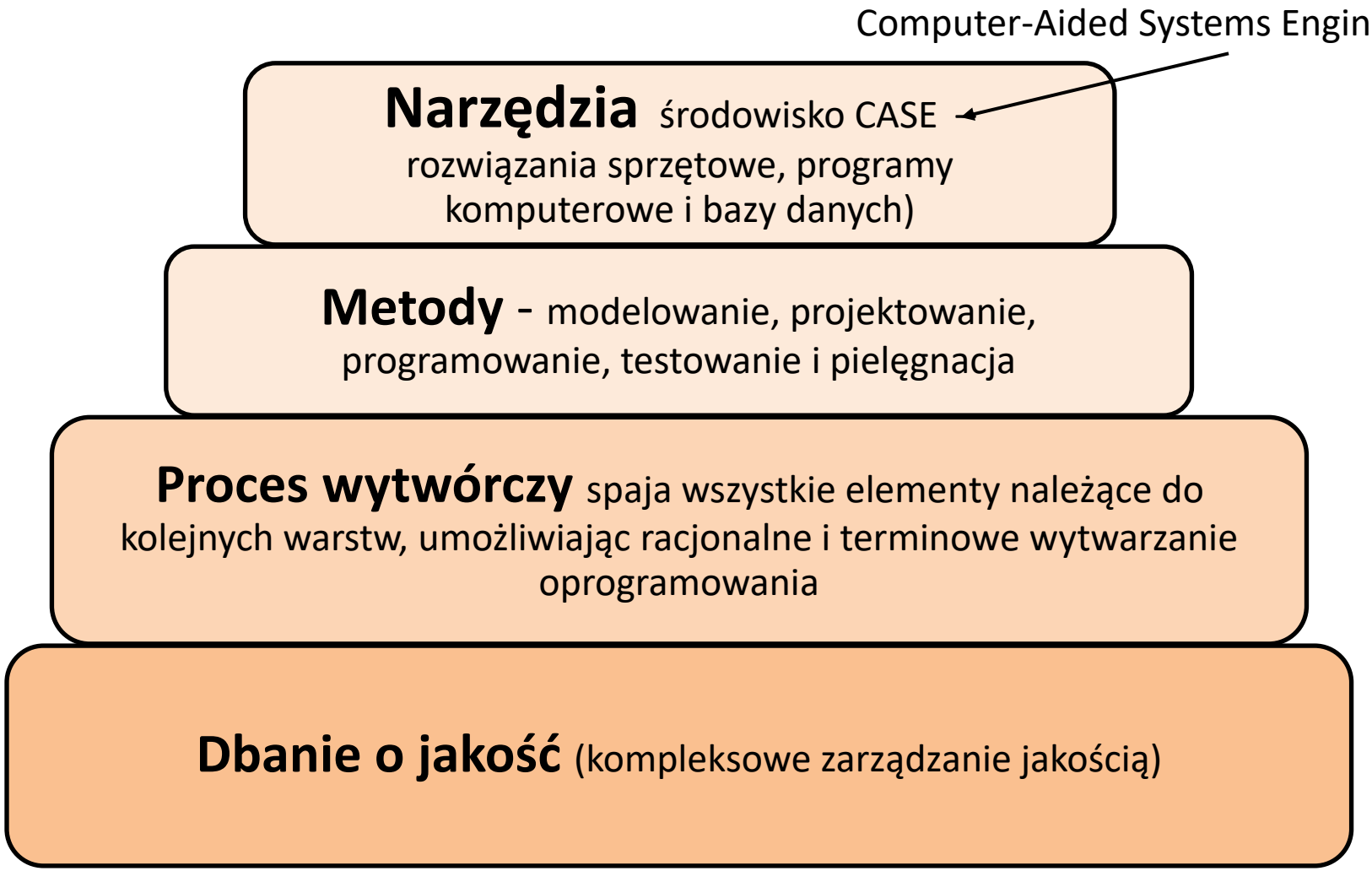
- [Fritz Bauer]

  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(cd). Warstwowe podejście w inżynierii oprogramowania [1LU]



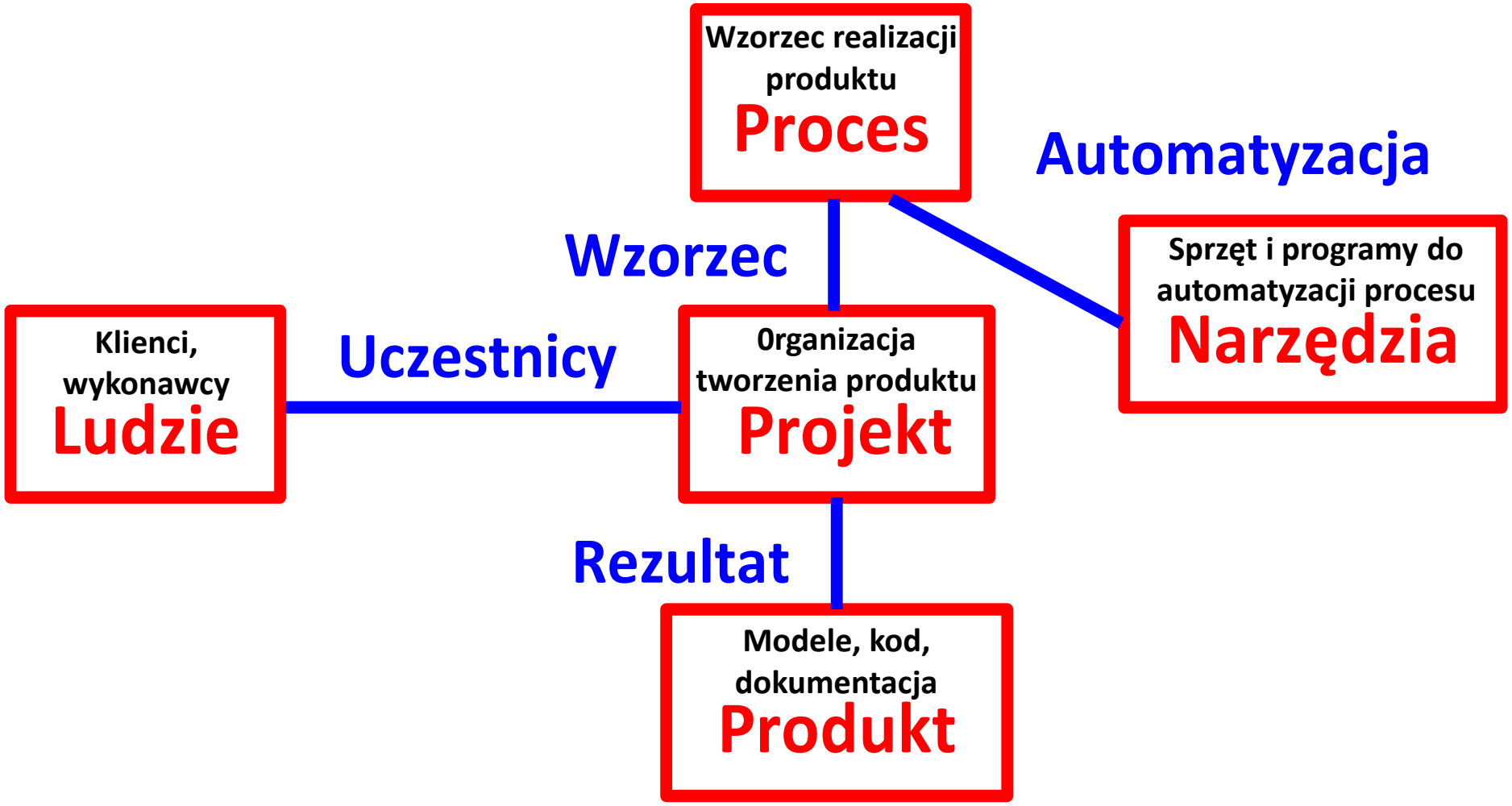
2(cd). Ogólne spojrzenie na inżynierię oprogramowania [1LU]

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)

Wprowadzenie, konsekwencje stosowania modelowania w projektach programistycznych

1. System informatyczny
2. Inżynieria oprogramowania
3. Tworzenie oprogramowania – podstawowe elementy

3. Elementy tworzenia oprogramowania – struktura [3LU]

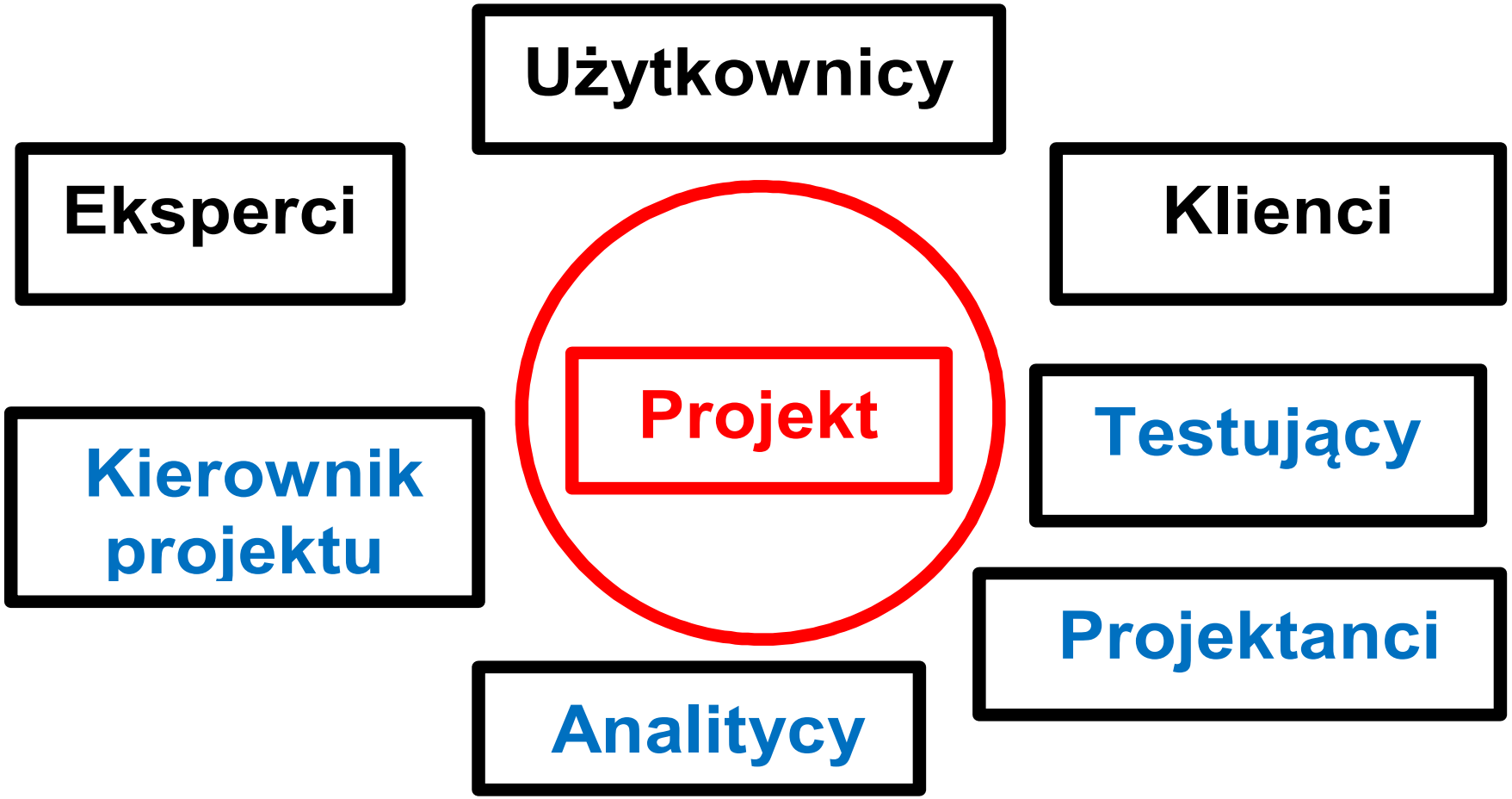


Wprowadzenie, konsekwencje stosowania modelowania w projektach programistycznych

1. System informatyczny
2. Inżynieria oprogramowania
3. Tworzenie oprogramowania –podstawowe elementy

   3.1. Ludzie

3.1. Ludzie [3LU]



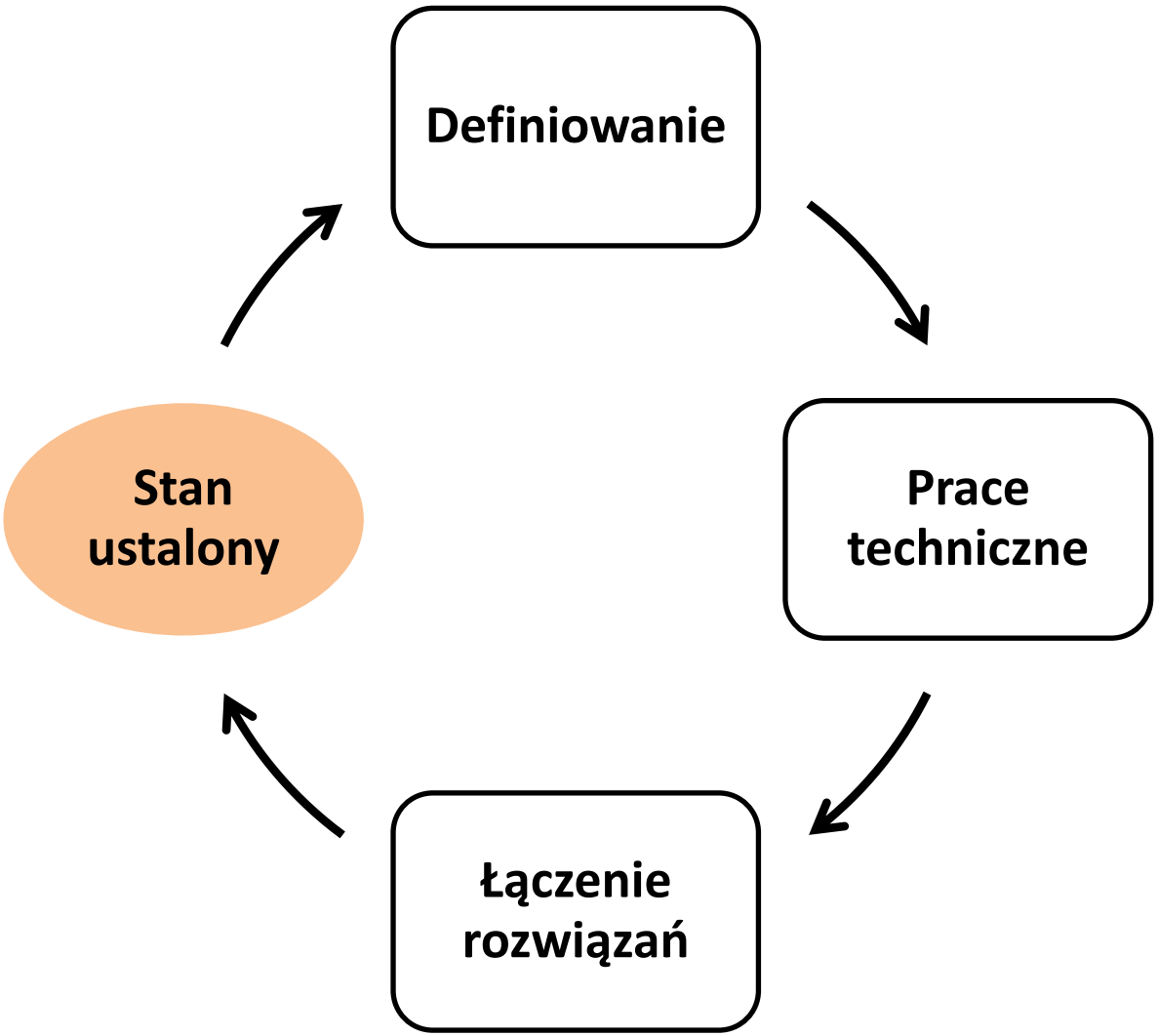
Wprowadzenie, konsekwencje stosowania modelowania w projektach programistycznych

