Projektowanie i programowanie

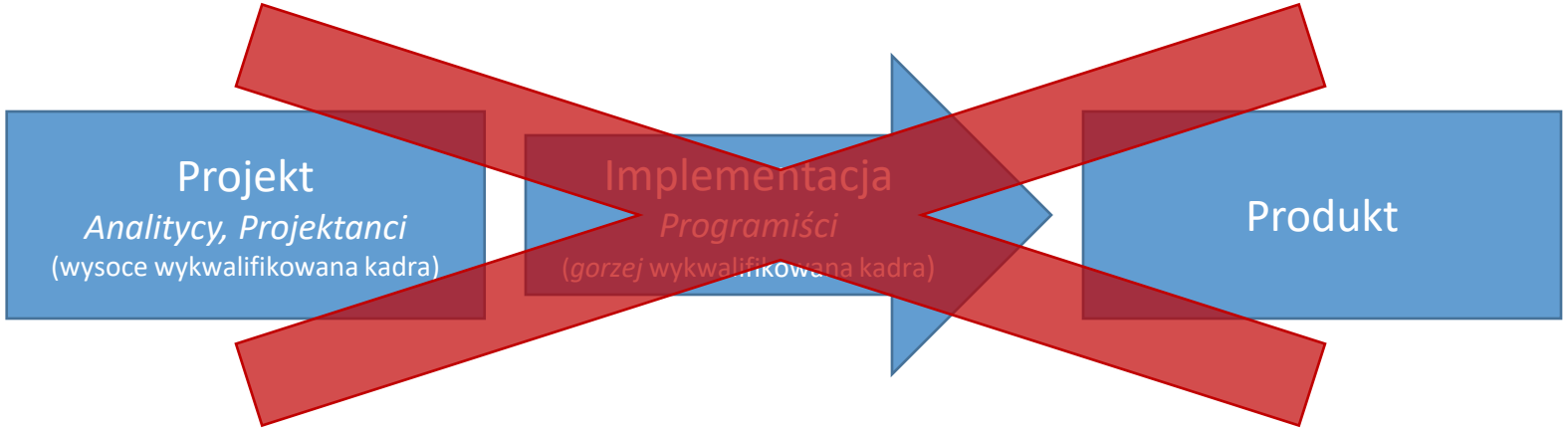
Sztuka Wytwarzania Oprogramowania, w. 4 cz. 2

Konrad Grochowski

Instytut Informatyki, Politechnika Warszawska, 2025 ©

Co to jest projekt?

- Niekiedy rozumiany jako
  „wysokopoziomowa architektura rozwiązania”
- Co niektórych prowadzi do takiego procesu:



- Jest to *bardzo* błędne rozumienie procesu inżynieryjnego
- A w szczególności efekt zmiksowania niepoprawnych wizji procesu budowlanego i linii produkcyjnej

Projekt

- Projekt to wszystko *po drodze* do rzeczywistego produktu
- Kiedy aplikacja staje się rzeczywistością?
- Wtedy gdy zabiera się do jej stworzenia kompilator/interpreter
- Czyli – *kod jest projektem aplikacji*
- Poprawniej – *kod **też** jest częścią projektu aplikacji*
- O **kluczowym** wpływie na to, czym aplikacja będzie, ale częścią
- Jak ktoś potrzebuje analogii – *projekt wykonawczy,*
  a może nawet *projekt **po**wykonawczy*

Proces inżynieryjny (zarys)

- To bardzo ogólny zarys, będą o tym kolejne wykłady
- I całe przedmioty, bo dziedzina jest ogromna i bardzo ważna
- *Inżynieria Oprogramowania / Software Engineering*

- Dwukierunkowość można osiągnąć np. współdzieląc osoby
- *Obowiązkiem* inżyniera, jest pilnować jakości procesu (*zmieniać* firmę)

Projekt i kod

- Kod nie jest idealny do łatwego demonstrowania architektury
- (Ale ona *musi* być *w* kodzie – jak się nie zgadzają, to architektura może iść do kosza, bo to kod efektywnie „idzie do klienta”)
- Rysunek bywa najefektywniejszą formą przekazu informacji
- Ale trzeba pamiętać, że w razie wątpliwości – *sprawdzać kod*
- Zabiegi pomocnicze:
  - Nazewnictwo oparte domenę i architekturę
  - Struktura pakietów/folderów/plików zgodna z architekturą
  - *Rozsądna* dokumentacja kodu

W czym rysować?

