

# Realizacja Projektu Informatycznego



***Jakub Miler***

*Katedra Inżynierii Oprogramowania  
Wydział Elektroniki, Telekomunikacji i Informatyki  
Politechnika Gdańskia  
jakubm@eti.pg.edu.pl*

*Materiały pomocnicze do wykładu na Wydziale ETI Politechniki Gdańskiej.  
Wykorzystanie materiałów w innym celu i ich rozpowszechnianie bez zgody WETI PG zabronione.*

# O mnie



dr inż. Jakub Miler

Katedra Inżynierii Oprogramowania  
Wydział Elektroniki, Telekomunikacji i Informatyki  
Politechnika Gdańskia  
jakubm@eti.pg.gda.pl  
tel. 58 347 26 97

## Projekty



nor-sta

2010-2014



PIPS

Personalized Information Platform  
for Life & Health Services

2004-2008



2009-2010

## Projekty studenckie – dyplomowe, grupowe

ok. 80 projektów od 2004r.



2014-



Advanced Networked embedded platform  
as a Gateway to Enhance quality of Life

2006-2009



DRug in Virtual Enterprise

2002-2003

## Certyfikaty



## Funkcje



Zastępca kierownika  
Katedry Inżynierii  
Oprogramowania  
2013-2015

Opiekun koła naukowego



[www.zarzadzanieit.com](http://www.zarzadzanieit.com)

# Koło naukowe Zarządzanie IT



GŁÓWNA  
PROJEKTY  
BLOG  
WSPÓŁPRACA ▾  
ZESPÓŁ ▾  
KONTAKT



[www.zarzadzanieit.com](http://www.zarzadzanieit.com)

## NASZE PROJEKTY



### Konferencja Inżynierii Oprogramowania beIT

to:

- eksperci z całej Polski
- wysoki poziom merytoryczny



### Pomosty Kariery

Pomagamy członkom Koła:

- zdobyć wymarzoną pracę
- zyskać rozwojowy staż



### Praktycy na Politechnice

Zapraszamy do prowadzenia warsztatów:

- praktyków z firm
- startupowców
- ludzi z pasją



### Wspólne Certyfikowanie

Przygotowujemy się do certyfikatów z:

- analizy biznesowej
- zarządzania projektami



## ŚCIEŻKI

be UX  
Designer

beAgile

beAnalyst1

be AI  
Engineer

beAnalyst2

beLeader

be Product  
Manager

beTester

# Motywacje i cele przedmiotu

*„Wszystkie metodyki są dobre, lecz do różnych projektów”*

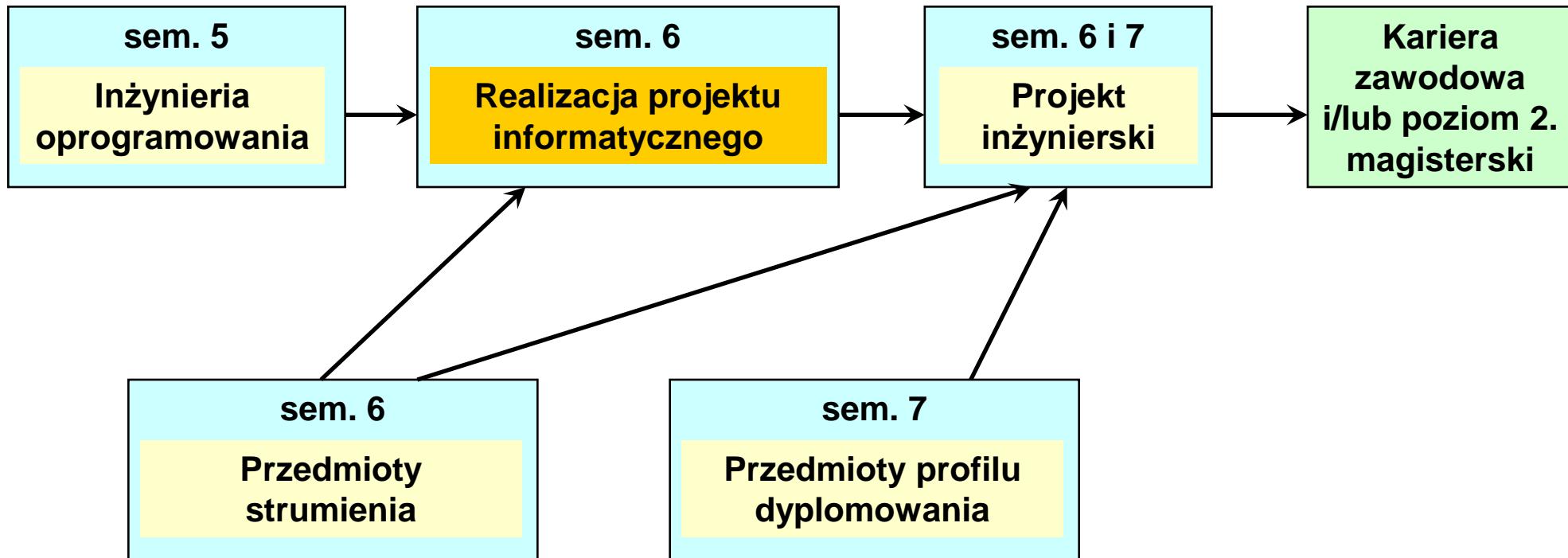
## Motywacje:

- Metodyki pozwalają uporządkować (nie skomplikować!) projekt
- Nie ma najlepszej metodyki do wszystkiego – projekt to twórczość
- Żadna metodyka nie jest „srebrnym pociskiem” gwarantującym sukces
- Realizując projekt powinniśmy dobierać środki do celu

## CELE PRZEDMIOTU:

1. Zapoznanie się z wybranymi metodami wytwarzania oprogramowania
2. Zrozumienie silnych i słabych stron poszczególnych metod
3. Nabycie umiejętności doboru metodyki i jej adaptacji do potrzeb projektu

# RPI w ścieżce edukacyjnej inżyniera informatyka



# Organizacja przedmiotu

## ➤ Wykład:

- **2 godziny w tygodniu do połowy semestru**
- **25.02.2021 - Wprowadzenie, infrastruktura projektu, przykłady projektów, metodyki**
- **4.03.2021 - Scrum: Framework, wartości, filary, role**
- **11.03.2021 - Scrum: Backlog produktu, planowanie sprintu, backlog sprintu**
- **18.03.2021 - Scrum: Codzienny Scrum, przegląd sprintu, retrospekcja, zasady**
- **25.03.2021 - Kanban, eXtreme Programming (XP), Rational Unified Process (RUP)**
- **1.04.2021 - Scrum of Scrums, DevOps, Scaled Agile Framework (SAFe) - prowadzi Katarzyna Łukasiewicz**
- **8.04.2021 - Dobór i adaptacja metodyki**
- **15.04.2021 - Wykład zaproszony**

## ➤ Zasady zaliczenia

- **egzamin z wykładu – 40%, projekt – 60%**
- **trzeba zaliczyć osobno wykład i projekt (min. wykład: 21/40 pkt, projekt 31/60 pkt)**
- **o ocenie decyduje suma punktów z egzaminu i projektu**
- **skala ocen: 0-51,5: 2.0, 52-59,5: 3.0, 60-69,5: 3.5, 70-79,5: 4.0, 80-89,5: 4.5, 90-100: 5.0**

# Organizacja przedmiotu (2)

## ➤ Projekt:

- 2 godziny w tygodniu co 2 tygodnie
- 6 zadań – wybrane zagadnienia organizacji projektów, Scruma oraz dobór metodyki
- ~~w tym 3 warsztaty – praca na miejscu bez oddawania raportu~~
- prowadzący:
  - dr inż. Katarzyna Łukasiewicz, p. 648, [katarzyna.lukasiewicz@eti.pg.edu.pl](mailto:katarzyna.lukasiewicz@eti.pg.edu.pl)
  - dr inż. Jakub Miler, p. 648, [jakubm@eti.pg.edu.pl](mailto:jakubm@eti.pg.edu.pl)
  - mgr inż. Małgorzata Pykała, p. 318, [malpykal@pg.edu.pl](mailto:malpykal@pg.edu.pl)
- terminy zajęć:
  - poniedziałek 10-14 - 2 grupy projektowe - prowadzi Małgorzata Pykała
  - środa 15-17 - 2 grupy projektowe - prowadzi Katarzyna Łukasiewicz
  - środa 15-17 - 1 grupa projektowa - prowadzi Jakub Miler

- Rozpoczęcie projektu wspólne dla wszystkich 3.03.2021 g. 15:15 na eNauczaniu
- Projekt można realizować w ramach IOS (awansem)
- Egzamin kontaktowy tylko dla osób na liście przedmiotu w mojaPG

# Ważne zasady

- Celem przedmiotu jest poznanie głównych współczesnych metodyk wytwarzania oprogramowania i nabycie umiejętności realizacji projektu zgodnie z pasującą, odpowiednio i świadomie zaadaptowaną metodyką, a także łączenie elementów różnych metodyk w spójny proces organizacji pracy.
- Na wykładzie w omówieniu slajdów przekazywane są zasady, reguły, przyczyny, skutki, cele, związki pozwalające zrozumieć istotę zagadnień, co jest celem przedmiotu i co podlega weryfikacji w egzaminie.
- Slajdy to tylko materiał pomocniczy zawierający suche informacje. Samo ich przeczytanie i zapamiętanie nie dostarczy wymaganej wiedzy.
- Proszę korzystać z aktualnych materiałów pomocniczych (slajdów) – mogą być istotne zmiany względem poprzednich lat.
- Obecność na wykładaach jest obowiązkowa, choć nie musi być sprawdzana.
- Będą organizowane tylko dwa terminy egzaminu (podstawowy i poprawkowy). Niezdanie w obu terminach powoduje opóźnienie dyplomu inżynierskiego. Proszę wziąć to pod uwagę na początku semestru i po pierwszym niezdanym egzaminie.
- Zapraszam na konsultacje najpóźniej po pierwszym niezdanym egzaminie.

# Literatura

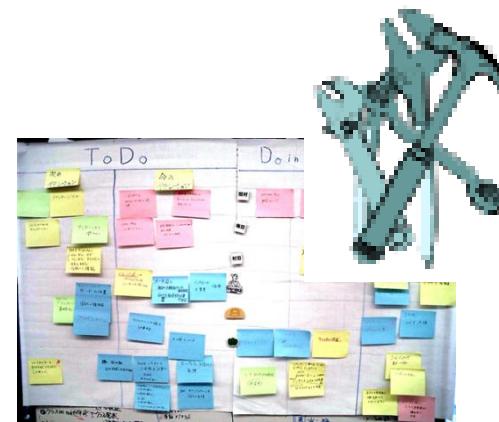
## ➤ Książki

- A. Koszlańda, *Zarządzanie Projektami IT Przewodnik po Metodykach*, Helion, 2010
- K. Schwaber, J. Sutherland, *The Scrum Guide, The Definitive Guide to Scrum: The Rules of the Game*, Scrum.org, 2020  
[wersja polska: *The Scrum Guide, Przewodnik po Scrumie: Reguły Gry*, Scrum.org, 2020]
- M. Lacey, *Scrum. Praktyczny przewodnik dla początkujących*, Helion, 2014
- M. Chrapko, *Scrum. O zwinnym zarządzaniu projektami*, Helion, 2012, wyd. 2, 2014
- K. S. Rubin, *Scrum. Praktyczny przewodnik po najpopularniejszej metodyce Agile*, Helion, 2013
- J. Rasmussen, *Zwinnie samuraj. Jak programują mistrzowie zwinności*, Helion, 2012
- A. Cockburn, *Agile Software Development. Gra zespołowa*, wyd. II, Helion, 2008
- K. Beck, C. Andres, *Wydajne programowanie. Extreme Programming*, wyd. II, MIKOM, 2006
- P. Kruchten, *The Rational Unified Process: An Introduction*, 3rd edition, Addison-Wesley Professional, 2003  
[wyd. polskie: *Rational Unified Process od strony teoretycznej*, WNT, 2006]

## ➤ Źródła internetowe

- Manifest Zwinnego Tworzenia Oprogramowania - [www.agilemanifesto.org/iso/pl/](http://www.agilemanifesto.org/iso/pl/)
- [www.scrum.org](http://www.scrum.org), [www.scrumalliance.org](http://www.scrumalliance.org)

# Infrastruktura projektu



# Infrastruktura projektu

- **Infrastruktura - ludzie**
  - ludzie, zespoły, role
- **Infrastruktura komunikacyjna**
  - komunikacja wewnętrzna i zewnętrzna
- **Infrastruktura dokumentacyjna**
  - dokumentowanie wytwarzania, zarządzania
  - składowanie kodu i innych produktów projektu
  - zarządzanie konfiguracją
- **Narzędzia**
  - modelowanie, wytwarzanie, testowanie
  - zarządzanie, komunikacja

# Infrastruktura - ludzie

- **Skład zespołu**
- **Dane kontaktowe**
- **Doświadczenie i kompetencje**
- **Odpowiedzialności**
  - za obszary kompetencyjne
  - za konkretne działania
  - za konkretne produkty, artefakty
  - za elementy infrastruktury (np. repozytoria)
- **Struktura organizacyjna zespołu**
  - hierarchia, zespół równorzędny, wyróżnione komórki/funkcje
- **Gdy znamy już konkretną metodykę realizacji projektu to również obsada ról zdefiniowanych przez tę metodykę**

# Infrastruktura – ludzie – przykład

Imię i nazwisko	Kontakt	Kompetencje	Odpowiedzialność
Jan Kowalski	kowalski@projekt, tel. 1234, skype: jkowalski	-analiza -modelowanie UML -programowanie C#	pozyskiwanie wymagań od klienta, analiza, modelowanie
Marek Bednarski	marek@projekt, tel. 4321, skype: marekbed	-projektowanie -programowanie Java -testowanie	wybór architektury, dobór technologii, programowanie
Beata Nowak	beata@projekt, tel. 2341, skype: beatanowak	-przywództwo -analiza -programowanie	lider zespołu, kontakt z klientem, programowanie
Agata Zielińska	agata@projekt, tel. 1432, skype: agatazie	-programowanie Java -testowanie -administracja siecią	programowanie, infrastruktura, dokumentacja

# Infrastruktura - komunikacja

## ➤ Komunikacja wewnętrzna

- stopień sformalizowania komunikacji: kontakt osobisty (rozmowa), mejle, pisma
- kanały komunikacji (kto z kim powinien się komunikować w danej sprawie)

## ➤ Komunikacja zewnętrzna

- w jakich okolicznościach powinny odbywać się spotkania z otoczeniem projektu (udziałowcami, klientem, reprezentantami użytkowników)
- kto reprezentuje projekt?

## ➤ Spotkania wewnętrzne i zewnętrzne

- jak często, gdzie, kto będzie zwoływał, plan spotkań
- jeżeli ustalona już metodyka, to powinno to być zgodne z założeniami metodyki

## ➤ Komunikacja w rozproszonym zespole

- kod i dokumentacja dostępne on-line (w narzędziach lub repozytorium)
- email, komunikatory, Skype
- spotkania co najmniej raz w tygodniu w celu omówienia stanu projektu, poczynienia uzgodnień i wymyśleniu rozwiązań co trudniejszych problemów

# Infrastruktura – kod

## ➤ Możliwości współdzielenia kodu i dokumentacji

- współzielony folder w sieci lokalnej (ewentualnie dostęp przez VPN)
- rozsyłanie emailem (bardzo zły pomysł)
- uzgodnienie interfejsów i silna separacja rozwoju fragmentów produktu
- repozytorium z kontrolą wersji (zalecane): Git, SVN, Mercurial, MS Team Foundation Server

## ➤ Zalety systemu kontroli wersji

- automatyczne zachowywanie poprzednich wersji plików
- możliwość cofnięcia zmian (cofnięcia się do poprzedniej wersji)
- automatyczne łączenie (niektórych) zmian w (niektórych) plikach
- możliwość wydzielania osobnych linii rozwojowych
- repozytorium na serwerze pozwala na łatwe przenoszenie stanowiska pracy
- możliwość ustalenia uprawnień

# Infrastruktura – dokumentacja

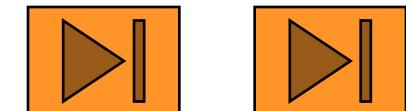
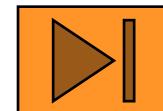
## ➤ Standardy dokumentacyjne

- nazewnictwo plików np. [Projekt]\_[Etap]\_[Id]\_[Nazwa]\_[wersja].docx
- szablon dokumentu: identyfikatory, formaty, stała część początkowa („metryczka”)
- podstawowe informacje w dokumencie: projekt, nazwa dokumentu, wersja, autorzy, data powstania, etap projektu

## ➤ Sposób wersjonowania

- za pomocą systemu kontroli wersji
- ręczny, poprzez tworzenie nowych plików  
i wskazywanie wersji w nazwie pliku

Szablon



Szablony z projektu  
NOR-STA

## ➤ Produkty wewnętrzne i zewnętrzne

- zasady przekazywania produktów na zewnątrz

## ➤ Sposób aktualizacji zatwierdzonych dokumentów bazowych

- procedura aktualizacji (propozycja, zatwierdzenie i wprowadzenie zmiany)

## ➤ Uprawnienia dostępu do repozytoriów

# Infrastruktura – narzędzia IO

## ➤ Wymagania



Rational DOORS, edytory tekstu i arkusze

## ➤ Modelowanie



Enterprise Architect, Visual Paradigm,  
PoseidonUML, MS Visio

## ➤ Programowanie



Eclipse, NetbeansIDE, MS Visual Studio

## ➤ Testowanie



NUnit, JUnit, Selenium

## ➤ Komunikacja



MS Teams, Skype, Zoom, e-mail, Git, tablica

## ➤ Dokumentacja



Google Docs, Sharepoint, Wiki, Confluence,  
edytory tekstu

## ➤ Zarządzanie

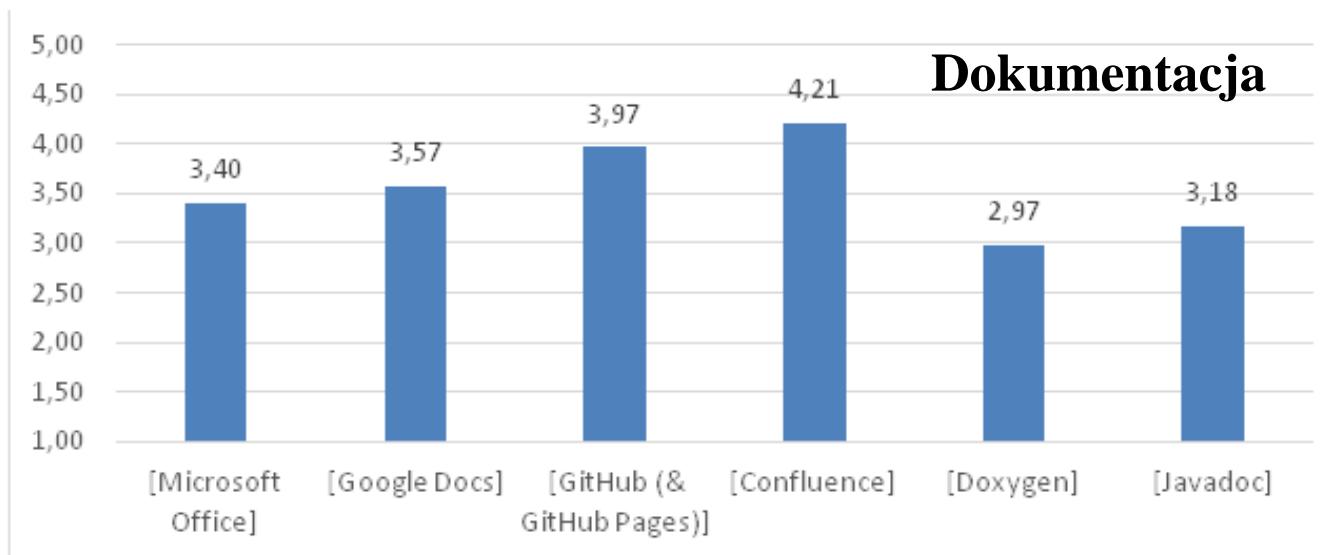
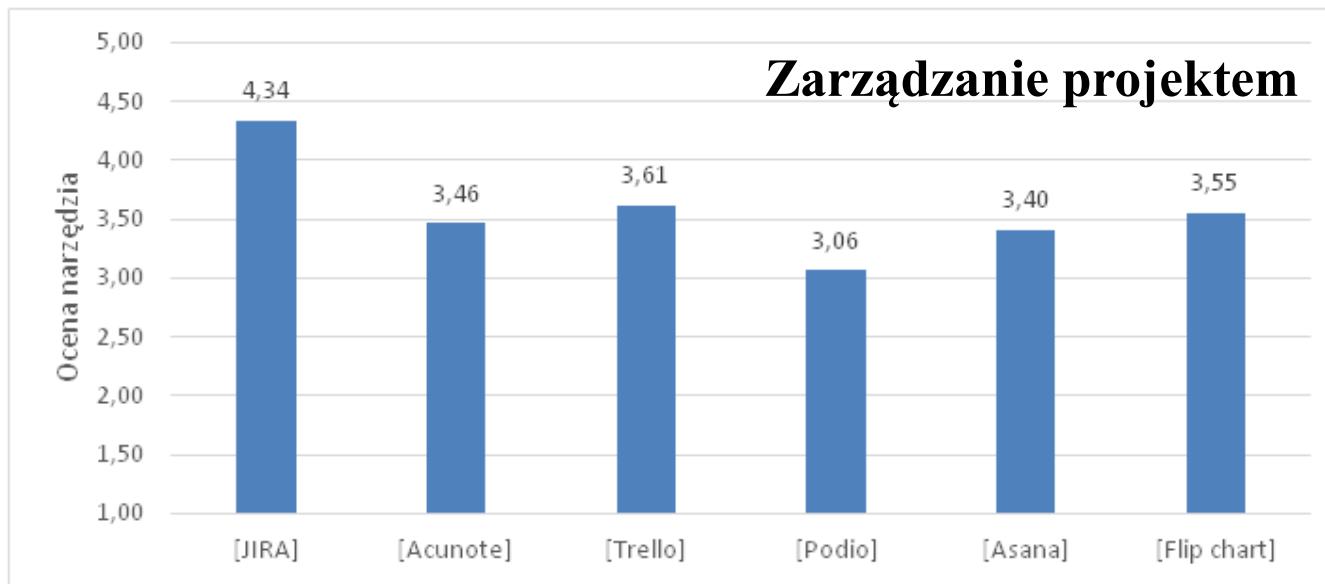


MS Project, GanttProject, Open Workbench  
JIRA

# Infrastruktura – narzędzia Agile

- JIRA Cloud – <https://www.atlassian.com/software/jira/free>
- Azure DevOps (dawniej Microsoft Team Foundation Service (TFS))
- ScrumDesk - [www.scrumdesk.com](http://www.scrumdesk.com)
- Trello – [www.trello.com](http://www.trello.com)
- Yodiz - [www.yodiz.com](http://www.yodiz.com)
- Acunote - [www.acunote.com](http://www.acunote.com)
- Agilo for Trac - [www.agilofortrac.com](http://www.agilofortrac.com)
- ScrumWorksPro - <https://www.collab.net/products/scrumworks>
- CA Agile Central (dawniej Rally) - [www.ca.com/us/products/ca-agile-central.html](http://www.ca.com/us/products/ca-agile-central.html)
- Mingle - <https://www.thoughtworks.com/mingle/>

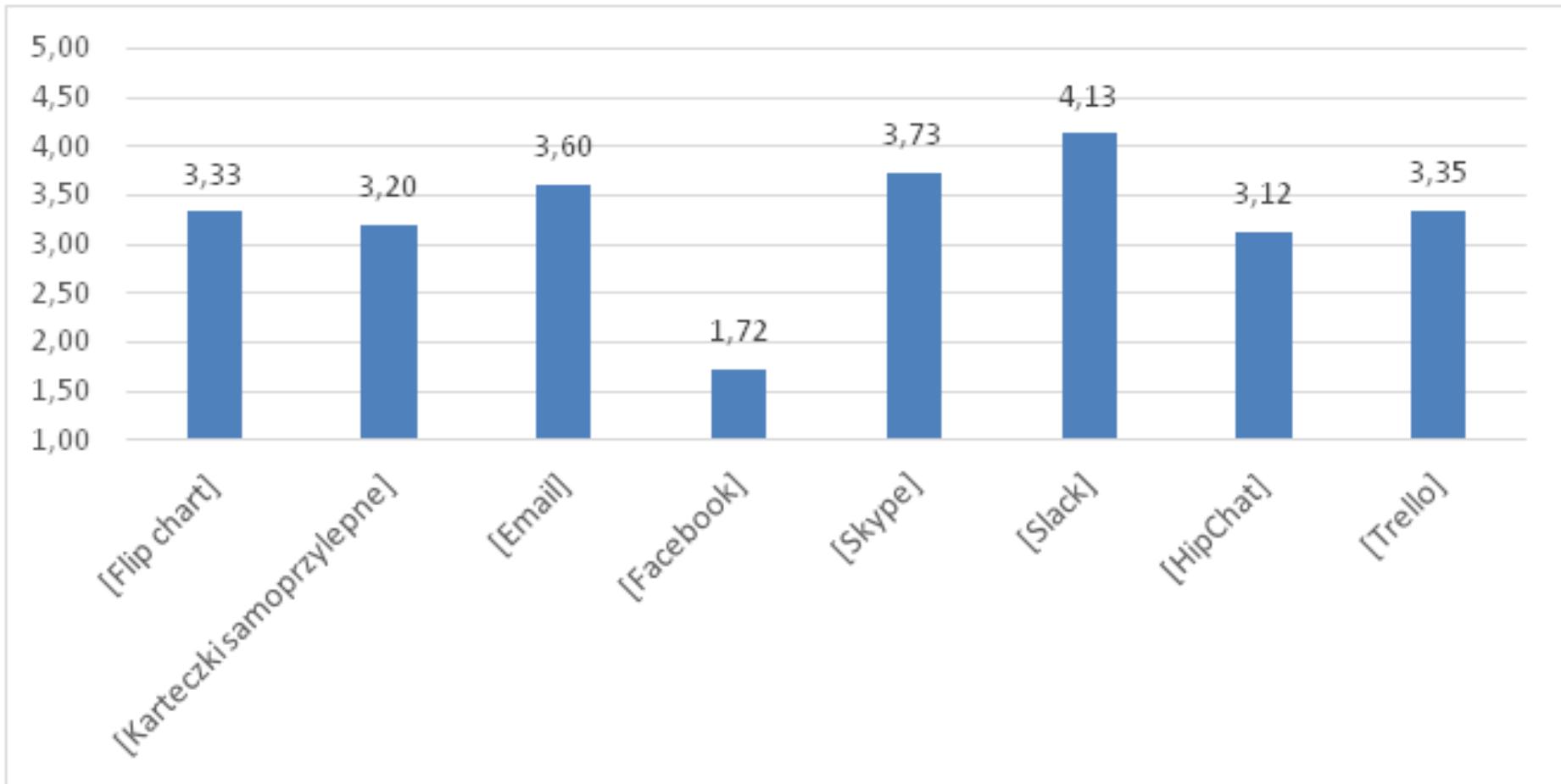
# Rankingi narzędzi dla Scruma



Źródło: Tyberiusz Kriger, *Analiza narzędzi wspomagających projekty prowadzone metodą Scrum*, praca magisterska, opiekun: Jakub Miler, WETI, Politechnika Gdańskia, 2017

# Rankingi narzędzi dla Scruma (2)

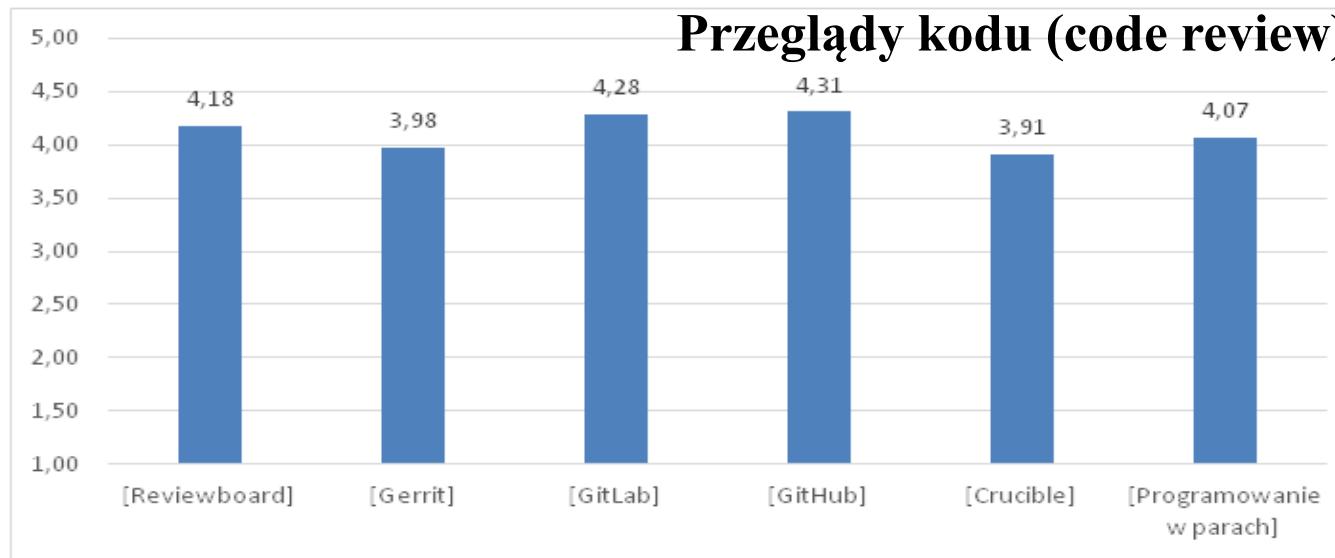
## Komunikacja i współpraca



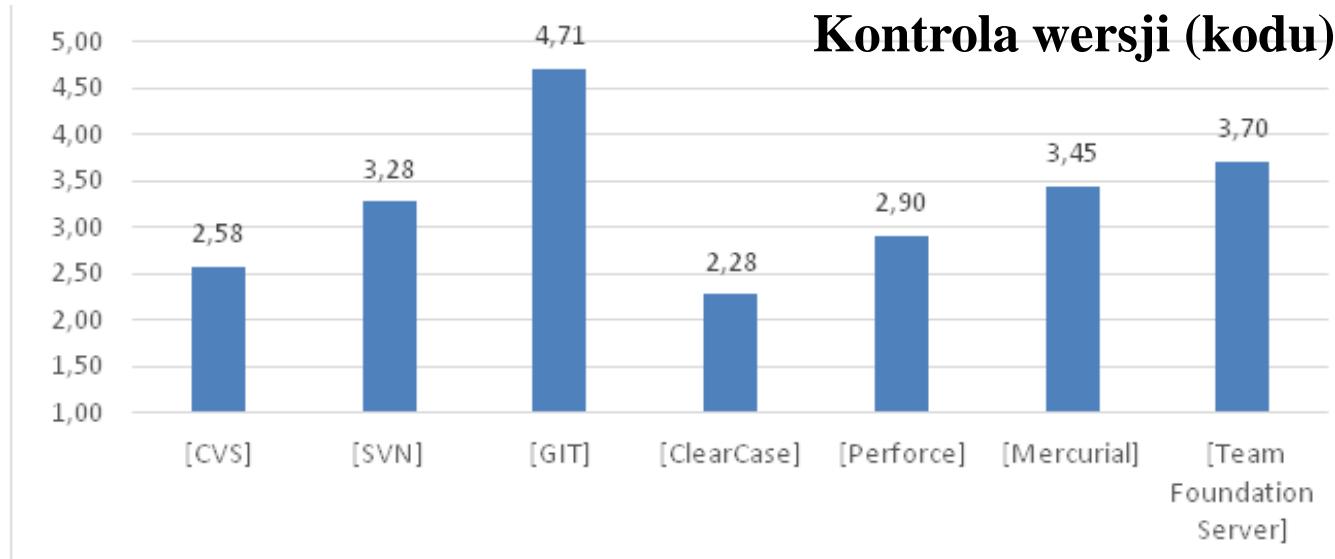
Źródło: Tyberiusz Kriger, *Analiza narzędzi wspomagających projekty prowadzone metodą Scrum*, praca magisterska, opiekun: Jakub Miler, WETI, Politechnika Gdańskia, 2017

# Rankingi narzędzi dla Scruma (3)

## Przeglądy kodu (code review)



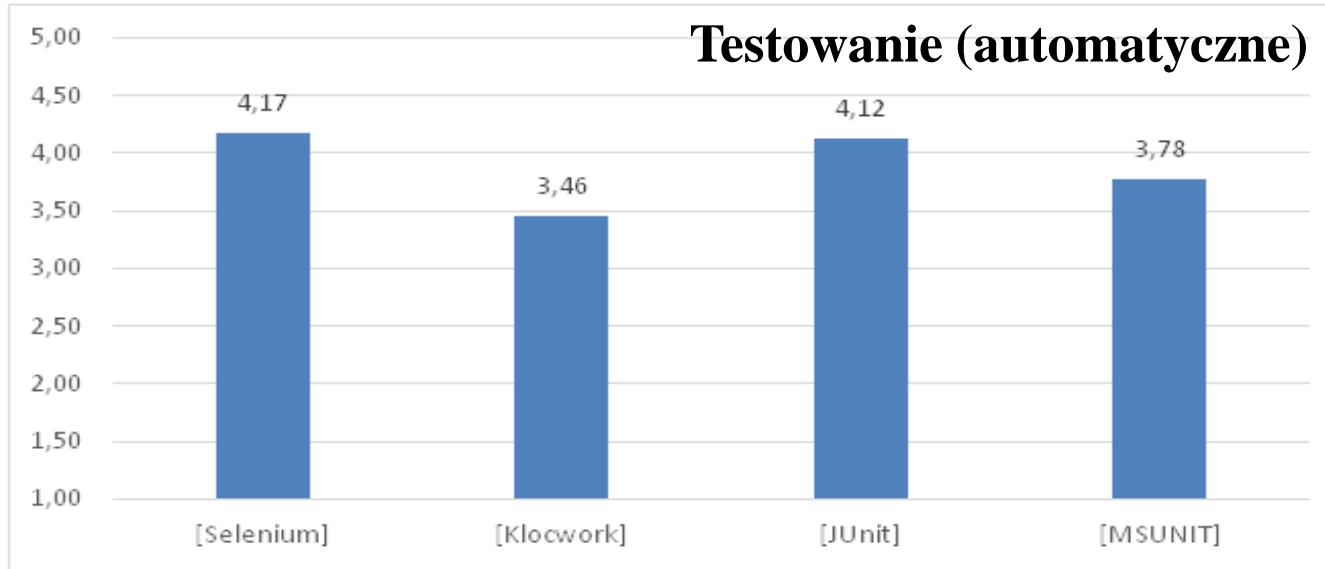
## Kontrola wersji (kodu)



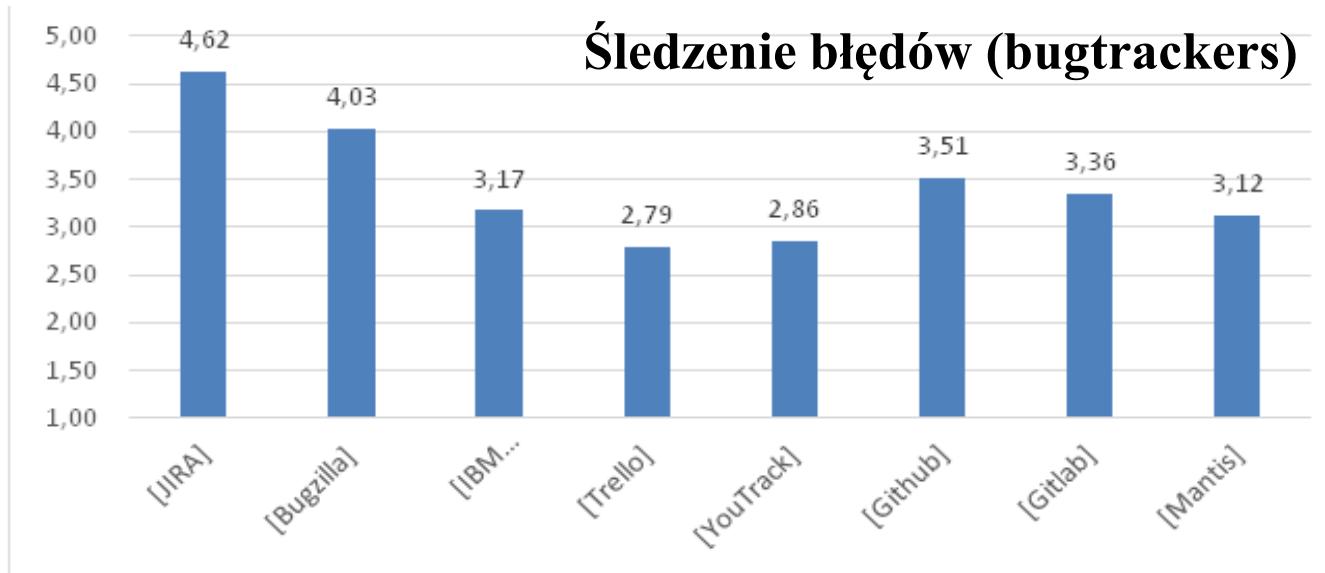
Źródło: Tyberiusz Kriger, *Analiza narzędzi wspomagających projekty prowadzone metodą Scrum*, praca magisterska, opiekun: Jakub Miler, WETI, Politechnika Gdańskia, 2017