1. System informatyczny
2. Inżynieria oprogramowania
3. Tworzenie oprogramowania –podstawowe elementy
   - 3.1. Ludzie
   - 3.2. Proces

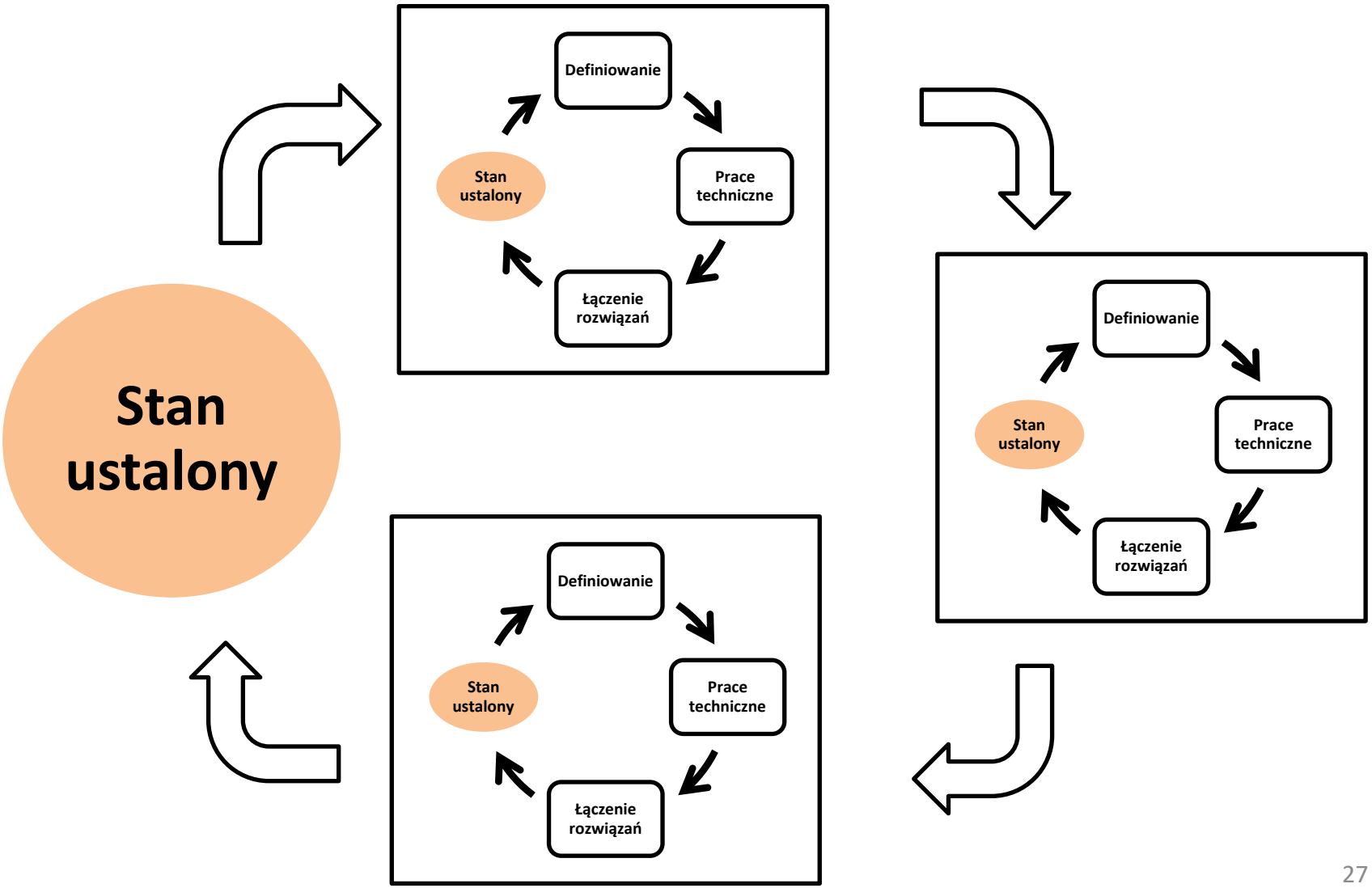
3.2. Proces tworzenia oprogramowania [3LU]

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**
   - Czynniki techniczne

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



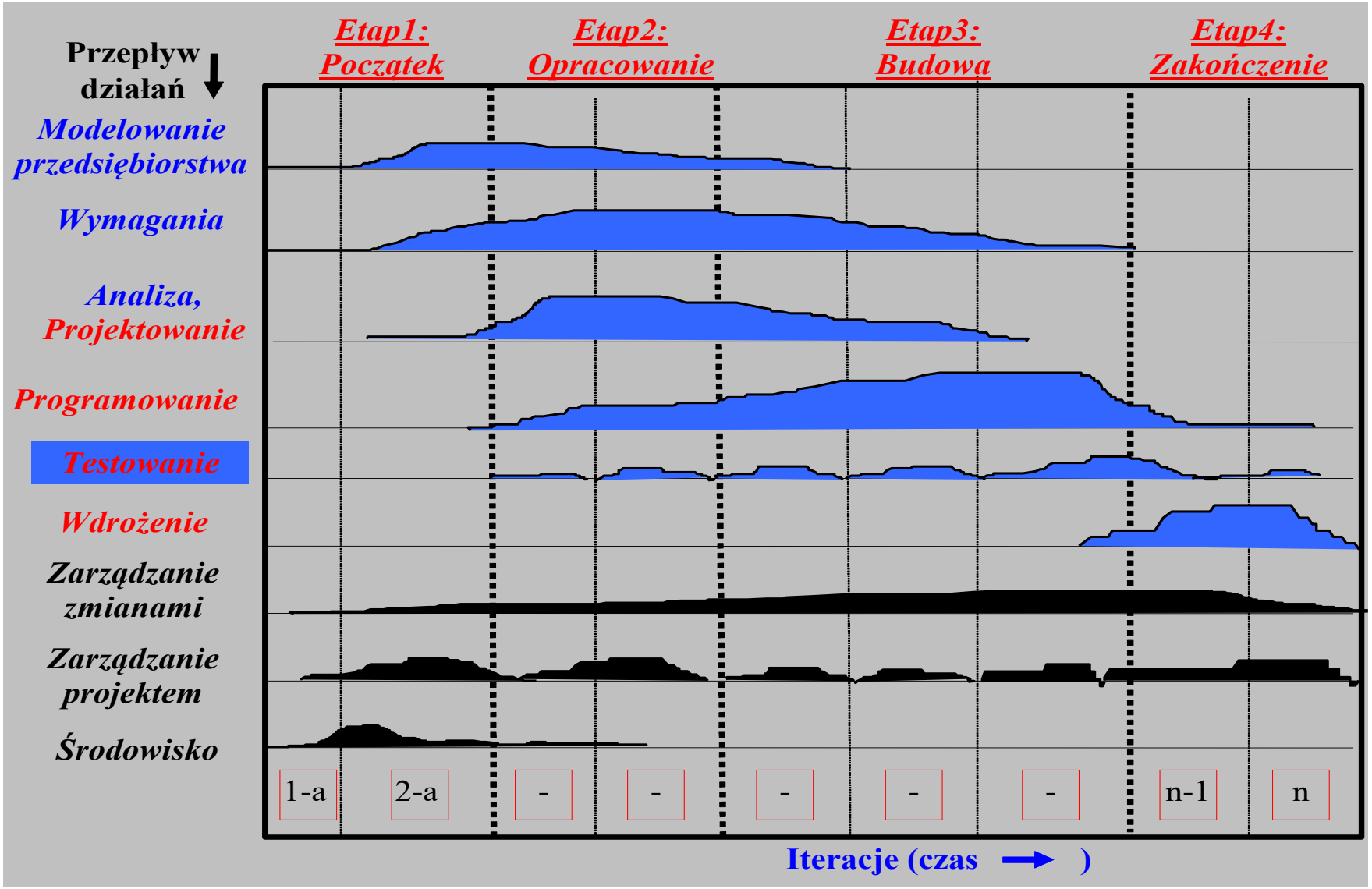
3.2.2. Proces - model iteracyjnego cyklu życia procesu tworzenia oprogramowania [1LU]



3.2.3. Proces - *model* procesu wytwarzania obiektowego oprogramowania - czyli *model* cyklu życia oprogramowania [3LU, 2LPW]

<table>
<tr><th colspan="1">Definiowanie
Modelowanie struktury i dynamiki systemu</th><th colspan="2">Prace techniczne, łączenie rozwiazań
Implementacja systemu, struktury i dynamiki generowanie kodu</th></tr>
<tr><td>Perspektywa koncepcji
co należy wykonać?</td><td>Perspektywa specyfikacji
jak należy używać?</td><td>Perspektywa implementacji
jak należy wykonać?</td></tr>
<tr><td>• model problemu np. przedsiębiorstwa
• wymagania
• analiza (model konceptualny)
• testy modelu</td><td>• projektowanie (model projektowy: architektura sprzętu i oprogramowania; dostęp użytkownika; przechowywanie danych)
• testy projektu</td><td>• programowanie (specyfikacja programu : deklaracje, definicje; dodatkowe struktury danych: struktury „pojemnikowe”, pliki, bazy danych)
• testy oprogramowania
• wdrażanie
• testy wdrażania</td></tr>
</table>

3.2.4. Proces - zunifikowany iteracyjno- przyrostowy proces tworzenia obiektowego oprogramowania – kiedy należy wykonać? [3LU]



3.2.4(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 niezbędnej do opracowania systemu

3.2.5. Proces tworzenia obiektowego oprogramowania

Perspektywy tworzenia obiektowych systemów informacyjnych - perspektywy identyfikacji obiektów [2LPW]

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

  - obiekt jest zbiorem różnego rodzaju odpowiedzialności

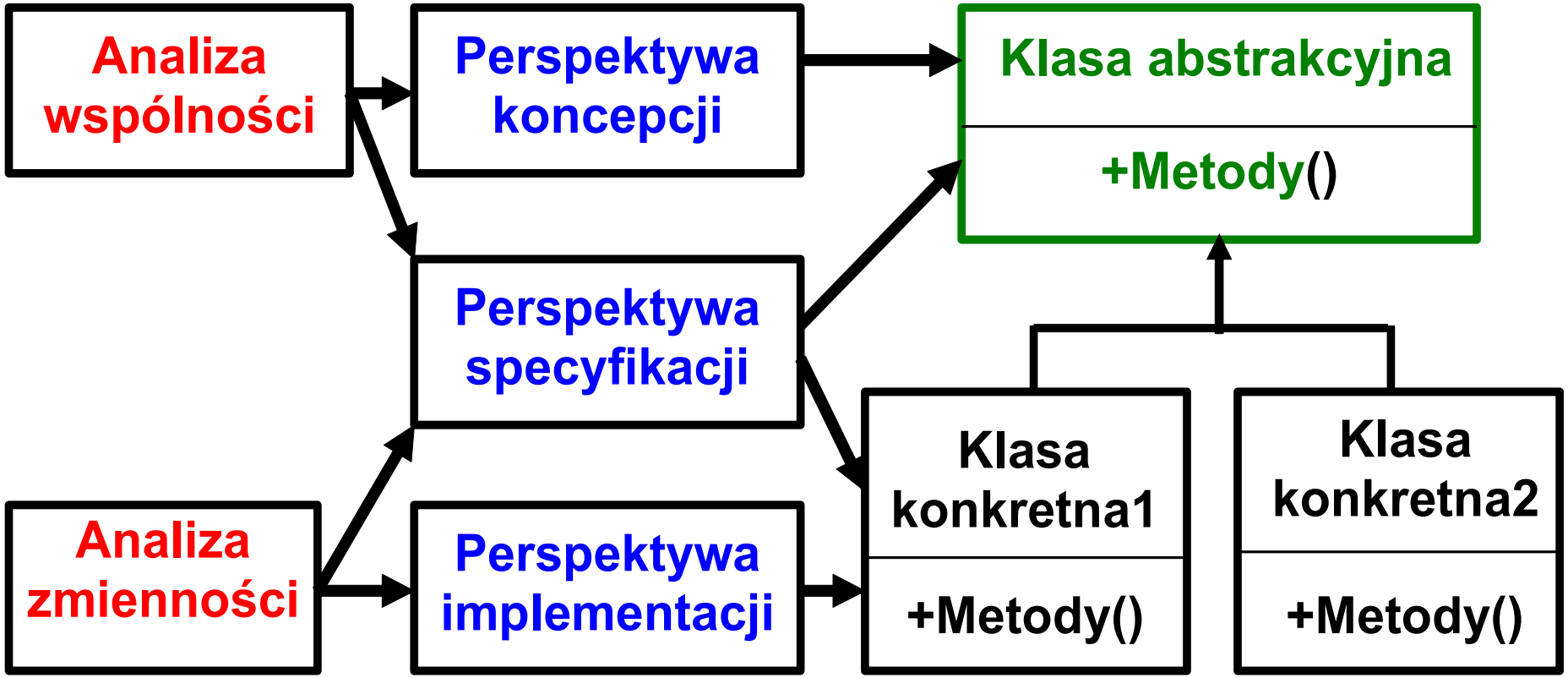
- **Perspektywa specyfikacji** (*modelu* projektowego)