- UML (Unified Modeling Language)
- UML 1.0 (1997)
  - Grady Booch, James Rumbaugh, Ivar Jacobson
- Object Management Group (OMG)
  - UML 1.1 (1997)
  - UML 1.3 (2000)
  - UML 1.5 (2003)
  - UML 2.0 (2004)
  - UML 2.4 (2011)
  - UML 2.5 (2015)

Slajdy o UML zostały oparte na materiałach dra inż. Marcina Szlenka

Czym UML jest a czym nie jest?

UML JEST

- językiem specyfikacji, projektowania i dokumentowania systemów IT

UML NIE JEST

- metodyką
- narzędziem modelowania
- językiem programowania

UML is messy, imprecise, complex and sprawling. That is both a fault and a virtue. Anything intended for such widespread usage is going to be messy.

J. Rumbaugh, I. Jacobson, G. Booch

The Unified Modeling Language Reference Manual

Wady UML i jak go wykorzystać

- Wady UML
  - obszerność (ilość pojęć, 1000 stron specyfikacji)
  - niestabilność (próbuje nadążać za nowymi technologiami)
  - nieprecyzyjność (trzeba zajrzeć do kodu, żeby zrozumieć model)
- Jak korzystać?
  - nie trzeba znać całości UMLa, żeby z niego korzystać
    - tylko mały podzbiór notacji UML jest szeroko stosowany
  - nie należy modelować wszystkiego, a tylko istotne fragmenty systemu
    - dobry do przekazania *zarysów* budowy i działania systemu
  - dobry do szybkiego przekazania **idei** niektórych złożonych rozwiązań
  - z Clean Code: im bardziej szczegółowy diagram tym krócej powinien żyć

Narzędzia wspierające

- Jako edytor diagramów (rysunków) wystarczy MS Visio
  - Są też wygodne edytory on-line (nadal tylko „rysują”, nie „modelują”) https://www.lucidchart.com https://draw.io itp.
- Nie ma narzędzia w 100% implementującego standard UML
- Funkcjonalność zaawansowanych narzędzi
  - inżynieria wprzód i wstecz dla różnych języków programowania
  - rozszerzalność np. poprzez definiowanie własnych profili (*stereotypów)*
- Przykładowe środowiska komercyjne:
  - Sparx Enterprise Architect, Rational Software Architect, Borland Together
  - Wprowadzają własne „dialekty” (*stereotypy*) UML
  - Wersja open-source próbująca być alternatywą: Papyrus

Diagramy UML

OPISU STRUKTURY

- **klas**
- obiektów
- pakietów
- struktury złożonej
- komponentów
- wdrożenia

OPISU ZACHOWANIA

- przypadków użycia
- maszyny stanowej
- czynności
- **sekwencji**
- komunikacji
- przeglądu interakcji
- czasowy