# Rankingi narzędzi dla Scruma (4)

## Testowanie (automatyczne)



## Śledzenie błędów (bugtrackers)



Źródło: Tyberiusz Kriger, *Analiza narzędzi wspomagających projekty prowadzone metodą Scrum*, praca magisterska, opiekun: Jakub Miler, WETI, Politechnika Gdańskia, 2017

# Rekomendacje narzędzi dla Scruma

Kategoria	Rekomendowane narzędzie	Średnia ocena przydatności	Liczba ocen
Zarządzanie projektem	JIRA	4,34	91
	Trello	3,61	63
Współpraca projektowa	Slack	4,13	75
	Skype	3,73	100
Tworzenie dokumentacji	Confluence	4,12	59
	GitHub (& GitHub Pages)	3,97	76
Przeglądy kodu	GitHub	4,31	75
	GitLab	4,28	42
Testowanie	Selenium	4,17	56
	JUnit	4,12	47
Systemy kontroli wersji	GIT	4,71	94
	Team Foundation Server	3,70	27
Bugtrackery	JIRA	4,62	83
	Bugzilla	4,03	44

# Przykłady projektów

**GraPM – sukces zwinności**

**System motywacyjny dla pacjentów – sukces dyscypliny**

**RiskGuide 3.0 – Scrum z problemami**

**TCT/NOR-STA – sukces adaptacji**

# GraPM – sukces zwinności

- **Projekt dyplomowy inżynierski pt.**  
*„Gra edukacyjna ucząca zarządzania projektami”*
- **Czas realizacji całego projektu dyplomowego: marzec 2014 – grudzień 2014**
- **Projekt realizowany zgodnie z metodyką Scrum, zespół 4-osobowy**
- **Klient – opiekun projektu dr inż. Jakub Miler**

# GraPM - produkt

[grapm.eti.pg.gda.pl](http://grapm.eti.pg.gda.pl)

Projekt: Mrowisko Punkty: -100

Czas:	dzień: 0	Klient:	<div style="width: 50%;"> </div>
Zakres:	2%	Zespół:	<div style="width: 100%;"> </div>
Jakość:	<div style="width: 10%;"> </div>	Ryzyko:	<div style="width: 20%;"> </div>

Zasoby

Robotnice	10 ↗
Opiekunki	3 ↗
Wartownicy	1 ↗
Zbieracze	6 ↗
Hodowcy	5 ↗

Ryzyka

Niski przyrost naturalny	70% !!
--------------------------	--------

Produkt

Komora królowej	10 ⚡ 2 ★ 30 ↗
Magazyn jedzenia świeżego	6 ⚡ 1 ★ 40 ↗
Magazyn jedzenia martwego	3 ⚡ 0 ★ 20 ↗
Wylegarnia zapasowa	8 ⚡ 0 ★ 20 ↗
Korytarze główne	8 ⚡ 1 ★ 40 ↗
Korytarze boczne	3 ⚡ -2 ★ 30 ↗
Kanały wentylacyjne	8 ⚡ 1 ★ 20 ↗
Wejścia do mrowiska	5 ⚡ -3 ★ 30 ↗
Izba odpoczynku	1 ⚡ 2 ★ 20 ↗
Hodowla grzybów	6 ⚡ 2 ★ 30 ↗
Pokoje parad	2 ⚡ -2 ★ 10 ↗

Zadania

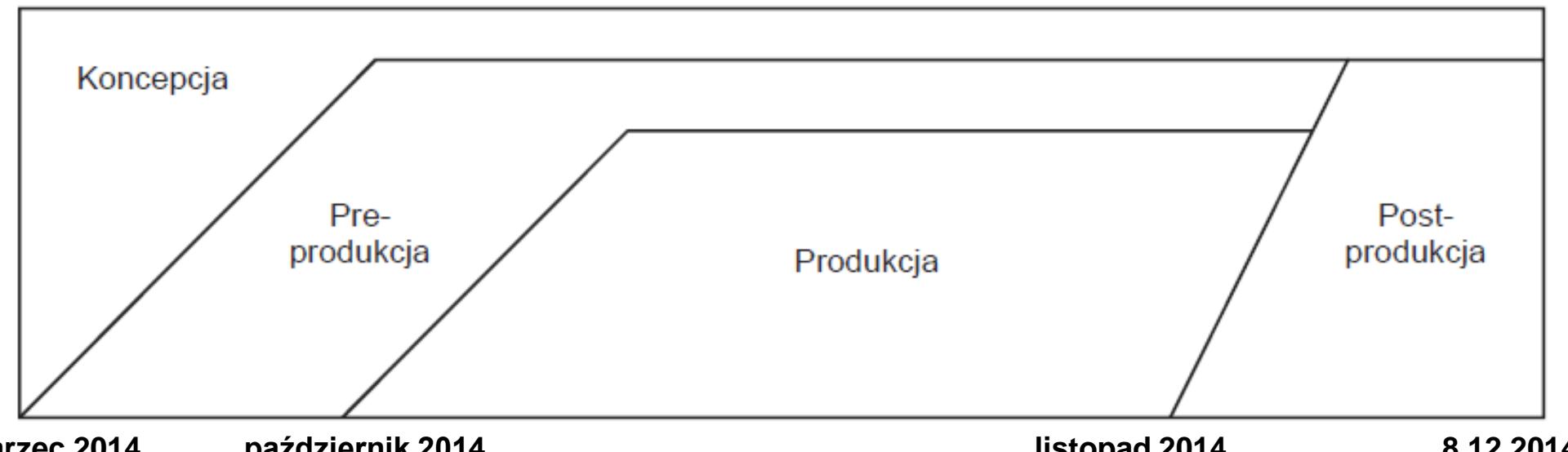
Wylegarnia wielka (Rob)	10 ⚡ 1 ★ 34 ↗ x
Szkoła robotnic (Opi)	6 ⚡ 0 ★ 8 ↗ x
Koszary wartowników (War)	4 ⚡ 0 ★ 9 ↗ x
Otwory ewakuacyjne (Zbi)	3 ⚡ 3 ★ 37 ↗ x

Działania wobec ryzyka

Wysłanie zwiadowców	8 ⚡
Testowanie mrowiska	10 ⚡
Poszukiwanie nowych grzybów	6 ⚡
Powiększenie salonu gier i zabaw	12 ⚡
Zgromadzenie zapasów jedzenia	16 ⚡
Wystawienie licznych wartowników	8 ⚡
Powiększenie magazynów	16 ⚡
Przeszkolenie opiekunek	12 ⚡
Powiększenie korytarzy	10 ⚡
Zmniejszenie tempa pracy	6 ⚡

# GraPM – fazy projektu

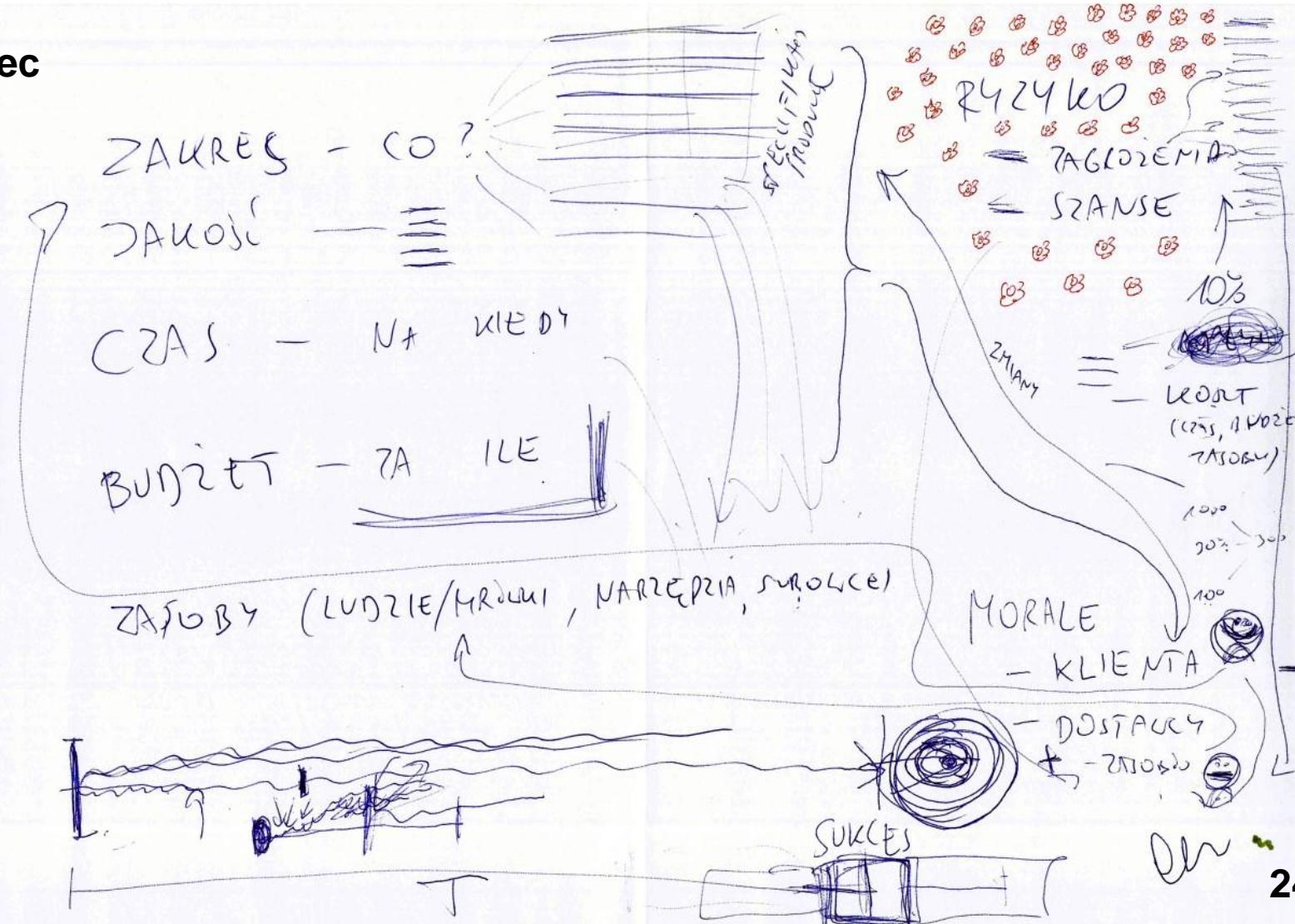
- **Faza koncepcyjna** – marzec-czerwiec 2014
- **Faza pre-produkcji** – czerwiec-październik 2014 (w tym sprint 1)
- **Faza produkcji** – październik-listopad 2014 (sprinty 2 i 3)
- **Faza post-produkcji** – listopad-grudzień 2014 (w tym sprint 4)
- **Scrum: 4 sprinty 2-tygodniowe: 7.10.2014 – 2.12.2014**



# GraPM – faza koncepcyjna

## Projekt gry (wstępny)

marzec  
-czerwiec  
2014



24.06.2014

# GraPM – faza pre-produkcji

## Rejestr produktu w Acunote (fragment)

lipiec  
2014

Sprint 5 - dokumentacja						More Actions				
	No.	Description	Owner	Status	Pri.	Est.	Rem.			
	14	Cel gry - ukończenie projektu zgodnie z celem projektu, z jak najwyższym wskaźnikiem sukcesu - punktami (wiki)	-	Not Started	P1	5	5			
	15	Cel projektu - docelowe wartości poszczególnych atrybutów projektu: czas, koszt, zakres, jakość, zadowolenie klienta, zadowolenie zespołu dostawcy (wiki)	-	Not Started	P0	5	5			
	16	Atrybuty projektu	-	Not Started	P0	23	23			
	17	Zakres (łączna wartość biznesowa wykonanych elementów produktu)	-	Not Started	P0	2	2			
	18	Jakość (wiki)	-	Not Started	P0	2	2			
	19	Czas (wiki)	-	Not Started	P1	2	2			
	113	Ryzyko - ogólny poziom ryzyka projektu (wiki)	-	Not Started	P1	5	5			
	20	Koszt	-	Not Started	P2	2	2			
	21	Zadowolenie Klienta - wyliczane z czasu, kosztu, zakresu i jakości (wiki, 2 comments)	-	Not Started	P0	5	5			
	22	Zadowolenie zespołu dostawcy - wyliczane z czasu, kosztu, zakresu, jakości (wiki)	-	Not Started	P1	5	5			
	23	Bieżący wskaźnik sukcesu łączący bieżące wartości poszczególnych atrybutów projektu (wiki)	-	Not Started	P0	8	8			
	24	Prezentacja docelowych i bieżących atrybutów projektu (składowych celu i wskaźnika sukcesu) (wiki)	-	Not Started	P0	8	8			
	25	Różne kryteria końca gry (wiki)	-	Not Started	P1	17	17			
	26	Realizacja pełnego zakresu projektu (wiki)	-	Not Started	P1	3	3			
	114	Realizacja jak najlepiej projektu w ograniczonym czasie (wiki)	-	Not Started	P1	3	3			
	27	Osiągnięcie określonej wartości biznesowej produktu	-	Not Started	P2	3	3			
	28	Osiągnięcie maksymalnego zadowolenia klienta	-	Not Started	P1	5	5			
	29	Wybór kryterium końca projektu przy starcie gry (wiki)	-	Not Started	P1	3	3			
	30	Zakres i jakość produktu - lista elementów/cech produktu do wytworzenia	-	Not Started	P0	62	62			
	31	Atrybuty elementów produktu	-	Not Started	P0	36	36			
	32	Nazwa elementu zależna od tematu gry	-	Not Started	P0	8	8			
	33	Wartość biznesowa - przyrost zakresu	-	Not Started	P0	5	5			
	34	Jakość - przyrost lub utrata jakości	-	Not Started	P1	5	5			
	35	Rozmiar elementu - nakład pracy zasobów	-	Not Started	P0	13	13			
	36	Specjalizacja zasobu - wymagane specjalne zasoby	-	Not Started	P2	5	5			
	37	Elementy konieczne do wykonania wcześniej	-	Not Started	P2	-	-			
	38	Zmieniający się zakres produktu w poszczególnych turach gry	-	Not Started	P1	13	13			
	39	Nowe elementy/cechy produktu - klient dodaje	-	Not Started	P1	5	5			
	40	Znikanie elementów/cech produktu - klient rezygnuje	-	Not Started	P1	5	5			
	41	Zmieniające się atrybuty elementów/cech produktu	-	Not Started	P2	3	3			
	42	Wizualizacja graficzna elementów zakresu i jakości produktu	-	Not Started	P1	13	13			

**Źródło:** Rafał Piechowski, Damian Płatek, Anna Wasik, Sylwia Grabowska, *Gra edukacyjna ucząca zarządzania projektami*, projekt dyplomowy inżynierski, opiekun: Jakub Miler, WETI, Politechnika Gdańskia, 2014

# GraPM – faza pre-produkcji

## Role gamedev / Scrum

Osoba	Rola gamedev	Rola Scrum
Jakub Miler	Główny projektant Scenarzysta Projektant interfejsu	Właściciel Produktu
Jakub Miler	Producent	Scrum Master
Jakub Miler	Tester	Deweloper
Rafał Piechowski	Główny programista Programista mechaniki gry Programista interfejsu	Deweloper w Zespole Deweloperskim
Damian Płatek	Programista mechaniki gry Programista interfejsu Zastępca producenta	Deweloper w Zespole Deweloperskim
Anna Wasik	Programista mechaniki gry Programista interfejsu Tester	Deweloper w Zespole Deweloperskim
Sylwia Grabowska	Programista mechaniki gry Twórca interfejsu Programista interfejsu	Deweloper w Zespole Deweloperskim

# GraPM – faza pre-produkcji Prototyp

Przyrost po sprincie 1



Źródło: Rafał Piechowski, Damian Płatek, Anna Wasik, Sylwia Grabowska, *Gra edukacyjna ucząca zarządzania projektami*, projekt dyplomowy inżynierski, opiekun: Jakub Miler, WETI, Politechnika Gdańskia, 2014

# GraPM – faza produkcji

## Wersja 1 gry

Przyrost po sprincie 2

### Gra PM

#### Produkty

Wejścia do mrowiska	-3 5 30
Otwory ewakuacyjne	3 3 40
Izba odpoczynku	2 1 20
Pokój narad	-2 2 10
Salon gier i zabaw	-2 8 30
Sypialnie	1 6 30

#### Zasoby

Robotnice
Opiekunki
Wartownicy
Tłumacz

#### Zadania

Magazyn jedzenia świeżego	1 6 40
Hodowla grzybów	2 6 30
Maskowanie mrowiska	2 3 50

#### Ryzyka

(Empty)

#### Działania zapobiegawcze

Wysłanie zwiadówców	8
Testowanie mrowiska	10
Poszukiwanie nowych grzybów	6
Powielczanie szkolenia zatrudnionych	12

Wykonano: 0%   Wynik: 0   Satysfakcja klienta: 3   Zadowolenie zespołu: 3   Dzień 0. Godzina - 08:23.



Źródło: Rafał Piechowski, Damian Płatek, Anna Wasik, Sylwia Grabowska, *Gra edukacyjna ucząca zarządzania projektami*, projekt dyplomowy inżynierski, opiekun: Jakub Miler, WETI, Politechnika Gdańskia, 2014

- 33 -

© WETI PG

# GraPM – faza produkcji

## Wersja 2 gry

Przyrost po sprincie 3

 Zarządzanie Projektem : Mrowisko

Zakres: 0% Klient:   Wynik: 0

Jakość:   Zespół:  

Czas: Dzień: 0 Ryzyko:   10%

Zasoby

Robotnice	10
Opiekunki	3
Wartownicy	1
Zbieracze	6
Hodowcy	5
Zwiadowcy	2

Ryzyka

Produkty

Komora królowej	10 ⚡ 2 ★ 30 ↗
Magazyn jedzenia świeżego	6 ⚡ 1 ★ 40 ↗
Magazyn jedzenia martwego	3 ⚡ 0 ★ 20 ↗
Wylegarnia wielka	10 ⚡ 1 ★ 40 ↗
Wylegarnia zapasowa	8 ⚡ 0 ★ 20 ↗
Szkoła robotnic	6 ⚡ 0 ★ 10 ↗
Koszary wartowników	4 ⚡ 1 ★ 10 ↗
Korytarze główne	8 ⚡ 1 ★ 40 ↗
Korytarze boczne	3 ⚡ -2 ★ 30 ↗
Kanały wentylacyjne	8 ⚡ 1 ★ 20 ↗
Wejścia do mrowiska	5 ⚡ -3 ★ 30 ↗
Otwory ewakuacyjne	3 ⚡ 3 ★ 40 ↗
Izba odpoczynku	1 ⚡ 2 ★ 20 ↗

Zadania

▶

Działania wobec ryzyka

Wysłanie zwiadowców	6 ✅
Testowanie mrowiska	6 ✅
Poszukiwanie nowych grzybów	4 ✅
Powiększenie salonu gier i zabaw	8 ✅
Zgromadzenie zapasów jedzenia	10 ✅
Wystawienie licznych wartowników	6 ✅
Powiększenie magazynów	10 ✅
Przeszkolenie opiekunek	8 ✅

# GraPM – faza post-produkcji

## Testy i optymalizacja

Sprint 4

Tabela 8.1 Zgłoszenia w Sprincie 4

wszystkich zgłoszeń:	53
propozycje zmian:	7
błędy:	46

Tabela 8.2 Rodzaje błędów w Sprincie 4

mylne zgłoszenia błędów:	6
błędy o priorytecie P0	12
błędy o priorytecie P1	14
błędy o priorytecie P2	13
błędy naprawione:	39

# GraPM – faza post-produkcji

## Wersja finalna gry

**Przyrost po sprincie 4**

Projekt: Mrowisko      Punkty: -100

Czas:	dzień: 0	Klient:	
Zakres:	2%	Zespół:	
Jakość:		Ryzyko:	20%

**Zasoby**

Robotnice	10
Opiekunki	3
Wartownicy	1
Zbieracze	6
Hodowcy	5

**Ryzyka**

- Niski przyrost naturalny 70%

**Produkt**

Komora królowej	10  2  30
Magazyn jedzenia świeżego	6  1  40
Magazyn jedzenia martwego	3  0  20
Wylegarnia zapasowa	8  0  20
Korytarze główne	8  1  40
Korytarze boczne	3  -2  30
Kanały wentylacyjne	8  1  20
Wejścia do mrowiska	5  -3  30
Izba odpoczynku	1  2  20
Hodowla grzybów	6  2  30
Pokój narad	2  -2  10

**Zadania**

Wylegarnia wielka (Rob)	10  1  34
Szkoła robotnic (Opi)	6  0  8
Koszary wartowników (War)	4  0  9
Otwory ewakuacyjne (Zbi)	3  3  37

**Działania wobec ryzyka**

Wysłanie zwiadowców	8
Testowanie mrowiska	10
Poszukiwanie nowych grzybów	6
Powiększenie salonu gier i zabaw	12
Zgromadzenie zapasów jedzenia	16
Wystawienie licznych wartowników	8
Powiększenie magazynów	16
Przeszkolenie opiekunek	12
Powiększenie korytarzy	10
Zmniejszenie tempa pracy	6

# System motywacyjny dla pacjentów – sukces dyscypliny

- Projekt dyplomowy inżynierski pt. „*System motywacyjny dla pacjentów*”
- Czas realizacji całego projektu dyplomowego: kwiecień 2011 – grudzień 2011
- Etapy projektu
  - kwiecień – wrzesień:
    - wizja, organizacja infrastruktury, przegląd technologii, zbieranie wymagań, dokumentacja przypadków użycia
    - odpowiada RUP Inception
  - październik:
    - dokończenie zbierania wymagań, projekt bazy danych i architektury, „trzon” aplikacji
    - odpowiada RUP Elaboration (2 iteracje)
  - listopad:
    - implementacja i testowanie ustalonego zakresu funkcjonalnego
    - odpowiada RUP Construction (5 iteracji)
  - grudzień:
    - finalizacja, testy akceptacyjne, zamykanie dokumentacji
    - odpowiada RUP Transition

# System motywacyjny dla pacjentów – cechy i wnioski

## ➤ Cechy projektu

- 4 osoby w zespole, w tym 1 znająca już technologię Oracle ADF
- wielu udziałowców, wiele źródeł wymagań, brak jednego reprezentanta dziedziny
- zrealizowano funkcje dla 3 aktorów, 11 przypadków użycia
- 8 wymagań pozafunkcjonalnych
- baza danych Oracle 11.2
- framework Oracle Application Development Framework (ADF)
- środowisko Oracle JDeveloper Studio Edition 11.1.2
- infrastruktura: DokuWiki z rozszerzeniami (kalendarz, backup), VirtualSVN, Enterprise Architect
- nie zrealizowano części zakresu zidentyfikowanego w analizie

## ➤ Wnioski

- sporo czasu zajmuje interakcja między osobami w zespole i wspólne podejmowanie decyzji
- trzeba uwzględnić czas na uczenie się nowej technologii
- zdarzają się awarie infrastruktury
- dobrze pracować w jednym miejscu – szybsze rozwiązywanie problemów

# RiskGuide 3.0 – Scrum z problemami

- **Projekt grupowy pt. „Internetowe narzędzie wspomagające zarządzanie ryzykiem w projekcie informatycznym”**
- **Czas realizacji: marzec 2010 – luty 2011**
- **Zespół projektu**
  - Klient – Jakub Miler
  - Wykonawcy – 3 studentów 8 i 9 sem. studiów jednolitych magisterskich
- **Cechy projektu**
  - Istniejący sprawdzony produkt RiskGuide 2
  - Cel: nowa generacja narzędzia w nowych technologiach
  - Założenie technologii: .NET/MS SQL lub Java/PostgreSQL
  - Bardzo kompetentny i zaangażowany klient
  - Zespół wykonawców znał się wcześniej, częściowo pracował razem w projektach
  - Wysokie wymagania co do ochrony danych, ergonomii, elastyczności wspieranego procesu
  - Oczekiwany produkt o potencjale komercyjnym
  - Założony termin wdrożenia – koniec stycznia 2011

# RiskGuide 3.0 - produkt

## RiskGuide<sup>3</sup> Risk Management Tool

Strona główna Moje konto (Jan Nowak) Moje projekty EN Wyloguj się

RiskGuide > Przeglądanie listy projektów >>

**Przeglądy Raporty Zespół**

**Identyfikacja Szacowanie Analiza Raport**

Projekt: Przykładowy projekt  
 Przegląd: Przykładowy przegląd  
 Raport:

**+ Nowy projekt**

Data	Projekt	Maksymalne oszacowanie	Domyślny?
Sortuj	Sortuj	Sortuj	Sortuj
Mon, 31 Jan 2011	<b>Przykładowy projekt</b> Jan Nowak Opis przykładowego projektu Pokaż/ukryj kategorie ryzyk <b>+</b>	10	<input checked="" type="checkbox"/>
Mon, 31 Jan 2011	<b>Kolejny przykładowy projekt</b> Jan Nowak Opis tegoż projektu. Pokaż/ukryj kategorie ryzyk <b>+</b>	6	<input type="checkbox"/>

Zespół Projektowy KIO © 2010  
 Risk Guide ver 3.0

# RiskGuide 3.0 – podsumowanie

## ➤ Rezultat projektu

- pokrycie większości wymagań kluczowych, bardzo niewielu ponadto
- wdrożenie opóźnione o ok. 2 miesiące
- użyta technologia to PHP (z PHPTal), baza danych MySQL – utrzymanie?
- brak testów akceptacyjnych – czy rzeczywiście działa?
- brak dokumentacji projektowej (jedynie podręczniki użytkownika i administratora)

## ➤ Problemy

- chaotyczne zaangażowanie jednego z członków Zespołu (głównego projektanta i programisty interfejsu użytkownika)
- osobne repozytorium jednej z osób – problemy ze scalaniem kodu
- błędny projekt, szczególnie interfejsu użytkownika – konieczne długotrwałe poprawki

## ➤ Wnioski

- Scrum nie był dobrą decyzją – były stabilne wymagania, zespół nie w pełni znał technologię, występowali duże problemy komunikacyjne w zespole
- przy stabilnych wymaganiach konieczna jest szczegółowa specyfikacja na początku
- wspólna infrastruktura to absolutna podstawa współpracy w zespole
- zwinność wymaga dobrej znajomości technologii
- lepsze są krótsze iteracje

# Projekt TCT/NOR-STA – sukces adaptacji

- Cel projektu: dostarczyć narzędzie wspomagające zespołowe tworzenie i utrzymywanie argumentacji (drzew z załącznikami)
- Prowadzony przez zespół w Katedrze Inżynierii Oprogramowania
- Cechy projektu:
  - na początku produkt badawczy
  - różne obszary zastosowań argumentacji
  - podprojekt różnych projektów badawczych
  - długa historia – start grudzień 2005
  - mały zespół (ok. 5 osób)
  - stopniowe zmiany w składzie zespołu
  - ponad 100 użytkowników
  - wysokie wymagania jakościowe
- Obecnie produkt rozwijany jako platforma NOR-STA w spółce spin-off Argevide sp. z o.o.



# TCT / NOR-STA - przebieg wydań (1)

- **Wydanie 1.0 – czerwiec 2006**
  - dobrze znane wymagania, duża niepewność technologiczna, charyzmatyczny kierownik
  - proces pełny, zdyscyplinowany, sterowany planem
- **Wydanie 2.0 – marzec 2007**
  - nowe wymagania, zmiana projektu kodu, zmiana składu zespołu (w tym kierownika)
  - proces łączący zwinność i dyscyplinę, różne praktyki
- **Wydanie 3.0 – marzec 2008**
  - praca magisterska, duża niepewność wymagań, nieznane kompetencje wykonawcy
  - proces z przewagą zwinności
- **Wydanie 4.0 – grudzień 2010**
  - stabilne wymagania i zespół, dojrzała technologia, wysokie wymagania jakościowe
  - proces z przewagą dyscypliny
- **Wydanie 5.0 – październik 2012**
  - stabilny, profesjonalny zespół, reimplementacja aplikacji klienta w nowej technologii
  - proces z przewagą dyscypliny poprzedzony wnikliwym prototypowaniem

# TCT / NOR-STA - przebieg wydań (2)

## ➤ Wydanie 6.0 – październik 2013

- stabilny, profesjonalny zespół
- ostatnie wydanie przed komercjalizacją
- proces z przewagą dyscypliny

## ➤ Wydania 6.1 – 6.5 – co ok. 2 miesiące od kwietnia 2014

- wydania rozwijane w spółce spin-off Argevide Sp. z o.o.
- przejście na krótsze wydania jak w podejściu zwinnym
- zmiana infrastruktury (JIRA zamiast Mantisa)
- zachowany zespół projektu
- intensywna praca programistów, małe zadania, krótkie iteracje
- elementy dyscypliny: specyfikacje, fazy, role
- proces łączący zwinność i dyscyplinę

## ➤ Wydanie 6.6 – grudzień 2015

- wydanie rozwijane w spółce spin-off Argevide Sp. z o.o.
- budowa nowego modułu administracji
- elementy dyscypliny: specyfikacje, projekt, rozbudowane testowanie
- proces z przewagą dyscypliny

# Platforma NOR-STA - produkt

Project View Account Help
Log out

- [i] Conformance of NOR-STA project to the "Guide On Promoting Projects Financed Within Operational Programme Innovative
- [i] Introduction to the standard
- [i] Conformance argument
  - [i] NOR-STA conformance to the Guide
    - [i] Argument by demonstrating conformity to all requirements from the Guide
      - ✓ Requirements of the Guide
      - ✓ Labelling
        - [i] Argument by conformity with requirements concerning labelling all kinds of products
          - ✓ Requirements on labelling all kinds of products with uniform visual identification
          - ✓ Labelling the boards
          - ✓ Labelling of large-size products
          - ✓ Labelling of small-size products
        - ✓ Uniformity of the visual identification
          - [i] Argument by reference to the requirements
            - ✓ Requirements on visual identification elements
            - ✓ Text content and form
            - ✓ Colouring of the project products' labelling
            - ✓ Font of the project products' labelling
          - ✓ Placing information and memorial boards
          - ✓ Informing the public about the project being co-financed from the EU funds
          - ✓ Promotion costs do not exceed the limit
      - [i] References to the original text of the Guide

Details

Assessment

Belief:

Disbelief:

Uncertainty:

Confidence:

with high confidence

Decision:

opposable

Assessment was calculated automatically.

Links
Versions

[03-02-2013 13:41:37] Node opened Uniformity of the visual identification

# Projekt TCT / NOR-STA - podsumowanie

## ➤ Rezultat

- aplikacja w użyciu od pierwszego wydania
- utrzymywana wysoka jakość (w tym niezawodność, wydajność, dostępność)
- obecnie produkt skomercjalizowany i rozwijany w spółce spin-off

## ➤ Źródła sukcesu

- właściwy dobór organizacji projektu do okoliczności danego wydania
- udane połączenie dyscypliny i zwinności
- aktualizowana dokumentacja projektowa
- skuteczna i trwała infrastruktura (m.in. SVN, Mantis, JIRA)
- dobre kluczowe decyzje technologiczne
- skuteczne zarządzanie projektem
- stary zespół, powolne i kontrolowane zmiany w składzie zespołu

# Realizacja Projektu Informatycznego



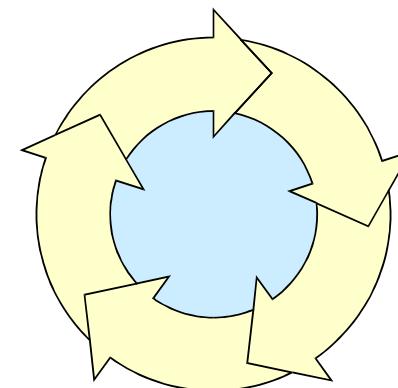
***Jakub Miler***

*Katedra Inżynierii Oprogramowania  
Wydział Elektroniki, Telekomunikacji i Informatyki  
Politechnika Gdańskia  
jakubm@eti.pg.edu.pl*

*Materiały pomocnicze do wykładu na Wydziale ETI Politechniki Gdańskiej.  
Wykorzystanie materiałów w innym celu i ich rozpowszechnianie bez zgody WETI PG zabronione.*

# Metodyki wytwarzania oprogramowania

## Wprowadzenie



# Wyzwania współczesnych projektów

- Krótkie terminy
- Małe budżety
- Częste i daleko idące zmiany
- Duża złożoność
- Zróżnicowane, szybko zmieniające się technologie
- Wysokie wymagania kompetencji, specjalizacja
- Praca zespołowa i rotacja personelu
  
- Skutki: ok. 50% projektów ma problemy, ok. 20% nie udaje się  
[Standish Group]

# Po co metodyka?

## Metodyka, metoda, framework

Zestaw wartości, zasad, praktyk i narzędzi mający na celu skutecną realizację określonego zadania

- Metodyki porządkują proces wytwarzania oprogramowania
- Wyznaczają główne zasady organizacji pracy w projekcie
- Pozwalają efektywniej pracować w zespole i lepiej dzielić się pracą
- Zawierają liczne wskazówki, jak realizować konkretne zadania – praktyki
- Dla najpopularniejszych metodyk istnieje wiele narzędzi wspomagających

# Manifest Zwinnego Tworzenia Oprogramowania

Wytwarzając oprogramowanie i pomagając innym w tym zakresie,  
odkrywamy lepsze sposoby wykonywania tej pracy.

W wyniku tych doświadczeń przedkładamy:

**Ludzi i interakcje** ponad procesy i narzędzia.

**Działające oprogramowanie** ponad obszerną dokumentację.

**Współpracę z klientem** ponad formalne ustalenia.

**Reagowanie na zmiany** ponad podążanie za planem.

Doceniamy to, co wymieniono po prawej stronie,  
jednak bardziej cenimy to, co po lewej.

Źródło: [www.agilemanifesto.org/iso/pl](http://www.agilemanifesto.org/iso/pl)

# Zasady kryjące się za Manifestem Zwinnego Wytwarzania Oprogramowania

*Kierujemy się następującymi zasadami:*

Najważniejsze dla nas jest **zadowolenie Klienta** wynikające z wcześnie rozpoczętego i ciągłego dostarczania wartościowego oprogramowania.

Bądź otwarty na **zmieniające się wymagania** nawet na zaawansowanym etapie projektu. Zwinne procesy wykorzystują zmiany dla uzyskania przewagi konkurencyjnej Klienta.

**Często dostarczaj działające oprogramowanie** od kilku tygodni do paru miesięcy, im krócej tym lepiej z preferencją krótszych terminów.

**Współpraca** między ludźmi biznesu i programistami musi odbywać się **codziennie** w trakcie trwania projektu.

Źródło: <http://agilemanifesto.org/iso/pl/principles.html>

## Zasady kryjące się za Manifestem Zwinnego Wytwarzania Oprogramowania (2)

Twórz projekty wokół **zmotywowanych osób**. Daj im środowisko i wsparcie, którego potrzebują i ufaj im, że wykonają swoją pracę.

Najwydajniejszym i najskuteczniejszym sposobem przekazywania informacji do i ramach zespołu jest **rozmowa twarzą w twarz**

Podstawową i najważniejszą miarą postępu jest  
**działające oprogramowanie.**

Zwinne procesy tworzą środowisko do równomiernego rozwijania oprogramowania. **Równomierne tempo** powinno być nieustannie utrzymywane poprzez sponsorów, programistów oraz użytkowników.

Źródło: <http://agilemanifesto.org/iso/pl/principles.html>

# Zasady kryjące się za Manifestem Zwinnego Wytwarzania Oprogramowania (3)

Poprzez ciągłe skupienie na **technicznej doskonałości** i dobremu zaprojektowaniu oprogramowania zwiększa zwinność.

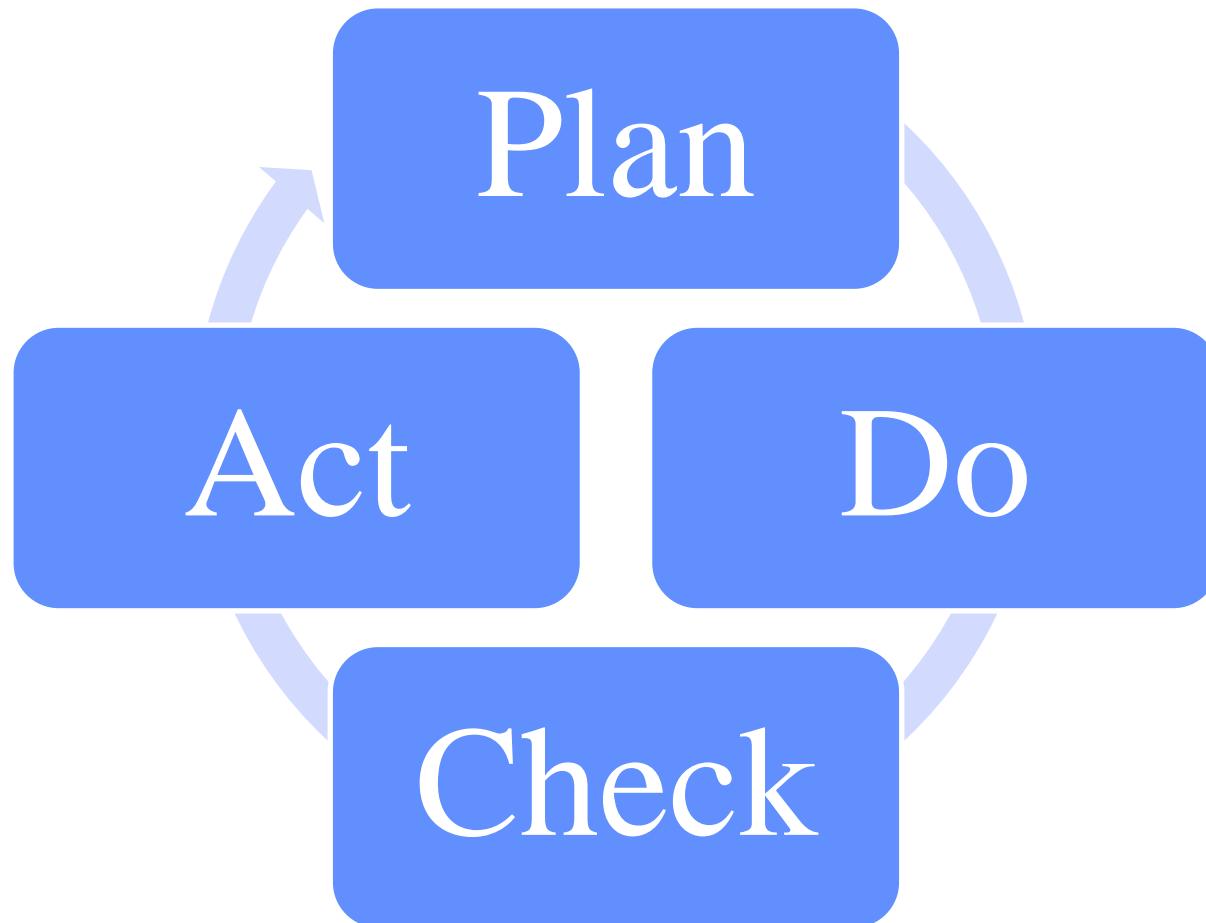
**Prostota** – sztuka maksymalizacji pracy niewykonanej – jest zasadnicza.