  - 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

3.2.5. Proces tworzenia obiektowego oprogramowania - metoda identyfikacji obiektów i klas [2LPW]



Związki między perspektywą specyfikacji, koncepcji i implementacji

3.2.5. Proces tworzenia obiektowego oprogramowania - jak wykonać? [2LPW]

Perspektywy tworzenia obiektowych systemów informacyjnych - uzupełnienie

- **Perspektywa tworzenia i zarządzania obiektami**

(model implementacji) **-** obiekt A w roli fabryki obiektów tworzy obiekt B i/lub zarządza obiektem

- **Perspektywa używania obiektów**

(model implementacji) - obiekt A tylko używa obiektu B – nie może go jednocześnie tworzyć. Opiera się na hermetyzacji i polimorfizmie obiektu B

3.2.6. Proces - rozłożenie pracy w czasie (1LU)

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

3.2.7. Proces - *modele* procesów wytwórczych (1LU)

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

Wprowadzenie, konsekwencje stosowania modelowania w projektach programistycznych

1. System informatyczny
2. Inżynieria oprogramowania
3. Tworzenie oprogramowania –podstawowe elementy
   - 3.1. Ludzie
   - 3.2. Proces
   - 3.3. Produkt

3.3. 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**

3.3.1. Produkt - oprogramowanie – byt logiczny

Oprogramowanie to:

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

3.3.2. 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 komponentów jest tworzona od nowa

3.3.3. 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.

3.3.4. Produkt - dziedziny zastosowań oprogramowania

- Oprogramowanie systemowe
- Systemy czasu rzeczywistego
- Systemy informacyjne dla przedsiębiorstw
- Oprogramowanie inżynierskie i naukowe
- Systemy wbudowane
- Oprogramowanie komputerów osobistych
- Oprogramowanie internetowe
- Sztuczna inteligencja

3.3.5. Produkt – architektura oprogramowania

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

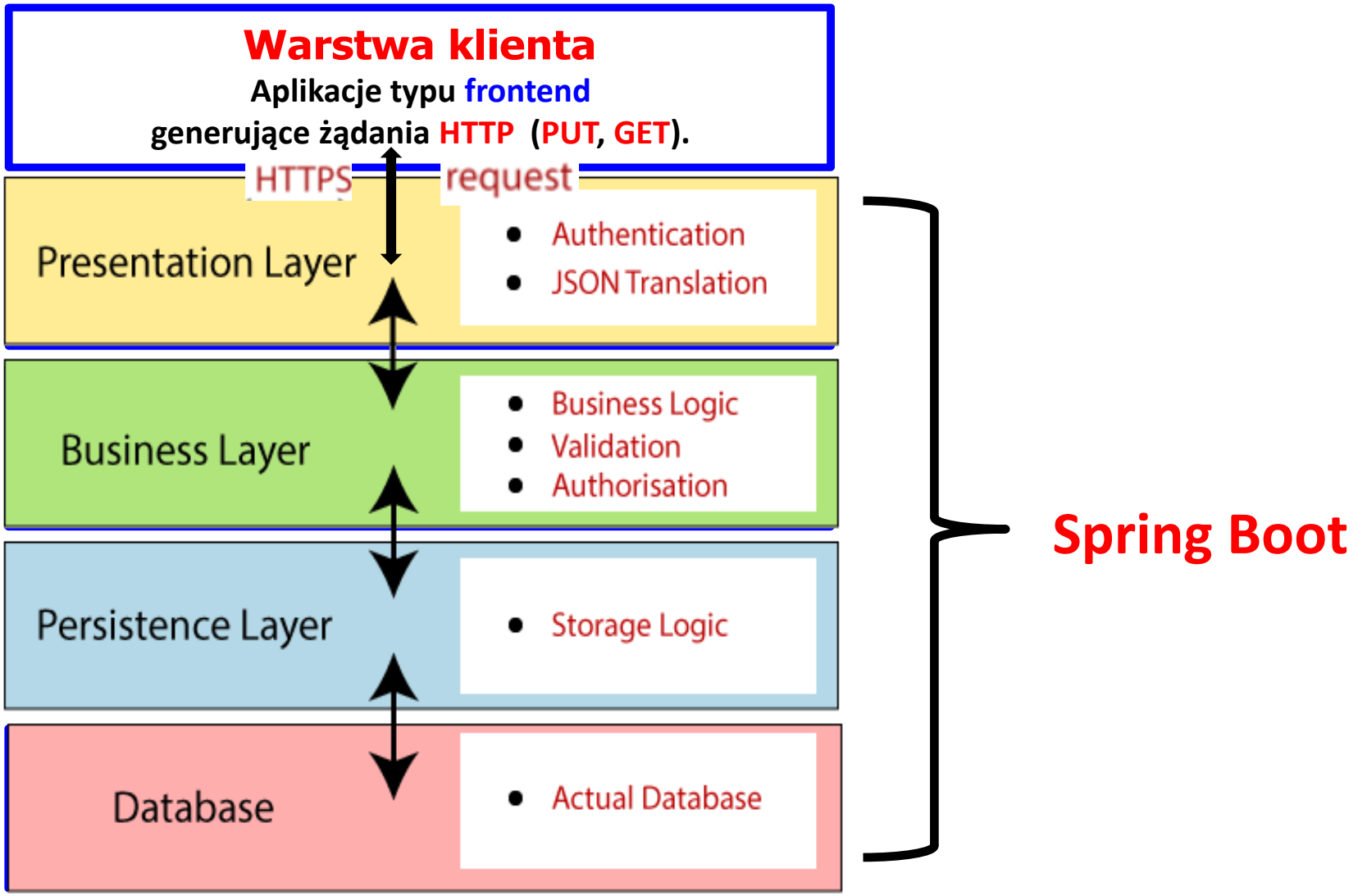
Adaptery zasobów, systemy zewnętrzne, mechanizmy zasobów, przepływ sterowania

Warstwa zasobów
Bazy danych, systemy zewnętrzne i pozostałe zasoby

Zasoby, dane i usługi zewnętrzne

3.3.5. (cd) Produkt – architektura oprogramowania

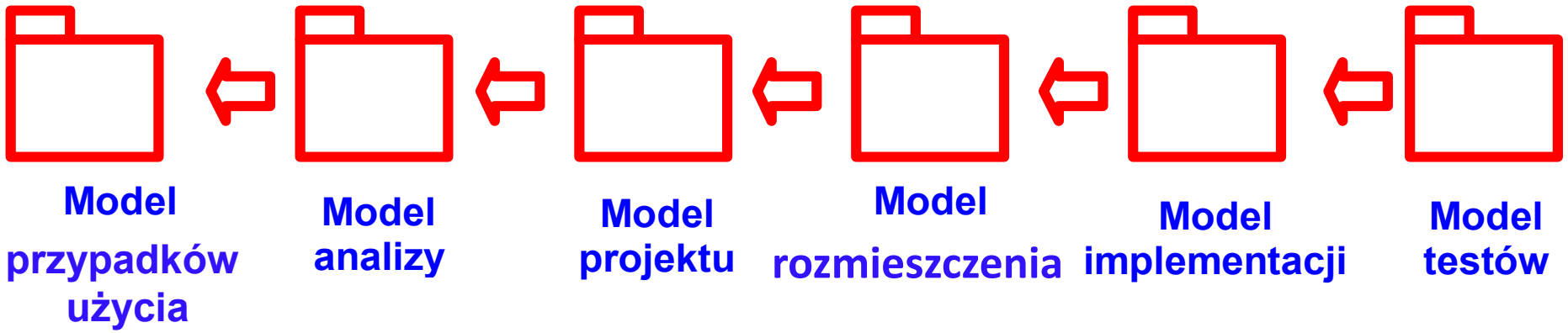
Pięciowarstwowy model logicznego rozdzielania zadań aplikacji
(wg. Spring Boot Architecture - javatpoint)



3.3.6. Produkt – modele [3LU]

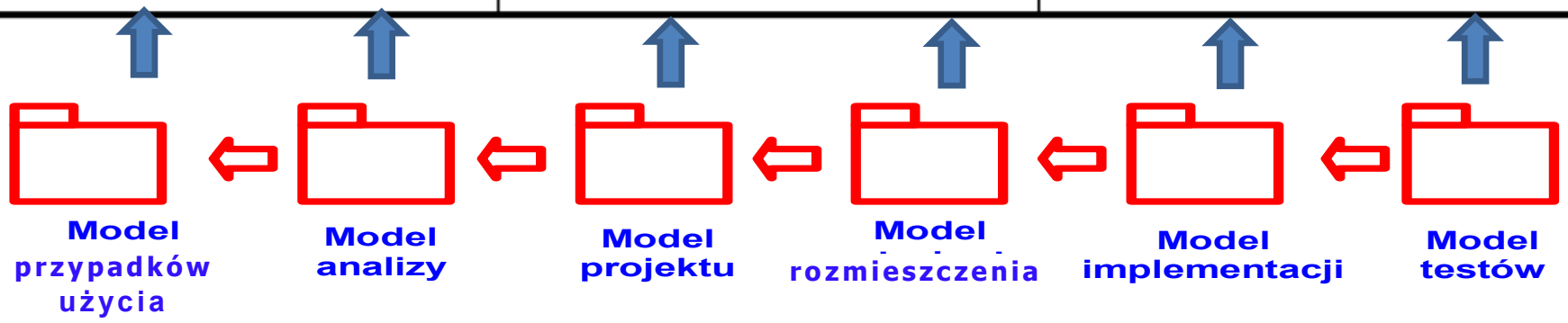
Modele:

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



3.3.7. Produkt – modele i proces tworzenia obiektowego oprogramowania [3LU]

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



3.3.8. Produkt - języki modelowania

1. **Języki modelowania poprzedzające UML jako odpowiedź na obiektowe języki 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**
2. **Modelowanie struktury aplikacji, zachowania i architektury, ale również procesów biznesowych i struktury danych - Unified Modeling Language UML**
   1. zunifikowany pół-formalny język modelowania **powstał w 1997,**
   2. autorzy: Grady Booch, James Rumbaugh, Ivar Jacobson
   3. obecnie rozwijany przez OMG (Object Management Group) – **UML 2.5.1** http://www.uml.org (od 2017)

3.3.8(cd). Produkt - języki modelowania

3. Modelowanie usług sieciowych

WSBPEL (*Web Services Business Process Execution Language)* - model procesu współpracy zbioru aplikacji (model interakcji usług sieci Web) w celu osiągnięcia celu biznesowego, który jednocześnie jest usługą sieciową. Oparty jest na następujących technologiach:

SOAP definiuje protokół XML przesyłania wiadomości zapewniający podstawową interoperacyjność usług, WSDL (*Web Services Description Language*) wprowadza wspólną gramatykę do opisu usług, UDDI (*Universal Description, Discovery, and Integration*) zapewnia infrastrukturę wymaganą do realizacji usług.

Komitet Techniczny (TC) organizacji OASIS (*Organization for the Advancement of Structured Information Standards*) zarządza procesem rozwoju języka WSBPEL:

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

4. Modelowanie procesów biznesowych

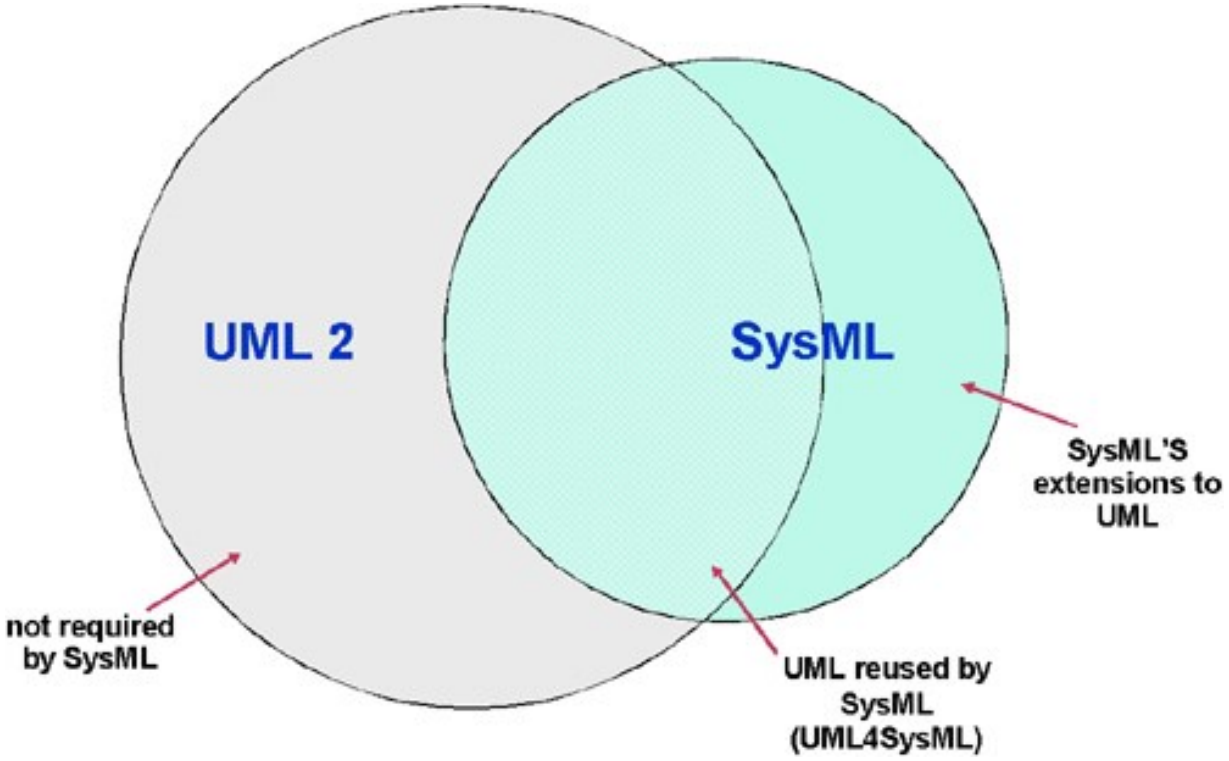
BPMN (*Business Process Model and Notation*) rozwijany przez OMG http://www.bpmn.org do modelowania procesów biznesowych z użyciem diagramów przepływu procesów **BPD** (*Business Process Diagram*) podobnych do diagramów aktywności UML.