Diagram klas

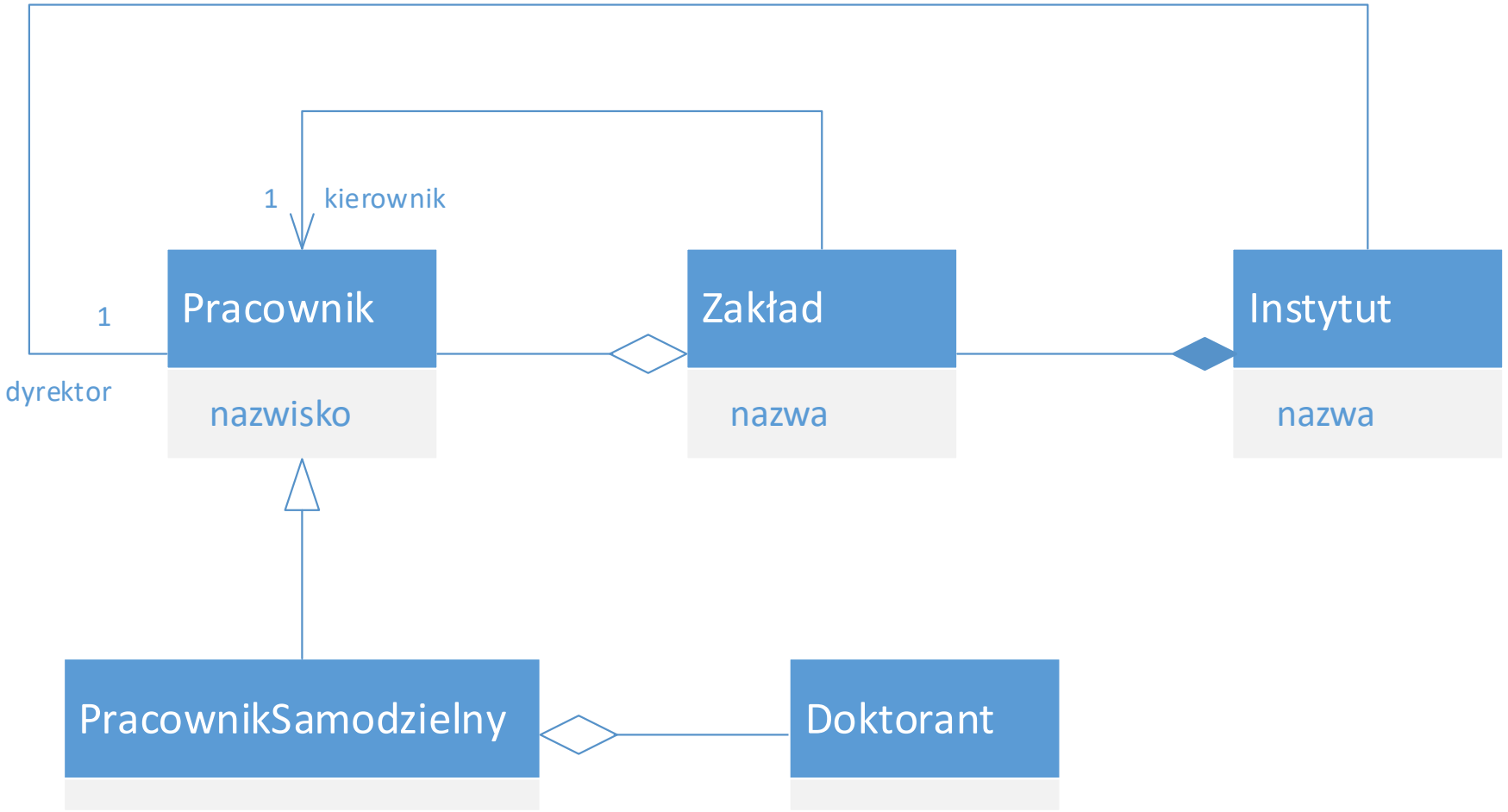


Diagram klas – bez ozdobników