Najlepsze architektury, wymagania i projekty powstają w **samoorganizujących się zespołach**.

W regularnych odstępach czasu zespół zastanawia się **jak poprawić swoją efektywność**, dostosowuje lub zmienia swoje zachowanie.

Źródło: <http://agilemanifesto.org/iso/pl/principles.html>

# Proces empiryczny

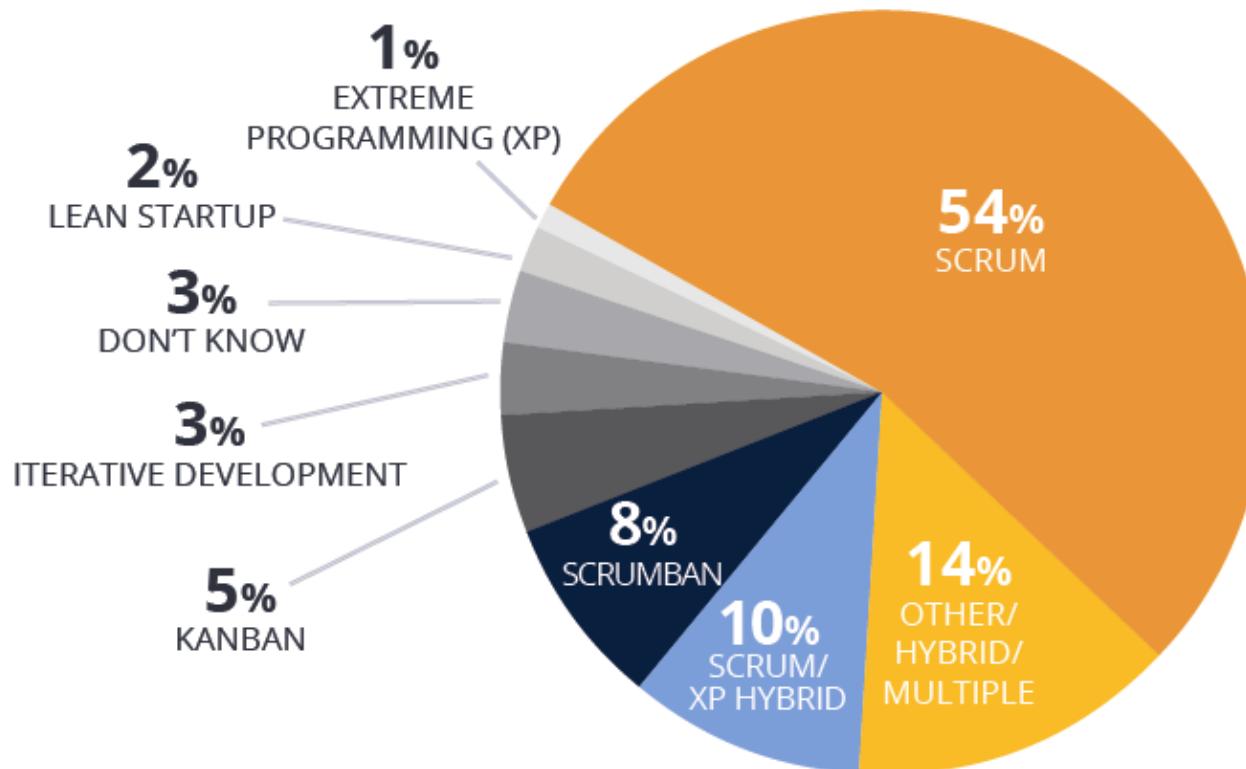


- Cykl Deminga
- Krótkie iteracje

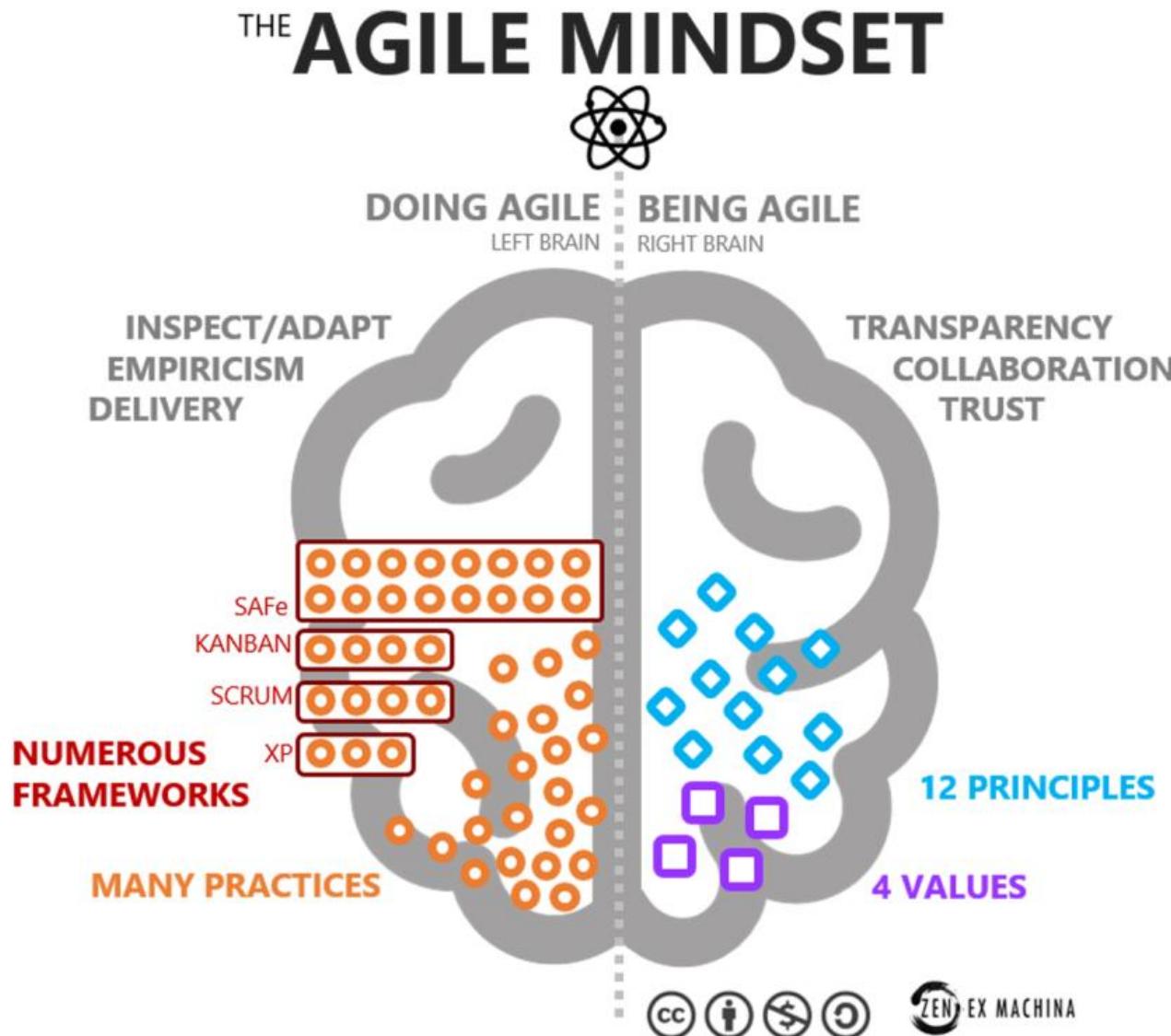
# Metody zwinne

## *Agile Methodologies Used*

Scrum and Scrum/XP Hybrid (64%) continue to be the most common agile methodologies used by respondents' organizations.



# Zwinność to sposób myślenia



# Najważniejsze elementy agile mindset

Identyfikator elementu	Element agile mindset	Średnia ocena
RZ_AM_9	Szukanie rozwiązania problemu zamiast winnych	4,44
CI_AM_3	Bycie zmotywowanym	4,44
RZ_AM_4	Wzajemna pomoc/wsparcie	4,40
RZ_AM_7	Wzajemne słuchanie się	4,37
RZ_AM_3	Nastawienie na osiągnięcie wspólnego, a nie własnego celu	4,29
CI_AM_5	Otwartość na krytykę/feedback	4,23
OP_AM_4	Współdzielenie się wiedzą i wynikami	4,21
RZ_AM_6	Wzajemny szacunek	4,11
RZ_AM_5	Szczerość	4,10
RZ_AM_1	Wzajemne zaufanie	4,10
CI_AM_1	Ciągłe doskonalenie się	4,08
OP_AM_7	Przezroczystość w podejmowaniu decyzji i działań, transparentność	4,08
OP_AM_1	Samoorganizacja	4,04
CI_AM_2	Otwartość na zmiany	4,00

# Metodyki a sposób myślenia (mindset)

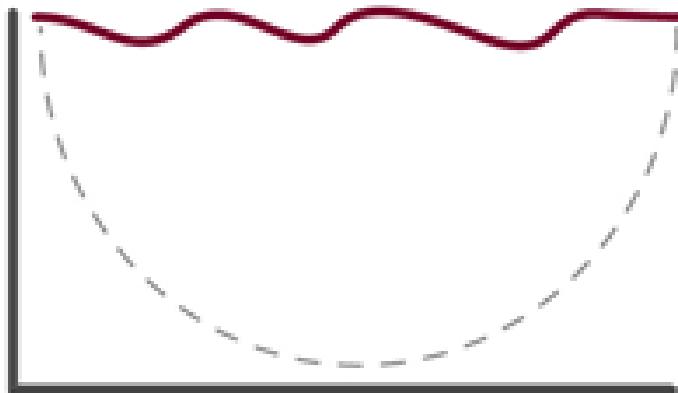
Osoby z Fixed Mindset:	Osoby z Growth (Agile) Mindset:
Unikają wyzwań	Chętnie podejmują nowe wyzwania
Inteligencję uznają jako element stały, bez możliwości udoskonalenia i rozwoju	Inteligencję uznają jako element, który można stale rozwijać i udoskonalać
Uważają, że starania i włożony wysiłek są bezwocne	Uważają, że starania i włożony wysiłek są drogą do osiągnięcia sukcesu
Pragną udowadniać swoją wiedze i inteligencję	Pragną się stale uczyć i rozwijać swoją wiedzę i inteligencję
Ignorują konstruktywną krytykę oraz opinie zwrotne	Uczą się i wyciągają wnioski z konstruktywnej krytyki oraz opinii zwrotnych
Łatwo się poddają	„Stawiają czoła” porażkom, nie poddają się
Czują się zagrożeni przez sukces innych osób	Sukces innych osób jest dla nich inspiracją
Są zamknieni na zmiany	Są otwarci na zmiany

**Dyscyplina  
Waterfall**

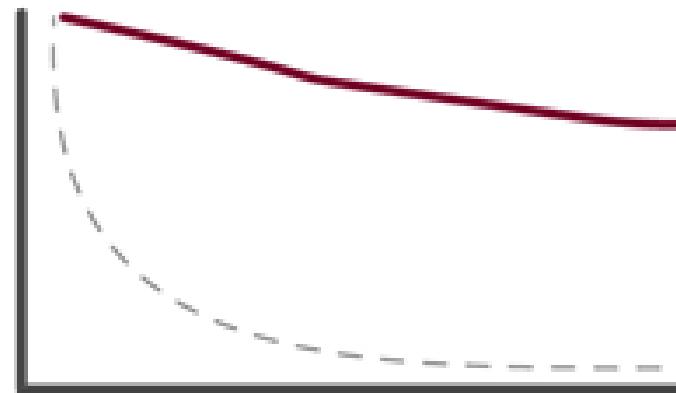
**Agile**

# Podejście zwinne a klasyczne (kaskadowe)

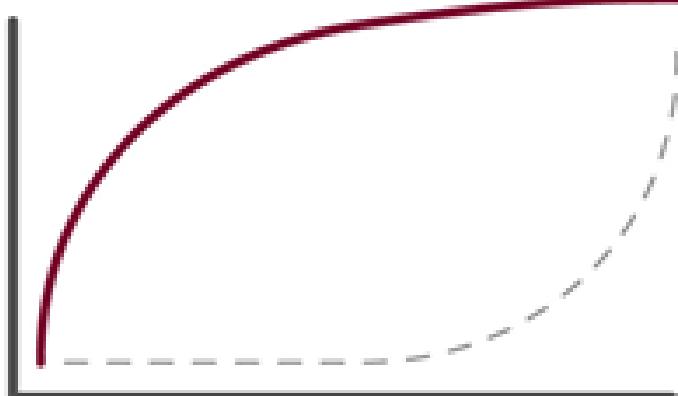
VISIBILITY



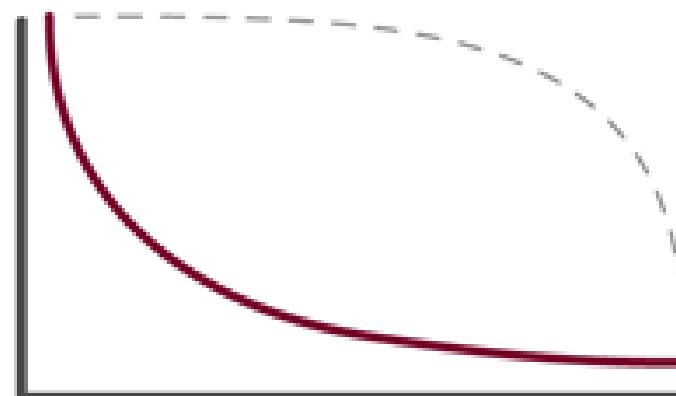
ADAPTABILITY



BUSINESS VALUE



RISK



— AGILE DEVELOPMENT

- - - TRADITIONAL DEVELOPMENT

# Klasyczne podejście do wydań

## BIZNES - ZARZĄDZANIE

Kierownik projektu, kierownik produktu, właściciel produktu

Wydanie

## IT - WYTWARZANIE

Deweloperzy, testerzy, architekci, analitycy

Przyrost

Przyrost

Przyrost

# Zwinne podejście do wydań

## BIZNES - ZARZĄDZANIE

Kierownik projektu, kierownik produktu, właściciel produktu



## IT - WYTWARZANIE

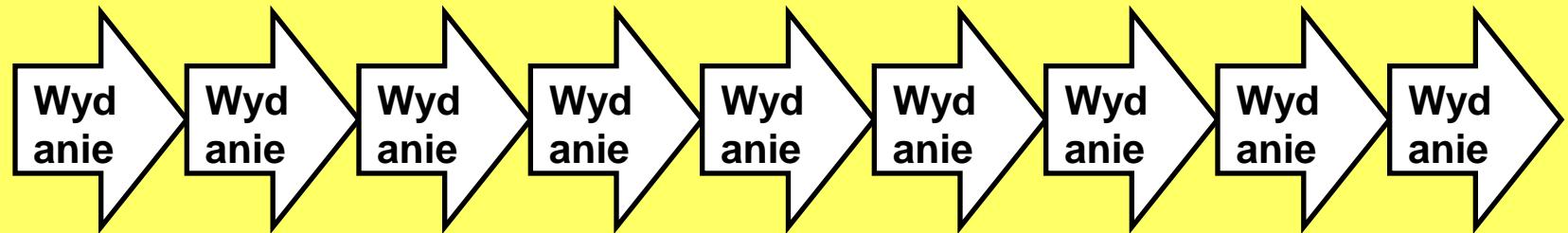
Deweloperzy, testerzy, architekci, analitycy



# Ciągłe wdrażanie (continuous deployment)

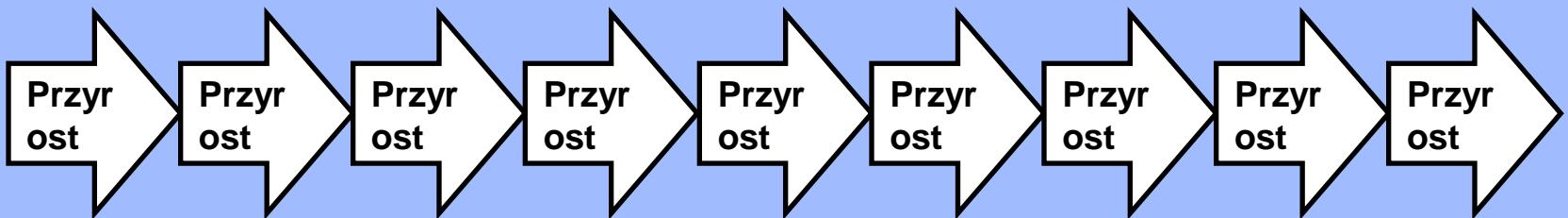
## BIZNES - ZARZĄDZANIE

Kierownik projektu, kierownik produktu, właściciel produktu



## IT - WYTWARZANIE

Deweloperzy, testerzy, architekci, analitycy



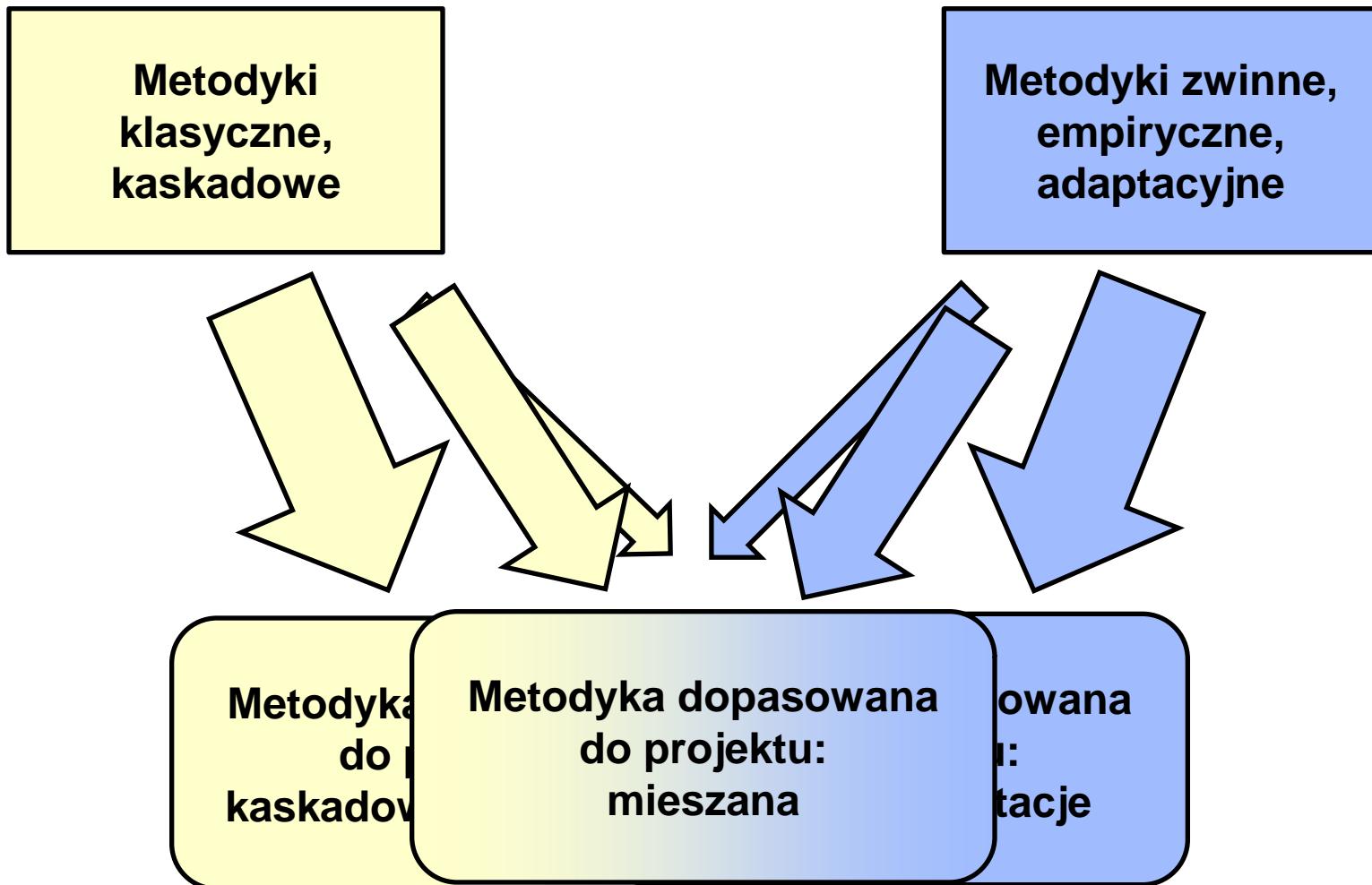
# Kluczowe cechy współczesnych procesów wytwarzania

- iteracyjność, wydania
- współpraca z interesariuszami projektów
- częsta weryfikacja pracy
- częsta aktualizacja planów/zakresu
- praca zespołowa
- frameworki/reuse
- dopasowanie metodyki/praktyk do potrzeb projektu
- ograniczenie nakładów, zwiększenie wydajności

**Nie ma metodyki idealnej – sprawdzającej się w każdym projekcie**

**Są różne pomysły na dojście do „idealnej” metodyki w danym projekcie!**

# Idea doboru metodyki



# Realizacja Projektu Informatycznego



***Jakub Miler***

*Katedra Inżynierii Oprogramowania  
Wydział Elektroniki, Telekomunikacji i Informatyki  
Politechnika Gdańskia  
jakubm@eti.pg.edu.pl*

*Materiały pomocnicze do wykładu na Wydziale ETI Politechniki Gdańskiej.  
Wykorzystanie materiałów w innym celu i ich rozpowszechnianie bez zgody WETI PG zabronione.*

## Wykład 2

# Scrum: Podstawy, wartości, filary, role



# Scrum Guide

Ken Schwaber & Jeff Sutherland

## Przewodnik po Scrumie

Przewodnik po Scrumie: opis reguł

Listopad 2020

# Scrum framework

## The Agile Scrum Framework at a glance

Inputs from  
Customers, Team,  
Managers, Execs



  
**Product Owner**

1	Prioritized list of what is required: features, bugs to fix...
2	
3	
4	
5	
6	
7	
8	

**Product Backlog**



**Developers**

Developers select starting at top as much as they can commit to deliver by end of Sprint

**Sprint Planning Meeting**

Task Breakout

**Sprint Backlog**

  
**Scrum Master**



**Burn Down/Up Chart**

24 Hour Sprint

1-4 Week Sprint

Sprint end date and team deliverable do not change



**Daily Standup Meeting**



**Sprint Review**



**Finished Work**



**Sprint Retrospective**



 AGILE FOR ALL  
Making Agile a Reality®



# Główne elementy Scrum

- Projekt prowadzony jest w Zespole Scrumowym  
(ang. *Scrum Team*)
  
- Wartości
- Filary
- Role
- Artefakty
- Zobowiązania
- Wydarzenia

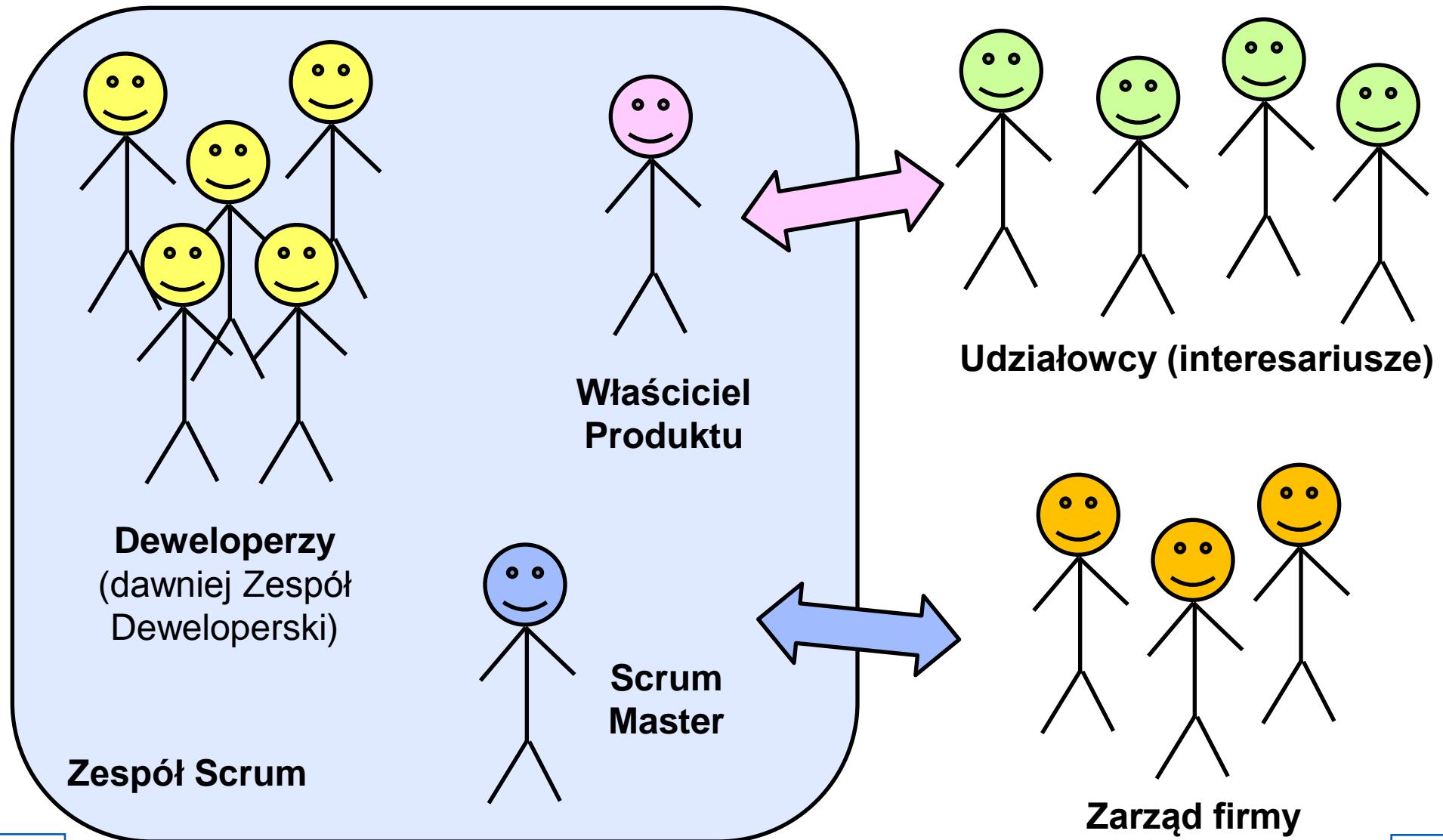
# Scrum - wartości

- **zaangażowanie** – wszyscy osobiście zobowiązują się osiągać cele Zespołu Scrumowego
- **skupienie** – wszyscy skupiają się na pracy w Sprincie i celach Zespołu Scrumowego
- **otwartość** – na wszystkie aspekty wykonywanej pracy i związane z nią wyzwania
- **szacunek** – wzajemny szacunek członków Zespołu Scrumowego
- **odwaga** – członkowie Zespołu Scrumowego mają odwagę postępować właściwie i przewyciążać trudności

# Filary Scruma - podstawowe zasady

- **Przejrzystość** – wszystkie istotne aspekty procesu są widoczne dla osób odpowiedzialnych, są tak samo rozumiane przez wszystkie zaangażowane osoby: wspólne nazewnictwo, wspólna Definicja Ukończenia
- **Inspekcja** – sprawdzanie artefaktów oraz postępów, nie powinna być zbyt częsta
- **Adaptacja** – korekta procesu lub artefaktów w wyniku stwierdzenia odbiegania ich od oczekiwania, co mogłoby doprowadzić do nieakceptownego produktu

# Uczestnicy procesu Scrum



# Role w Zespole Scrumowym

- **Scrum Master** – mentor procesu Scrum, pomaga w pracy zespołu promując zasady Scruma i wspierając jego stosowanie, usuwa przeszkody, wspiera spotkania, dopasowuje i ocenia Scrum
- **Właściciel produktu** (ang. *Product Owner*) – reprezentuje odbiorcę produktu i jego przyszłych użytkowników, współdefiniuje produkt, wypowiada wymagania, określa priorytety, zatwierdza zakres sprintu
- **Deweloperzy** (ang. *Developers*) – osoby wytwarzające produkt zgodnie z metodą Scrum, określają plan sprintu, zadania, programują i testują produkt

Rozmiar Zespołu Scrum – 10 osób lub mniej

# Scrum Master

- Ma duży wkład w sukces projektu
- Nie jest kierownikiem projektu! Jest przywódcą służebnym (ang. *servant-leader*)
- Wspiera Właściciela Produktu
  - w technikach prowadzenia Backlogu Produktu
  - w promowaniu w Zespole Scrum potrzeby jasnego i zwięzkiego formułowania elementów Backlogu Produktu
  - w rozumieniu i praktykowaniu podejścia empirycznego
  - we współpracy z interesariuszami, gdy jest taka prośba lub potrzeba

# Scrum Master (2)

## ➤ Wspiera cały Zespół Scrum

- w wykorzystaniu samoorganizacji, samozarządzania i interdyscyplinarności
- pomaga tworzyć wartościowe Przyrosty zgodnie z Definicja Ukończenia
- usuwa przeszkody, przyczyny ograniczające postępy
- dbając, aby wydarzenia odbywały się, były konstruktywne i produktywne oraz mieściły się w ramach czasowych (timebox)

## ➤ Wspiera organizację (firmę prowadzącą projekty)

- przewodzi wdrażaniu Scruma, uczy osoby – podstawowa technika uczenia to coaching
- w planowaniu wykorzystania Scruma
- pracowników i interesariuszy w rozumieniu i stosowaniu podejścia empirycznego
- usuwa bariery pomiędzy interesariuszami a Zespołami Scrum

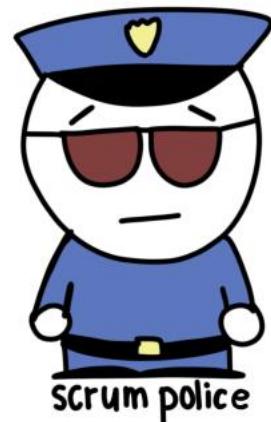
# Rola Scrum Mastera

- Rolą Scrum Mastera nie jest "pilnowanie" zespołu, ale:
  - pomaganie,
  - doszkalanie,
  - usuwanie barier,
  - ochrona zespołu,
  - zbieranie metryk (dla zarządu),
  - katalizowanie komunikacji,
  - doradzanie w organizacji pracy,
  - sugerowanie zmian w praktykach,
  - ciągła adaptacja procesu do stanu projektu uwzględniając również ryzyko

# Właściwe postawy Scrum Mastera



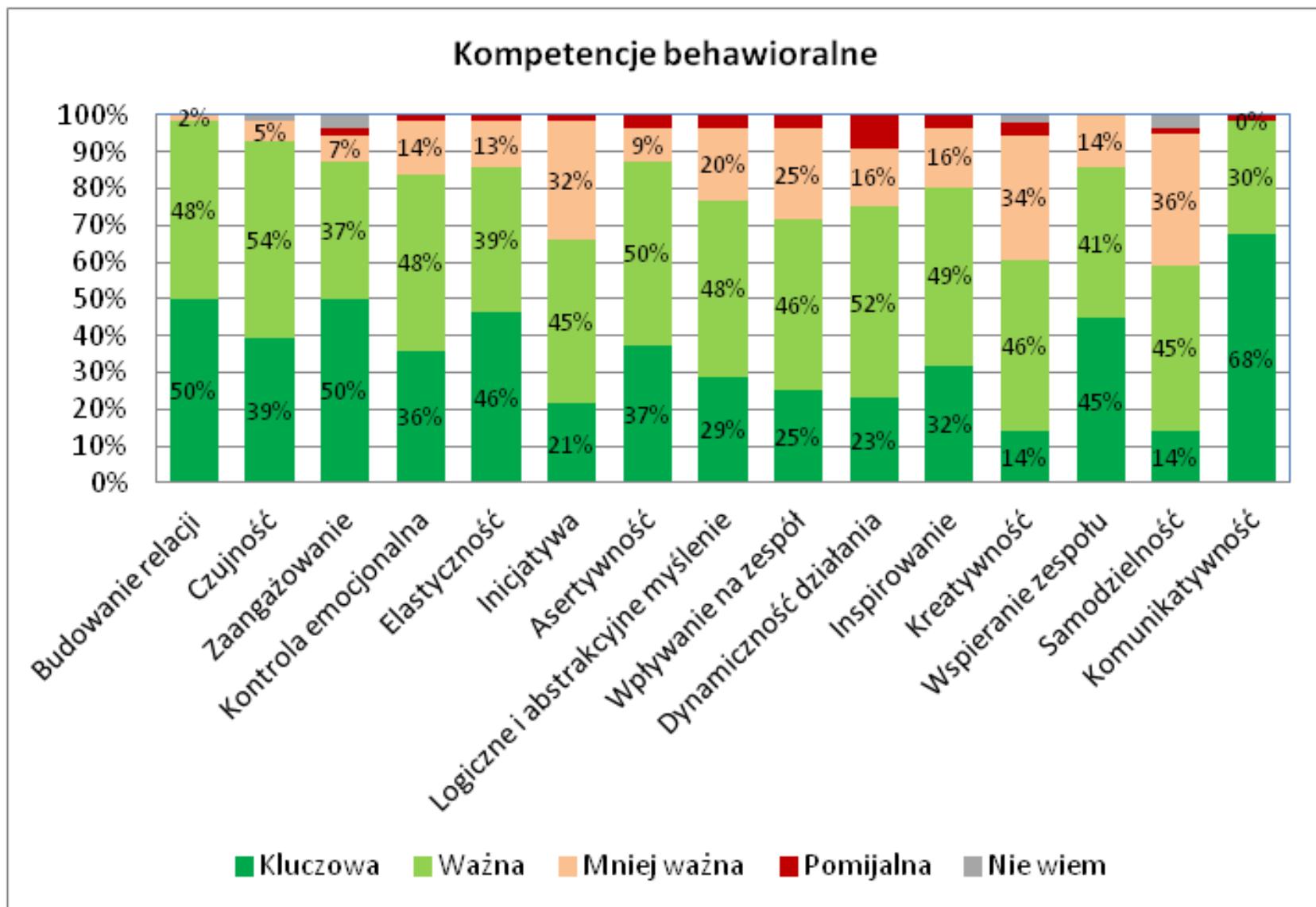
# Antypostawy Scrum Mastera



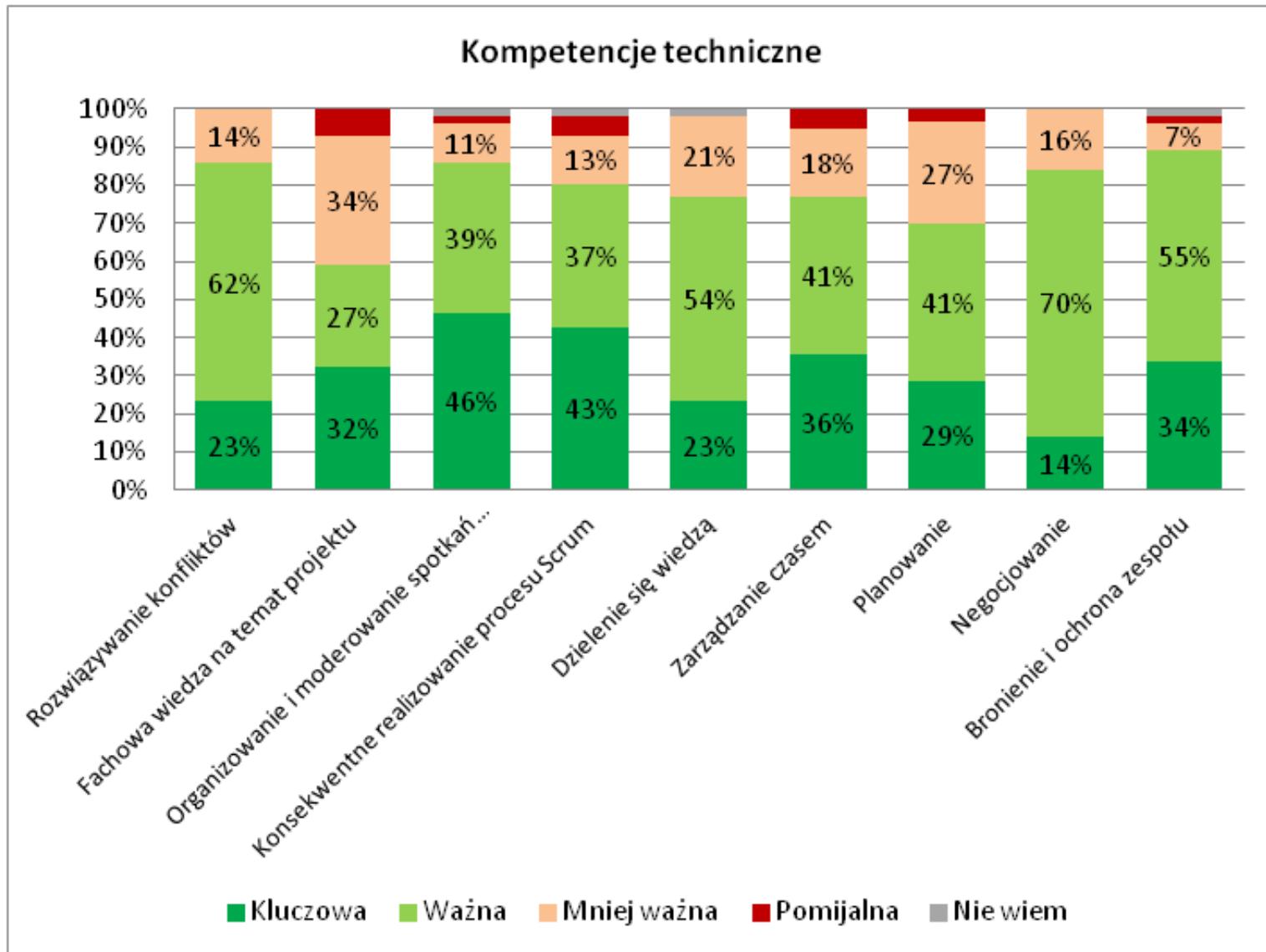
SCRUM MASTER ?