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

3.3.8(cd). Produkt - języki modelowania

6. Modelowanie złożonych systemów zawierających sprzęt, oprogramowanie, informację, personel, procedury i ułatwienia - SysML V2.0 Beta 1 (lipiec 2023) jako podzbiór języka UML (p.2).

https://www.omgsysml.org/index.htm



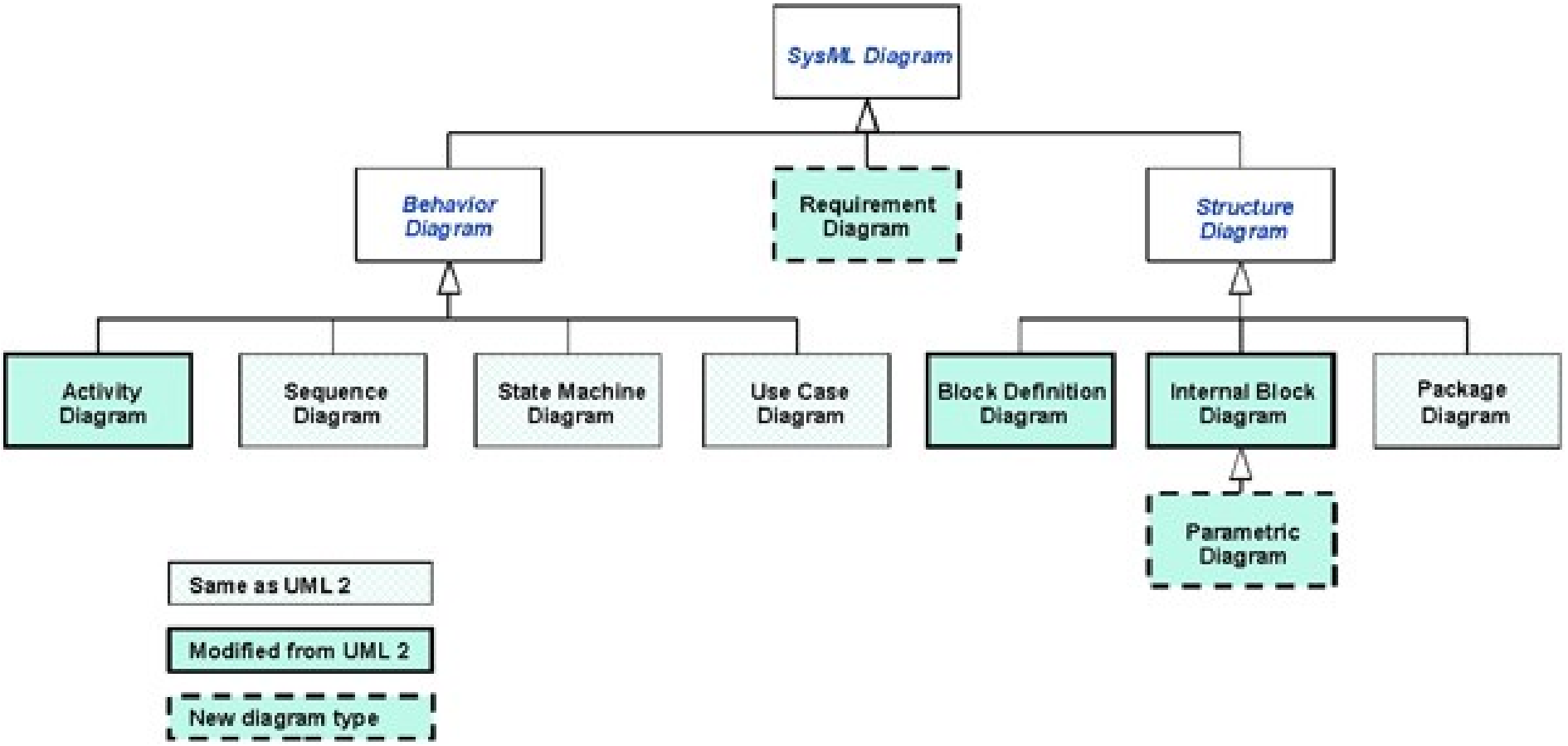
MBSE(*Model-Based Systems Engineering*) – modelowanie za pomocą języka modelowania **SysML** (***Systems Modeling Language***), obecnie OMG SysML.

SysML jest dialektem języka UML

https://www.omgsysml.org/what-is-sysml.htm

3.3.8(cd). Produkt - języki modelowania

6. (cd) diagramy SysML v2.0 BETA 1

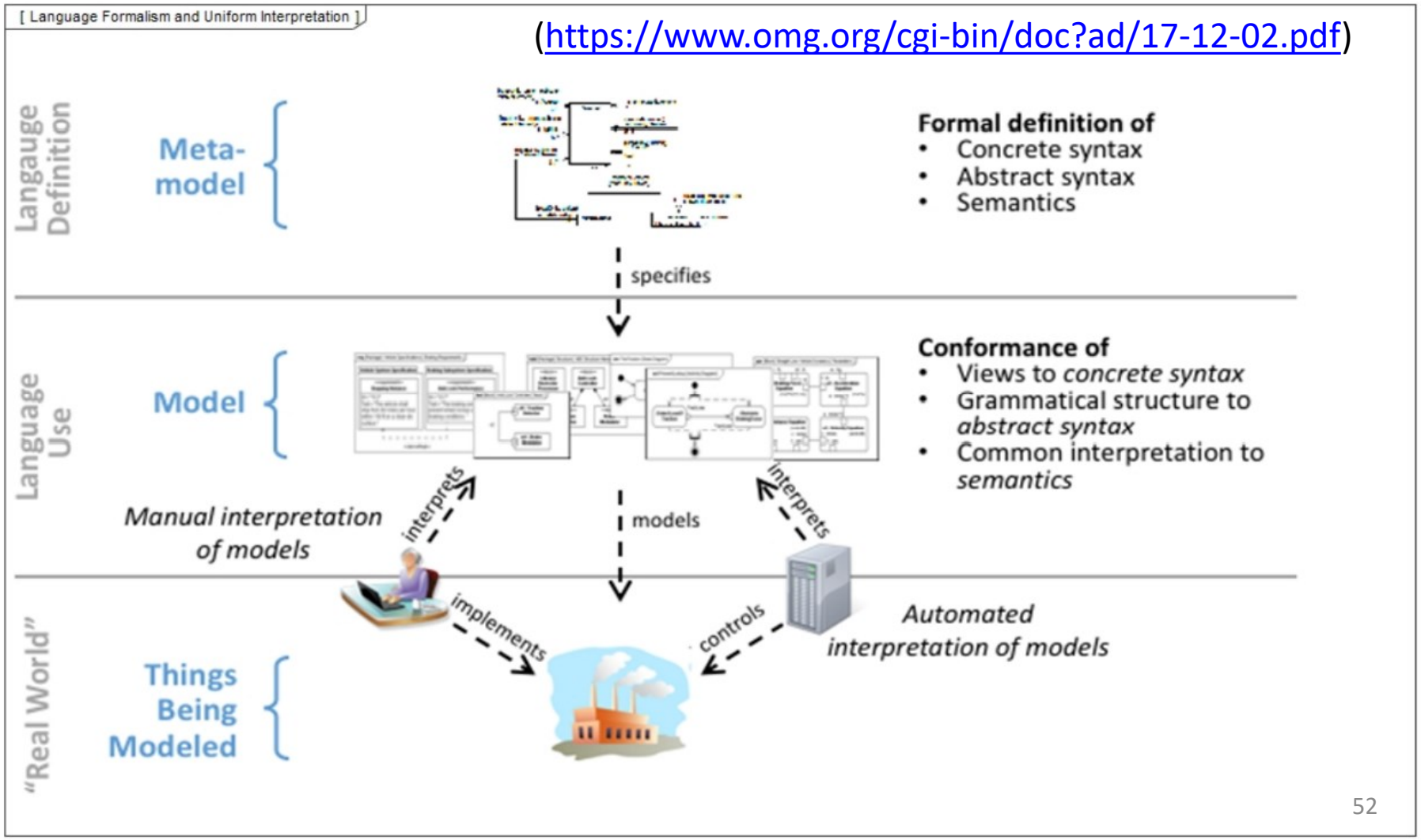


https://www.omgsysml.org/what-is-sysml.htm

3.3.8(cd). Produkt – języki modelowania

6. (cd) Formalizm językowy i interpretacja SysML v2

Związek między formalizmem językowym SysML a modelowanymi rzeczami ze „świata rzeczywistego”.



3.3.9. Produkt – język modelowania UML 2.5.1 (3.3.8 – p.2)

Diagramy UML wspierające zunifikowany iteracyjno - przyrostowy proces tworzenia obiektowego oprogramowania [1LPU]

https://www.omg.org/spec/category/software-engineering/

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

3.3.9. (cd) Produkt – język modelowania UML 2.5.1 (3.3.8 – p.2)

Diagramy UML wspierające zunifikowany iteracyjno - przyrostowy proces tworzenia oprogramowania [1LPU]

Diagramy UML modelowania zachowania

2.1. Diagramy przypadków użycia

2.2. Diagramy aktywności

2.3. Diagramy stanów

2.4. Diagramy komunikacji

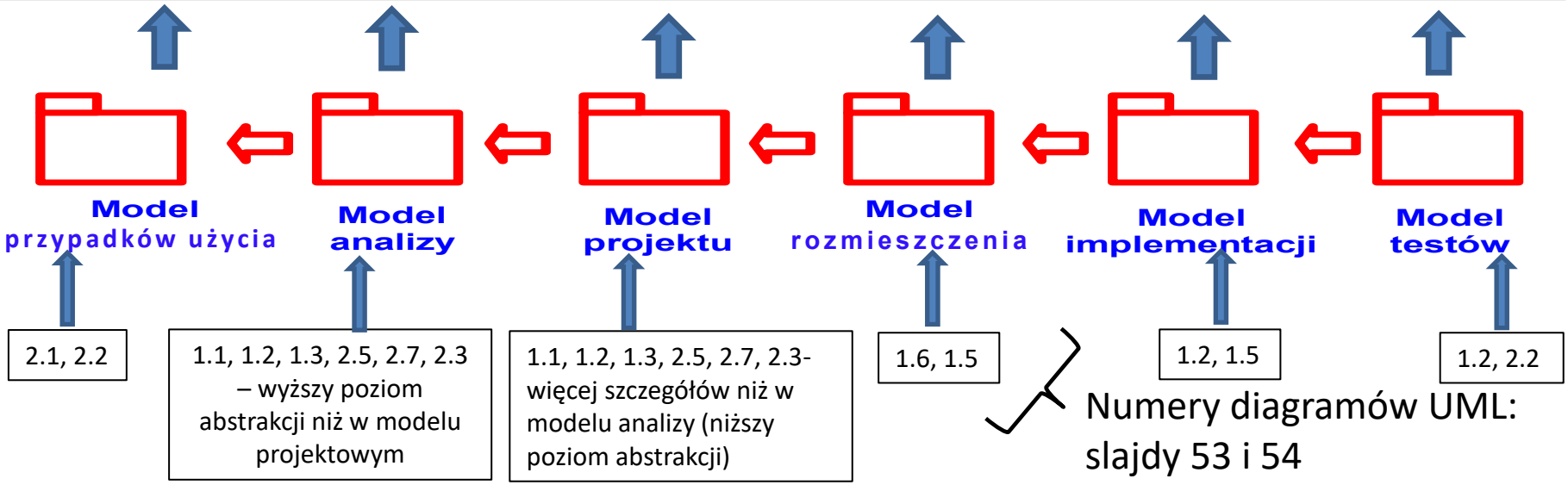
2.5. Diagramy sekwencji

2.6. Diagramy czasu

2.7. Diagramy interakcji

3.3.9. (cd). Produkt - diagramy UML – modele i proces tworzenia obiektowego oprogramowania

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



3.3.9. (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)

3.3.10. Produkt - język UML do modelowania systemów wbudowanych czasu rzeczywistego

jako UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded Systems v 1.3 (RTES) - June 2023

1) Modelowanie oprogramowania wbudowanego działającego w czasie rzeczywistym (RTE) uwzględniając jego niefunkcjonalne właściwości (Non-Functional Properties - NEP) np. czas, energia
2) Modelowanie aspektów sprzętu RTE
3) Architektura systemu - uwzględnia 1) i 2) i modeluje ich alokacje
4) Analiza wydajności RTES
5) Analiza harmonogramu RTES
6) Dostawca infrastruktury – np. określa potrzebne usługi RTES, tworzy biblioteki niezbędne do modelowania RT na wyższym poziomie abstrakcji
7) Metodologia – dobór właściwych narzędzi do modelowania MARTE

https://www.omg.org/spec/MARTE/1.3/PDF

3.3.10. (cd) Produkt - języki modelowania

UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded Systems v 1.3 (https://www.omg.org/spec/MARTE/1.3/PDF, June 2023)