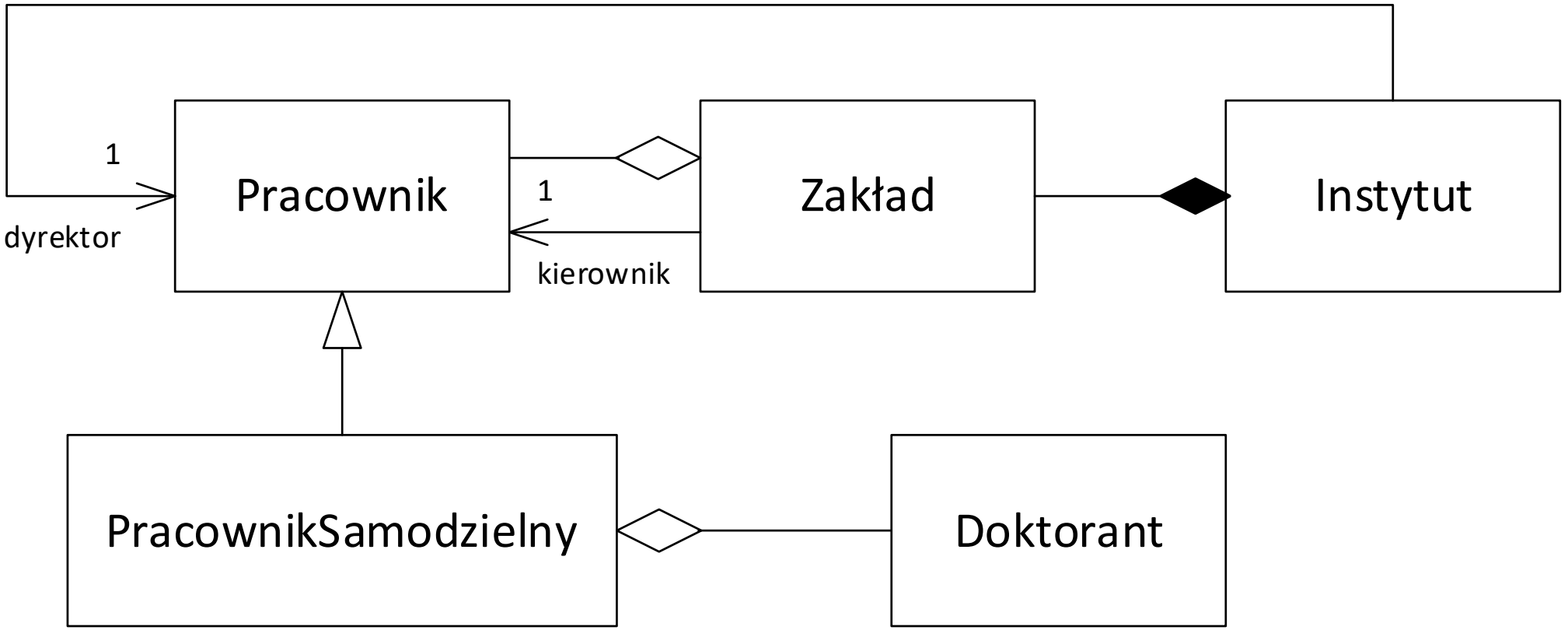
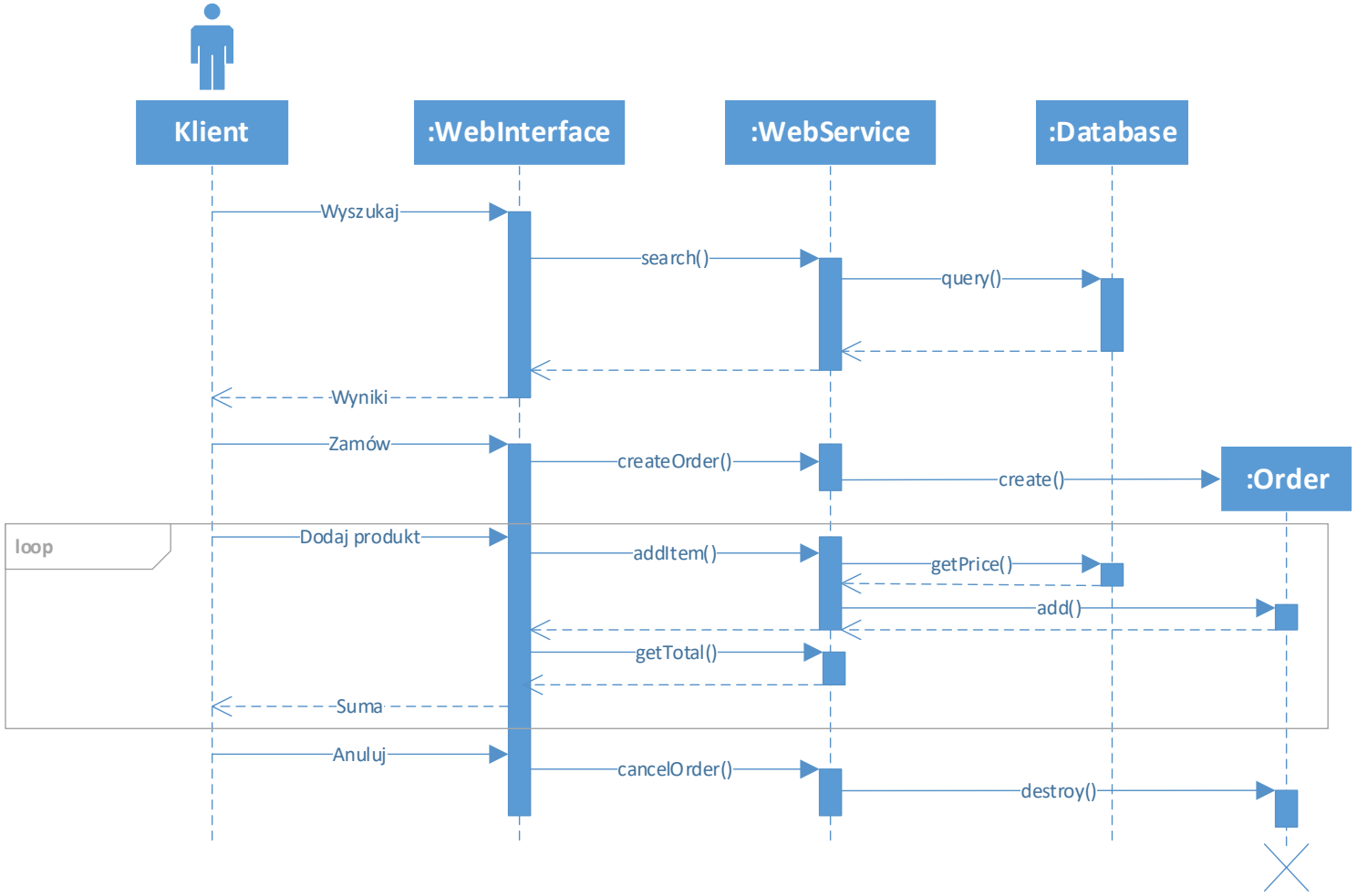