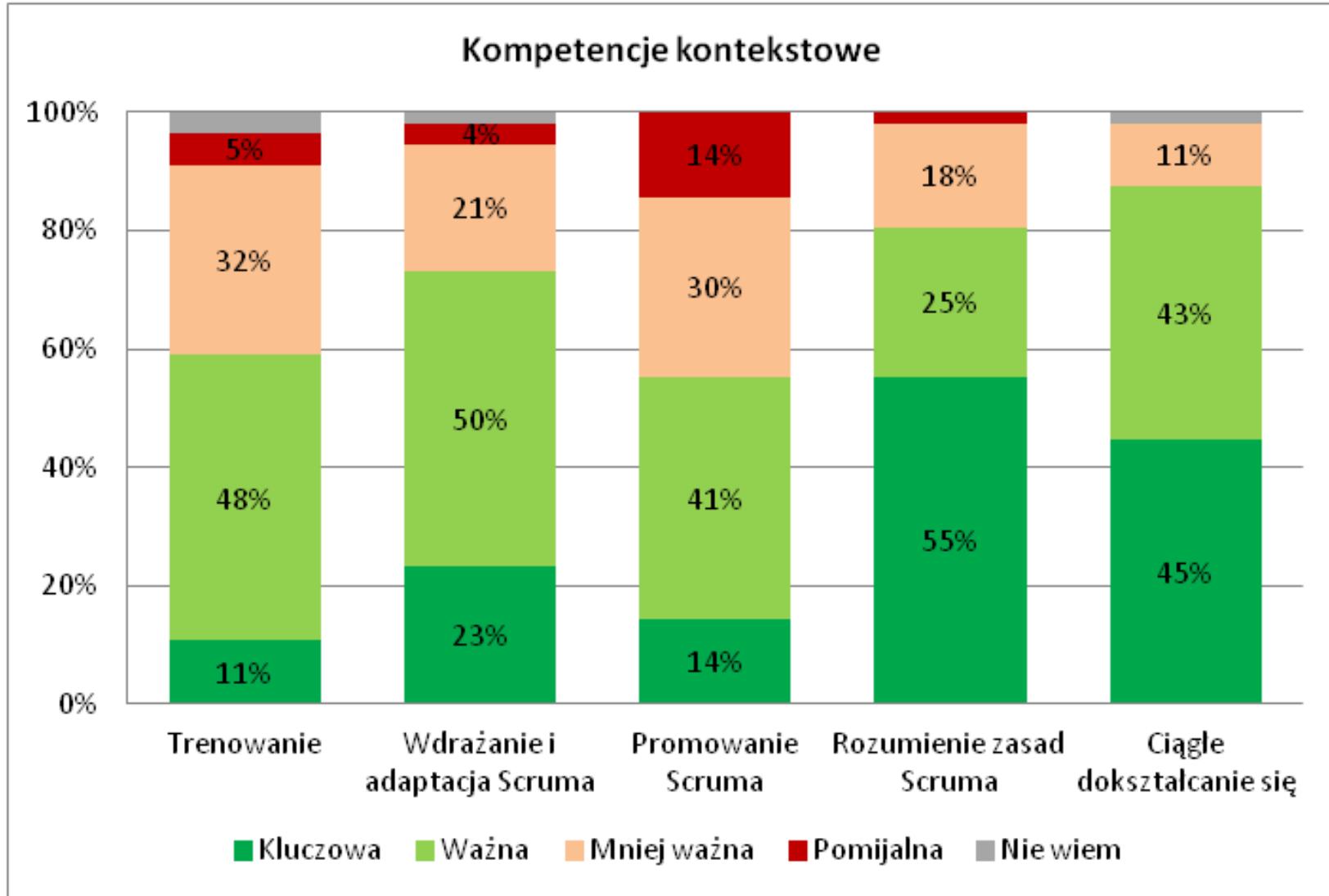
# Kompetencje Scrum Mastera (1)



# Kompetencje Scrum Mastera (2)



# Kompetencje Scrum Mastera (3)



# Certyfikat Professional Scrum Master I

## Details

- Fee: \$150 USD per attempt
- Passing score: 85%
- Time limit: 60 minutes
- Number of Questions: 80
- Format: Multiple Choice, Multiple Answer and True/False
- Difficulty: Intermediate
- Language: English only
- Scrum Resources
- Required course: None
- Recommended courses: [Professional Scrum Foundations](#) or [Professional Scrum Master](#)
- [PSM Subject Areas](#)
- Practice Assessment: [Scrum Open](#)
- Ways to [Learn More](#) to help you prepare
- Passwords have no expiration date, but are valid for one attempt only

## Certification

If you pass the PSM I assessment you will receive the industry-recognized "PSM I" certification, along with a PSM I logo that you can use to identify your achievement. In addition, your name will be posted publicly for colleagues, managers, and potential employers to see.

Unlike other Scrum certifications that require only class attendance, Scrum.org certification requires a minimum score on an online assessment. Attending a course is neither required nor sufficient for certification. This gives Scrum.org certification teeth and ensures that it has true value in the marketplace.

[Search the list of Professional Scrum Certificate Holders](#)

# Właściciel Produktu

- Odpowiada za maksymalizację wartości produktu będącego efektem pracy Zespołu Scrum
- Zarządza Backlogiem Produktu (sam odpowiada za niego)
- To pojedyncza osoba, nie komitet
- Źródłem wymagań są udziałowcy, Właściciel Produktu reprezentuje udziałowców (interesariuszy) produktu
- Tłumaczy dziedzinę biznesową na język Deweloperów
- Deweloperzy mogą kontaktować się z interesariuszami, ale wykonują tylko prace uzgodnione z Właścicielem Produktu

# Kim może być Właściciel Produktu?

- Możliwe jest różne obsadzenie tej roli przez tradycyjne role
- Wyznaczony od strony dostawcy
  - Aналитик бизнеса
  - Керівник продукту
- Wyznaczony od strony odbiorcy/klienta
  - Головний користувач
  - Керівник постачання

# Kompetencje Właściciela Produktu

**Table 12.** Product Owner's behavioral competencies

ID	Competence	E	Act.	Dep.
PO.B1	Communication skills	5	11	5
PO.B2	Creating product vision	4	8	6
PO.B3	Decisiveness	2	6	4
PO.B4	Inspiration	2	6	3
PO.B5	Leadership	2	6	3
PO.B6	Responsibility	1	6	2
PO.B7	Negotiating	1	4	3

**E** – poziom ważności kompetencji

**Act.** – liczba aktywności wspieranych przez kompetencję

**Dep.** – liczba kompetencji zależnych od danej kompetencji

**Table 13.** Product Owner's technical competencies

ID	Competence	E	Act.	Dep.
PO.T1	Product backlog management	5	14	6
PO.T2	Requirements management	4	9	5
PO.T3	Taking care of the product value	3	8	4
PO.T4	Customer relationship management	3	6	5
PO.T5	Business context	2	6	4
PO.T6	Project measurement	2	6	3
PO.T7	Planning	2	6	3
PO.T8	Product domain specialist	2	6	3
PO.T9	Project administration	1	5	2

# Certyfikat Professional Scrum Product Owner I

## Details

- Fee: \$200 USD per attempt
- Passing score: 85%
- Time limit: 60 minutes
- Number of Questions: 80
- Format: Multiple Choice, Multiple Answer and True/False
- Difficulty: Intermediate
- Language: English only
- [PSPO Subject Areas](#)
- Required course: None
- Recommended course: [Professional Scrum Product Owner](#)
- Practice assessments: [Scrum Open](#) and [Product Owner Open](#)
- Ways to [Learn More](#) to help you prepare
- Passwords have no expiration date, but are valid for one attempt only

## Certification

If you pass the PSPO I assessment you will receive the industry-recognized "PSPO I" certification, along with a PSPO I logo that you can use to identify your achievement. In addition, your name will be listed on Scrum.org.

Unlike other Scrum certifications that require only class attendance, Scrum.org certification requires a minimum score on an online assessment. This gives Scrum.org certification teeth and ensures that it has true value in the marketplace.

[Search the list of Professional Scrum Certificate Holders](#)

# Deweloperzy

- Odpowiadają za wytworzenie i dostawę produktu (w Przyrostach)
- Wzajemnie egzekwują od siebie odpowiedzialność zawodową
- Są samoorganizujący się – nawet Scrum Master nie może mówić Deweloperom, jak mają wytwarzac produkt
- Deweloperzy powinni mieć odpowiednie kompetencje techniczne w zakresie projektowania, programowania, testowania
- Scrum nie wyróżnia jawnie ról np. Analityk, Projektant, Programista, Tester – wszyscy są Deweloperami
- Zwinność wymaga dostosowania się do potrzeb i zmiany rodzaju wykonywanej pracy

# Certyfikat Professional Scrum Developer I

## Details

- Fee: \$200 USD per attempt
- Passing score: 85%
- Time limit: 60 minutes
- Number of Questions: 80
- Format: Multiple Choice, Multiple Answer and True/False
- Difficulty: Intermediate
- Language: English only
- [Scrum Developer Subject Areas](#)
- Ways to [Learn More](#) to help you prepare
- Required course: None
- Recommended course: [Professional Scrum Developer](#)
- Practice assessments: [Scrum Open](#) and [Scrum Developer Open](#)
- Passwords have no expiration date, but are valid for one attempt only

## Certification

If you pass the PSD assessment you will receive the industry-recognized "PSD" certification, along with a PSD logo that you can use to identify your achievement. In addition, your name will be listed on Scrum.org.

Unlike other Scrum certifications that require only class attendance, Scrum.org's certification requires a minimum score on a rigorous online assessment. This gives Scrum.org certification teeth and ensures that it has true value in the marketplace.

[Search the list of Professional Scrum Certificate Holders](#)

# Zagadnienia na certyfikat PSD I

- The Scrum Framework
- Scrum Theory and Principles
- Cross-functional, Self-organizing Development
- Analysis
- Emergent Architecture
- Programming
- Test First Development
- Standards
- Testing
- ALM - Application Lifecycle Management

# Agile mindset a zasady Agile i kompetencje ról

Table 16. Comparison of agile mindset, agile principles and competence models

No.	ID	Agile Mindset Element	P#	SM	PO	AA
1	T9	Searching for a solution to the problem instead of finding the guilty				A.B2
2	I3	Being motivated	5	SM.B4		
3	T4	Helping each other		SM.B7		
4	T7	Mutual listening				A.B6
5	T3	Focus on achieving common goal			PO.B2	A.B7
6	I5	Openness to criticism and feedback		SM.B8		A.T1
7	O4	Sharing knowledge and results		SM.T6		A.T2
8	T6	Mutual respect				
9	T1	Mutual trust	5			A.B11
10	T5	Sincerity				A.B13
11	I1	Continuous improvement and learning	12	SM.C1		
12	O7	Transparency in decision-making and actions			PO.B3	A.B14
13	O1	Self-organization	11	SM.B15		
14	I2	Openness to change	2	SM.B6		A.B10
15	G1	Continuous delivery of a valuable product in short intervals	1, 3, 7		PO.T3	
16	G3	Attitude towards customer satisfaction and needs	1		PO.T3	A.B15

P# – zasada Agile

SM – kompetencja Scrum Mastera

PO – kompetencja Właściciela Produktu

AA – kompetencja zwanego analityka

# Agile mindset a zasady Agile i kompetencje ról (2)

17	G2	Cooperation with the customer based on partnership	4	SM.B2	PO.T4	A.B5
18	I6	Openness to others				A.B4
19	O6	Finishing the current task before taking the next one				
20	I4	Positive attitude				A.B20
21	O3	Ability to collaborate		SM.B12		A.B12
22	O5	Asking questions in case of insufficient knowledge				A.B1
23	T2	Direct communication - face to face conversations	6	SM.B1	PO.B1	A.T11
24	T8	Equality in the team				
25	T10	Team responsibility			PO.B6	A.B18
26	O2	Maintaining a steady pace of work	8	SM.T7	PO.T7	A.B16

**P#** – zasada Agile

**SM** – kompetencja Scrum Mastera

**PO** – kompetencja Właściciela Produktu

**AA** – kompetencja zwanego analityka

# Realizacja Projektu Informatycznego



***Jakub Miler***

*Katedra Inżynierii Oprogramowania  
Wydział Elektroniki, Telekomunikacji i Informatyki  
Politechnika Gdańskia  
jakubm@eti.pg.edu.pl*

*Materiały pomocnicze do wykładu na Wydziale ETI Politechniki Gdańskiej.  
Wykorzystanie materiałów w innym celu i ich rozpowszechnianie bez zgody WETI PG zabronione.*

## Wykład 3

# Scrum: Backlog produktu, planowanie sprintu, backlog sprintu



# Persony – archetypy użytkowników

- Hipotetyczne archetypy rzeczywistych użytkowników
- Powinny być wystarczająco precyzyjne, aby być wiarygodne – warto mieć jakieś źródłowe dane
- Pomagają wytworzyć empatię do użytkowników i lepiej ich zrozumieć
- Zakres treści w opisie persony
  - imię i nazwisko
  - wizerunek, zdjęcie
  - stan cywilny, rodzinny
  - wykształcenie
  - zawód
  - co lubi robić, jak pracować, relaksować się, spędzać czas
  - zaawansowanie w korzystaniu z IT
  - problemy
  - potrzeby



Przykład

# Scenariusze biznesowe (ang. *Business scenario*)

- Opowieści w języku naturalnym o wykorzystaniu powstającego systemu w docelowym kontekście przez użytkownika w celu rozwiązania jego problemów
- Opisuje pracę z systemem z punktu widzenia użytkownika
- Obejmuje wykorzystanie wielu funkcji systemu w logicznym ciągu dającym realną wartość użytkownikowi – rozwiązującą jego problem
- Dla czytelności w opisie scenariusza można wyróżnić:
  - role użytkowników
  - dane wejściowe i wyjściowe
  - kroki scenariusza
  - wykorzystywane usługi systemu
- Powstają poprzez wywiady z użytkownikami lub techniką „burzy mózgów”, gdy użytkownik nie jest dostępny
- Powinny być zweryfikowane i zatwierdzone przez użytkownika
- Dla danego produktu może być wiele scenariuszy użycia
- Scenariusze stanowią podstawę identyfikacji cech produktu

# Dobry scenariusz biznesowy

## ➤ Produkt: „Przypominacz” - terminarz

Tęgomir dostał od swojego kolegi Kwasibrzucha nowy program - Przypominacz. Zainstalował go na swoim komputerze i uruchomił. Szybko stworzył nowe konto użytkownika - podał login, hasło, swój adres email oraz numer i operatora telefonu komórkowego z którego korzysta.

Jako że Kwasibrzuch ma z Tęgomirem sporo wspólnych znajomych, wyeksportował mu większość kontaktów, co znacznie ułatwiło Tęgomirowi pracę związaną z uzupełnianiem danych w programie.

Tęgomir wraca do domu z uczelni. Włącza komputer żeby dodać termin oddawania projektu z RPI (i oczywiście przypomnienie na dzień wcześniej, że powinien się jednak zabrać do roboty) i dowiaduje się, że jego dobra znajoma ma jutro urodziny. Niestety, w komunikacie programu jest również informacja o jutrzejszym kolokwium z matematyki. Wybiega więc szybko na miasto w poszukiwaniu prezentu, ale jako że nie ma pomysłu, traci na to dużo czasu. Do nauki siada dopiero późnym wieczorem. Nagle dostaje smsa - okazuje się, że jest północ i nadszedł dzień urodzin znajomej. Czym przedzej dzwoni do niej z życzeniami. Ustawił sobie taką godzinę przypominania, bo uważa, że to zabawne zaskakiwać tak solenizantów. Znajoma jest zachwycona.

Nazajutrz, po zajęciach, siada do komputera. „Przypominacz” informuje o umówionej wizycie u dentysty na 16.00 oraz oczywiście o imprezie urodzinowej u znajomej na 20.00. Nigdy jeszcze u niej nie był, ale na szczęście ma w programie zapisany dokładny adres. Sprawdza jeszcze tylko czy nie ma jakichś terminów związanych z uczelnią w przeciągu dwóch kolejnych dni. Na szczęście nie. Może więc spokojnie poddać się urokowi studiowania. Chwilę zapomnienia przerywa jednak brutalnie sygnał z komórki - już najwyższy czas wychodzić do dentysty. Wieczorem Tęgomir bardzo dobrze bawi się u znajomej. Jego śnieżnobiały uśmiech dostrzega jedna z koleżanek solenizantki.

Następnego dnia, po południowej pobudce, Tęgomir z radością dodaje w „Przypominaczu” nowy kontakt oraz termin spotkania. Wie, że nie może się na nie spóźnić.

W końcu ma czas na odpoczynek. Ma pewność, że nie ciąży na nim żaden zapomniany termin.

# Zły scenariusz biznesowy

## „Logowanie do systemu”

Dlaczego?

- Użytkownik (najczęściej) nie ma potrzeby po prostu logować się od czasu do czasu do systemu
- W „scenariuszu” „Logowanie do systemu” nie jest realizowana potrzeba użytkownika ze świata rzeczywistego, autoryzacja użytkownika jest jedynie pierwszym krokiem scenariusza

**Scenariusz to nie przypadek użycia!**

# Zły scenariusz biznesowy (2)

## ➤ Zbyt dużo informacji kontekstowej – za mało informacji o systemie

Padało. Dzień był szary i nieprzyjemny. Marek jak zwykle stał w porannym korku. Wszyscy inni mieli zielone światło, ale nie on. Radio sączyło kolejną porcję wiadomości. Już miał wrażenie, że nigdy nie dotrze do biura.

[...]

W końcu znalazł się w swoim pokoju. Włączył komputer i uruchomił nowy system TimeMan PRO. Przejrzał dostępne informacje, wyłączył go i zajął się odpisywaniem na mejle.

## ➤ Przesadny humor – rozprasza, utrudnia zrozumienie cech systemu

Garbaty kopnął Leniwego w tę część ciała, gdzie zaczynają się nogi i krzyknął:

- Ej, frajerce, jeszcze nie uruchomłeś tego nowego rewelacyjnego wszystkomogącego narzędzia AKuKuTraLaLa MAXI PRO?

- Odczep się, pajacu – wypalił Leniwy – właśnie gram w Wiedźmina 10.3. A co ten ABuBuNaNaNa TAXI FRO potrafi?

- A no ma kilka takich bajerów, na przykład można się zalogować.

[...]

**Tu scenariusz służy specyfikacji systemu, a nie kręceniu filmu!**

# Jak pisać scenariusze biznesowe?

- Opisuje użytkowanie produktu od potrzeby w świecie realnym aż do zaspokojenia tej potrzeby
- Scenariusz obejmuje (najczęściej) wykorzystywanie „po drodze” wielu różnych funkcji produktu
- Dla prostszych produktów będzie tylko jeden scenariusz
- Lepiej mieć mniej a większych scenariuszy niż wiele drobnych
- Scenariusze można scalać - jeżeli funkcjonalność wykorzystywana w jakimś scenariuszu w pełni mieści się w innych scenariuszach, to taki scenariusz jest nadmiarowy i można go usunąć
- Scenariusz pisze się z perspektywy użytkownika – co użytkownik widzi, jakie polecenie wykonuje, znowu co widzi itd. Przydaje się trochę kontekstu (życia).
- Narracja scenariusza w pierwszej („ja”) lub w trzeciej osobie („on”). Dla realizmu scenariusza warto nadać bohaterom imiona.

# Artefakt: Backlog produktu

## ➤ Backlog Produktu (Rejestr Produktu) (ang. *Product Backlog*)

- „ewoluująca, uporządkowana lista tego, co jest konieczne do ulepszenia produktu” [Scrum Guide]
- jedynie źródło pracy podejmowanej przez Zespół Scrum
- za jego zawartość, dostępność i uporządkowanie odpowiada Właściciel Produktu

## ➤ Zobowiązanie: Cel Produktu

- „opisuje przyszły stan produktu, który może posłużyć Scrum Teamowi jako punkt odniesienia w procesie planowania” [Scrum Guide]
- jest ujęty w backlogu produktu
- „*Produkt to sposób na dostarczenie wartości. Ma jasno określone granice, znanych interesariuszy, dobrze zdefiniowanych użytkowników lub klientów. Produkt może być usługą, fizycznym produktem bądź czymś bardziej abstrakcyjnym.*”

# Backlog produktu (projekt A) – ok. 100 elementów

	No.	Description	Owner All	Status All	Pri. P1	Est. 5	Rem. 5
>	14	Cel gry - ukończenie projektu zgodnie z celem projektu, z jak najwyższym wskaźnikiem sukcesu - punktami (wiki)	-	Not Started	P1	5	5
	15	Cel projektu - docelowe wartości poszczególnych atrybutów projektu: czas, koszt, zakres, jakość, zadowolenie klienta, zadowolenie zespołu dostawcy (wiki)	-	Not Started	P0	5	5
	16	Atrybuty projektu	-	Not Started	P0	23	23
	17	Zakres (łączna wartość biznesowa wykonanych elementów produktu)	-	Not Started	P0	2	2
	18	Jakość (wiki)	-	Not Started	P0	2	2
	19	Czas (wiki)	-	Not Started	P1	2	2
	113	Rzyzyko - ogólny poziom ryzyka projektu (wiki)	-	Not Started	P1	5	5
	20	Koszt	-	Not Started	P2	2	2
	21	Zadowolenie klienta - wyliczane z czasu, kosztu, zakresu i jakości (wiki, 2 comments)	-	Not Started	P0	5	5
	22	Zadowolenie zespołu dostawcy - wyliczane z czasu, kosztu, zakresu, jakości (wiki)	-	Not Started	P1	5	5
	23	Bieżący wskaźnik sukcesu łączący bieżące wartości poszczególnych atrybutów projektu (wiki)	-	Not Started	P0	8	8
	24	Prezentacja docelowych i bieżących atrybutów projektu (składowych celu i wskaźnika sukcesu) (wiki)	-	Not Started	P0	8	8
	25	Różne kryteria końca gry (wiki)	-	Not Started	P1	17	17
	26	Realizacja pełnego zakresu projektu (wiki)	-	Not Started	P1	3	3
	114	Realizacja jak najlepiej projektu w ograniczonym czasie (wiki)	-	Not Started	P1	3	3
	27	Osiągnięcie określonej wartości biznesowej produktu	-	Not Started	P2	3	3
	28	Osiągnięcie maksymalnego zadowolenia klienta	-	Not Started	P1	5	5
	29	Wybór kryterium końca projektu przy startie gry (wiki)	-	Not Started	P1	3	3
	30	Zakres i jakość produktu - lista elementów/cech produktu do wytworzenia	-	Not Started	P0	62	62
	31	Atrybuty elementów produktu	-	Not Started	P0	36	36
	32	Nazwa elementu zależna od tematu gry	-	Not Started	P0	8	8
	33	Wartość biznesowa - przyrost zakresu	-	Not Started	P0	5	5
	34	Jakość - przyrost lub utrata jakości	-	Not Started	P1	5	5
	35	Rozmiar elementu - nakład pracy zasobów	-	Not Started	P0	13	13
	36	Specjalizacja zasobu - wymagane specjalne zasoby	-	Not Started	P2	5	5
	37	Elementy konieczne do wykonania wcześniej	-	Not Started	P2	-	-
	38	Zmieniający się zakres produktu w poszczególnych turach gry	-	Not Started	P1	13	13

Źródło: Rafał Piechowski, Damian Płatek, Anna Wasik, Sylwia Grabowska, *Gra edukacyjna ucząca zarządzania projektami*, projekt dyplomowy inżynierski, opiekun J. Miler, WETI, PG, 2014

# Backlog produktu (projekt B) – 25 elementów

Features \* Backlog items

Backlog Board

New | Create query | Column options |

Order	ID	Work Item Type	Title	State	Effort	Business Value	Iteration Path
1	1	Product Backlog...	Rejestr ryzyk z wieloma polami	Committed	13	3	TeamRiskAid\Sprint 1
2	2	Product Backlog...	Wpiswanie nowych pozycji w rejestrze ryzyk	Committed	2	3	TeamRiskAid\Sprint 1
3	3	Product Backlog...	Edycja wpisów w rejestrze ryzyk	Committed	1	3	TeamRiskAid\Sprint 1
4	4	Product Backlog...	Przeglądanie historycznego stanu rejestru ryzyk	New	13	2	TeamRiskAid
5	5	Product Backlog...	Nazywanie dat	New	2	1	TeamRiskAid
6	6	Product Backlog...	Sortowanie rejestru po różnych kolumnach	Committed	5	3	TeamRiskAid\Sprint 1
7	14	Product Backlog...	Komentowanie wpisów w rejestrze	New	5	2	TeamRiskAid
8	7	Product Backlog...	Czytelna prezentacja rejestru	Komentowanie wpisów w rejestrze			TeamRiskAid
9	8	Product Backlog...	Usuwanie wpisów z rejestru	New	1	2	TeamRiskAid
10	9	Product Backlog...	Zakładanie kont przez admina	New	3	3	TeamRiskAid
11	10	Product Backlog...	Samodzielne zakładanie kont - rejestracja	Committed	3	2	TeamRiskAid\Sprint 1
12	11	Product Backlog...	Zakładanie projektów	Committed	3	3	TeamRiskAid\Sprint 1
13	12	Product Backlog...	Przypisywanie użytkowników do projektów	New	3	3	TeamRiskAid
14	13	Product Backlog...	Konfiguracja ustawień projektu: skale, kategorie, próg akceptacji	New	5	2	TeamRiskAid
15	15	Product Backlog...	Konfigurowanie uprawnień uczestników projektu	New	5	3	TeamRiskAid
16	21	Product Backlog...	Kombinacje uprawnień jako role	New	1	1	TeamRiskAid
17	16	Product Backlog...	Rysowanie profilu ryzyka	New	2	2	TeamRiskAid
18	17	Product Backlog...	Generowanie edytowalnego i drukowalnego raportu z wybranych ryzyk	New	3	3	TeamRiskAid
19	18	Product Backlog...	Zespołowe szacowanie ryzyka - głosowanie	New	1	1	TeamRiskAid
20	19	Product Backlog...	Filtrowanie rejestru	Committed	3	2	TeamRiskAid\Sprint 1
21	20	Product Backlog...	Dedykowane widoki rejestru	New	2	2	TeamRiskAid
22	22	Product Backlog...	Logowanie i wylogowywanie się, utrzymywanie sesji	Committed	1	3	TeamRiskAid\Sprint 1
23	23	Product Backlog...	Otwieranie projektu domyślnego po zalogowaniu	New	2	2	TeamRiskAid
24	24	Product Backlog...	Utrzymanie wyboru projektu w czasie sesji	New	1	3	TeamRiskAid
25	25	Product Backlog...	Zmiana aktywnego projektu	New	2	3	TeamRiskAid

Źródło: Krzysztof Kułkowski, Maciej Maroszczyk, Marcin Szczypka, Tomasz Truszkowski, *TeamRiskAid – System komunikacji na temat ryzyka i problemów w zespole projektowym*, projekt dyplomowy inżynierski, opiekun J. Miler, WETI, PG, 2013

# Backlog Produktu z epikami (projekt C)

Jira Software   Twoja praca   Projekty **Utwórz**   Wyszukaj   JM   KC   JM

Projekty / Zręczna zręcznościówka / Tablica ZZ

## Backlog

Udostępnij

WERSJE	EPIKI	Utwórz epik
	Wszystkie zgłoszenia	
>	Optymalizacja	
>	Testowanie	
>	Środowisko	
>	GUI produktu/ustawienia	
>	GUI w trakcie gry	
>	Rozwój postaci	
>	Fabuła	
>	Grafika	
>	Audio	
>	Plansze	

**Backlog** 19 zgłosz.

**Utwórz sprint**

Lektor	Audio	ZZ-91
Easter eggs	Fabuła	ZZ-100
Punktacja gracza	Fabuła	ZZ-79
Animacja paska ładowania	Grafika	ZZ-261
Animacja postaci	Grafika	ZZ-63
Model postaci	Grafika	ZZ-62
Steam	GUI produktu/ustawienia	ZZ-210
Ranking	GUI produktu/ustawienia	ZZ-87
Profil gracza	GUI produktu/ustawienia	Quickstart

**Źródło:** P. Kriger, K. Chmielewski, K. Ćwikliński, Gra zręczna zręcznościowa 3D zrealizowana z użyciem zaadaptowanej metody Scrum, Projekt dyplomowy inżynierski, Katedra Inżynierii Oprogramowania, Politechnika Gdańskia, 2020

# Zakres produktu – elementy PB

- Epiki (ang. *epics*) – większe funkcje, moduły, zbiory cech/historyjek
- Cechy (ang. *features*) – funkcjonalne i pozafunkcjonalne własności produktu oczekiwane przez użytkownika
  - przykład: „przeglądanie złożonych zamówień”
- Historyjki (ang. *user stories*, koncepcja z eXtreme Programming) - wypowiadane z punktu widzenia użytkownika umotywowane oczekiwania względem produktu
  - format dla historyjki (M. Cohn): Jako (rola) chcę (czego) by osiągnąć (co)
  - przykład: „Jako student chcę zakupić kartę wjazdu na parking by móc przyjeżdżać samochodem na uczelnię”
  - **UWAGA: Nie mylić historyjki ze scenariuszem! Scenariusz obejmuje wiele historyjek.**
- Ulepszenia (ang. *improvement*)
  - przykład: „dodać obsługę wyjątków do klasy Login”

Na początku projektu identyfikuje się główne (oczywiste) elementy produktu, gdyż i tak będzie ich sporo więcej niż na pierwszy sprint

# Priorytety

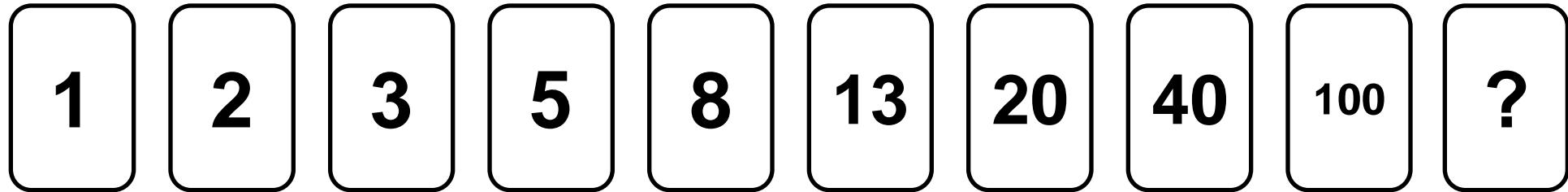
- Określają ważność (a nawet wartość biznesową) elementu z punktu widzenia interesariuszy (klienta/użytkownika i jego biznesu)
- Praktyczne poziomy priorytetów elementów:
  - kluczowe – element jest niezbędny, bez niej produkt nie ma wartości biznesowej
  - przydatne – element zwiększa wartość biznesową, lecz nie jest niezbędny
  - dodatkowe – element nieznacznie zwiększa wartość biznesową, lecz nie musi być realizowany natychmiast, a stanowi raczej propozycję dla dalszych wydań
- Technika MoSCoW:
  - kluczowe – must have
  - przydatne – should have
  - dodatkowe – could have
  - niepotrzebne – won't have
- Stosowane w projekcie poziomy priorytetów powinny być uzgodnione w zespole
- Priorytety w backlogu produktu wpisuje Właściciel produktu

# Rozmiar

- Szacowany w Story Points (SP) – SP to arbitralna miara rozmiaru (złożoności/trudności, a nie czasochłonności) dla elementu PB
- Najczęściej przyjmuje się oszacowanie w skali M. Cohna:  
**1, 2, 3, 5, 8, 13, 20, 40, 100 punktów**
- W praktyce wybiera się najłatwiejszy element i przydziela mu 2 punkty
- Kolejne elementy szacuje się względem tego pierwszego
- Story point jest jednostką względną i będzie miał inny „rozmiar” dla różnych produktów – czym jest 1 story point jest kwestią umowy w każdym zespole
- Story points nie odnoszą się bezpośrednio do godzin pracy – w różnych zespołach będzie różne tempo wytwarzania części produktu odpowiadającej 1 punktowi

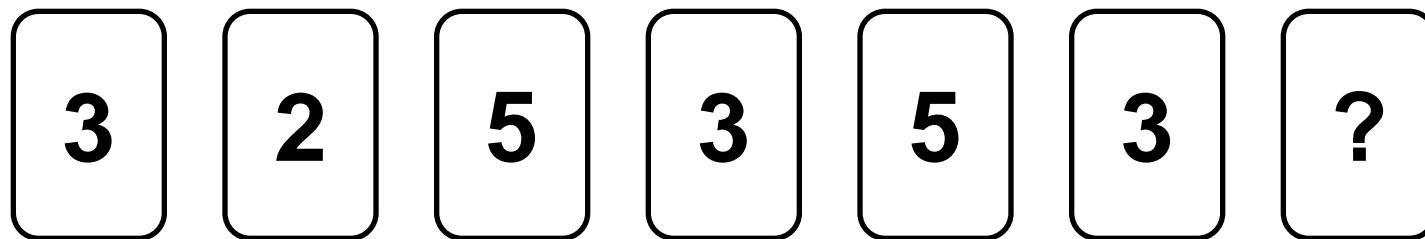
# Planning Poker (technika z XP)

- Każdy Deweloper ma do dyspozycji karty z liczbami ze skali story points np. 1, 2, 3, 5, 8, 13, 20, 40, 100 + karta „?”
- Przy szacowaniu danego elementu PB wszyscy naraz wykładają kartę ze swoim oszacowaniem
- Wszyscy przyglądają się, jakie karty (oszacowania) padły. Jeżeli jest zgoda, to jest przyjmowane takie oszacowanie. Jeżeli nie ma zgody to ponownie wykładane są naraz karty (można zmienić swoje oszacowanie).

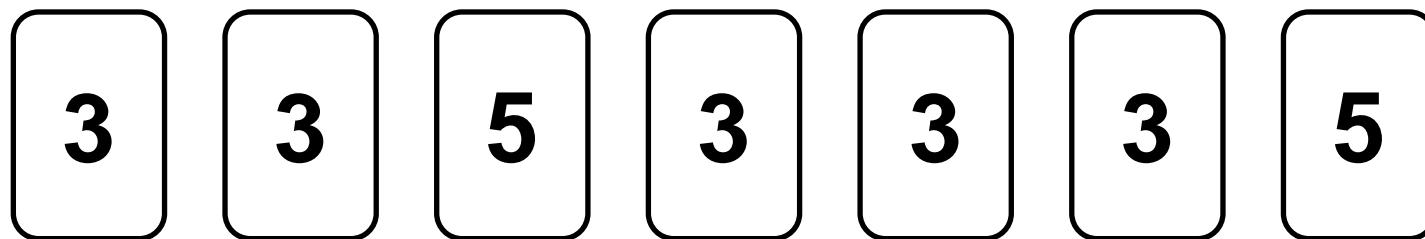


# Planning Poker - przykład

➤ Tura 1



➤ Tura 2



➤ Tura 3 albo uzgodnione oszacowanie: 3

# Backlog Produktu - Product backlog

- Zawiera elementy (ang. *items*), a nie zadania (*tasks*)
- Element może reprezentować:
  - wymaganie (cechę)
  - historyjkę
  - ulepszenie
  - błąd do usunięcia
- Elementy posortowane po priorytetach
- Każdy element ma identyfikator, priorytet, oszacowanie (w story points)

	No.	Description	Owner	Status	Pri.	Est.	Rem.
➤	14	Cel gry - ukończenie projektu zgodnie z celem projektu, z jak najwyższym wskaźnikiem sukcesu - punktami (wiki)	-	Not Started	P1	5	5
➤	15	Cel projektu - docelowe wartości poszczególnych atrybutów projektu: czas, koszt, zakres, jakość, zadowolenie klienta, zadowolenie zespołu dostawcy (wiki)	-	Not Started	P0	5	5
➤	16	Atrybuty projektu	-	Not Started	P0	23	23
➤	17	Zakres (łączna wartość biznesowa wykonanych elementów produktu)	-	Not Started	P0	2	2
➤	18	Jakość (wiki)	-	Not Started	P0	2	2
➤	19	Czas (wiki)	-	Not Started	P1	2	2
➤	113	Rzyko - ogólny poziom ryzyka projektu (wiki)	-	Not Started	P1	5	5
➤	20	Koszt	-	Not Started	P2	2	2
➤	21	Zadowolenie klienta - wyliczane z czasu, kosztu, zakresu i jakości (wiki, 2 comments)	-	Not Started	P0	5	5
➤	22	Zadowolenie zespołu dostawcy - wyliczane z czasu, kosztu, zakresu, jakości (wiki)	-	Not Started	P1	5	5
➤	23	Bieżący wskaźnik sukcesu łączący bieżące wartości poszczególnych atrybutów projektu (wiki)	-	Not Started	P0	8	8
➤	24	Prezentacja docelowych i bieżących atrybutów projektu (składowych celu i wskaźnika sukcesu) (wiki)	-	Not Started	P0	8	8
➤	25	Różne kryteria końca gry (wiki)	-	Not Started	P1	17	17
➤	26	Realizacja pełnego zakresu projektu (wiki)	-	Not Started	P1	3	3
➤	114	Realizacja jak najlepiej projektu w ograniczonym czasie (wiki)	-	Not Started	P1	3	3
➤	27	Osiągnięcie określonej wartości biznesowej produktu	-	Not Started	P2	3	3
➤	28	Osiągnięcie maksymalnego zadowolenia klienta	-	Not Started	P1	5	5
➤	29	Wybór kryterium końca projektu przy starcie gry (wiki)	-	Not Started	P1	3	3
➤	30	Zakres i jakość produktu - lista elementów/cech produktu do wytworzenia	-	Not Started	P0	62	62
➤	31	Atrybuty elementów produktu	-	Not Started	P0	36	36
➤	32	Nazwa elementu zależna od tematu gry	-	Not Started	P0	8	8
➤	33	Wartość biznesowa - przyrost zakresu	-	Not Started	P0	5	5
➤	34	Jakość - przyrost lub utrata jakości	-	Not Started	P1	5	5
➤	35	Rozmiar elementu - nakład pracy zasobów	-	Not Started	P0	13	13
➤	36	Specjalizacja zasobu - wymagane specjalne zasoby	-	Not Started	P2	5	5
➤	37	Elementy konieczne do wykonania wcześniej	-	Not Started	P2	-	-
➤	38	Zmieniający się zakres produktu w poszczególnych turach gry	-	Not Started	P1	13	13

Punkt widzenia Właściciela produktu i interesariuszy!