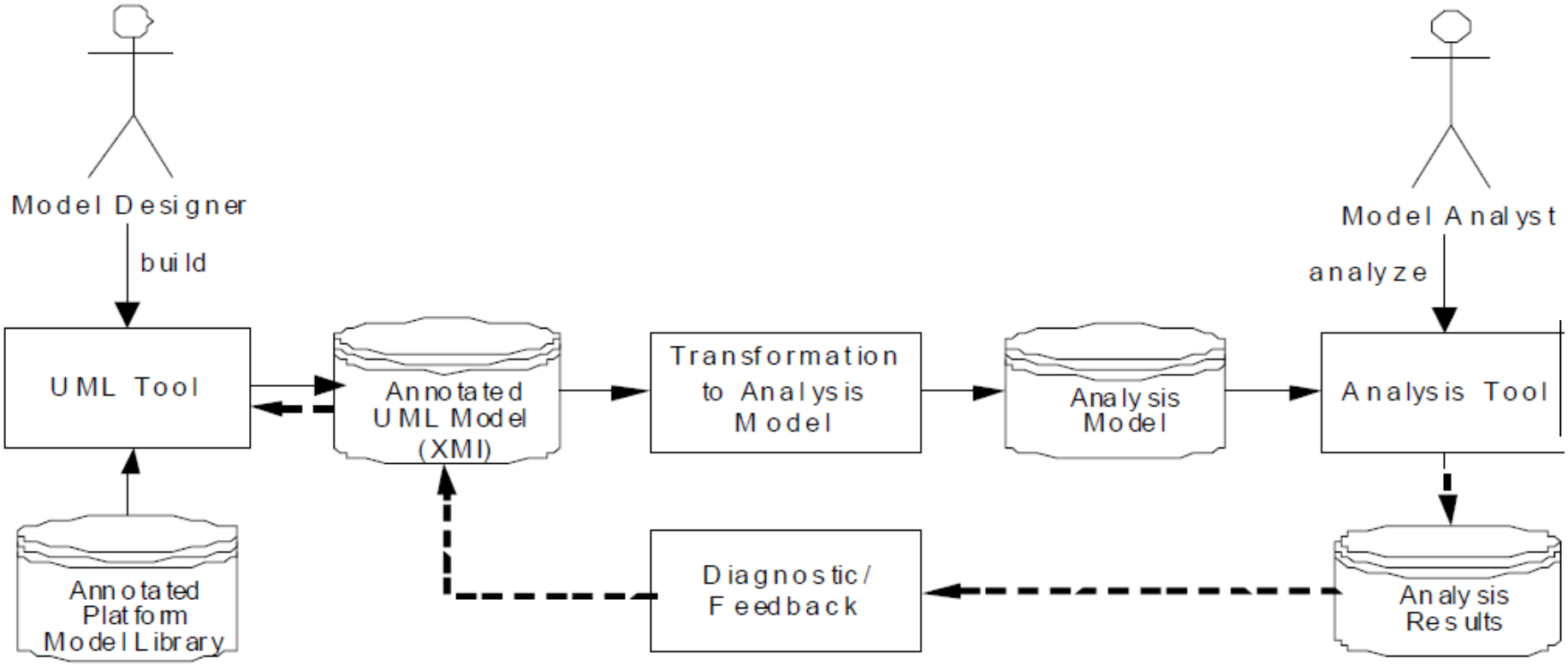


Figure 6.5 - A Tool Chain for Carrying out Analysis of a Model

3.3.10. (cd) Produkt - języki modelowania

UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded Systems v 1.3

Korzyści wynikające z korzystania z tego profilu to między innymi:

1) Zapewnienie wspólnego sposobu modelowania aspektów sprzętowych i programowych RTES w celu ulepszenia komunikacji między programistami.
2) Umożliwienie interoperacyjności między narzędziami programistycznymi używanymi do specyfikacji, projektowania, weryfikacji, generowania kodu itp.
3) Wspieranie budowy modeli, które mogą być wykorzystane do prognozowania ilościowego w czasie rzeczywistym i wbudowane funkcje systemów uwzględniające zarówno cechy sprzętu, jak i oprogramowania.

3.3.11. Produkt a proces wytwórczy [1LU]

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

3.3.12. Produkt - opinie z ankiet o stosowaniu UML

(An industrial case study on the use of UML in software maintenance and its perceived benefits and hurdles,
Ana M. Fernández-Sáez1, Michel R.V. Chaudron2, Marcela Genero
Published online: 16 March 2018, Springer Science+Business Media, LLC, part of Springer Nature 2018)

Zastosowania UML	Rezultaty - model programu i dokumentacja
1. Praktyki związane z korzystania UML związane z tworzeniem i konserwacją oprogramowania: • komunikacja i uzyskanie przeglądu projektu, • tworzenie projektu i jego zrozumienie,	• Ograniczona aktualizacja modeli do aktualnej wersji UML • Ważniejszy jest „cel projektowy” niż „plan projektowy” – ważne są kluczowe części projektu, stąd abstrahowanie od szczegółów projektu • Ulepszanie projektu przedwdrożeniowego może prowadzić do pogorszenia synchronizacji modelu z wykonanym już kodem • Stosowanie jedynie modeli inżynierii odwrotnej może zniekształcać główne wymagania oprogramowania
2. Ocena korzyści z korzystania z diagramów UML związanych z tworzeniem i konserwacją oprogramowania	**Korzyści dotyczące procesu tworzenia programów** • Lepsza komunikacja, szczególnie podczas globalnego tworzenia programu lub stosowania outsourcing • Ulepszanie projektu przed wdrożeniem – łatwość recenzowania • Pozwala utrwalać wiedzę dotyczącą utworzonego programu • Umożliwia diagnozowanie problemów programu **Korzyści dotyczące jakości oprogramowania** • 89% inżynierów uważa, że UML poprawia jakość programu dzięki korzyściom wynikającym z procesu jego tworzenia.

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

Zastosowania UML	Rezultat - model programu i dokumentacja
3. Przeszkody dotyczące stosowania UML w tworzeniu i konserwacji oprogramowania	• Błędna interpretacja wytycznych Manifestu Agile („działające oprogramowanie zamiast obszernej dokumentacji”, „kod źródłowy to dokumentacja”) • Niepoprawnie organizowane zespoły do tworzenia programów – brak projektantów • Podział ról w tworzeniu oprogramowania: projektanci tworzą modele UML, a programiści kod, stąd różnice w zrozumieniu i roli diagramów UML uczestników zespołu • Trudności w ustaleniu poziomu szczegółowości diagramów UML i ich odwzorowania w kodzie programu
4. Najlepsze praktyki dotyczące modelowania i korzystania z diagramów UML	• Cel - użycie diagramów UML i wynikający z niego poziom szczegółowości • Proces – aktualizacja dokumentacji powiązanej ze stanem aktualnego kodu • Narzędzia – umożliwiające poprawną i szybką aktualizację dokumentacji powiązanej z aktualnym stanem kodu • Szkolenia związane z UML– szczególnie programistów • Standaryzacja i zarządzanie - na poziomie centralnym firmy w celu ujednolicenia praktyk stosowania UML

3.3.12 (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, Published in EESSMod@MoDELS 2013)

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*)
	Nie egzekwuje oddzielania tego, **co należy zrobić** od tego, **jak należy zrobić**

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

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 46, 47 i 55).

Wprowadzenie, konsekwencje stosowania modelowania w projektach programistycznych

1. System informatyczny
2. Inżynieria oprogramowania
3. Tworzenie oprogramowania –podstawowe elementy
   - 3.1. Ludzie
   - 3.2. Proces
   - 3.3. Produkt
   - 3.4. Narzędzia

3.4. 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

Wprowadzenie, konsekwencje stosowania modelowania w projektach programistycznych

1. System informatyczny
2. Inżynieria oprogramowania
3. Tworzenie oprogramowania –podstawowe elementy
   - 3.1. Ludzie
   - 3.2. Proces
   - 3.3. Produkt
   - 3.4. Narzędzia
   - 3.5. Projekt

3.5. 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

Wprowadzenie, konsekwencje stosowania modelowania w projektach programistycznych

1. System informatyczny
2. Inżynieria oprogramowania
3. Tworzenie oprogramowania – podstawowe elementy
4. Korzyści wynikające ze stosowania inżynierii oprogramowania

4. Korzyści wynikające ze stosowania zasad inżynierii oprogramowania

na przykładzie zastosowania

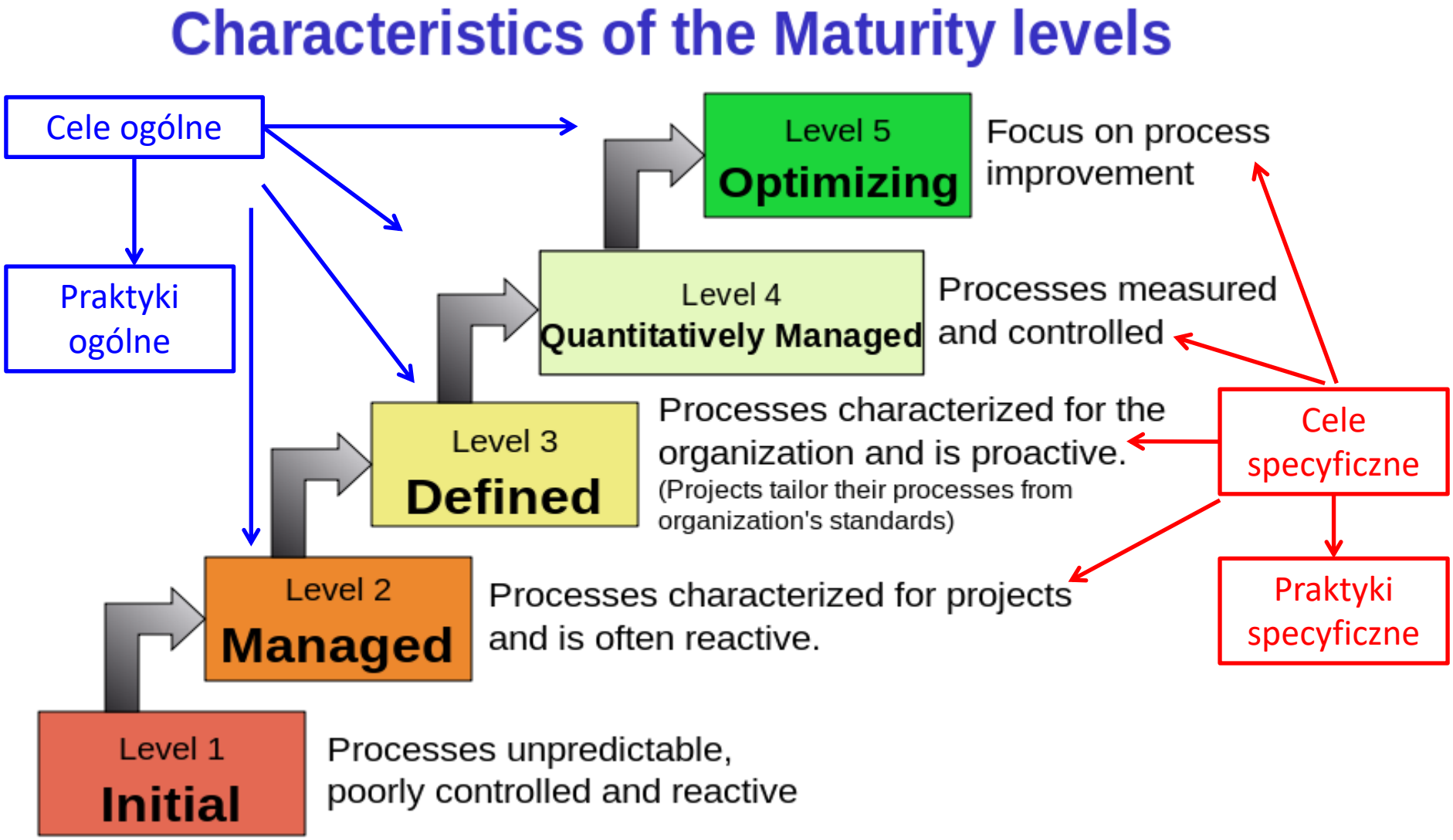
CMMI (Capability Maturity Model Integration) - wytycznych dla poprawy jakości **produktu** i integracji **procesu** z wykorzystaniem **UML**

1. https://cmmiinstitute.com/cmmi/intro
2. https://www.tutorialspoint.com/cmmi/
3. https://en.wikipedia.org/wiki/Capability_Maturity_Model_Integration
4. https://docs.microsoft.com/en-us/azure/devops/boards/work-items/guidance/cmmi/guidance-background-to-cmmi

4.1. CMMI - Capability Maturity Model Integration

- **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 framework 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