Diagram sekwencji



Rysowanie a programowanie

- Diagram, który może zostać aplikacją, musi być jednoznaczny, kompletny i konkretny
- Mamy wtedy do czynienia z *graficznym językiem* ***programowania***
- Czyli obowiązują wszystkie reguły „tworzenia kodu” i podobny sposób myślenia
  - Szczegółowe diagramy mają sens tylko jeśli stają się „plikami źródłowymi”
- Marzenia o „rysowaniu programów przez nie-programistów” są raczej nierealne
  - A przynajmniej *nie każdego* programu

Modelowanie w przemyśle

- MBSE – Model Based Software/System Engineering
- Szczególnie przydatny, gdy nie wszystkie elementy systemu to oprogramowanie
- Opiera się o języki o sformalizowanych właściwościach, np.:
  - AADL
  - SDL
  - ASN.1
  - SysML (skonkretyzowany dialekt UML)
  - Arcadia (Capella)
- Jak ktoś chce, może poszukać TASTE ESA (https://taste.tools) i zobaczyć, jak Europejska Agencja Kosmiczna próbuje znaleźć metody na „wyklikiwanie” satelit :)

Czatowanie a programowanie

- Ostatnie osiągnięcia AI znowu przywróciły rozmowę o:
  - Programowaniu bez programistów
  - Tworzenia programów bez „ścisłych umysłów”
- Argumenty z poprzednich slajdów pozostają bez zmian
- Tworzenie wspomagane AI wymaga:
  - Dobrej umiejętności definiowania potrzeb („ścisłość”)
  - Przewidywania skutków działań (programowanie)
  - Rozumienia do czego się poszczególne wygenerowane klocki składają (inżynieria)
- Ale będą (?) problemy z przekształcaniem *juniorów* w *seniorów*

Dziękuję za uwagę

Konrad.Grochowski@pw.edu.pl