# Doskonalenie Backlogu Produktu (ang. refinement)

- Regularny przegląd zawartości Backlogu Produktu przez Właściciela Produktu i Deweloperów
- Zwiększanie przejrzystości i zrozumiałości elementów backlogu, a także podział dużych elementów na mniejsze
- Uszczegółowienie elementów
- Dodawanie nowych elementów
- Aktualizacja priorytetów
- Aktualizacja oszacowań
  
- Jest realizowane w trakcie sprintu (zabiera nie więcej niż 10% czasu sprintu)
- *W projekcie dyplomowym inżynierskim, przy 2-tygodniowych sprintach, na cotygodniowym spotkaniu w środku sprintu → duża korzyść dla projektu*

# Rozwinięcie opisu funkcji produktu

- Właściciel produktu i/lub analitycy rozwijają opisy elementów produktu na NASTĘPNY sprint, aby były bardziej zrozumiałe dla Deweloperów
- Możliwe techniki, sposoby opisu
  - opis w języku naturalnym
  - formuły, algorytmy
  - kryteria akceptacji
  - przypadki testowe

Task 21 created Fri Jul 11, 2014 13:35 by jakubm

No.	Description	Owner	Status	Pri.	Est.	Rem.
21	Zadowolenie klienta - wyliczane z czasu, kosztu, zakresu i jakości <small>(wiki, 2 comments)</small>	Platonow	Completed	P0	5	0

**Wiki** Edit

Zadowolenie klienta wyrażane jest w skali 0-100% rzutowanej na 5 poziomów: bardzo niskie 0-20%, niskie 21-40%, średnie 41-60%, wysokie 61-80%, bardzo wysokie 81-100%.

Wyliczane jest następująco: (( procent wykonanej wartości biznesowej produktu / procent upłyniętego czasu projektu ) + procent maksymalnej jakości produktu + (100% - ryzyko porażki projektu)) / 3

Czyli jest to średnia z: zzawansowania projektu, jakości produktu oraz szansy sukcesu projektu (100% - ryzyko porażki). Ryzyka porażki projektu nie ma jeszcze w sprintie 2, więc na razie określamy je jako 0%.

Zadowolenie klienta obliczane jest co określoną jednostkę czasu gry zwaną taktiem projektu TAKT.

**Źródło:** R. Piechowski, D. Płatek, A. Wasik, S. Grabowska, Gra edukacyjna ucząca zarządzania projektami.

Projekt dyplomowy inżynierski, Katedra Inżynierii Oprogramowania, Politechnika Gdańskia, 2014

# Artefakty: Backlog sprintu

## ➤ Backlog Sprintu (Rejestr Sprintu) (ang. *Sprint Backlog*)

- „Na Sprint Backlog składają się: Cel Sprintu (po co), elementy Product Backlogu wybrane do realizacji w Sprincie (co) oraz wykonalny plan dostarczenia Incrementu (jak).” [Scrum Guide]
- plan przygotowany przez Developerów dla nich samych
- za jego opracowanie i prowadzenie odpowiadają Deweloperzy

## ➤ Zobowiązanie: Cel Sprintu (ang. *Sprint Goal*)

- wyrażona jednym zdaniem główna wartość do dostarczenia w Sprincie
- jedyny cel w sprincie
- „pozostawia swobodę pod względem tego, co dokładnie należy zrobić, aby go osiągnąć”
- zapewnia spójność i skupienie

# Zobowiązanie: Cel Sprintu

## ➤ Przykłady dobrego celu sprintu

- możliwość umieszczania towarów w koszyku oraz złożenia zamówienia
- analiza i przygotowanie zbioru danych do trenowania sieci neuronowej
- zbudowany i zbadany model klasyfikacyjny oparty na losowym lesie

## ➤ Przykłady złego celu sprintu

- pierwsza wersja systemu z najważniejszymi funkcjami (zbyt ogólny cel pasujący do każdego produktu)
- kluczowe funkcje umożliwiające wykonywanie podstawowych czynności (zbyt ogólny cel pasujący do każdego sprintu)
- możliwość przeglądania listy towarów, szczegółów towarów, dodawanie do koszyka, usuwanie z koszyka, zamawianie, płatność i dostawa, potwierdzenie, przeglądanie zamówień (zbyt rozbudowany, szczegółowy cel sprintu)

# Backlog Sprintu - Sprint backlog

- Backlog sprintu reprezentuje **zakres produktu i pracy do wykonania w ramach danego sprintu** na podstawie ustaleń z Właścicielem produktu
- Backlog sprintu zawiera elementy z backlogu produktu (głównie te o aktualnie najwyższym priorytecie) podjęte do wykonania przez Deweloperów
- Elementy bardziej złożone (np. user stories) rozbijane są na zadania
- **Zadania powinny być jednoosobowe, jeden dzień roboczy lub mniejsze**
- **Zadania szacowane są już w godzinach, a nie w story points**

**Punkt widzenia  
Deweloperów**

Sprint 3. Plug in the Real Weather		
Story ID	Story/task	0
10	Fetch one day temperature data from the weather provider system Make our server connect and authenticate to the provider system Read provider's data directory Parse the current temperature out of the data Push the temperature data to the client	63
11	Fetch rain, snow, etc details from the provider Parse snow/rain data from the provider's data Push the snow/rain data to the client Redesign client screen a bit Refactor the server code	16
12	Fetch several days data from the provider Parse the weather data in day packs	4
13	Auto-refresh feature Make the client ping server once per 4 hours Make the server update the client	4

# Przykład Backlogu Sprintu

## Zgłoszenia zakończone

						<a href="#">Wyświetl w Nawigatorze zgłoszeń</a>
Klucz	Podsumowanie	Typ zgłoszenia	Priorytet	Status	Liczba zgłoszeń (15)	
ZZ-14 *	Wczytywanie wejścia gracza	Feature/story	↑ Showstopper	GOTOWE	1	
ZZ-16 *	Chodzenie	Feature/story	↑ Showstopper	GOTOWE	1	
ZZ-17 *	Bieganie	Feature/story	↑ Wysoki	GOTOWE	1	
ZZ-18 *	Skakanie	Feature/story	↑ Showstopper	GOTOWE	1	
ZZ-19 *	Bieganie po ścianie	Feature/story	↑ Średni	GOTOWE	1	
ZZ-22 *	Ruchome platformy	Feature/story	↑ Wysoki	GOTOWE	1	
ZZ-43 *	Plik z ustawieniami gry	Feature/story	↑ Showstopper	GOTOWE	1	
ZZ-48 *	Poruszanie się w powietrzu	Feature/story	↑ Średni	GOTOWE	1	
ZZ-50 *	Kucanie	Feature/story	↑ Średni	GOTOWE	1	
ZZ-68 *	Stworzenie repozytorium	Zadanie	↑ Showstopper	GOTOWE	1	
ZZ-69 *	Ustawienia .gitignore	Zadanie	↑ Wysoki	GOTOWE	1	
ZZ-70 *	Ustawienie narzędzia do rozwiązywania konfliktów plików .meta	Zadanie	↑ Wysoki	GOTOWE	1	
ZZ-72 *	Plansza treningowa	Feature/story	↑ Showstopper	GOTOWE	1	
ZZ-78 *	Muzyka w tle	Feature/story	↑ Średni	GOTOWE	1	
ZZ-103 *	Wdrożyć program do modyfikacji plansz	Zadanie	↑ Showstopper	GOTOWE	1	

## Zgłoszenia usunięte ze sprintu

						<a href="#">Wyświetl w Nawigatorze zgłoszeń</a>
Klucz	Podsumowanie	Typ zgłoszenia	Priorytet	Status	Liczba zgłoszeń (1)	
ZZ-23 *	Rozpadający się teren	Feature/story	↑ Wysoki	NEW	1	

***Uwaga: to jest stan Backlogu Sprintu po zakończeniu sprintu***

**Źródło:** P. Kriger, K. Chmielewski, K. Ćwikliński, Gra zręcznościowa 3D zrealizowana z użyciem zaadaptowanej metody Scrum, Projekt dyplomowy inżynierski, Katedra Inżynierii Oprogramowania, Politechnika Gdańsk, 2020

# Przykład Backlogu Sprintu (sprint 2)

**Uwaga: to jest stan Backlogu Sprintu po zakończeniu sprintu**

**Źródło:** M. Błaszkowski, M. Dargacz, K. Żmijewski,  
 System wspomagający komunikację w szkole nauki jazdy.  
 Projekt dyplomowy inżynierski, Katedra Inżynierii  
 Oprogramowania, Politechnika Gdańskia, 2012

No.	Description	Owner All	Status All	Pri. All	Est. All	Rem.
44	Formularz rejestracji - zapisy on-line	mblaszkowski	In Progress	P1	6	1
154	Dodanie możliwości zapisu na kurs + rejestracja w dwóch krokach	mblaszkowski	Completed	-	2	1
100	Wykonanie formularza (1 comment)	mblaszkowski	Completed	-	4	0
45	Logowanie	Hannibal_DT	Completed	P1	19	5
104	Wykonanie formularza logowania	Hannibal_DT	Completed	-	2	0
105	Weryfikacja wprowadzonych danych	Hannibal_DT	Completed	-	2	0
112	Autoryzacja. Odróżnianie poziomu dostępu.	Hannibal_DT	Completed	-	10	0
106	Opracowanie funkcjonalności przypominania hasła	Hannibal_DT	Not Started	-	5	5
65	Tworzenie nowych kont	dargi	Completed	P1	6	0
126	Formularz tworzenia konta dla pracowników biurowych. (1 comment)	dargi	Completed	-	3	0
127	Formularz tworzenia konta dla instruktora. (1 comment)	dargi	Completed	-	3	0
68	Przeglądanie listy kursantów/pracowników/instruktorów	mblaszkowski	Completed	P1	6	0
155	Lista kursantów	mblaszkowski	Completed	-	1	0
156	Lista pracowników	mblaszkowski	Completed	-	1	0
157	Lista instruktorów	mblaszkowski	Completed	-	1	0
158	Obsługa paginacji strony	mblaszkowski	Completed	-	3	0
113	Stworzenie szkieletu modułu administracyjnego (po zalogowaniu na konto administracyjne wyświetla się strona do zarządzania systemem)	Hannibal_DT	Completed	-	1	0
134	Aktualizacja konta klienta, który zarejestrował się on-line (1 comment)	dargi	Completed	P2	9	0
146	Lista nieaktywnych kont (z kryterium czasowym) z możliwością przejścia do aktywacji konta (1 comment)	dargi	Completed	-	3	0
144	Stworzenie formularza z wypełnionymi danymi pobranymi z bazy danych	dargi	Completed	-	2	0
145	Możliwość edycji wcześniej wpisanych (przy zapisu klienta on-line) danych	dargi	Completed	-	4	0
135	Dodawanie nowych kursów (1 comment)	mblaszkowski	Completed	P1	10	0
139	Zmiany w projekcie bazy danych (nowa tabela + flaga w tabeli Drives)	mblaszkowski	Completed	-	1	0
140	Zmiany w modelu klas we framework Zend	mblaszkowski	Completed	-	2	0
141	Stworzenie formularza do tworzenia informacji o nowych kursach (rodzaj kursu, data pierwszego spotkania)	mblaszkowski	Completed	-	4	0
142	Przeglądanie listy wszystkich dodanych kursów	mblaszkowski	Completed	-	3	0
53	Przeglądanie terminów swoich jazd we własnym kalendarzu (1 comment)	dargi	Not Started	P1	7	7
138	Research Zbadanie, czy można dodawać jazdy bezpośrednio na kalendarzu (2 comments)	dargi	Not Started	-	1	1
137	Pobieranie potrzebnych informacji z bazy danych i wyświetlenie ich na kalendarzu	dargi	Not Started	P1	3	3
136	Stworzenie kalendarza z możliwością podglądu miesięczego/dnia	dargi	Not Started	-	3	3
76	Anulowanie jazd (1 comment)	dargi	Not Started	P1	3	3
148	Blokada - brak możliwości anulowania jazdy przy określonych warunkach (np. można anułować jazdę najpóźniej 24 godzin przed jazdą)	dargi	Not Started	-	2	2
149	Zmiana statusu jazdy w bazie danych (wyświetlanie takiej jazdy innym kolorem na kalendarzu)	dargi	Not Started	-	1	1
150	Stworzenie layoutu dla modułu dla kursanta.	dargi	Completed	-	5	0
152	Zaprojektowanie wizualne layoutu.	dargi	Completed	-	2	0
151	Stworzenie szkieletu strony.	dargi	Completed	-	3	0
	Click here to add a new task			Total: 72	16	
	New Task					© WETI PG

# Planowanie sprintu

- Planowanie sprintu (ang. *Sprint planning meeting*)
- Temat 1 – Dlaczego ten sprint ma wartość?
  - Właściciel Produktu i Deweloperzy ustalają Cel Sprintu
- Temat 2 – Co może zostać ukończone w tym Sprincie? (wybór zakresu sprintu)
  - Deweloperzy wraz z Właścicielem Produktu uzgadniają, jakie elementy z Backlogu Produktu zostaną wykonane w sprincie
- Temat 3 – W jaki sposób zostanie wykonana praca? (ustalenie sposobu przekształcenia wymagań w Przyrost)
  - Deweloperzy opracowuje zadania wymagane do dostarczenia każdego wybranego elementu produktu w Przyroście
  - Właściciel Produktu może doradzać, ale nie decyduje
- Całe spotkanie trwa maks. 8h dla 4-tygodniowego sprintu, maks. 4h dla 2-tygodniowego sprintu
- Cel sprintu, wybrane do realizacji w danym sprincie elementy backlogu produktu oraz zadania deweloperskie razem tworzą Backlog Sprintu

# Dobór zakresu sprintu

- Jak dużo *story points* elementów produktu wybrać łącznie do sprintu?
- Nie wiadomo, ile *story points* na godzinę (na sprint) wykona zespół
- Pierwszy sprint
  - obliczyć liczbę sprintów – czas na projekt / długość sprintu
  - obliczyć średnią liczbę *story points* na sprint
  - wybrać elementy produktu o takiej łącznej liczbie *story points*
  - oszacować zadania w godzinach
  - jeżeli suma (znacznie) przekracza dostępne godziny (pojemność) zredukować zakres
- Kolejne sprints
  - po poprzednich sprintach znane będzie coraz lepsze przybliżenie wydajności zespołu (**prędkości, ang. team velocity = story points / godzinę**)
  - przyjąć dostępną liczbę godzin zespołu w sprintie (**pojemność zespołu, ang. team capacity**)
  - obliczyć ile *story points* zespół może zrealizować i tyle wybrać
  - w kolejnych sprintach wydajność (prędkość) powinna rosnąć
    - rzeczywisty przykład:
      - 1. sprint –  $0,38 \text{ SP} / \text{h} = 13 \text{ SP} / 34 \text{ h}$
      - 2. sprint –  $0,47 \text{ SP} / \text{h} = 32 \text{ SP} / 68 \text{ h}$
      - 3. sprint –  $0,59 \text{ SP} / \text{h} = 25 \text{ SP} / 42 \text{ h}$

Decyzję podejmują  
Deweloperzy

# Dobór zakresu sprintu - przykład

## ➤ Dane wejściowe

- czas na projekt – 4 miesiące = 18 tygodni
- długość sprintu – 2 tygodnie
- łączna liczba *story points* w backlogu produktu – 250SP
- zakładana liczba sprintów =  $18/2 = 9$
- zakładana średnia prędkość (liczba SP / sprint) = ok. 28 SP = **250 SP / 9**

## ➤ 1. sprint - szacowanie

- zakres = średnia prędkość = ok. 28 SP
- oszacowana łączna liczba godzin zadań – 140 h
- szacowana prędkość =  $0,2 \text{ SP} / \text{h} = 28 \text{ SP} / 140 \text{ h}$

## ➤ 1. sprint - realizacja

- faktycznie zrealizowany zakres 1. sprintu – 24 SP
- faktyczna liczba godzin zużyta na 1. sprint – 160 h
- faktyczna prędkość =  $0,15 \text{ SP} / \text{h} = 24 \text{ SP} / 160 \text{ h}$

## ➤ 2. sprint – szacowanie

- dostępna liczba godzin zespołu w sprincie – 120 h
- zakres  $\geq 18 \text{ SP} = 120 \text{ h} * 0,15 \text{ SP} / \text{h}$

Podkreślenie oznacza fikcyjne, przykładowe dane użyte do obliczeń

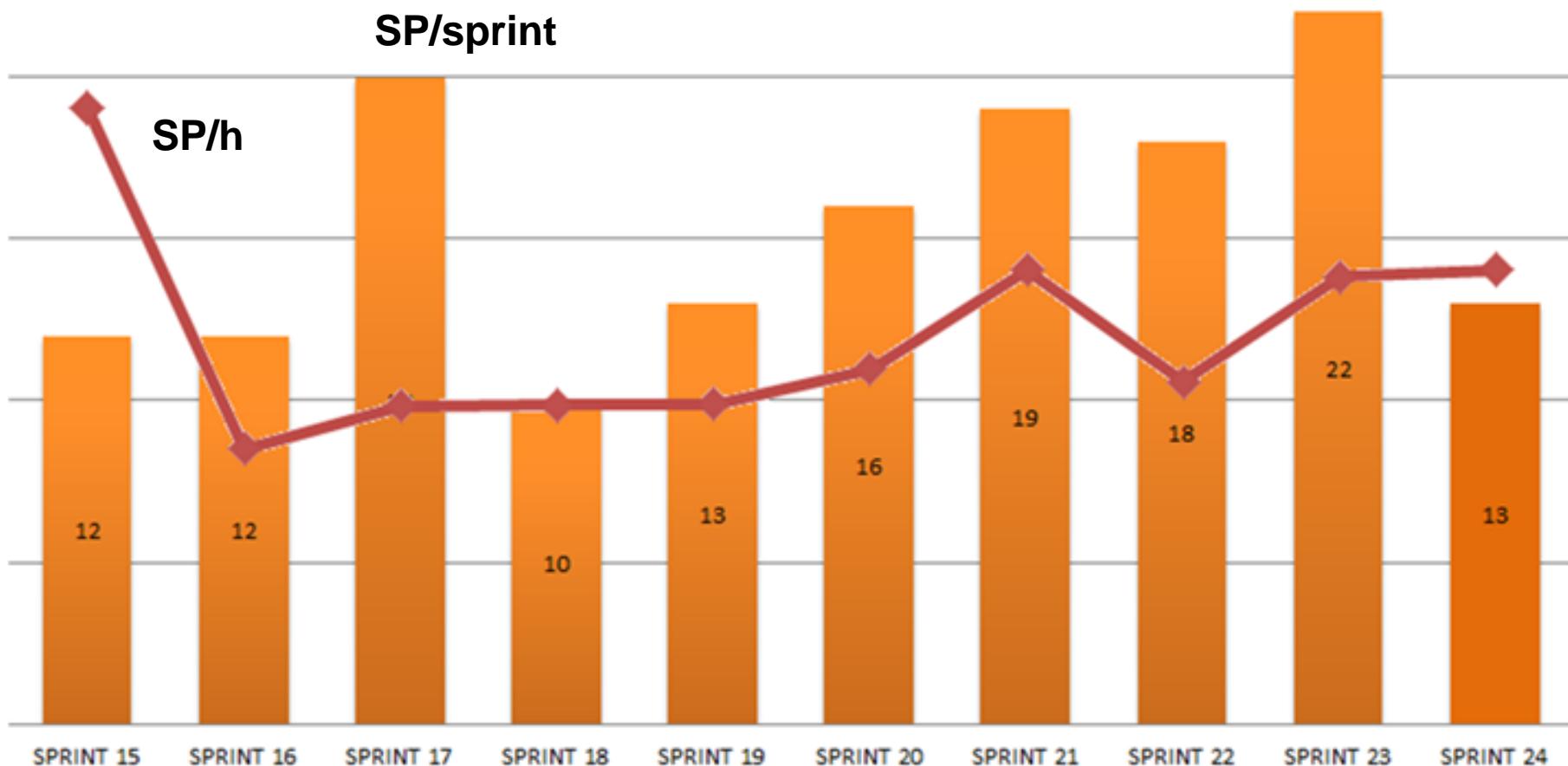
# Szacowanie pracy w Sprincie – pojemność zespołu

Tabela 2: Utrata czasu przeznaczonego na pracę w sprincie (wartości w godzinach)

Typowy czas pracy	Czas wyłączony ze sprintu					<b>Pozostało</b>
	Dzień ustawowo wolny	Urlopy	Szkolenia	“Szum”		
Pracownik 1	80	8	8	4	50%	
Pracownik 2	80	8	0	8	50%	
Pracownik 3	80	8	16	0	50%	
Pracownik 4	80	8	40	0	50%	
Pracownik 5	80	8	0	8	50%	
Zespół	<b>400</b>	40	64	20	50%	<b>138</b>

Źródło: Marcin Przypek, *Wspomaganie szacowania czasu i kosztu w zwinnych projektach informatycznych*, praca dyplomowa magisterska, opiekun J. Miler, WETI, PG, 2013

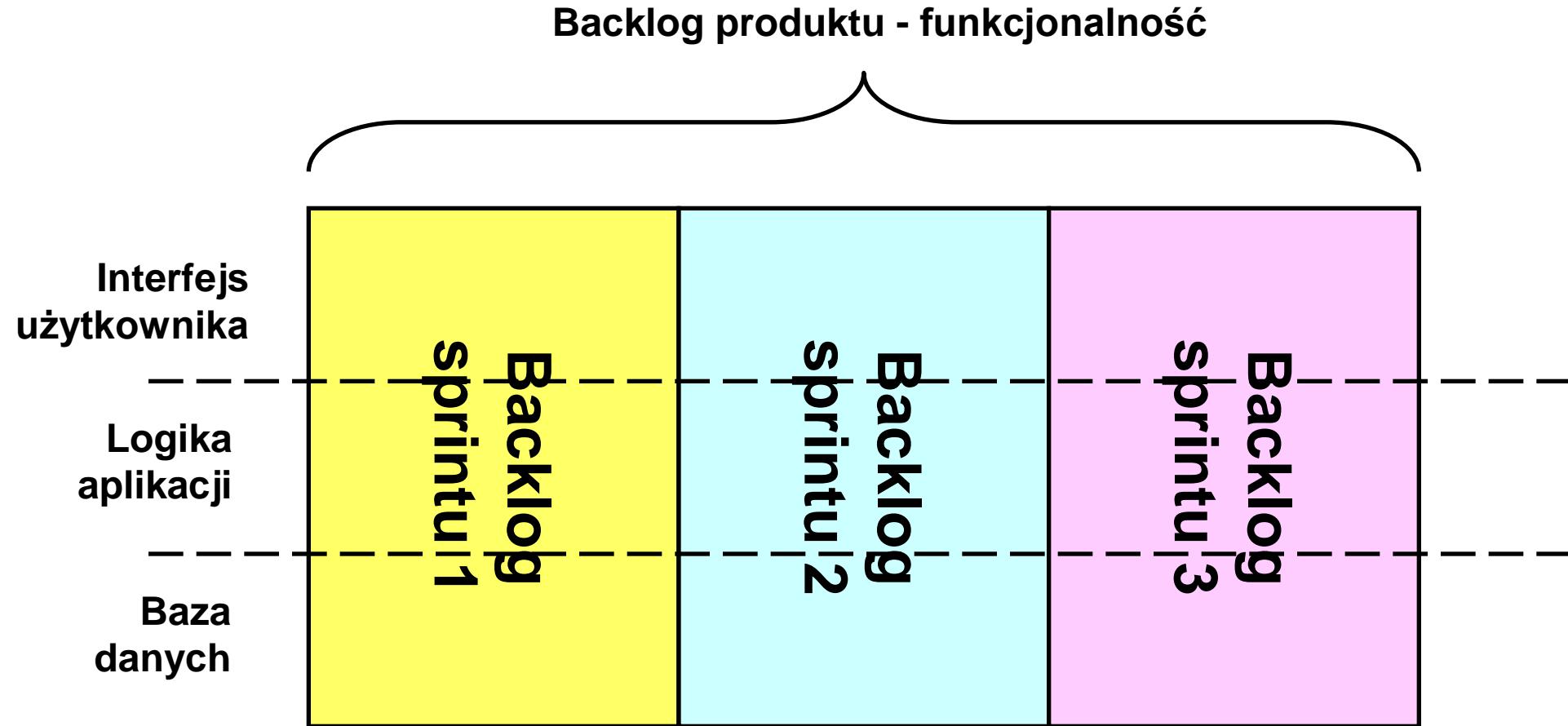
# Szacowanie pracy w Sprincie – prędkość zespołu



Rysunek 4: Analiza szybkości zespołu

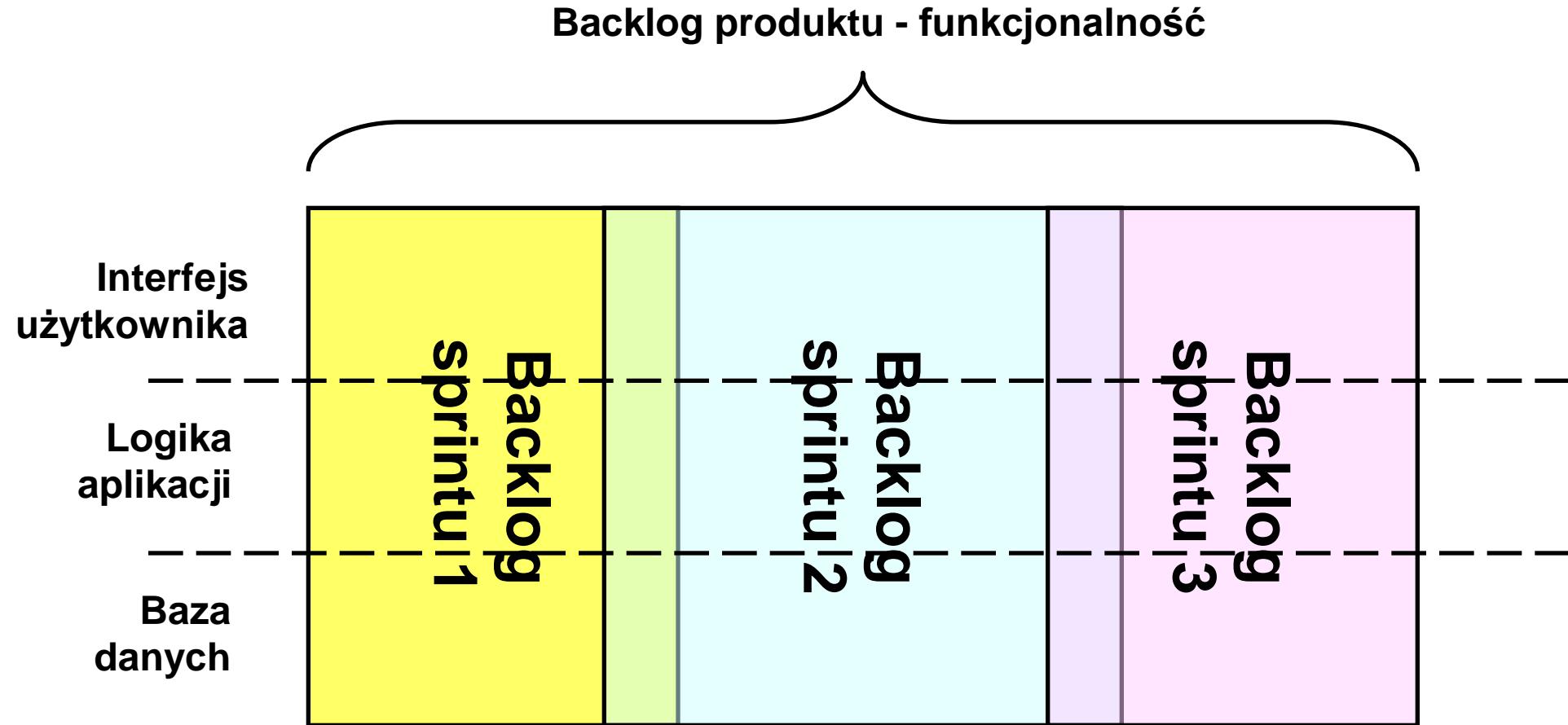
Źródło: Marcin Przypek, *Wspomaganie szacowania czasu i kosztu w zwinnych projektach informatycznych*, praca dyplomowa magisterska, opiekun J. Miler, WETI, PG, 2013

# Backlog produktu a backlog sprintu



- Backlog produktu – funkcje z punktu widzenia Właściciela produktu
- Backlog sprintu – wytwarzanie wybranych funkcji z punktu widzenia Deweloperów

# Backlog produktu a backlog sprintu (2)



- Może się zdarzyć, że nie wszystkie zadania dla danej cechy/funkcji produktu zostaną zrealizowane w danym sprintie – przechodzą wtedy do innego sprintu (niekoniecznie następnego i jeżeli potwierdzi to Właściciel produktu)

# Artefakty: Przyrost (*ang. Increment*)

## ➤ Przyrost (*ang. Increment*)

- „**Increment to konkretny krok w kierunku osiągnięcia Celu Produktu**” [Scrum Guide]
- „**Aby dostarczyć wartość, Increment musi być użyteczny**”
- **przyrost musi być „ukończony”, używalny, niezależnie od tego czy Właściciel Produktu zdecyduje się go wydać**
- „**W trakcie Sprintu może zostać wytworzonych wiele Incrementów.**”

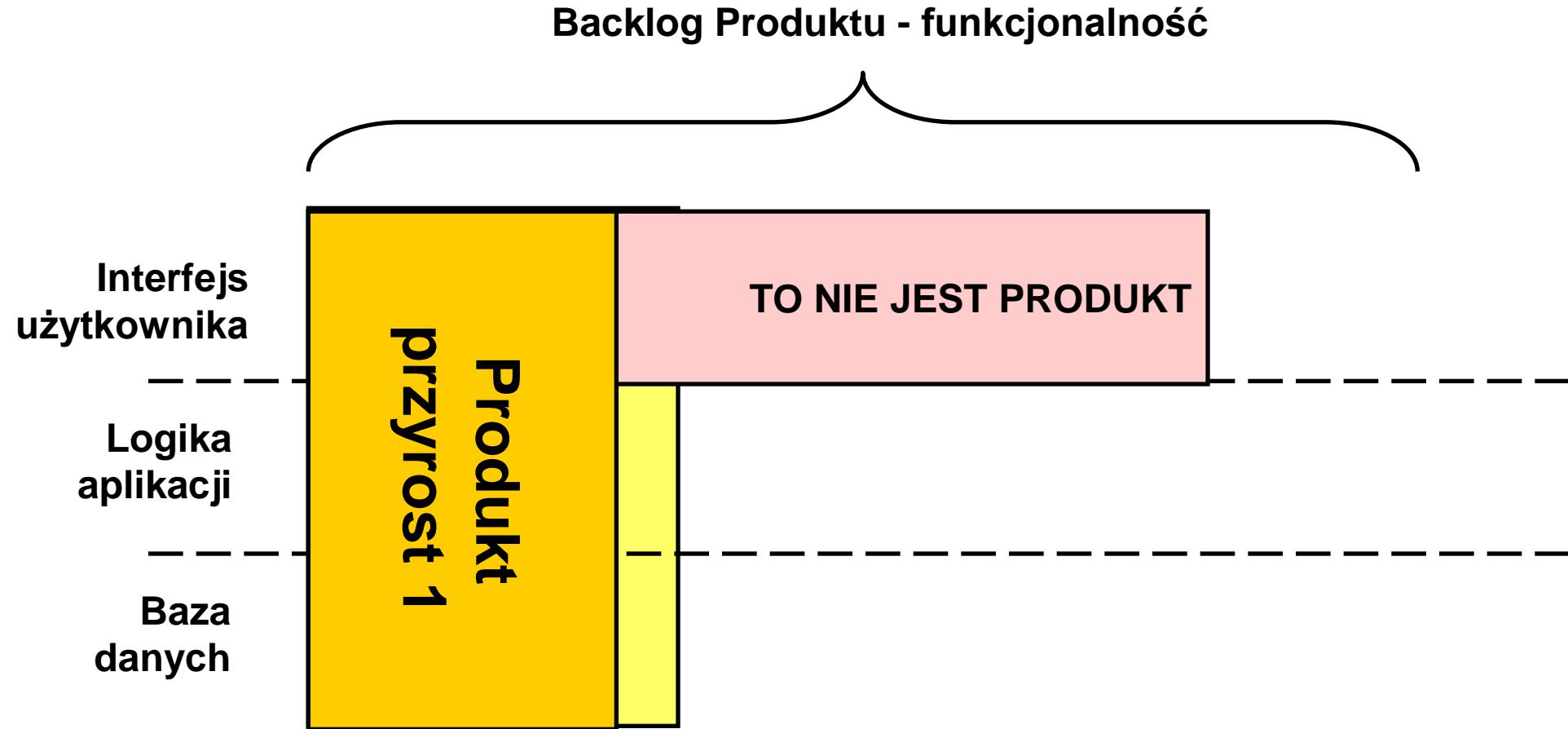
## ➤ Zobowiązanie: Definicja Ukończenia (*ang. Definition of Done*)

- „**formalny opis stanu Incrementu, w którym spełnia on kryteria jakościowe wymagane dla produktu**”

# Definicja ukończenia (ang. *Definition of Done*)

- Przykład: Element produktu jest ukończony gdy:
  - Napisano kod
  - Napisano testy
  - Kod przetestowano i poprawiono błędy
  - Wykonano testy integracyjne z Przyrostem
  - Kod i testy umieszczone w repozytorium
  - Zaktualizowano backlog sprintu
- „Kiedy element Product Backlogu osiąga zgodność z Definicją Ukończenia, powstaje Increment.”
- Definicja ukończenia pomaga Deweloperom w doborze zakresu sprintu
- Definicja ukończenia powinna być rozwijana w kolejnych projektach, aby Zespoły Scrumowe dostarczały produkty coraz wyższej jakości

# Produkt po sprincie - przyrost



- Przyrost udostępnia jakiś podzbiór funkcjonalności
- Sam interfejs użytkownika czy baza danych nie są przyrostem

# Przyrost 1 – produkt po sprincie 1



**Źródło:** R. Piechowski, D. Płatek, A. Wasik, S. Grabowska, Gra edukacyjna ucząca zarządzania projektami.  
 Projekt dyplomowy inżynierski, Katedra Inżynierii Oprogramowania, Politechnika Gdańskia, 2014

# Przyrost 2 – produkt po sprincie 2

## Gra PM

Produkty	
Wejścia do mrowiska	-3 5 30
Otwory ewakuacyjne	3 3 40
Izba odpoczynku	2 1 20
Pokój narad	-2 2 10
Salon gier i zabaw	-2 8 30
Sypialnie	1 6 30

Zasoby	
Robotnice	
Oplekunki	
Wartownicy	
Zbiorniki	

Zadania	
Magazyn jedzenia świeżego	1 6 40
Hodowla grzybów	2 6 30
Maskowanie mrowiska	2 3 50

Ryzyka	
--------	--

Działania zapobiegawcze	
Wysłanie zwiadówców	8
Testowanie mrowiska	10
Poszukiwanie nowych grzybów	6
Rozdzielanie salony gier i zabaw	12

Wykonano: 0% Wynik: 0 Satysfakcja klienta: 3 Zadowolenie zespołu: 3 Dzień 0. Godzina - 08:23.