4.2. Poziomy dojrzałości modelu CMMI



4.3. Lista organizacji korzystających z CMMI w procesie tworzenia oprogramowania

(wg CMMI Performance Report Slides (cmmiinstitute.com)) – April 2023

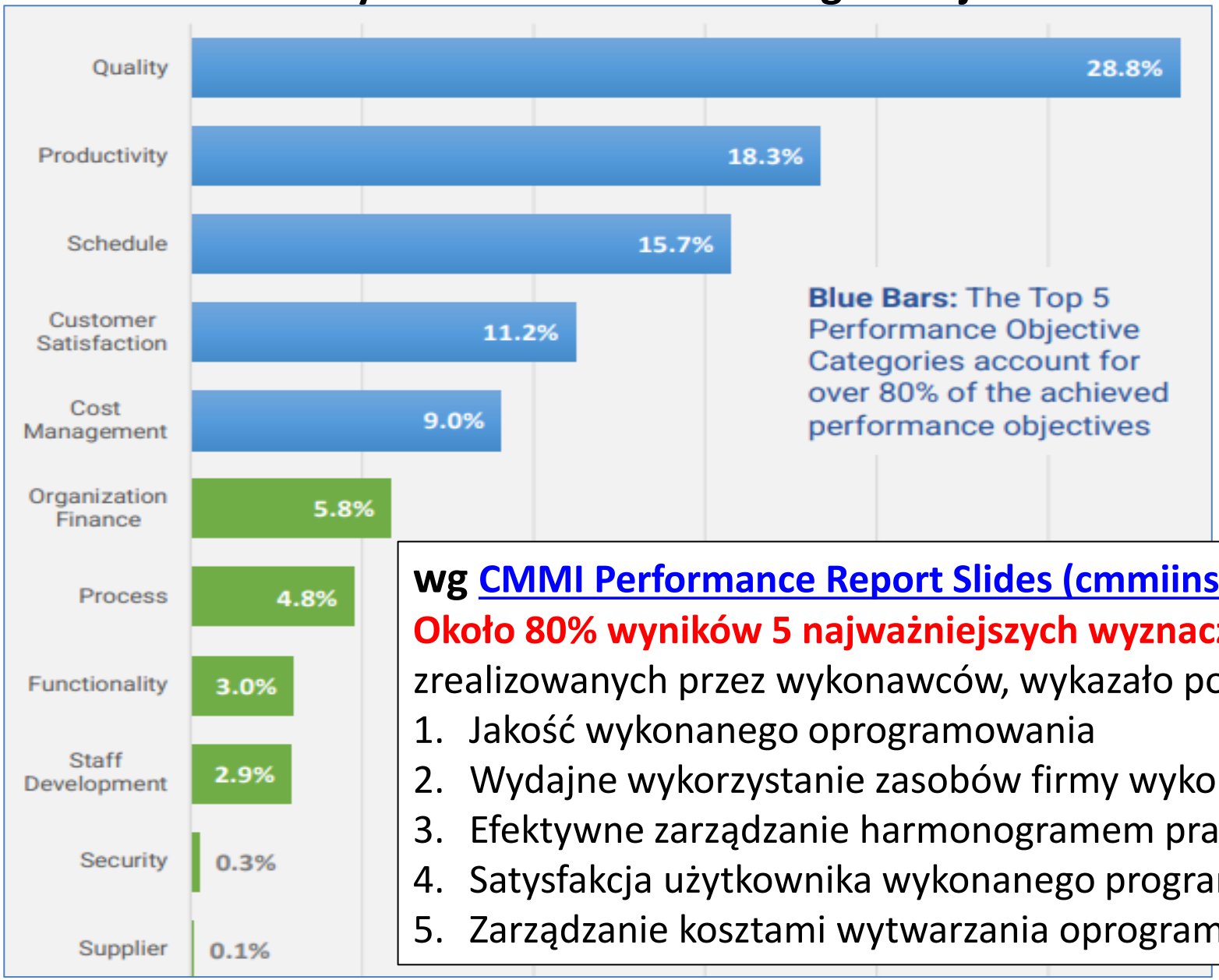
CMMI Institute – od 2016 r. współpracuje z globalnym stowarzyszeniem non-profit ***ISACA***

ISACA*-*Information Systems Audit and Control Association

(*10,000 organizacji w 106 krajach*)

4.4. Poprawa wyników celów procesu CMMI

- ocena 33 272 wyników celów w 8 866 organizacjach w latach 2019 – 2022



wg CMMI Performance Report Slides (cmmiinstitute.com)

Około 80% wyników 5 najważniejszych wyznaczonych celów, zrealizowanych przez wykonawców, wykazało poprawę:

1. Jakość wykonanego oprogramowania
2. Wydajne wykorzystanie zasobów firmy wykonawców
3. Efektywne zarządzanie harmonogramem prac
4. Satysfakcja użytkownika wykonanego programu
5. Zarządzanie kosztami wytwarzania oprogramowania

Wprowadzenie, konsekwencje stosowania modelowania w projektach programistycznych

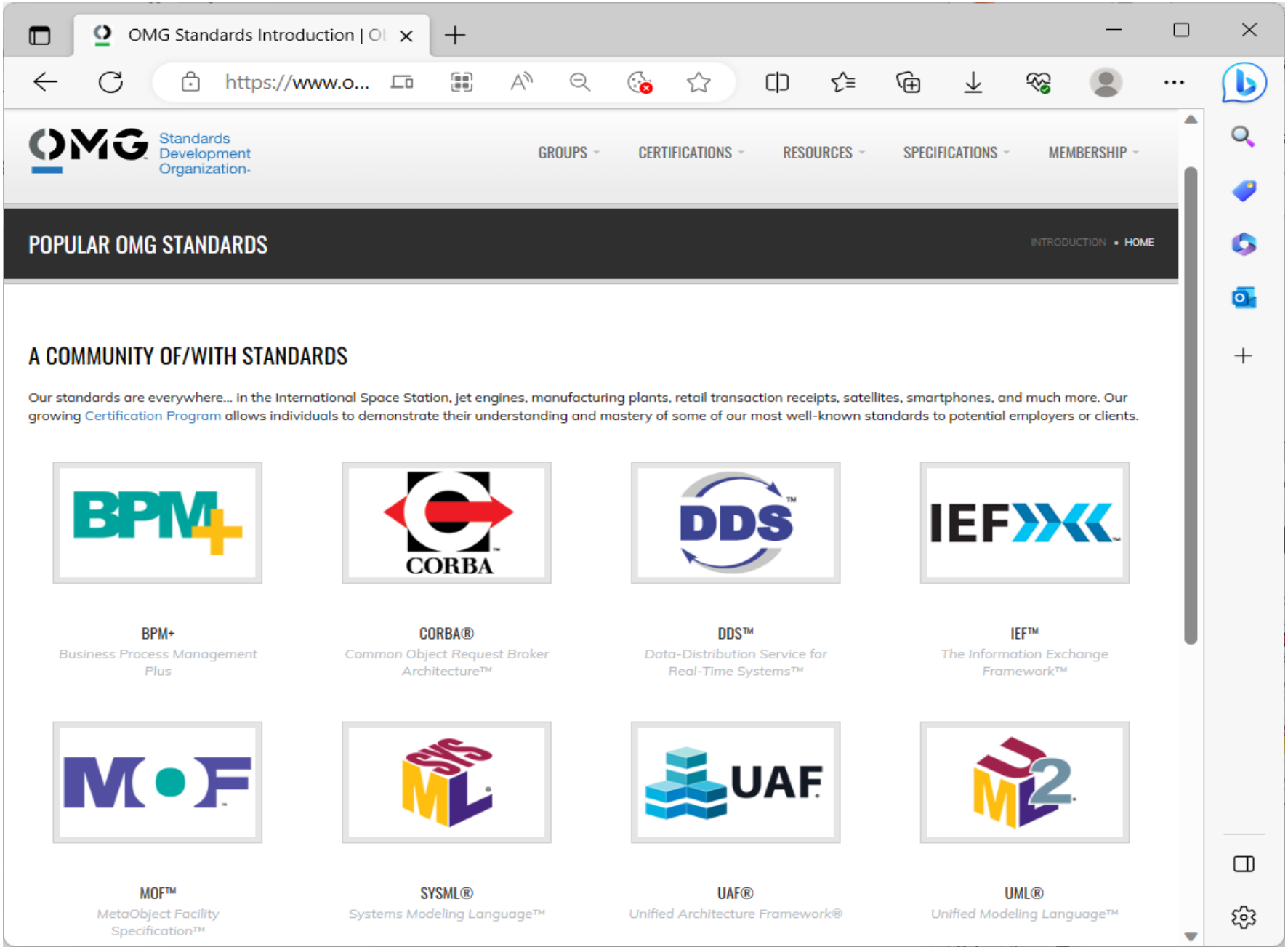
1. System informatyczny
2. Inżynieria oprogramowania
3. Tworzenie oprogramowania – podstawowe elementy
4. Korzyści wynikające ze stosowania zasad inżynierii oprogramowania
5. Dodatek do wykładu

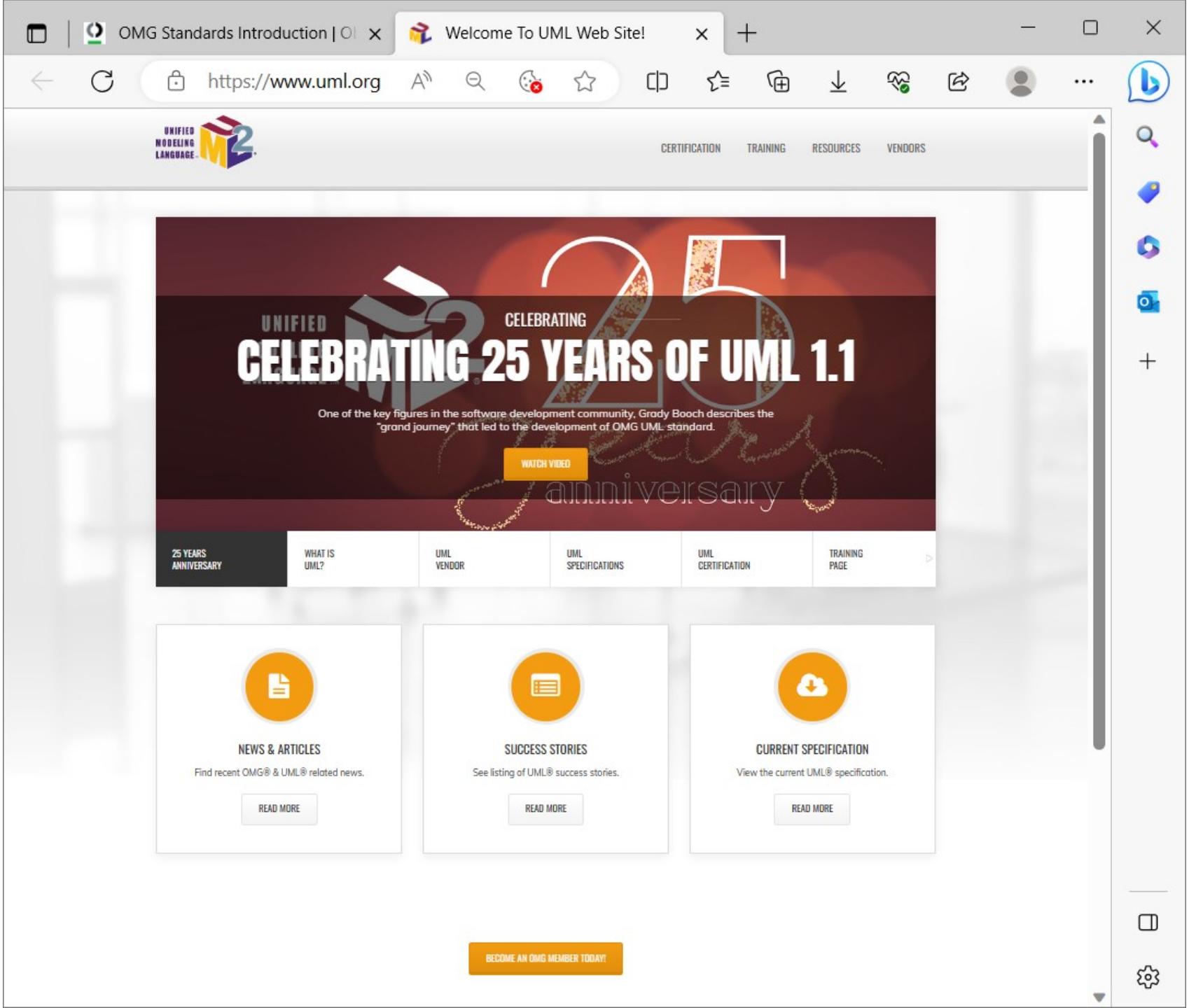
Strona OMG (Object Management Group)

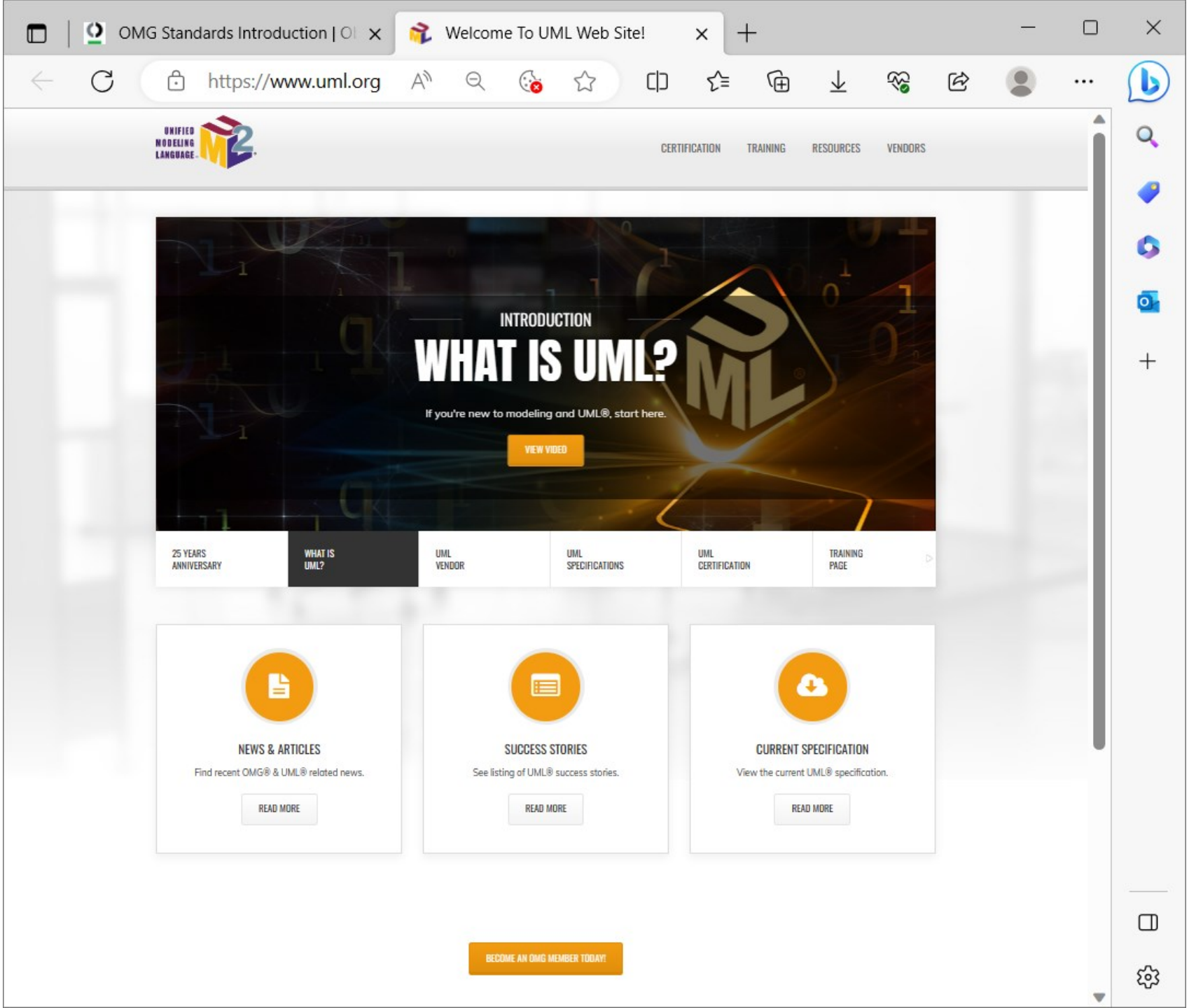
OMG | Object Management Group

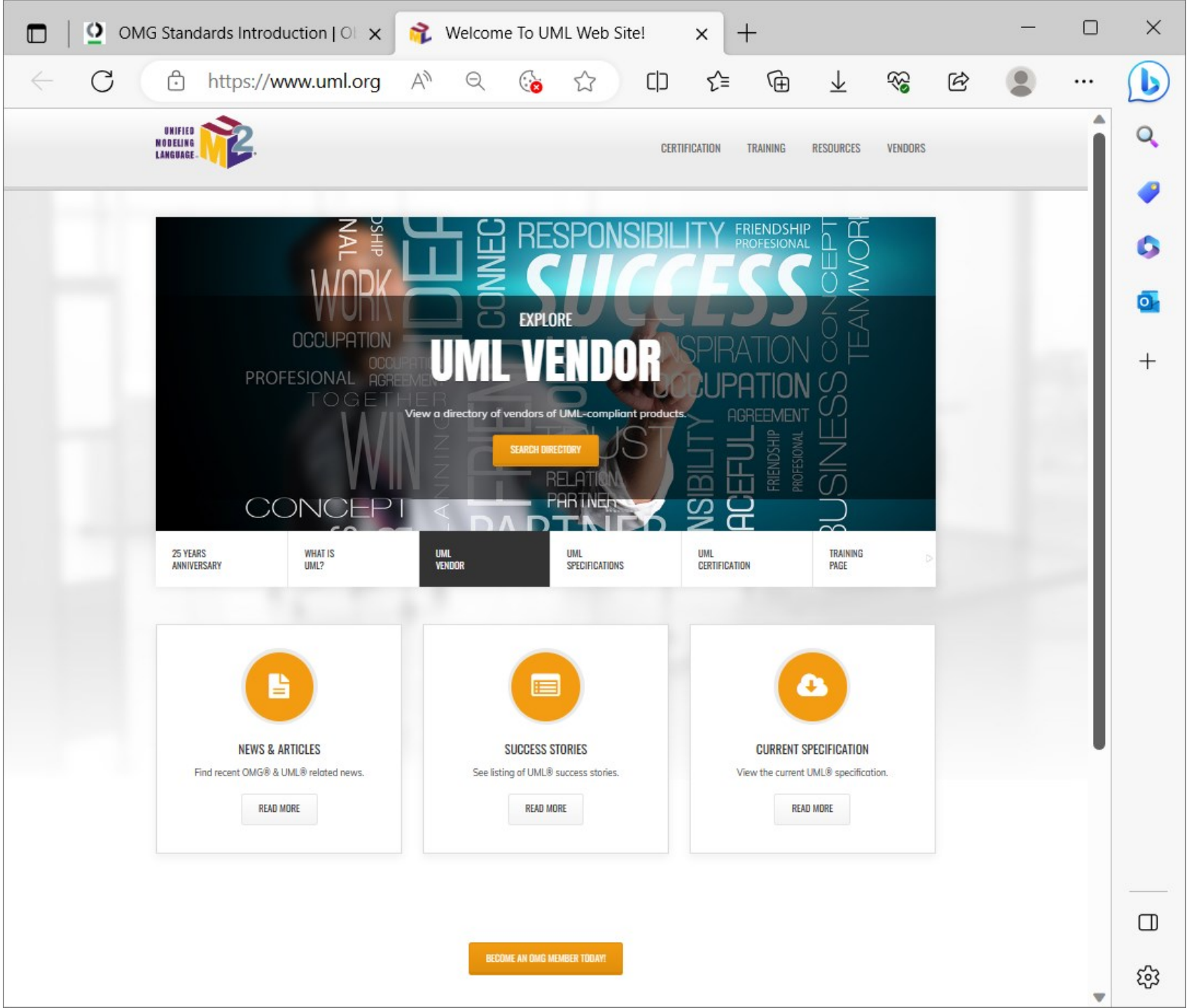
Grupa Object Management Group, Inc. (OMG), założona w 1989 r., jest konsorcjum standardów branży komputerowej typu non-profit z otwartym członkostwem, które produkuje i utrzymuje specyfikacje branży komputerowej dotyczące interoperacyjnych, przenośnych i wielokrotnego użytku aplikacji dla przedsiębiorstw w rozproszonych, heterogenicznych środowiskach. Członkostwo obejmuje dostawców technologii informatycznych, użytkowników końcowych, agencje rządowe i środowiska akademickie.

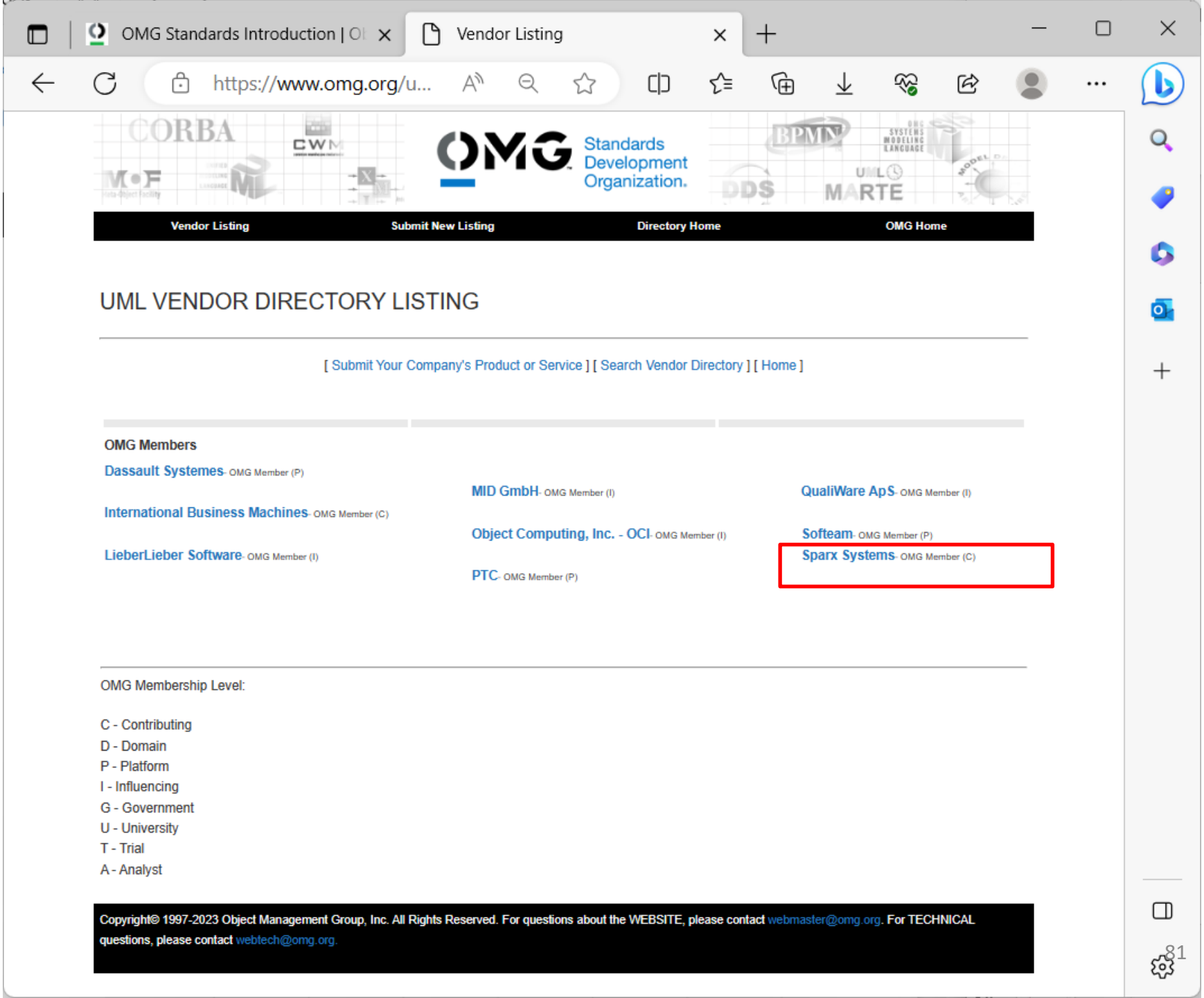
Popularne standardy grupy OMG

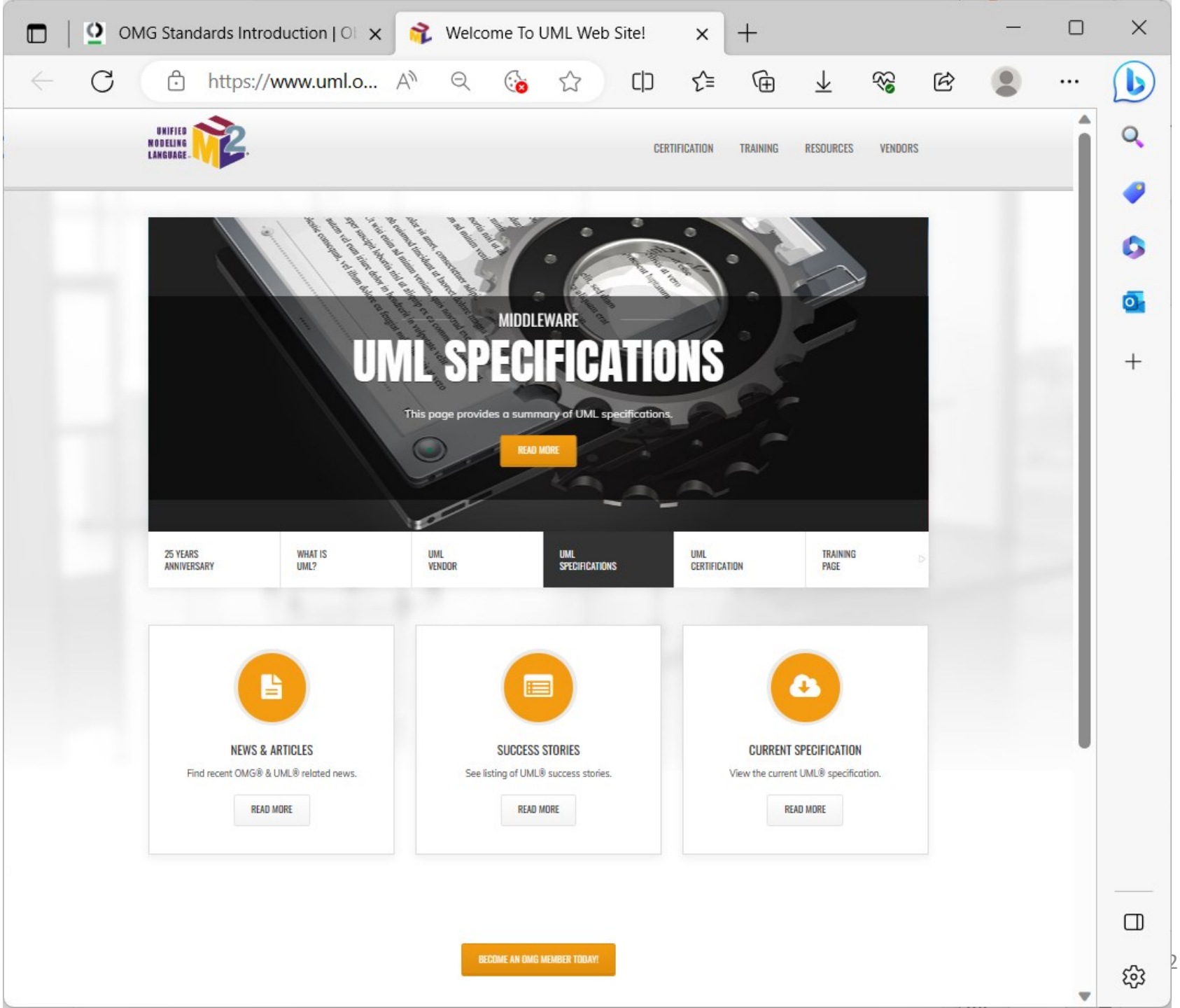



OMG Standards Introduction | O
Welcome To UML Web Site!
https://www.uml.org
UNIFIED MODELING LANGUAGE
CERTIFICATION
TRAINING
RESOURCES
VENDORS
CELEBRATING
CELEBRATING 25 YEARS OF UML 1.1
One of the key figures in the software development community, Grady Booch describes the "grand journey" that led to the development of OMG UML standard.
WATCH VIDEO
anniversary
25 YEARS ANNIVERSARY
WHAT IS UML?
UML VENDOR
UML SPECIFICATIONS
UML CERTIFICATION
TRAINING PAGE
NEWS & ARTICLES
Find recent OMG® & UML® related news.
READ MORE
SUCCESS STORIES
See listing of UML® success stories.
READ MORE
CURRENT SPECIFICATION
View the current UML® specification.
READ MORE
BECOME AN OMG MEMBER TODAY!


OMG Standards Introduction | O
Welcome To UML Web Site!
https://www.uml.org
UNIFIED MODELING LANGUAGE
CERTIFICATION
TRAINING
RESOURCES
VENDORS
INTRODUCTION
WHAT IS UML?
If you're new to modeling and UML®, start here.
VIEW VIDEO
25 YEARS ANNIVERSARY
WHAT IS UML?
UML VENDOR
UML SPECIFICATIONS
UML CERTIFICATION
TRAINING PAGE
NEWS & ARTICLES
Find recent OMG® & UML® related news.
READ MORE
SUCCESS STORIES
See listing of UML® success stories.
READ MORE
CURRENT SPECIFICATION
View the current UML® specification.
READ MORE
BECOME AN OMG MEMBER TODAY!