**Źródło:** R. Piechowski, D. Płatek, A. Wasik, S. Grabowska, Gra edukacyjna ucząca zarządzania projektami.  
Projekt dyplomowy inżynierski, Katedra Inżynierii Oprogramowania, Politechnika Gdańskia, 2014

# Przyrost 3 – produkt po sprincie 3

 Zarządzanie Projektem : Mrowisko

Zakres: 

0%

Jakość:

Czas: 

Dzień: 0

 Ryzyko: 

10%

Wynik: 0

Zasoby	
Robotnice	10
Opiekunki	3
Wartownicy	1
Zbieracze	6
Hodowcy	5
Zwiadowcy	2

Ryzyka

Produkty

Komora królowej	10	2	30	▲
Magazyn jedzenia świeżego	6	1	40	▲
Magazyn jedzenia martwego	3	0	20	▲
Wylegarnia wielka	10	1	40	▲
Wylegarnia zapasowa	8	0	20	▲
Szkoła robotnic	6	0	10	▲
Koszary wartowników	4	0	10	▲
Korytarze główne	8	1	40	▲
Korytarze boczne	3	-2	30	▲
Kanały wentylacyjne	8	1	20	▲
Wejścia do mrowiska	5	-3	30	▲
Otwory ewakuacyjne	3	3	40	▲
Izba odpoczynku	1	2	20	▲

Zadania

Działania wobec ryzyka

Wysłanie zwiadowców	6	✿
Testowanie mrowiska	6	✿
Poszukiwanie nowych grzybów	4	✿
Powiększenie salonu gier i zabaw	8	✿
Zgromadzenie zapasów jedzenia	10	✿
Wystawienie licznych wartowników	6	✿
Powiększenie magazynów	10	✿
Przeszkolenie opiekunek	8	✿

Źródło: R. Piechowski, D. Płatek, A. Wasik, S. Grabowska, Gra edukacyjna ucząca zarządzania projektami.  
 Projekt dyplomowy inżynierski, Katedra Inżynierii Oprogramowania, Politechnika Gdańskia, 2014

# Przyrost 4 – produkt po sprincie 4

 Projekt: Mrowisko

Czas:	dzień: 0	Klient:	<div style="width: 30%;"> </div>
Zakres:	2%	Zespół:	<div style="width: 100%;"> </div>
Jakość:	<div style="width: 10%;"> </div>	Ryzyko:	<div style="width: 20%;"> </div>

**Zasoby**

Robotnice	10 
Opiekunki	3 
Wartownicy	1 
Zbieracze	6 
Hodowcy	5 

**Ryzyka**

Niski przyrost naturalny 70% 

**Produkt**

Komora królowej	10  2  30 
Magazyn jedzenia świeżego	6  1  40 
Magazyn jedzenia martwego	3  0  20 
Wylegarnia zapasowa	8  0  20 
Korytarze główne	8  1  40 
Korytarze boczne	3  -2  30 
Kanaly wentylacyjne	8  1  20 
Wejścia do mrowiska	5  -3  30 
Izba odpoczynku	1  2  20 
Hodowla grzybów	6  2  30 
Pokój parady	2  -2  10 

**Zadania**

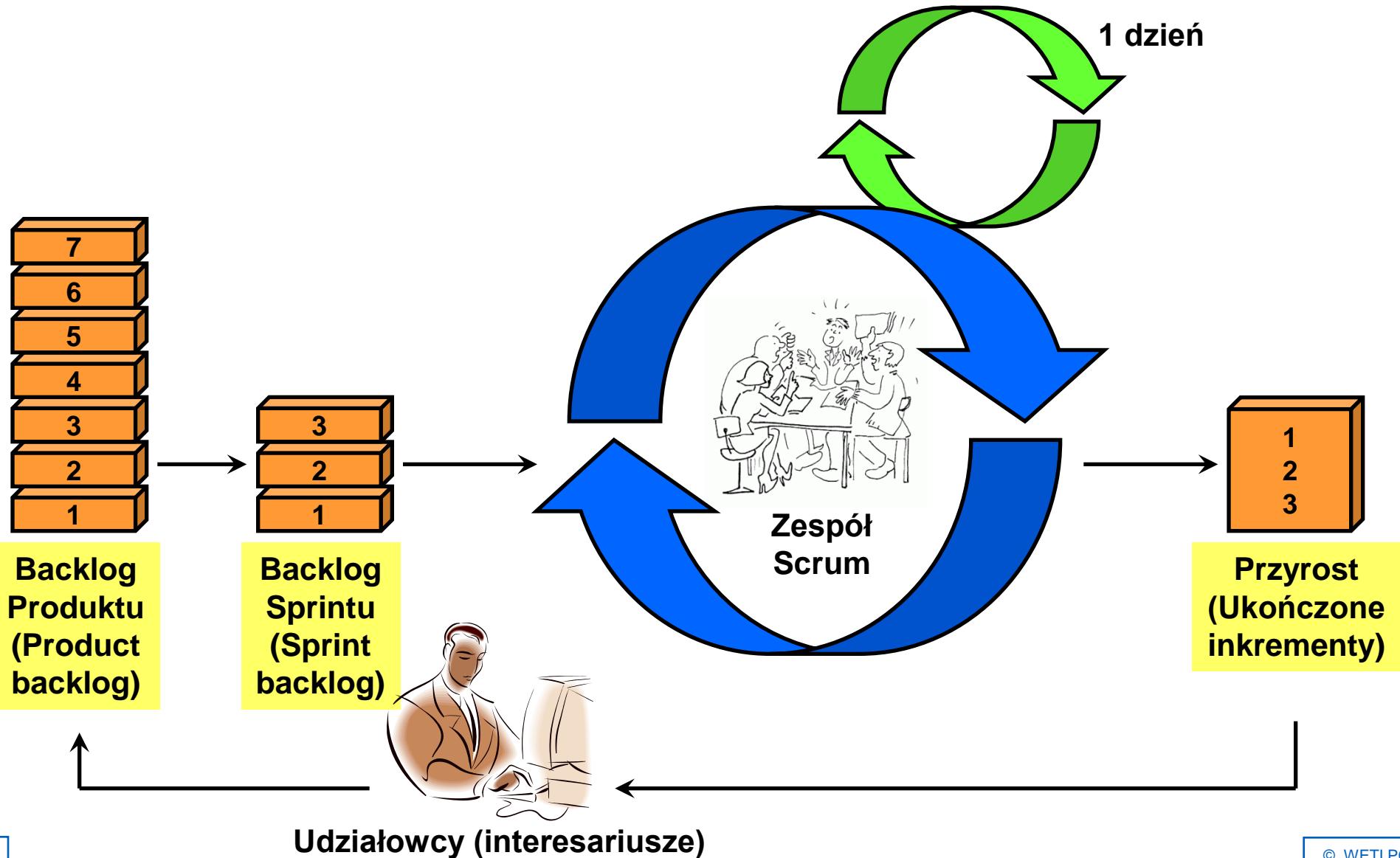
Wylegarnia wielka (Rob)	10  1  34 
Szkoła robotnic (Opi)	6  0  8 
Koszary wartowników (War)	4  0  9 
Otwory ewakuacyjne (Zbi)	3  3  37 

**Działania wobec ryzyka**

Wysłanie zwiadowców	8 
Testowanie mrowiska	10 
Poszukiwanie nowych grzybów	6 
Powiększenie salonu gier i zabaw	12 
Zgromadzenie zapasów jedzenia	16 
Wystawianie licznych wartowników	8 
Powiększenie magazynów	16 
Przeszkolenie opiekunek	12 
Powiększenie korytarzy	10 
Zmniejszenie tempa pracy	6 

**Źródło:** R. Piechowski, D. Płatek, A. Wasik, S. Grabowska, Gra edukacyjna ucząca zarządzania projektami.  
 Projekt dyplomowy inżynierski, Katedra Inżynierii Oprogramowania, Politechnika Gdańskia, 2014

# Scrum – proces



# Realizacja Projektu Informatycznego



***Jakub Miler***

*Katedra Inżynierii Oprogramowania  
Wydział Elektroniki, Telekomunikacji i Informatyki  
Politechnika Gdańskia  
jakubm@eti.pg.edu.pl*

*Materiały pomocnicze do wykładu na Wydziale ETI Politechniki Gdańskiej.  
Wykorzystanie materiałów w innym celu i ich rozpowszechnianie bez zgody WETI PG zabronione.*

## Wykład 4

# Scrum: Przyrost, Codzienny Scrum, Przegląd Sprintu, Retrospektyna



# Artefakty: Przyrost (*ang. Increment*)

## ➤ Przyrost (*ang. Increment*)

- „**Increment to konkretny krok w kierunku osiągnięcia Celu Produktu**” [Scrum Guide]
- „**Aby dostarczyć wartość, Increment musi być użyteczny**”
- **przyrost musi być „ukończony”, używalny, niezależnie od tego czy Właściciel Produktu zdecyduje się go wydać**
- „**W trakcie Sprintu może zostać wytworzonych wiele Incrementów.**”

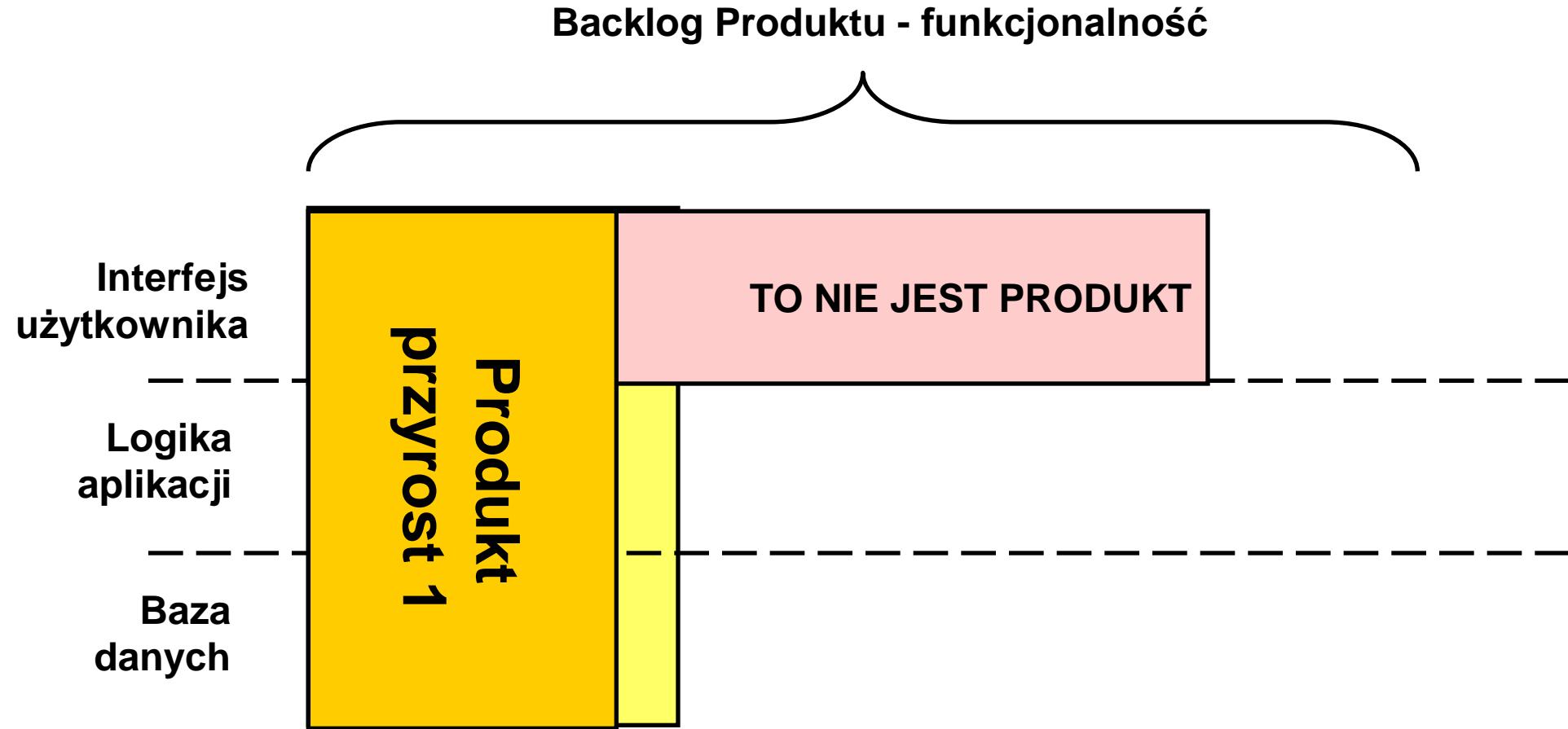
## ➤ Zobowiązanie: Definicja Ukończenia (*ang. Definition of Done*)

- „**formalny opis stanu Incrementu, w którym spełnia on kryteria jakościowe wymagane dla produktu**”

# Definicja ukończenia (ang. *Definition of Done*)

- Przykład: Element produktu jest ukończony gdy:
  - Napisano kod
  - Napisano testy
  - Kod przetestowano i poprawiono błędy
  - Wykonano testy integracyjne z Przyrostem
  - Kod i testy umieszczone w repozytorium
  - Zaktualizowano backlog sprintu
- „Kiedy element Product Backlogu osiąga zgodność z Definicją Ukończenia, powstaje Increment.”
- Definicja ukończenia pomaga Deweloperom w doborze zakresu sprintu
- Definicja ukończenia powinna być rozwijana w kolejnych projektach, aby Zespoły Scrumowe dostarczały produkty coraz wyższej jakości

# Produkt po sprincie - przyrost



- Przyrost udostępnia jakiś podzbiór funkcjonalności
- Sam interfejs użytkownika czy baza danych nie są przyrostem

# Przyrost 1 – produkt po sprincie 1



**Źródło:** R. Piechowski, D. Płatek, A. Wasik, S. Grabowska, Gra edukacyjna ucząca zarządzania projektami.  
 Projekt dyplomowy inżynierski, Katedra Inżynierii Oprogramowania, Politechnika Gdańskia, 2014

# Przyrost 2 – produkt po sprincie 2

## Gra PM

Produkty	
Wejścia do mrowiska	-3 5 30
Otwory ewakuacyjne	3 3 40
Izba odpoczynku	2 1 20
Pokój narad	-2 2 10
Salon gier i zabaw	-2 8 30
Sypialnie	1 6 30

Zasoby	
Robotnice	
Oplekunki	
Wartownicy	
Zbiorniki	

Zadania	
Magazyn jedzenia świeżego	1 6 40
Hodowla grzybów	2 6 30
Maskowanie mrowiska	2 3 50

Ryzyka	
--------	--

Działania zapobiegawcze	
Wysłanie zwiadówców	8
Testowanie mrowiska	10
Poszukiwanie nowych grzybów	6
Rozdzielanie salony gier i zabaw	12

Wykonano: 0% Wynik: 0 Satysfakcja klienta: 3 Zadowolenie zespołu: 3 Dzień 0. Godzina - 08:23.

**Źródło:** R. Piechowski, D. Płatek, A. Wasik, S. Grabowska, Gra edukacyjna ucząca zarządzania projektami.  
Projekt dyplomowy inżynierski, Katedra Inżynierii Oprogramowania, Politechnika Gdańskia, 2014

# Przyrost 3 – produkt po sprincie 3

 Zarządzanie Projektem : Mrowisko

Zakres:  0%

Jakość:  0%

Czas:  Dzień: 0

Ryzyko:  10%

Wynik: 0

Zasoby	
Robotnice	10
Opiekunki	3
Wartownicy	1
Zbieracze	6
Hodowcy	5
Zwiadowcy	2

Ryzyka

Produkty

Komora królowej	10	2	30
Magazyn jedzenia świeżego	6	1	40
Magazyn jedzenia martwego	3	0	20
Wylegarnia wielka	10	1	40
Wylegarnia zapasowa	8	0	20
Szkoła robotnic	6	0	10
Koszary wartowników	4	0	10
Korytarze główne	8	1	40
Korytarze boczne	3	-2	30
Kanały wentylacyjne	8	1	20
Wejścia do mrowiska	5	-3	30
Otwory ewakuacyjne	3	3	40
Izba odpoczynku	1	2	20

Zadania

Działania wobec ryzyka

Wysłanie zwiadowców	6
Testowanie mrowiska	6
Poszukiwanie nowych grzybów	4
Powiększenie salonu gier i zabaw	8
Zgromadzenie zapasów jedzenia	10
Wystawienie licznych wartowników	6
Powiększenie magazynów	10
Przeszkolenie opiekunek	8

Źródło: R. Piechowski, D. Płatek, A. Wasik, S. Grabowska, Gra edukacyjna ucząca zarządzania projektami.  
 Projekt dyplomowy inżynierski, Katedra Inżynierii Oprogramowania, Politechnika Gdańskia, 2014

# Przyrost 4 – produkt po sprincie 4

 Projekt: Mrowisko

Czas: dzień: 0

Zakres: 2%

Jakość: ███████

Punkty: -100

Klient:

Zespół:

Ryzyko:  20%

Zasoby

Robotnice	10	
Opiekunki	3	
Wartownicy	1	
Zbieracze	6	
Hodowcy	5	

Ryzyka

Niski przyrost naturalny 70% 

Produkt

Komora królowej	10	2	30	
Magazyn jedzenia świeżego	6	1	40	
Magazyn jedzenia martwego	3	0	20	
Wylegarnia zapasowa	8	0	20	
Korytarze główne	8	1	40	
Korytarze boczne	3	-2	30	
Kanaly wentylacyjne	8	1	20	
Wejścia do mrowiska	5	-3	30	
Izba odpoczynku	1	2	20	
Hodowla grzybów	6	2	30	
Pokój parady	2	-2	10	

Zadania

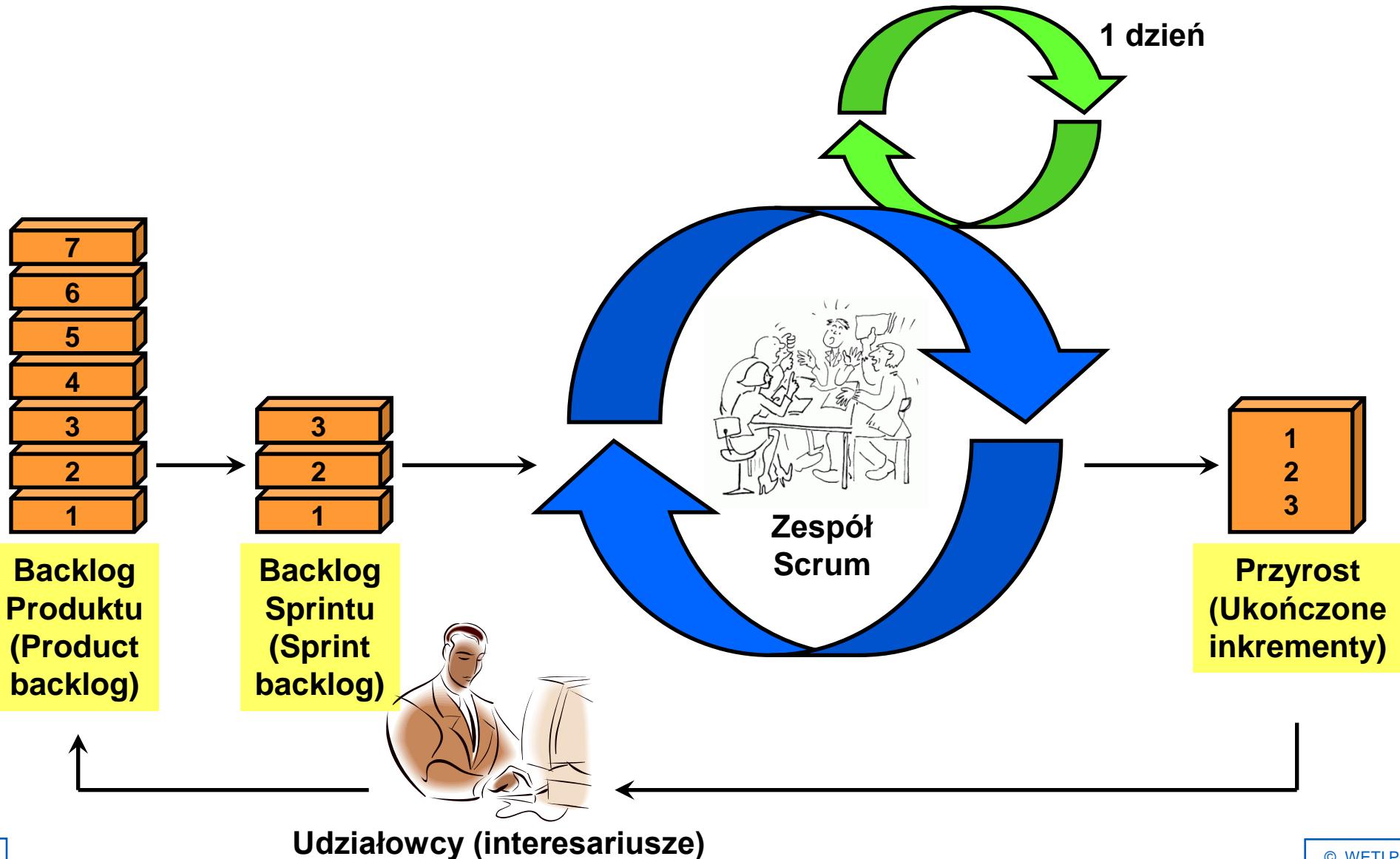
Wylegarnia wielka (Rob)	10	1	34	
Szkoła robotnic (Opi)	6	0	8	
Koszary wartowników (War)	4	0	9	
Otwory ewakuacyjne (Zbi)	3	3	37	



Działania wobec ryzyka

Wysłanie zwiadowców	8	
Testowanie mrowiska	10	
Poszukiwanie nowych grzybów	6	
Powiększenie salonu gier i zabaw	12	
Zgromadzenie zapasów jedzenia	16	
Wystawianie licznych wartowników	8	
Powiększenie magazynów	16	
Przeszkolenie opiekunek	12	
Powiększenie korytarzy	10	
Zmniejszenie tempa pracy	6	

# Scrum – artefakty

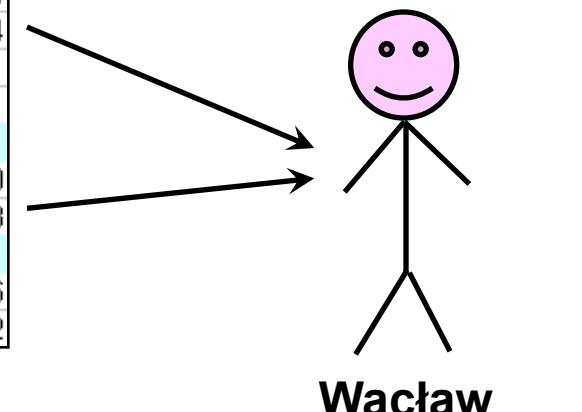
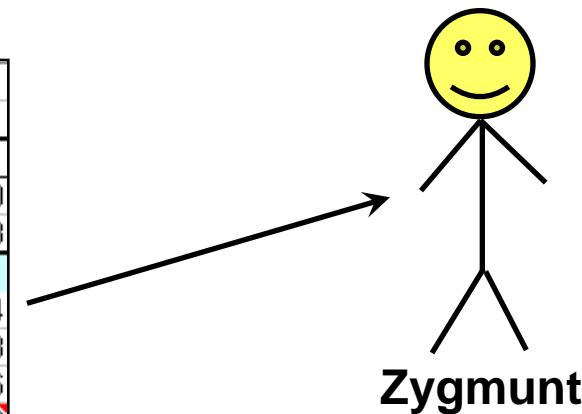


# Zasada przydziału pracy

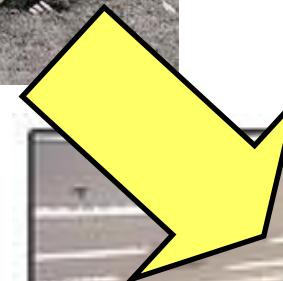
- Członkowie zespołu sami „pobierają” zadania z Backlogu Sprintu według własnego uznania – nie ma odgórnego planowania i przydziału pracy!

## Backlog sprintu:

Sprint 3. Plug in the Real Weather		
Story ID	Story/task	
10	<b>Fetch one day temperature data from the weather provider system</b> Make our server connect and authenticate to the provider system Read provider's data directory Parse the current temperature out of the data Push the temperature data to the client	0 63 4 8 6 16
11	<b>Fetch rain, snow, etc details from the provider</b> Parse snow/rain data from the provider's data Push the snow/rain data to the client Redesign client screen a bit Refactor the server code	4 4 4
12	<b>Fetch several days data from the provider</b> Parse the weather data in day packs Push several days data to the client	10 3
13	<b>Auto-refresh feature</b> Make the client ping server once per 4 hours Make the server update the client	6 2



# Codzienny Scrum – Daily Scrum, Daily



Sprint 3. Plug in the Real Weather		
Story ID	Story/task	
10	Fetch one day temperature data from the weather provider system	0
	Make our server connect and authenticate to the provider system	63
	Read provider's data directory	4
	Parse the current temperature out of the data	8
	Push the temperature data to the client	6
		16
11	Fetch rain, snow, etc details from the provider	
	Parse snow/rain data from the provider's data	4
	Push the snow/rain data to the client	4
	Redesign client screen a bit	
	Refactor the server code	
12	Fetch several days data from the provider	
	Parse the weather data in day packs	10
	Push several days data to the client	3
13	Auto-refresh feature	
	Make the client ping server once per 4 hours	6
	Make the server update the client	2



# Jak prowadzić Codzienny Scrum

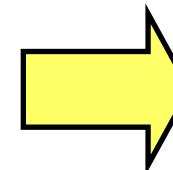
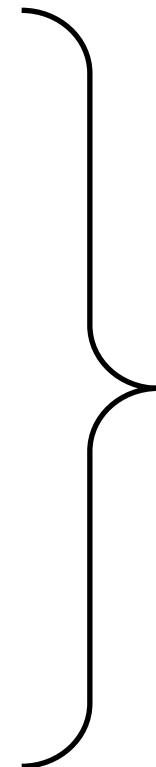
- **Spotkanie jest na stojąco**
  - trwa krócej
  - stanie przypomina, że spotkanie służy synchronizacji, a nie rozwiązywaniu skomplikowanych problemów
- **Ma być krótkie, maks. 15 minut**
  - wzajemna informacja, co kto robi – rozwiązywanie problemów w podgrupach później
- **Stoimy przed tablicą zadań, product backlogiem lub sprint backlogiem**
  - widzimy o czym rozmawiamy, możemy coś zmienić
- **Obecny jest cały zespół**
  - bezpośrednią komunikacją jest najskuteczniejsza
  - można wyznaczyć „zastępstwo”
  - jeżeli nie jest się obecnym, trzeba nadrobić wiedzę o stanie prac

# Jak prowadzić Codzienny Scrum (2)

- **Nie notujemy na komputerze**
  - kto ma komputer ma władzę – wygląda jak kierownik, a nie chcemy kierownika
  - notowanie przekręca słowa, notatki powinny być krótkie
- **Koncentrujemy się na drugim i trzecim pytaniu, nie na pierwszym**
  - 1. Co zrobiłem wczoraj, co pomogło Zespołowi Deweloperskiemu przybliżyć się do osiągnięcia Celu Sprintu?
  - 2. Co zrobię dzisiaj, co pomoże Zespołowi Deweloperskiemu przybliżyć się do osiągnięcia Celu Sprintu?
  - 3. Czy widzę jakiekolwiek przeszkody mogące uniemożliwić mi lub Zespołowi Deweloperskiemu osiągnięcie Celu Sprintu?
  - pierwsze pytanie to tylko kontekst dla drugiego i trzeciego
- **Nie „raportować” Scrum Masterowi**
  - jeżeli zespół zaczyna mówić do Scrum Mastera, ten powinien udawać, że nie słyszy albo nawet odejść nieco na bok
  - spotkanie jest dla Zespołu Deweloperskiego

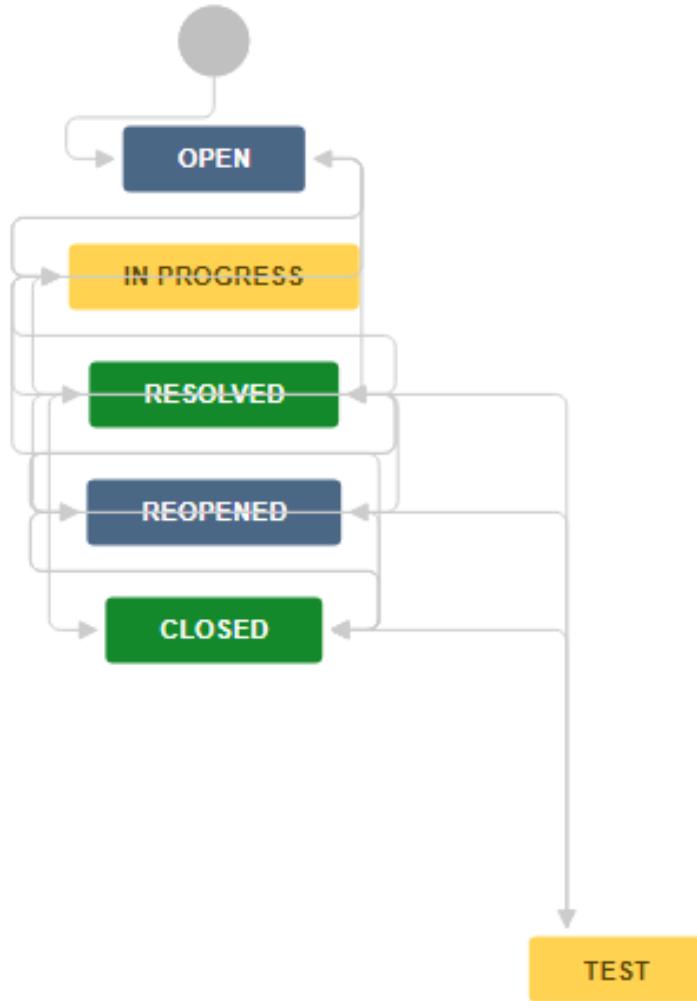
# Praca nad produktem

- Uściślanie wymagań użytkownika
- Projektowanie
- Programowanie
- Testowanie
  - jednostkowe
  - funkcjonalne
- Przeprojektowywanie
- Poprawianie błędów
- Integracja
- Dokumentowanie
  - wewnętrzne dokumenty robocze (mogą być nawet modele!)
  - aktualizacja *sprint backlog* i *sprint burndown chart*



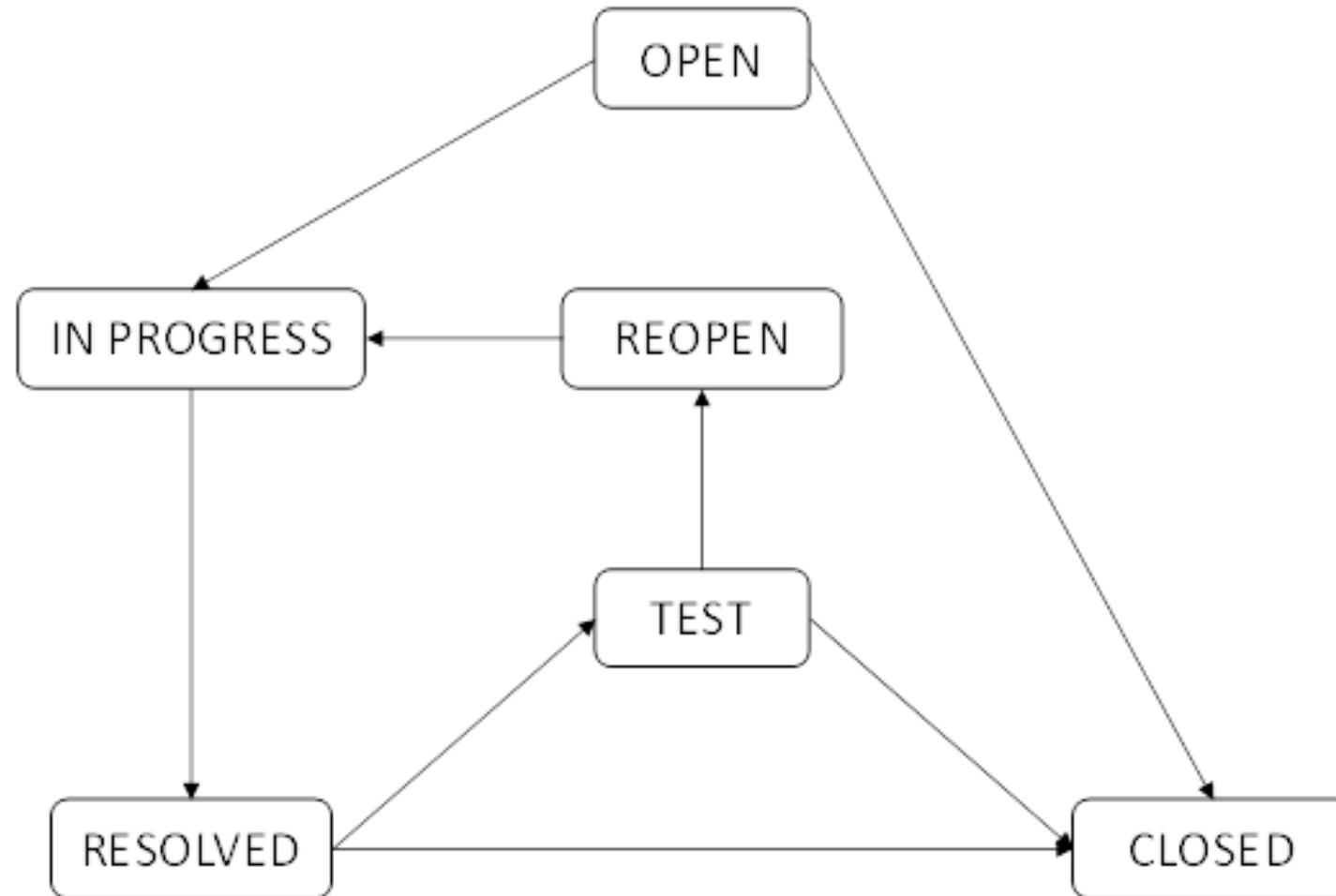
Praktyki  
eXtreme  
Programming

# Obieg zadań / zgłoszeń

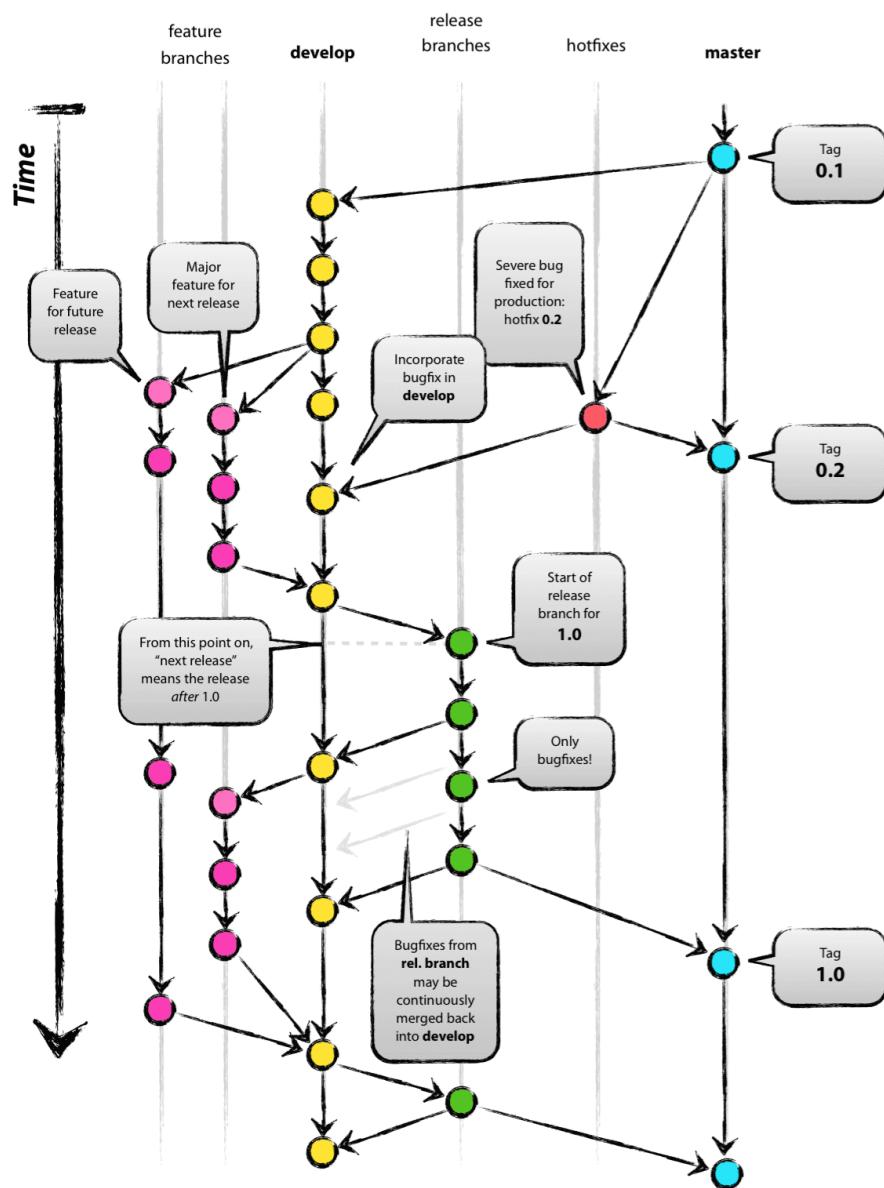


Step Name (id)	Linked Status	Transitions (id)
Open (1)	OPEN	Start Progress (4) >> IN PROGRESS Resolve Issue (5) >> RESOLVED Close Issue (2) >> CLOSED
In Progress (3)	IN PROGRESS	Stop Progress (301) >> OPEN Resolve Issue (5) >> RESOLVED Close Issue (2) >> CLOSED
Resolved (4)	RESOLVED	Close Issue (701) >> CLOSED Test Issue (711) >> TEST Reopen Issue (3) >> REOPENED
Reopened (5)	REOPENED	Resolve Issue (5) >> RESOLVED Close Issue (2) >> CLOSED Start Progress (4) >> IN PROGRESS
Closed (6)	CLOSED	Reopen Issue (3) >> REOPENED
In Testing (7)	TEST	Reopen Issue (721) >> REOPENED Close Issue (731) >> CLOSED