OMG Standards Introduction | O
Welcome To UML Web Site!
https://www.uml.org
UNIFIED MODELING LANGUAGE
CERTIFICATION
TRAINING
RESOURCES
VENDORS
EXPLORE
UML VENDOR
View a directory of vendors of UML-compliant products.
SEARCH DIRECTORY
25 YEARS ANNIVERSARY
WHAT IS UML?
UML VENDOR
UML SPECIFICATIONS
UML CERTIFICATION
TRAINING PAGE
NEWS & ARTICLES
Find recent OMG® & UML® related news.
READ MORE
SUCCESS STORIES
See listing of UML® success stories.
READ MORE
CURRENT SPECIFICATION
View the current UML® specification.
READ MORE
BECOME AN OMG MEMBER TODAY!


OMG Standards Introduction | O
Vendor Listing
https://www.omg.org/u...
CORBA
CWM
MOF
OMG
Standards
Development
Organization.
BPMN
DDS
MARTE
Vendor Listing
Submit New Listing
Directory Home
OMG Home
UML VENDOR DIRECTORY LISTING
[Submit Your Company's Product or Service] [Search Vendor Directory] [Home]
OMG Members
Dassault Systemes - OMG Member (P)
International Business Machines - OMG Member (C)
LieberLieber Software - OMG Member (I)
MID GmbH - OMG Member (I)
Object Computing, Inc. - OCI - OMG Member (I)
PTC - OMG Member (P)
QualiWare ApS - OMG Member (I)
Softeam - OMG Member (P)
Sparx Systems - OMG Member (C)
OMG Membership Level:
C - Contributing
D - Domain
P - Platform
I - Influencing
G - Government
U - University
T - Trial
A - Analyst
Copyright© 1997-2023 Object Management Group, Inc. All Rights Reserved. For questions about the WEBSITE, please contact webmaster@omg.org. For TECHNICAL questions, please contact webtech@omg.org.


OMG Standards Introduction | O
Welcome To UML Web Site!
https://www.uml.o...
UNIFIED MODELING LANGUAGE
CERTIFICATION
TRAINING
RESOURCES
VENDORS
MIDDLEWARE
UML SPECIFICATIONS
This page provides a summary of UML specifications.
READ MORE
25 YEARS ANNIVERSARY
WHAT IS UML?
UML VENDOR
UML SPECIFICATIONS
UML CERTIFICATION
TRAINING PAGE
NEWS & ARTICLES
Find recent OMG® & UML® related news.
READ MORE
SUCCESS STORIES
See listing of UML® success stories.
READ MORE
CURRENT SPECIFICATION
View the current UML® specification.
READ MORE
BECOME AN OMG MEMBER TODAY!

INFORMATIVE DOCUMENTS

DESCRIPTION	FORMAT	URL	OMG FILE ID
Specification changebar	PDF	UML/2.5.1/PDF/changebar	formal/17-12-06

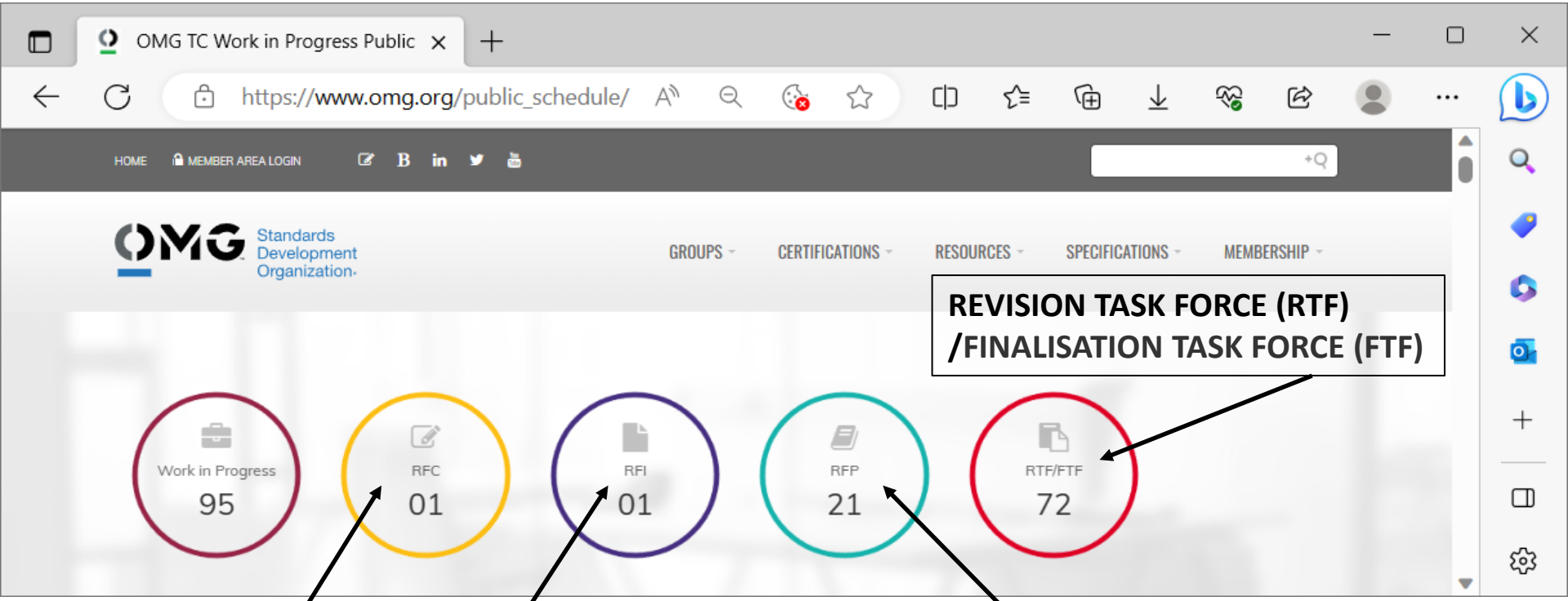
HISTORY

FORMAL VERSIONS

VERSION	ADOPTION DATE	URL
2.5.1	grudnia 2017	https://www.omg.org/spec/UML/2.5.1
2.4.1	lipca 2011	https://www.omg.org/spec/UML/2.4.1
2.3	maja 2010	https://www.omg.org/spec/UML/2.3
2.2	stycznia 2009	https://www.omg.org/spec/UML/2.2
2.1.2	października 2007	https://www.omg.org/spec/UML/2.1.2
2.0	lipca 2005	https://www.omg.org/spec/UML/2.0
1.5	marca 2003	https://www.omg.org/spec/UML/1.5
1.4	września 2001	https://www.omg.org/spec/UML/1.4
1.3	lutego 2000	https://www.omg.org/spec/UML/1.3
1.2	lipca 1999	https://www.omg.org/spec/UML/1.2
1.1	grudnia 1997	https://www.omg.org/spec/UML/1.1

OMG Transaction Capabilities (TC) Work in Progress Public

(fragment podanej strony)



REVISION TASK FORCE (RTF) /FINALISATION TASK FORCE (FTF)

REQUEST FOR COMMENTS (RFC)

REQUEST FOR INORMATION (RFI)

REQUEST FOR PROPOSAL (RFP)

OMG Transaction Capabilities (TC) Work in Progress Public

(wybrane informacje z podanej strony)

PENDING REQUESTS FOR PROPOSALS:

1. Agent Metamodel and Profile (AMP) RFP
2. DDS C# API Request For Proposal
3. Event Metamodel and Profile (EMP) RFP
4. Information Management Metamodel (IMM) RFP

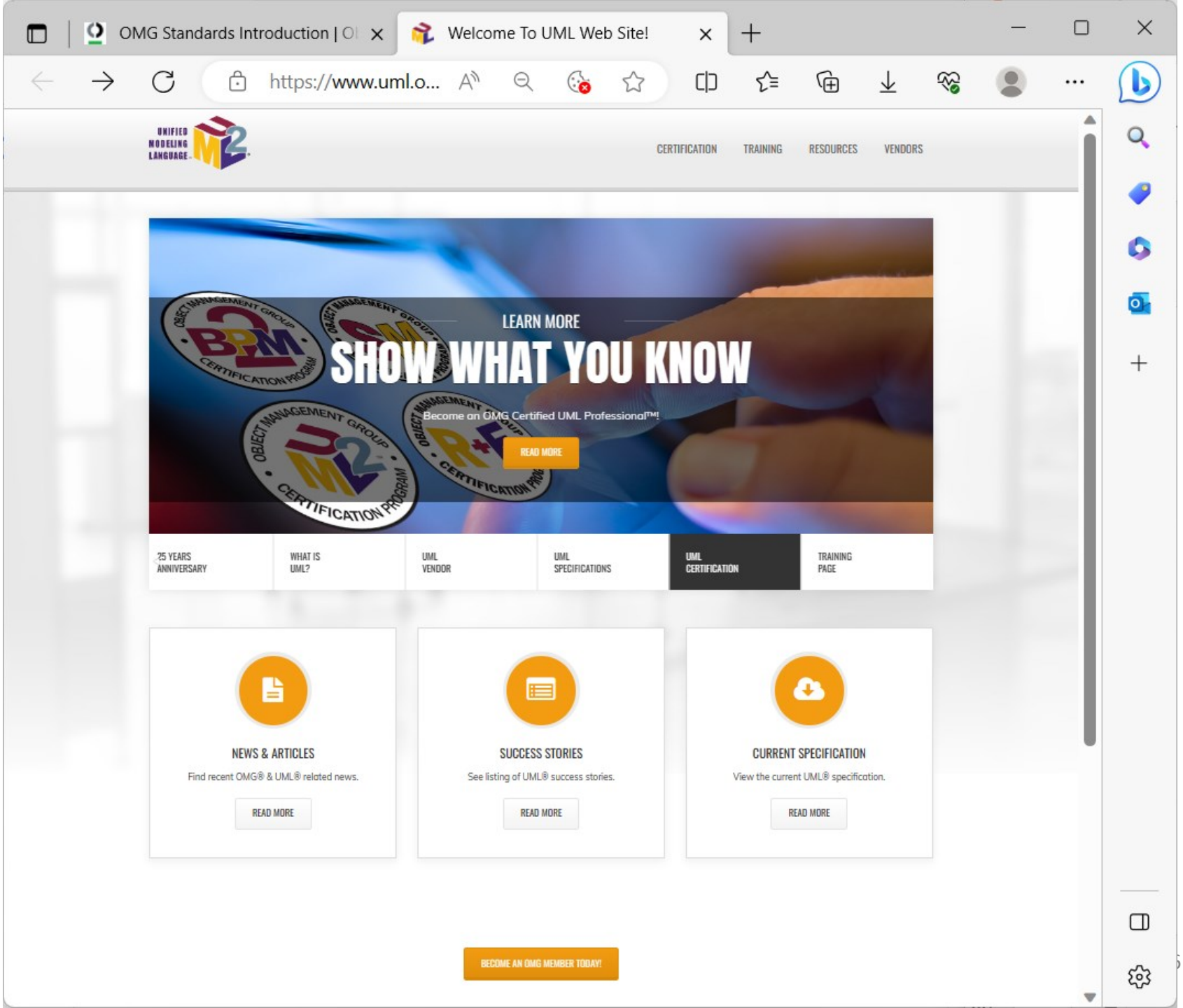
REQUEST FOR PROPOSAL (RFP)

.....

CURRENT SPECIFICATION REVISION PROCESSES (AVAILABLE TO MEMBERS ONLY):

1. Action Language for fUML 1.2 (ALF) RTF
2. DDS Consolidated JSON Syntax (DDS-JSON) 1.1 RTF
   DDS Consolidated XML Syntax (DDS-XML) 1.1 RTF
   DDS Extensible Types (DDS-XTYPES) 1.4 RTF
   DDS Extensions for Time Sensitive Networking (DDS-TSN) 1.0 FTF
   DDS Security 1.2 RTF
   DDS-OPC UA Gateway 1.1 RTF
   DDS-PSM-Cxx v1.1 RTF
   DDS-PSM-JAVA 1.1 RTF
   DDS-XRCE 1.1 RTF
3. Systems Modeling Language (SysML) 2.0 FTF
4. UML 2.6 RTF
   UML Profile for ROSETTA (UPR) 1.1 RTF
   UML Testing Profile 2 (UTP2) 2.2 RTF

REVISION TASK FORCE (RTF)


OMG Standards Introduction | O
Welcome To UML Web Site!
https://www.uml.o...
UNIFIED MODELING LANGUAGE
CERTIFICATION
TRAINING
RESOURCES
VENDORS
LEARN MORE
SHOW WHAT YOU KNOW
Become an OMG Certified UML Professional™!
READ MORE
25 YEARS ANNIVERSARY
WHAT IS UML?
UML VENDOR
UML SPECIFICATIONS
UML CERTIFICATION
TRAINING PAGE
NEWS & ARTICLES
Find recent OMG® & UML® related news.
READ MORE
SUCCESS STORIES
See listing of UML® success stories.
READ MORE
CURRENT SPECIFICATION
View the current UML® specification.
READ MORE
BECOME AN OMG MEMBER TODAY!

UML 2 CERTIFICATIONS

THE VALUE OF IT CERTIFICATIONS REPORT

- OMG Certifications
- Voucher Program
- Exam Prep Courses
- Join UML LinkedIn Group

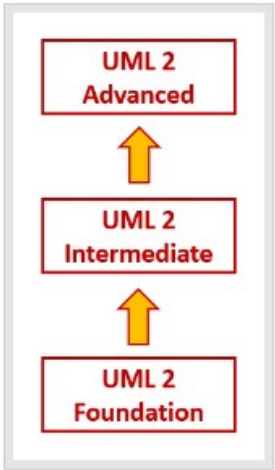
Receive and share your digital certification credentials immediately after passing your exam.

OMG-Certified UML Professional 2 (OCUP 2) exams test an individual's ability to properly interpret and construct UML model diagrams in the way UML is used today.

There are three UML 2 certifications exams: **Foundation**, **Intermediate** and **Advanced**.

To register for an **UML 2 Certifications** exam create a Pearson VUE account.

REGISTER TODAY!



Dziękuję za uwagę