# Typowy obieg zadań / zgłoszeń



# Zarządzanie kodem – Git Flow

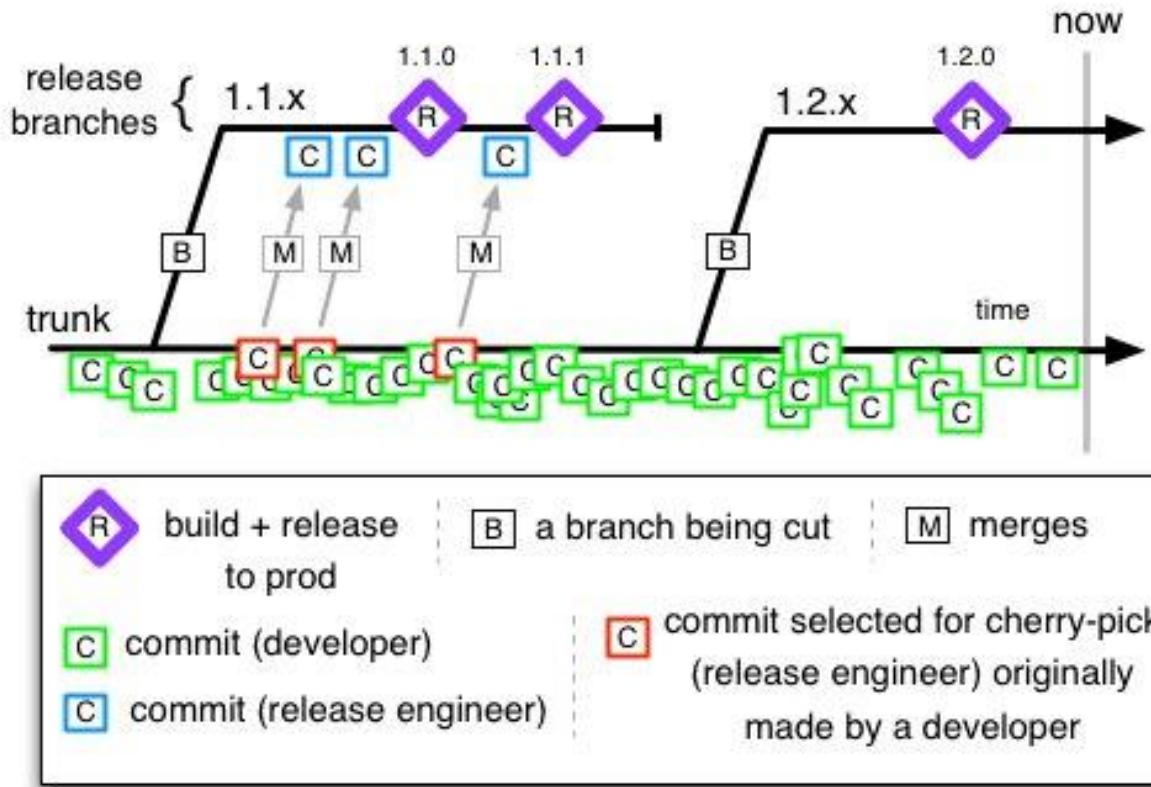


## ➤ Problemy

- konflikty przy scalaniu (merge conflicts)
- rozdzielenie funkcji (feature separation)
- nieprzewidziane wydanie

**Źródło:** Vincent Driessen, A successful Git branching model  
<https://nvie.com/posts/a-successful-git-branching-model/>

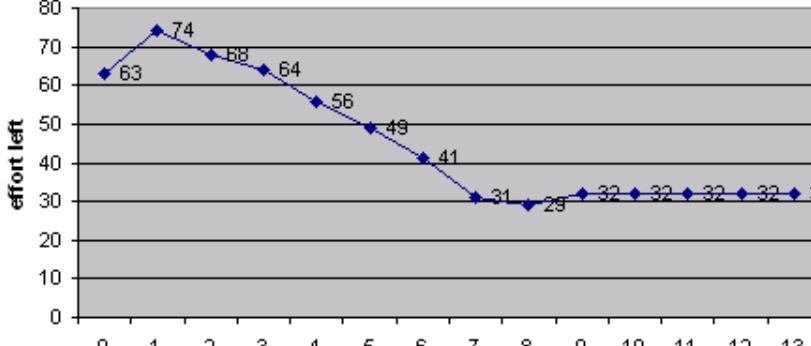
# Zarządzanie kodem – Trunk-Based Development



- Jedna główna gałąź (trunk aka master)
- Brak gałęzi develop i gałęzi dla funkcji (feature)
- Częste scalanie kodu z repozytorium lokalnego na trunk
- Funkcje są wydawane z trunk lub gałęzi wydaniowych

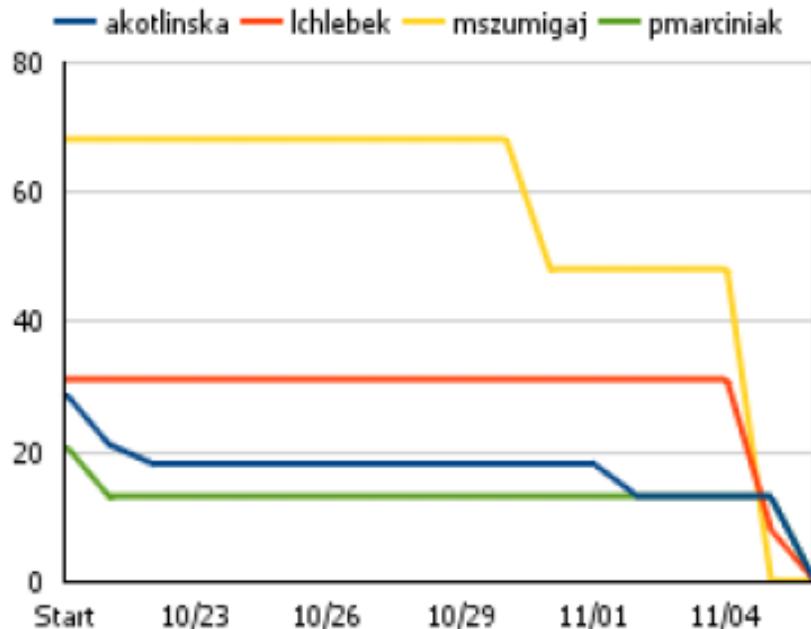
Źródło: <https://trunkbaseddevelopment.com/>

# Nadzór - burndown chart dla sprintu

Sprint 3. Plug in the Real Weather														
Story ID	Story/task	days in sprint / effort left												
		0	1	2	3	4	5	6	7	8	9	10	11	12
10	<b>Fetch one day temperature data from the weather provider system</b> Make our server connect and authenticate to the provider system Read provider's data directory Parse the current temperature out of the data Push the temperature data to the client	63	74	68	64	56	49	41	31	29	32	32	32	32
11	<b>Fetch rain, snow, etc details from the provider</b> Parse snow/rain data from the provider's data Push the snow/rain data to the client Redesign client screen a bit Refactor the server code	4	1	1	1	1	1	1	1	1	1	1	1	1
12	<b>Fetch several days data from the provider</b> Parse the weather data in day packs Push several days data to the client	16	16	16	16	16	16	16	16	16	16	16	16	16
13	<b>Auto-refresh feature</b> Make the client ping server once per 4 hours Make the server update the client	4	4	4	4	4	4	4	4	4	4	4	4	4
Effort left in sprint														
 Effort left: 63, 74, 68, 64, 56, 49, 41, 31, 29, 32, 32, 32, 32, 32														

# Przykład burndown chart dla sprintu 2

To jest stan po zakończeniu sprintu



▼ Sprint Results			
Person	Estimate	Completed	Remaining
akotlinska	29	29	0
Ichlebek	31	31	0
mszumigaj	68	68	0
pmarciniak	21	21	0
Team:	149	149	0

Źródło: P. Marciniak, M. Szumigaj, A. Kotlińska, Ł. Chlebek, MultiScrum - Rozbudowa wybranego narzędzia dla metodyki Scrum o zarządzanie wieloma projektami. Projekt dyplomowy inżynierski, Katedra Inżynierii Oprogramowania, Politechnika Gdańskia, 2012

# Przegląd Sprintu – Demo

 Projekt: Mrowisko

Czas: dzień: 0

Zakres: 2%

Jakość: ███████

Klient:

Zespół:

Punkty: -100

Ryzyko:  20%

Zasoby

Robotnice	10	
Opiekunki	3	
Wartownicy	1	
Zbieracze	6	
Hodowcy	5	

Ryzyka

Niski przyrost naturalny 70% 

Produkt

Komora królowej	10	🟡	2	🟢	30	
Magazyn jedzenia świeżego	6	🟡	1	🟢	40	
Magazyn jedzenia martwego	3	🟡	0	🟢	20	
Wylegarnia zapasowa	8	🟡	0	🟢	20	
Korytarze główne	8	🟡	1	🟢	40	
Korytarze boczne	3	🟡	-2	🟢	30	
Kanaly wentylacyjne	8	🟡	1	🟢	20	
Wejścia do mrowiska	5	🟡	-3	🟢	30	
Izba odpoczynku	1	🟡	2	🟢	20	
Hodowla grzybów	6	🟡	2	🟢	30	
Pokój parady	2	🟡	-2	🟢	10	

Zadania

Wylegarnia wielka (Rob)	10	🟡	1	🟢	34	
Szkoła robotnic (Opi)	6	🟡	0	🟢	8	
Koszary wartowników (War)	4	🟡	0	🟢	9	
Otwory ewakuacyjne (Zbi)	3	🟡	3	🟢	37	

Działania wobec ryzyka

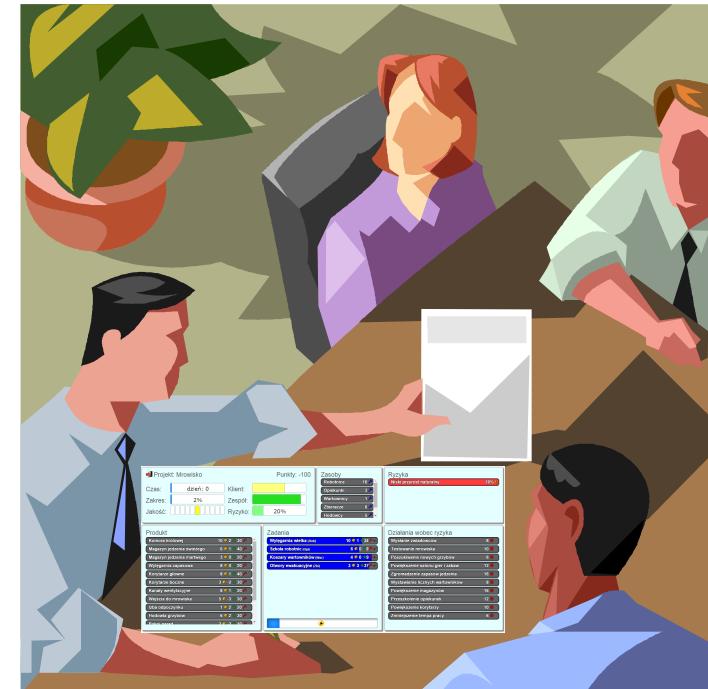
Wysłanie zwiadowców	8	
Testowanie mrowiska	10	
Poszukiwanie nowych grzybów	6	
Powiększenie salonu gier i zabaw	12	
Zgromadzenie zapasów jedzenia	16	
Wystawianie licznych wartowników	8	
Powiększenie magazynów	16	
Przeszkolenie opiekunek	12	
Powiększenie korytarzy	10	
Zmniejszenie tempa pracy	6	

# Przegląd Sprintu – ocena produktu

## ➤ Przegląd sprintu (ang. *Sprint review meeting, Review*)

- Zespół Scrum (w tym Właściciel Produktu) prezentuje Przyrost udziałowcom („na żywo”, slajdy są zabronione) – zebranie informacji zwrotnej
- ocena sprintu względem Celu Sprintu (niekoniecznie realizacja wszystkich elementów z Backlogu Sprintu!)

## ➤ Czas trwania spotkania: maks. 4 godz. dla 4-tygodniowego sprintu



# Przegląd Sprintu – rezultaty

- **Oczekiwania udziałowców:**
  - zmiana funkcji/cechy produktu
  - dodanie nowej funkcji/cechy do produktu
  - usunięcie funkcji/cechy z produktu
- **Właściciel Produktu zapisuje oczekiwania udziałowców w *Backlogu Produktu***
- **Elementy całkowicie niezrealizowane w sprincie wracają do *Backlogu Produktu***
- **Dla elementów zrealizowanych częściowo w sprincie niezrealizowane zadania, a zatem i część *story points* elementu wracają do *Backlogu Produktu***

# Przegląd sprintu – częste błędy

- Prezentacja niedziałającego produktu
- Prezentacja ekranów, slajdów, a nie produktu na żywo
- Niedopuszczenie interesariuszy do interakcji z produktem
- Brak interesariuszy na spotkaniu
- Prezentowanie przyrostu Właścicielowi Produktu
- Skupienie na rozliczaniu zadań z backlogu sprintu zamiast na celu sprintu
- Brak zbierania informacji zwrotnej od interesariuszy

# Retrospektyna – ocena procesu

## ➤ Retrospektyna (ang. *Retrospective, Retro*)

- przegląd procesu, ocena stosowanych praktyk, technik, narzędzi
- identyfikacja elementów do poprawy, wyznaczenie planu poprawy
- odbywa się po przeglądzie sprintu, a przed planowaniem następnego sprintu

## ➤ Cele usprawnienia (adaptacji) procesu

- zwiększenie efektywności pracy
- zwiększenie satysfakcji zespołu
- podniesienie jakości produktu

## ➤ Przykładowa agenda

- sukcesy – co poszło dobrze w sprincie
- problemy – co nie poszło dobrze w sprincie
- pomysły – co zrobić, żeby działać lepiej
- podziękowania

## ➤ Czas trwania spotkania: maks. 3h dla 4-tygodniowego sprintu

# Retrospektyna – action points

## ➤ Zadania z retrospektwy – Action points

- co zespół zamierza zmienić w swojej pracy w wyniku retrospektwy
- konkretne zadania możliwe do zrealizowania w ramach sprintów
- 2-3 zadania wybrane do wykonania w najbliższym sprincie

## ➤ Action points - przykłady

- Dodać nowy rodzaj zgłoszenia do JIRY – Security issue
- Zaprosić na następne sprint review kogoś z działu sprzedaży
- Przenieść Daily na godzinę 9
- Poszukać narzędzia do automatycznego generowania dokumentacji API z kodu
- Pilnować lepiej czasu na Daily

**Uwaga: nie brać zbyt dużo action points do ulepszenia na sprint!**

# Retrospekywa – częste błędy

- **Brak retrospektyw ;)**
- **Skupianie się tylko na negatywach**
- **Szukanie winnych**
- **Monotonne prowadzenie retrospektyw ciągle w ten sam sposób**
- **Niewykorzystywanie retrospektyw do szukania ulepszeń**
- **Brak wniosków z retrospektyw – brak action points**
- **Branie zbyt wielu action points do realizacji w jednym sprincie**
- **Nierealizowanie action points**

# Techniki wspierające retrospektyny

Technika	Ocena osób z doświadczeniem (odchylenie standardowe)	N <sub>d</sub>	Ocena osób bez doświadczenia (odchylenie standardowe)	N <sub>bd</sub>	Wartość p
<i>Starfish/Small Starfish</i>	4.40 (0.83)	27	3.44 (0.93)	27	0.017
<i>The Retrospective Game</i>	Brak ocen	-	2.75 (1.06)	40	b.d.
<i>Legospective</i>	4.00 (1.15)	4	2.37 (1.23)	41	0.035
<i>Mood-curve</i>	4.00 (0.87)	11	3.19 (1.19)	32	0.186
<i>360 degrees appreciation</i>	3.29 (1.20)	14	2.80 (1.29)	35	0.309
<i>Letters to the future</i>	3.29 (0.71)	2	1.85 (1.17)	41	0.035
<i>Hopes and corncerns</i>	3.95 (0.76)	20	4.15 (0.91)	26	0.859
<i>Hot-air balloon</i>	3.82 (0.88)	17	3.52 (1.01)	27	0.023
<i>Hot-air balloon – bad weather</i>	4.00 (1.00)	7	3.41 (1.02)	34	0.093
<i>KALM: Keep, add, more, less</i>	4.03 (0.72)	34	4.05 (0.90)	19	0.6
<i>Following up on action items</i>	4.11 (1.11)	35	3.00 (1.21)	16	0.004
<i>Mad/Sad/Glad</i>	4.04 (0.62)	24	3.58 (0.93)	24	0.032
<i>5Ls(3Ls)</i>	4.29 (0.64)	21	3.52 (1.15)	25	0.006
<i>RetroBox</i>	3.47 (1.15)	17	3.60 (1.39)	33	0.87

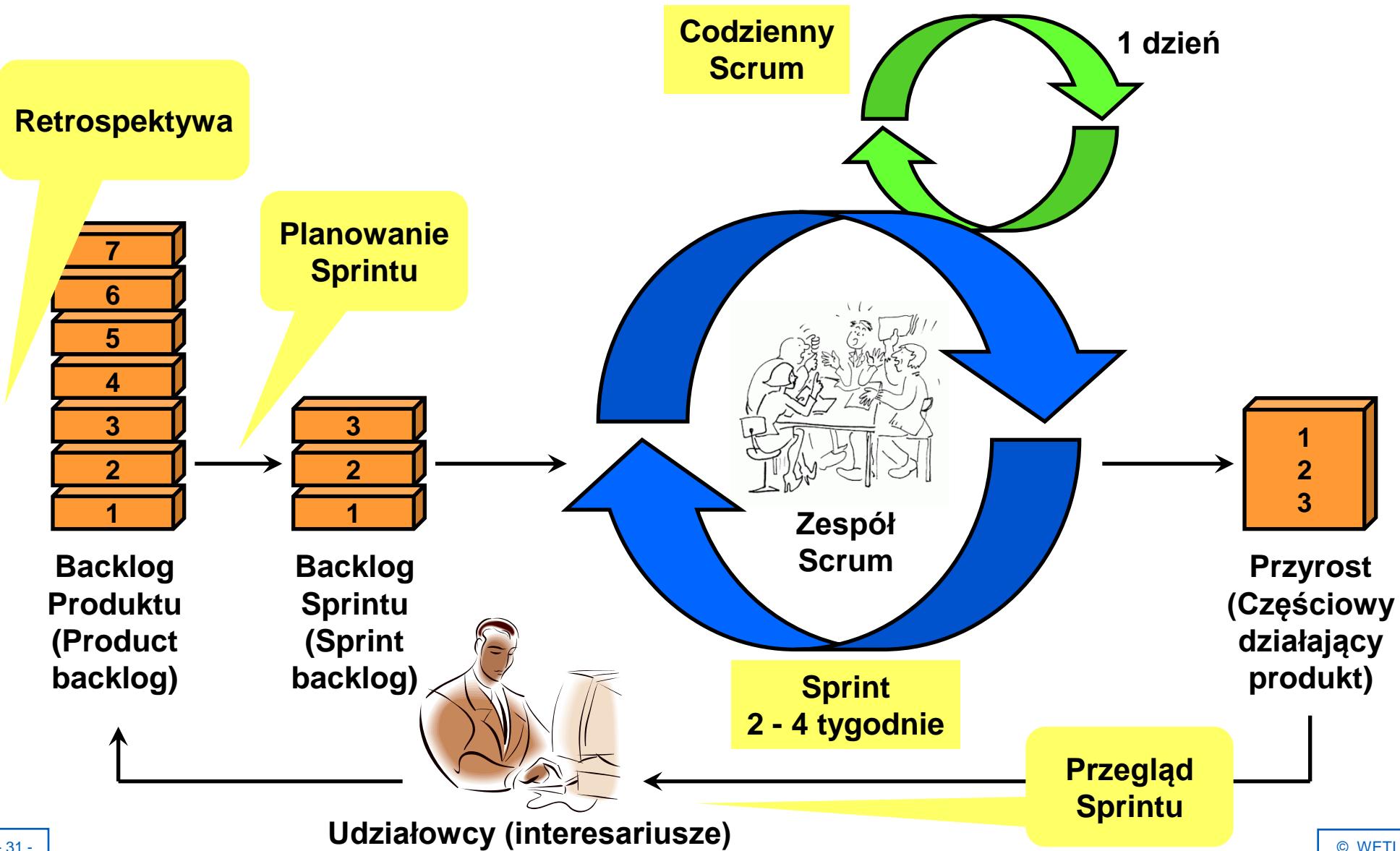
**Źródło:** Aleksandra Dzieciątek, Analiza technik prowadzenia retrospektyw w projektach realizowanych metodą Scrum, praca dyplomowa magisterska, opiekun J. Miler, WETI, PG, 2019.

# Techniki wspierające retrospektwy (2)

Lp	Technika	Średnia ocena satysfakcji (odchylenie standardowe)	N
1	<i>Legospektywa*</i>	4.75 (0.50)	4
2	<i>Following up on action item</i>	4.11 (0.68)	35
3	<i>5Ls (3Ls)</i>	4.10 (0.70)	21
4	<i>Letters to the future*</i> <i>360 degrees appreciation</i> <i>Starfish</i>	4.07 (0.71) (0.92) (0.85)	2 14 27
5	<i>Mad/Sad/Glad</i>	4.04 (1.00)	24
6	<i>KALM – Keep/Add/Less/More</i>	4.03 (0.72)	34
7	<i>Hot-air balloon</i>	4.00 (0.64)	17
8	<i>Hot-air balloon – bad weather*</i>	3.82 (1.00)	7
9	<i>Mood-curve</i>	3.73 (0.79)	11
10	<i>Hopes and concerns</i>	3.70 (0.73)	20
11	<i>Retrobox</i>	3.41 (1.33)	17

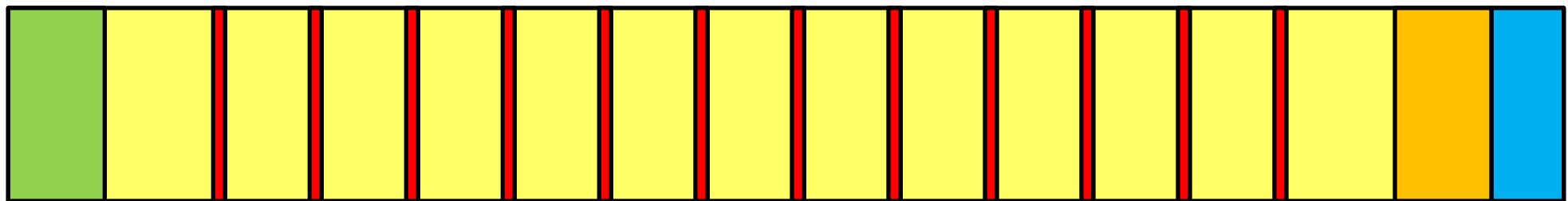
Źródło: Aleksandra Dzieciątek, Analiza technik prowadzenia retrospektyw w projektach realizowanych metodą Scrum, praca dyplomowa magisterska, opiekun J. Miler, WETI, PG, 2019.

# Scrum – wydarzenia



# Wydarzenia - podsumowanie

Sprint np. 2 tygodnie



Planowanie sprintu  
maks. 4h

Codzienny Scrum  
maks. 15 minut

Przegląd sprintu  
maks. 2h

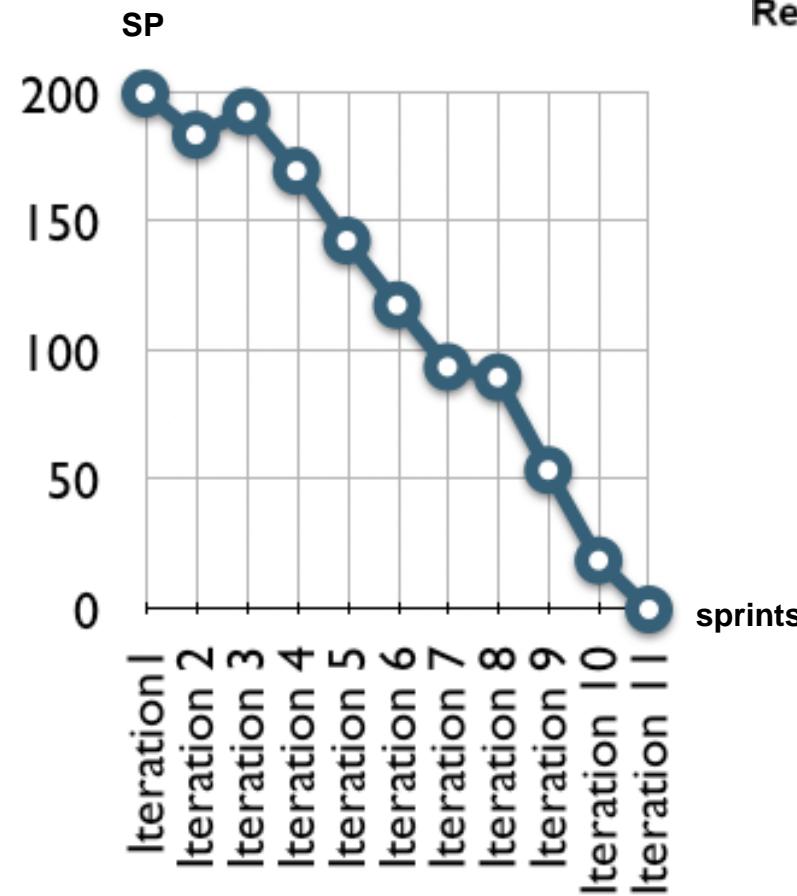
Retrospektwa sprintu  
maks. 1,5h

**Timebox – maksymalny czas**

# Kolejny sprint (w skrócie)

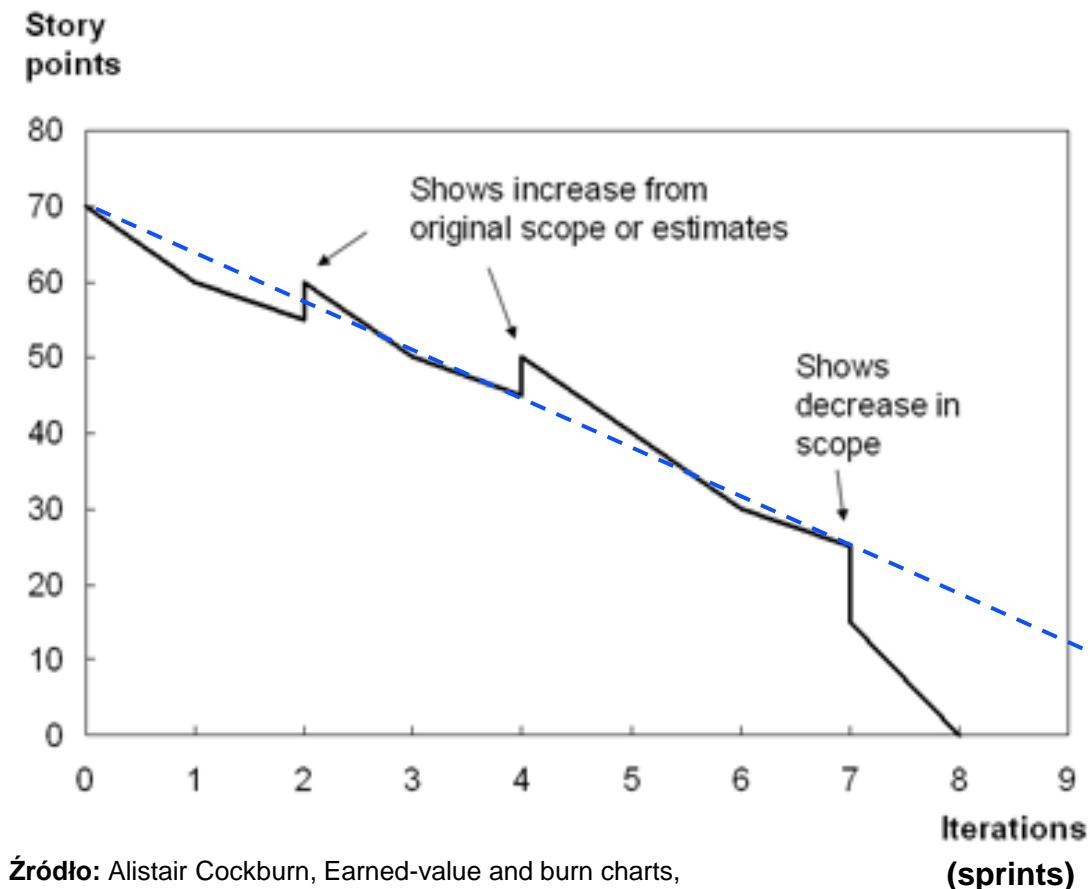
- Ponownie Deweloperzy ustalają z Właścicielem Produktu, które elementy z *backlogu produktu* wejdą z zakres sprintu
- Deweloperzy przygotowują dla siebie *backlog sprintu*, gdzie rozpisują ustalone z Właścicielem Produktu elementy PB na zadania
- Deweloperzy w dniu pracy pobierają zadania z *backlogu sprintu* i je wykonują
- Wykonanie zadania oznaczają wpisując 0 godzin jako oszacowanie pozostałej pracochłonności na kolejne dni sprintu
- Wykonane zadania można oznaczyć kolorem (np. zielonym) w *backlogu sprintu*, a na tablicy zadań przenieść do części „wykonane”
- Zadania są wykonywane aż do zakończenia sprintu – decyduje czas, a nie lista zadań w backlogu sprintu

# Nadzór - burndown chart dla produktu



Źródło: <http://www.mountaingoatsoftware.com/release-burndown>

Release Burndown Chart (showing scope changes)



Źródło: Alistair Cockburn, Earned-value and burn charts,  
<http://alistair.cockburn.us/Earned-value+and+burn+charts>

# Co pokazuje i jak powstaje burndown chart

- Pokazuje liczbę *story points* pozostałych do wykonania po każdym sprintie
- Pozostała liczba *story points* może:
  - zwiększyć się, jeżeli Właściciel Produktu doda elementy do *backlogu produktu* lub Zespół Deweloperski zwiększy oszacowanie istniejących elementów
  - zmniejszyć się, jeżeli Właściciel Produktu usunie elementy z *backlogu produktu* lub Zespół Deweloperski zmniejszy oszacowanie istniejących elementów
- W wersji uproszczonej (poprzedni slajd po lewej) pokazuje się tylko liczbę pozostałych *story points* przed kolejnymi sprintami uwzględniającą elementy wykonane, dodane, usunięte i przeszacowane
- W wersji dokładnej (poprzedni slajd po prawej) pokazuje się liczbę *story points* po zakończeniu każdego sprintu i przed rozpoczęciem następnego sprintu, a różnicę wynikającą z dodania, usunięcia i przeszacowania elementów oznacza się odcinkiem pionowym
  - Niebieską linią przerywaną zaznaczono trend średniej prędkości zespołu po 7 sprintach prognozując ukończenie projektu w ok. 11 sprintach. Właściciel Produktu wycofał ok. 10 SP, Zespół Deweloperski przyspieszył i ukończyli projekt w 8 sprintów.

# Problemy scrumowych zespołów rozproszonych (1)

<b>Id</b>	<b>Problem</b>	<b>Rozwiązanie</b>	<b>Ocena</b>
P1	Problem z komunikacją spowodowany brakiem kontaktu twarzą w twarz	Narzędzia dodatkowe podczas rozmów (shared desktop, interactive whiteboard).	4,14
		Używanie narzędzi do rozmów oraz video konferencji (Skype, Lync, HipChat).	4,09
P2	Problem wynikający z dużej różnicy czasu pomiędzy miejscami, w których znajduje się zespół	Obarczanie rozproszeniem cały zespół – jedni zostają później, ale za to drudzy przychodzą wcześniej.	3,40
P3	Problem z jednolitym stanem wiedzy wszystkich członków zespołu	Gdy odległość nie jest zbyt duża, spotkania 2-3 razy w tygodniu na żywo.	4,12
		Całodzienna komunikacja całego zespołu, stworzenie „wirtualnego pokoju”, rozwiązywanie problemów przez wszystkich członków zespołu, aby każdy wiedział, jak doszło się do rozwiązania.	3,37
P4	Problem braku zaangażowania podczas wirtualnej rozmowy	Interakcyjne prowadzenie Scruma/rozmów przez team leadera. Zadając pytanie, co ktoś o tym myśli, zmusza do bycia na bieżąco w rozmowie.	4,07
		Prowadzenie rozmów wirtualnych z obrazem. Widać, gdy robi się coś innego, dlatego każdy stara się uczestniczyć czynnie w spotkaniu.	3,72
P5	Problem z nieprzekazywaniem wszystkich istotnych informacji	Zapisywanie najważniejszych rzeczy do przekazania i prowadzenie rozmów zgodnie z listą.	4,15
		Oznaczanie osób, które mogą być zainteresowane poruszonym tematem zarówno w dokumentach lub mailu, lepszy przepływ informacji.	3,93

Źródło: Elżbieta Wierzbicka, *Wspomaganie zastosowania Scruma w zespołach rozproszonych*, praca dyplomowa magisterska, opiekun J. Miler, WETI, PG, 2016

# Problemy scrumowych zespołów rozproszonych (2)

<b>Id</b>	<b>Problem</b>	<b>Rozwiążanie</b>	<b>Ocena</b>
P6	Problem ze zrozumieniem siebie nawzajem	Otwarta komunikacja członków zespołu; nie bać się powiedzieć, że czegoś się nie rozumie.	4,50
		Poznanie człowieka na żywo, wizyty w zespołach między sobą co najmniej raz do roku, wyjazdy integracyjne.	4,36
P7	Nieszanowanie dni wolnych przez członków zespołów z innych krajów	Przygotowywanie się wcześniej poprzez sprawdzenie świąt oraz dni wolnych w danym kraju przed ustawieniem spotkań.	4,23
		Stworzenie kalendarza, gdzie każdy z członków zespołu zaznacza swoje święta i dni wolne.	4,22
P8	Problem z różnym zaangażowaniem oraz jakością pracy członków zespołu wynikający z charakteru	Wybranie odpowiedniego team leadera dbającego o pilnowanie zaangażowania i pomaganie osobom, które mają je mniejsze niż reszta.	4,11
		Dawanie różnych zadań, stawianie nowych wyzwań, ale pod ciągłą obserwacją team leadera.	3,50
P9	Problem ze zmniejszonym zaufaniem do innych członków zespołu	Uprzejmość, miłe podejście do ludzi, budowanie dobrych relacji.	4,43
		Wyjazdy oraz poznanie osobiste całego zespołu - budowa zaufania.	4,32
P10	Problem ze spotkaniami z odbiorcą produktu	Częste wyjazdy i spotykanie się z klientem na żywo.	3,98
		Tworzenie dokumentów możliwych do edycji zarówno przez team leadera, jak i odbiorcę produktu.	3,56
P11	Problem związany z różnicą produktywnością członków zespołu	Czujność team leadera na konflikty w zespole powodujące brak zaufania między członkami zespołu i zmniejszanie ich wydajności.	4,38
		Codzienny kontakt z przełożonym i/lub częścią zespołu.	3,77

Źródło: Elżbieta Wierzbicka, *Wspomaganie zastosowania Scruma w zespołach rozproszonych*, praca dyplomowa magisterska, opiekun J. Miler, WETI, PG, 2016

# Realizacja Projektu Informatycznego



***Jakub Miler***

*Katedra Inżynierii Oprogramowania  
Wydział Elektroniki, Telekomunikacji i Informatyki  
Politechnika Gdańskia  
jakubm@eti.pg.edu.pl*

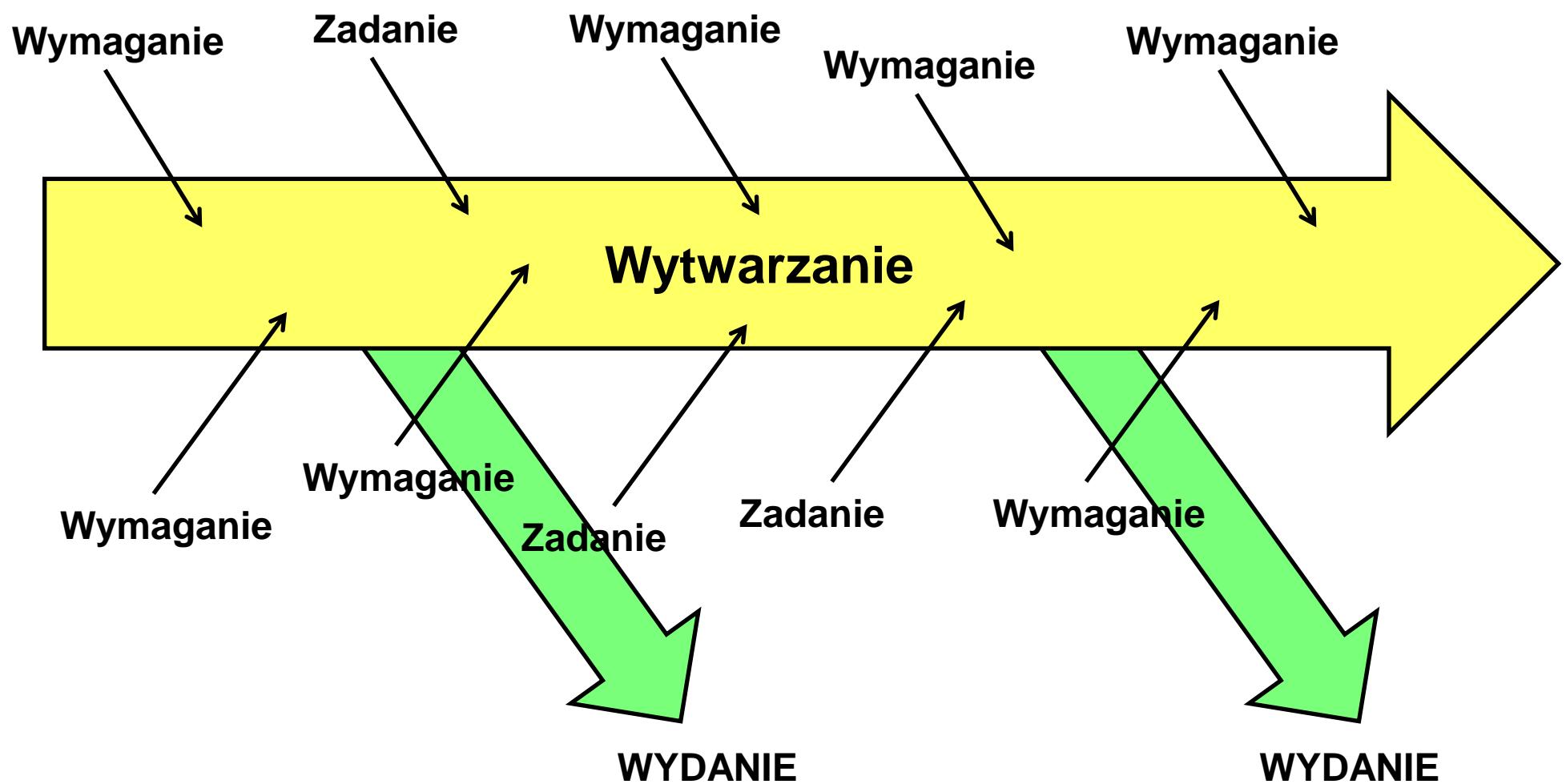
*Materiały pomocnicze do wykładu na Wydziale ETI Politechniki Gdańskiej.  
Wykorzystanie materiałów w innym celu i ich rozpowszechnianie bez zgody WETI PG zabronione.*

## Wykład 5

**Kanban, eXtreme Programming,  
Rational Unified Process**

# Kanban

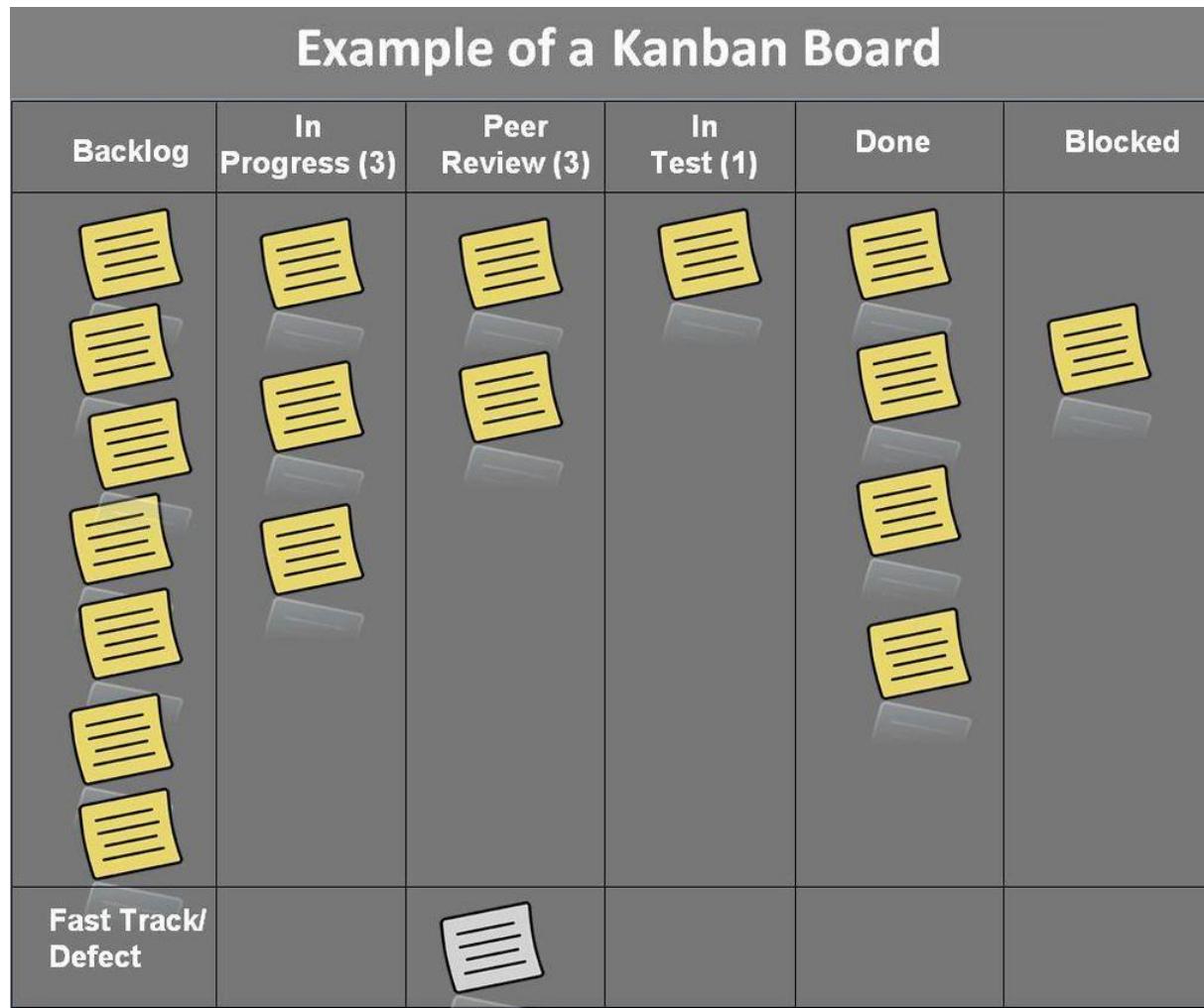
# Proces Kanban



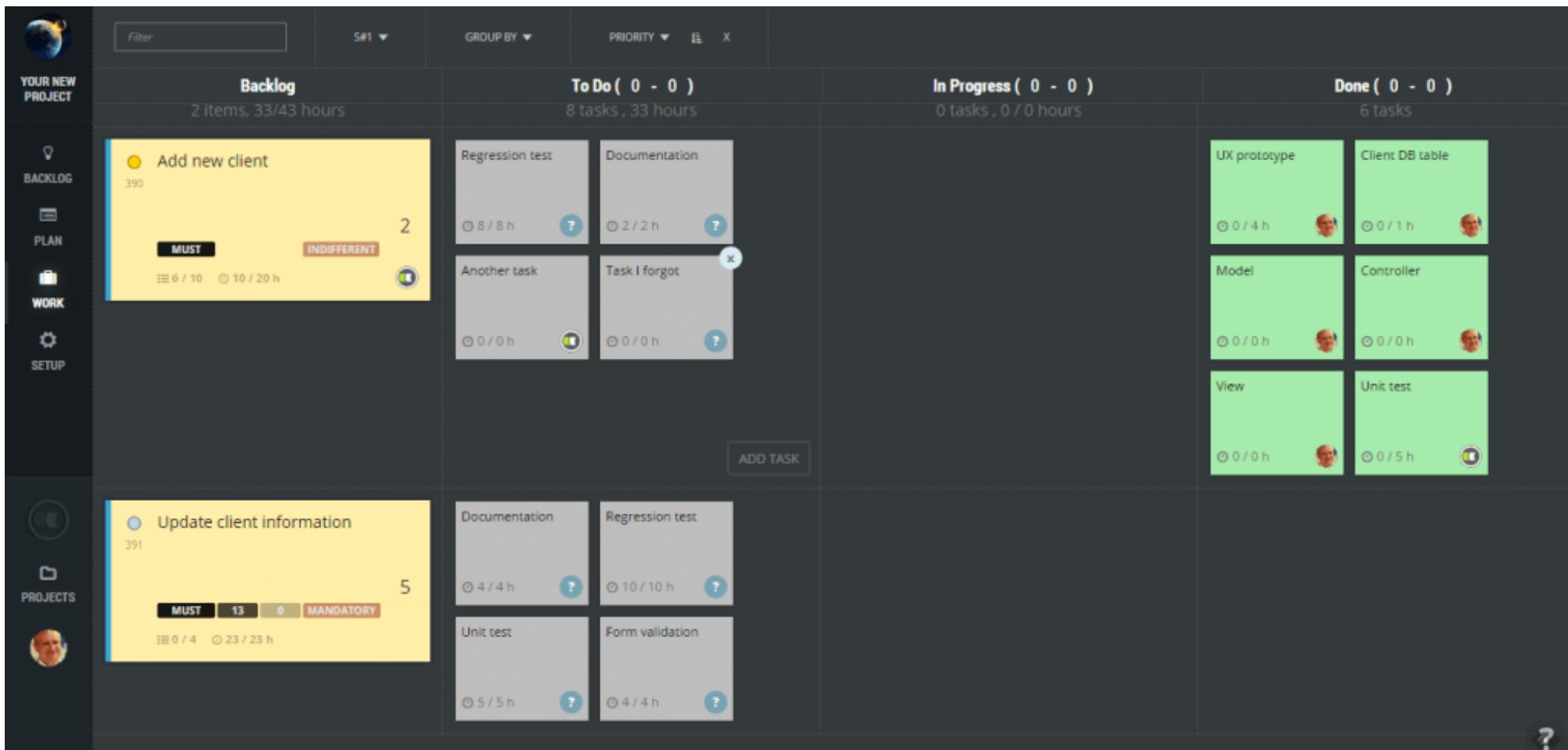
# Zasady Kanban

- **Wizualizacja** – metodyka ta zakłada korzystanie z tablicy oraz wizualizację każdego z działań zachodzącego w zespole. Dzięki wizualizacji można w prosty sposób sprawdzić, na jakim etapie jest projekt oraz zintegrować ze sobą członków zespołu.
- **Limity WIP** – Jest to jedna z cech która odróżnia Kanban od innych metodyk. Zakłada ona, że każdy z działów ma swoją maksymalną ilość zadań które może w danym czasie wykonywać. Realizacja kolejnego zadania jest możliwa, gdy jedno z zadań się zakończy i będzie miało miejsce na wykonanie kolejnego.
- **Zarządzanie przepływem** – Obserwacja i monitoring mierników oraz wskaźników przepływu realizacji zadań, by był on jak najbardziej efektywny
- **Transparentność zasad** – Zasady powinny się charakteryzować prostotą. Każdy z uczestników projektu oraz interesariuszy nie powinien mieć problemów z ich interpretacją
- **Efektywne kanały informacji zwrotnej** - Polega na przekazywaniu maksymalnie szybko oraz treściwie informacji, która ma za zadanie jak najszybciej wychwycić wszelkie nieprawidłowości w procesie
- **Kaizen** – Wszystkie zmiany w procesie powinny być dodawane w sposób ciągły. Proces powinien podlegać ciągłemu doskonaleniu. Celem jest osiągnięcie maksymalnych wyników (M. Konieczny 2014, s. 6)

# Tablica Kanban



# Tablica Kanban w ScrumDesk



The screenshot shows a Kanban board interface from ScrumDesk. The board is divided into four main columns: Backlog, To Do, In Progress, and Done.

- Backlog:** Contains 2 items, estimated at 33/43 hours. One item, "Add new client", is highlighted in yellow and has a priority of 2. It is categorized as MUST and INDIFFERENT. Another item, "Update client information", is highlighted in blue and has a priority of 5. It is categorized as MUST, 13, 0, and MANDATORY.
- To Do:** Contains 8 tasks, estimated at 33 hours. Tasks include "Regression test", "Documentation", "Another task", and "Task I forgot".
- In Progress:** Contains 0 tasks, 0/0 hours.
- Done:** Contains 6 tasks. Tasks include "UX prototype", "Client DB table", "Model", "Controller", "View", and "Unit test".

On the left side, there is a sidebar with navigation icons for BACKLOG, PLAN, WORK, SETUP, and PROJECTS. At the bottom right, there is a help icon (question mark). At the bottom center, there is a button labeled "ADD TASK".

# Porównanie Scruma i Kanbana

## Czynniki sukcesu lepiej wspierane przez Scruma

ID	Nazwa	$\bar{S}$
15.1	Organizowanie spotkań, podczas których zespół ma możliwość opowiedzenia o problemach i sukcesach z ostatniego czasu, a także opracowania planu usprawnień na kolejnych kilka tygodni pracy	4,51
1.1	Organizowanie codziennych spotkań zespołu	4,39
12.1	Komunikacja bezpośrednią, jeśli to możliwe	4,27
8.1	Zalecane 3-6, a maksymalnie ok. 20 członków zespołu	4,16
14.1	Praca lidera i zespołu w jednym miejscu	4,13
9.1	Lider zespołu interesujący się zespołem i jego postępami w pracy nad projektem, nienarzucający swojego zdania	4,09
4.2	Dokładne szacowanie zadań i jedna miara szacowania w całym projekcie	3,83

## Czynniki sukcesu lepiej wspierane przez Kanbana

ID	Nazwa	$\bar{K}$
14.2	Korzystanie z tablicy z zadaniami	4,60
13.1	Aktualne informacje zawieszone na ścianach - "radiatory informacji"	4,39
15.2	Praca w parach	3,64

**Źródło:** Natalia Kowalik, Analiza porównawcza organizacji zespołu w projektach informatycznych realizowanych metodami Scrum i Kanban, praca dyplomowa magisterska, opiekun J. Miler, WETI, PG, 2018.

# eXtreme Programming - XP

# Wybrane praktyki eXtreme Programming

- **Testy przed kodem**
- **Prosty projekt**
- **Refaktoryzacja**
- **Współwłasność kodu**
- **Programowanie w parach**
- **Standardy programowania**
- **Ciągła integracja**

# Testy przed kodem

- **Testy funkcjonalne**
  - przypadki testowe opisujące zachowanie aplikacji z punktu widzenia użytkownika
  - stanowią doprecyzowanie wymagań na produkt
  - trudne do automatyzacji
- **Testy jednostkowe**
  - testy zachowania poszczególnych klas
  - specyfikują zachowanie klasy
  - automatyzowane np. JUnit
- **Przypadki testowe dla testów funkcjonalnych pisane są przed rozpoczęciem projektowania**
- **Testy jednostkowe pisane są przed rozpoczęciem programowania klas**
- **Klasa jest zatwierdzana do repozytorium kodu tylko gdy przejdzie wszystkie testy jednostkowe**
- **Praktyka stanowi podstawę procesu Test Driven Development (TDD)**

# Przypadki testowe jako specyfikacja

<b>Test Case ID</b>	CutCopyPaste_ClipboardDetails
<b>Author</b>	Jakub Miler, Jakub Czyżnikiewicz
<b>Feature being tested</b>	Displaying different node details in “Node Clipboard” window
<b>Procedure</b>	<p><u>Initialization:</u>  Check if login page is displayed (see SystemFunctions_Login ).</p> <p><u>Actions:</u></p> <ol style="list-style-type: none"> <li>1. Log-in as admin/developer/viewer/assessor (project roles) (see SystemFunctions_Login).</li> <li>2. Open the first project.</li> <li>3. Add a new “information” node under “information” root node.</li> <li>4. Select the node and give it title as follows (after every change execute step 5 and 6): <ol style="list-style-type: none"> <li>a. “title”,</li> <li>b. non-empty string with HTML tags,</li> <li>c. non-empty string with HTML special characters (‘&amp;euro;’, ‘&amp;copy;’),</li> <li>d. “&lt;abcd&gt;”,</li> <li>e. “,.\\;’”[]{}-=+_!@#\$%^&amp;*()”,</li> <li>f. very long title (more than 500 characters).</li> </ol> </li> <li>5. Execute “Cut” option.</li> <li>6. Check displayed data in Clipboard.</li> <li>7. Delete added “information” node.</li> </ol> <p><u>Finalization:</u>  Do the same what in the “Initialization” step.</p>
<b>Intended result</b>	“title” is displayed literally, HTML tags aren’t applied to text, HTML special characters are interpreted, “<abcd>” shows as ‘< abcd>’, “,.\\;’”[]{}-=+_!@#\$%^&*()” shows literally, very long title is displayed in whole and causes vertical scrollbar to appear in window. For Viewer, Assessor: System doesn’t allow using this function.

# Testy przed kodem (3)

➤ Testowany kod:

```
public class MathOps {  
    public static double average (List l)  
    {  
        Iterator iter = l.iterator();  
        double sum = 0;  
        while (iter.hasNext())  
        {  
            Integer num = (Integer) iter.next();  
            sum += num.intValue();  
        }  
        return sum/l.size();  
    }  
}
```

➤ Test jednostkowy JUnit:

```
public class MathOpsTest extends public TestCase  
{  
    public void test_average_multiple ()  
    {  
        Vector nums = new Vector();  
        nums.add(new Integer(3));  
        nums.add(new Integer(6));  
        assertTrue(MathOps.average(nums) == 4.5);  
    }  
}
```

# Prosty projekt

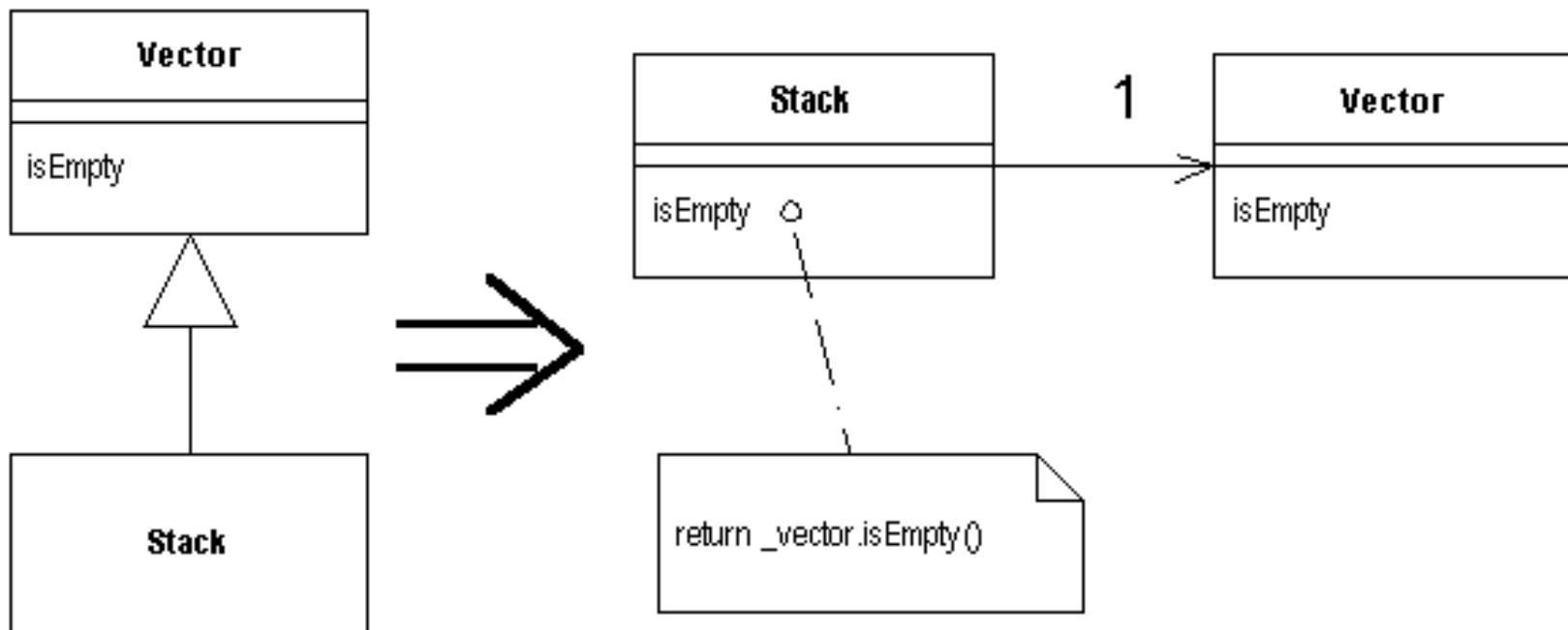
- Kod poprawnie przechodzi wszystkie testy jednostkowe
- Kod zawiera wszystkie potrzebne klasy dla realizacji funkcjonalności
- Nie ma powtórzeń kodu
- Nie ma niepotrzebnych klas i metod
- Nie wytwarzać kodu „na zapas” – zasada YAGNI (*You Aren’t Gonna Need It*)
- Jeżeli projekt przestaje być prosty → refaktoryzacja

# Refaktoryzacja

- Poszukiwanie w kodzie znamion „nieprostego” lub po prostu złego projektu – tak zwanych „złych zapachów” (ang. *bad smells* lub *code smells*)
- Zmiana struktury kodu poprzez kolejne „ściśle określone” przekształcenia
- Przykłady złych zapachów:
  - Powtórzony kod – identyczny lub bardzo podobny kod w wielu miejscach
  - Wielka metoda – metoda jest bardzo duża, powinna być rozbita na mniejsze
  - Wielka klasa – klasa skupia zbyt wiele zachowania (powstaje „obiekt bóg”)
  - Zazdrość o funkcjonalność – klasa wykorzystuje bardzo mocno metody innej klasy
  - Niepoprawna zażyłość – klasa zależy od szczegółów wewnętrznych innej klasy
  - Odrzucony spadek – klasa dziedzicząca nie zachowuje interfejsu klasy rodzica
  - Leniwa klasa – klasa jest bardzo mała
  - Klasa-dane – klasa przechowuje jedynie dane (ma jedynie atrybuty, getery i setery)
  - Łańcuchy komunikatów – klasa musi wywołać metodę innej klasy by otrzymać obiekt, następnie wywołać metodę tego obiektu by otrzymać kolejny obiekt itd.
  - Nieużywany kod – kod (np. metoda), który nie jest używana przez pozostały kod

# Refaktoryzacja (2)

- **Zły zapach: Odrzucony spadek**
- **Przekształcenie: Zastąp dziedziczenie delegowaniem**
- **Utwórz atrybut dla klasy rodzica, zmień metody by delegowały odpowiedzialność do nadklasy i usuń dziedziczenie**



# Współwłasność kodu

- Każdy członek zespołu może modyfikować dowolny fragment kodu: wprowadzać nową funkcjonalność, usuwać błędy, refaktoryzować
- Kod ma autora (współautorów), a nie właściciela
- Nie trzeba czekać na pierwszego autora, by wprowadził zmiany do „swojego” kodu
- Każdy kod ma testy jednostkowe, jeżeli zmiana kodu tego wymaga, trzeba też zmienić kod testu (i go wykonać)
- Pojedyncze osoby mogą odejść z zespołu, lecz wiedza na temat projektu i kodu jest rozproszona w zespole i nie odchodzi wraz z człowiekiem
- Osoba pracująca z „cudzym” kodem łatwiej zauważyci możliwości ulepszenia kodu (podniesienia jakości) przez refaktoryzację

# Programowanie w parach

- Programiści dobierają się w pary i pracują przy jednym komputerze
  - jedna osoba pełni rolę *programisty* – pisze kod
  - druga osoba pełni rolę *projektanta, konsultanta, przeglądającego* – siedzi obok i rozmawia, podpowiada, wyszukuje błędy, pomaga projektować, podsuwa możliwości refaktoryzacji
  - role mogą się zmieniać: klawiatura i myszka przechodzi z rąk do rąk
- Założenie: to się opłaca. Przecież dwa razy więcej osób pisze tę samą ilość kodu w jednostce czasu. Ale lepszego kodu i zysk przychodzi później.
- Rozmawianie umila pracę (bardzo wielu osobom, choć nie wszystkim)
- Pary zmieniają się często, co pozwala szybko rozprowadzić i uspójnić wiedzę na temat projektu w zespole
- Praca w parach ma walor edukacyjny: nowych lub mniej doświadczonych członków zespołu łączymy z bardziej doświadczonymi

# Czy programowanie w parach się opłaca?

**Table E-6 Pair Programming Results**

Study	Effects of Practice on				Baseline for Comparison	Length (Hrs.)
	Effort	Schedule	Defect Rate	Satisfaction		
<b>Pair Programming</b>						
Nosek 1998 <sup>30</sup>	+43%	-29%		High	Experiment vs. non-pairs	0.75
Williams 2000 <sup>31</sup>	+15%	-43%	-60%	High	Experiment vs. non-pairs	>10
Baheti 2002 <sup>32</sup>	+2%	-49%			Experiment: collocated	12–16
Baheti 2002	+14%	-43%			Experiment: distributed	12–16
Ciolkowski 2002 <sup>33</sup>	+9%	-46%		High	Experiment vs. non-pairs	14
Nawrocki 2001 <sup>34</sup>	+82%	-9%			Experiment vs. non-pairs	1–4
Rostaher 2002 <sup>35</sup>	+98%	-1%		High	Experiment: mixed PP-test	6
Arisholm 2002 <sup>36</sup>	+96%	-2%	-30%		Experiment vs. non-pairs	1

Źródło: B. Boehm, R. Turner, *Balancing Agility And Discipline. A Guide for the Perplexed*, Addison Wesley, 2004

# Standardy programowania

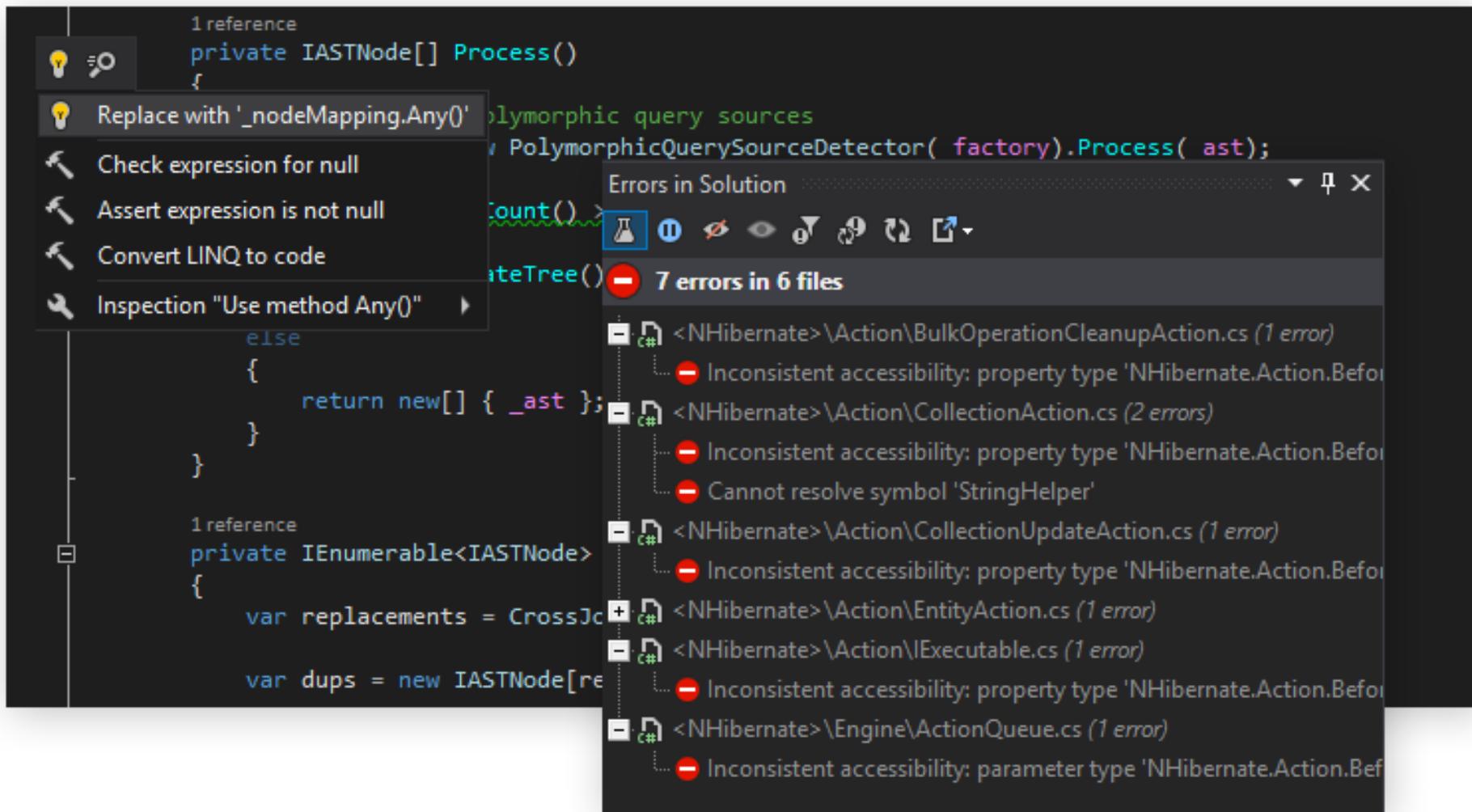
- Ustalone w zespole wspólne zasady pisania kodu obejmujące:
  - formatowanie kodu
  - komentarze
  - nazewnictwo klas, metod, zmiennych, pakietów
  - inne
- „Możesz pisać kod jak tylko chcesz, lecz nie w zespole” [Kent Beck]
- Kod służy (również) komunikacji w zespole na temat systemu
- Kod jest nośnikiem informacji o projekcie systemu, musi być zrozumiały dla wszystkich w zespole
- Standardy programowania są konieczne przy współwłasności kodu, programowaniu w parach i refaktoryzacji
- Standard komentowania pozwala automatycznie wygenerować dokumentację np. Javadoc

# Standardy programowania (2)

## Komentarz Javadoc

```
/**  
 * Returns an Image object that can then be painted on the screen.  
 * The url argument must specify an absolute {@link URL}. The name  
 * argument is a specifier that is relative to the url argument.  
 * <p>  
 * This method always returns immediately, whether or not the  
 * image exists. When this applet attempts to draw the image on  
 * the screen, the data will be loaded. The graphics primitives  
 * that draw the image will incrementally paint on the screen.  
 *  
 * @param url an absolute URL giving the base location of the image  
 * @param name the location of the image, relative to the url argument  
 * @return the image at the specified URL  
 * @see Image  
 */  
public Image getImage(URL url, String name) {  
    try {  
        return getImage(new URL(url, name));  
    } catch (MalformedURLException e) {  
        return null;  
    }  
}
```

# Czysty kod - ReSharper

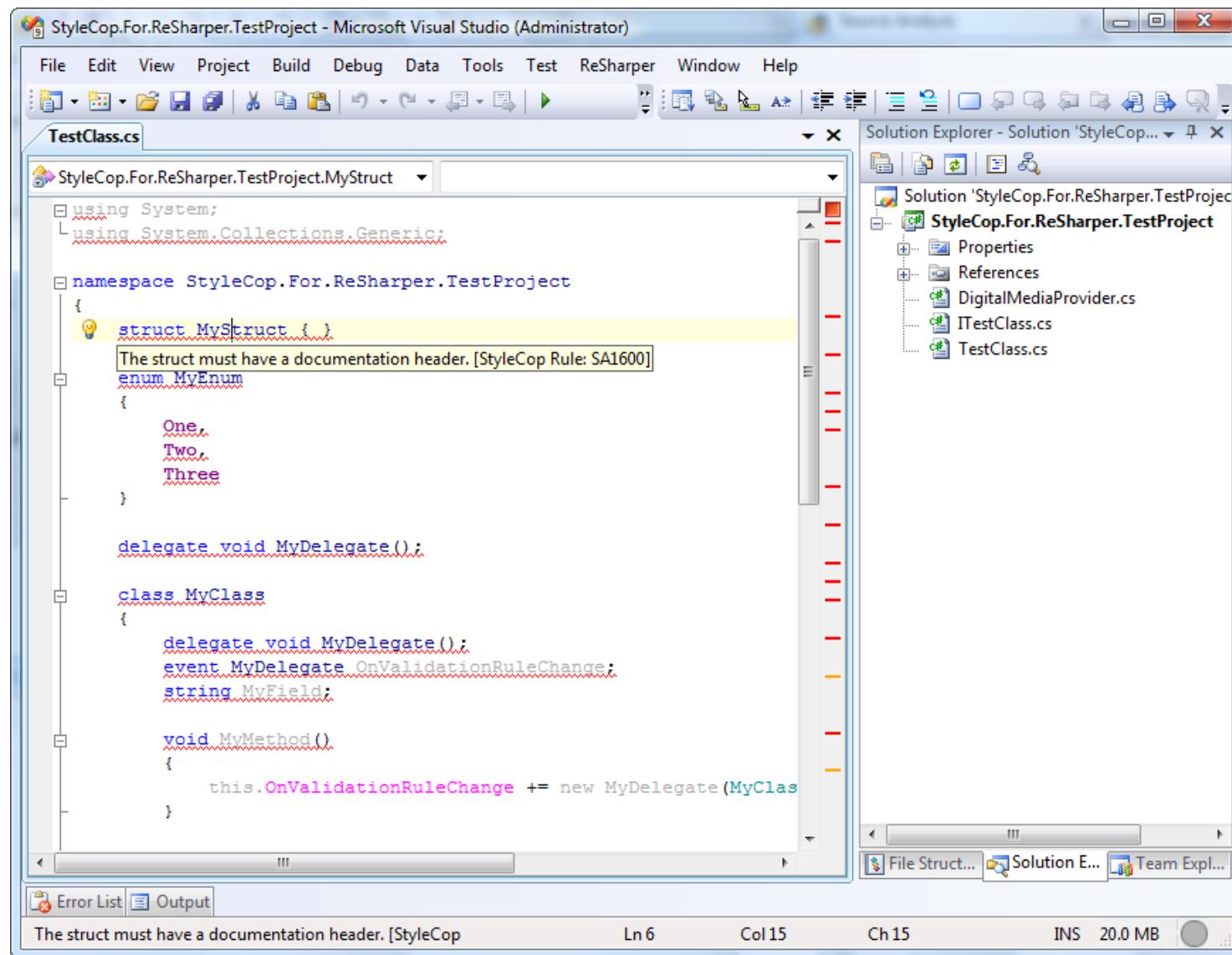


The screenshot shows the ReSharper IDE interface. On the left, there's a code editor with C# code. A context menu is open over a line of code, listing several inspection options: "Replace with '\_nodeMapping.Any()", "Check expression for null", "Assert expression is not null", "Convert LINQ to code", and "Inspection "Use method Any()"".

The main window displays the results of an inspection. At the top right, a status bar says "Errors in Solution" with "7 errors in 6 files". Below this is a tree view of errors across six files:

- <NHibernate>\Action\BulkOperationCleanupAction.cs (1 error)
  - Inconsistent accessibility: property type 'NHibernate.Action.Before
- <NHibernate>\Action\CollectionAction.cs (2 errors)
  - Inconsistent accessibility: property type 'NHibernate.Action.Before
  - Cannot resolve symbol 'StringHelper'
- <NHibernate>\Action\CollectionUpdateAction.cs (1 error)
  - Inconsistent accessibility: property type 'NHibernate.Action.Before
- <NHibernate>\Action\EntityAction.cs (1 error)
- <NHibernate>\Action\Executable.cs (1 error)
  - Inconsistent accessibility: property type 'NHibernate.Action.Before
- <NHibernate>\Engine\ActionQueue.cs (1 error)
  - Inconsistent accessibility: parameter type 'NHibernate.Action.Befor

# Czysty kod - StyleCop



The screenshot shows the Microsoft Visual Studio interface with the title bar "StyleCop.For.ReSharper.TestProject - Microsoft Visual Studio (Administrator)". The main window displays the file "TestClass.cs". A tooltip is visible over the "MyStruct" struct definition, stating: "The struct must have a documentation header. [StyleCop Rule: SA1600]". The Solution Explorer on the right shows the project structure:

- Solution 'StyleCop.For.ReSharper.TestProject'
  - StyleCop.For.ReSharper.TestProject
    - Properties
    - References
    - DigitalMediaProvider.cs
    - ITestClass.cs
    - TestClass.cs

The status bar at the bottom shows the message "The struct must have a documentation header. [StyleCop]" and coordinates Ln 6 Col 15 Ch 15 INS 20.0 MB.

# Ciągła integracja

- Przekazywanie kodu do repozytorium jak najczęściej i synchronizowanie lokalnej kopii roboczej z repozytorium
- Automatyczna integracja kodu – skrypty integracyjne
- Po każdej integracji uruchamiane są testy
- Jeżeli jakiś test przestał działać poprawnie, wiadomo, w której wersji kodu wprowadzono błąd
- Pozwala uniknąć „piekła integracji”, gdy próbujemy zintegrować kod wytwarzany niezależnie na końcu projektu
- Narzędzia wspomagające ciągłą integrację, np. Jenkins, QuickBuild

# Ciągła integracja - QuickBuild

qb Dashboards Grid Queue

**Default**

**All Projects**

	#Builds	#Requests	Latest build
Demo	0	0	No builds
Build Setup Tutorials	0	0	No builds
Custom Stats Example	1038	0	1.0.1038 ▾ 2 days ago ( 2s )
DotNet Report Examples	372	0	1.0.372 ▾ 2 days ago ( 4s )
JIRA Test	258	0	1.0.257 ▾ 1 day ago ( 1s )
Java Report Examples	631	0	1.0.734 ▾ 3 weeks ago ( 18s )
Maintenance	0	0	No builds
Open Source Projects	0	0	No builds
Organize By Departments	0	0	No builds
Organize By Projects	0	0	No builds
RestTest	1	0	1.0.0 ▾ 3 months ago ( 229 ms )

**Sample Artifacts :: 3.2.851**

Name	Size	Last Modified
LICENSE.txt	11.29 KB	2017-03-07 09:11:53
NOTICE.txt	187 bytes	2017-03-07 09:11:53
README.txt	1.59 KB	2017-03-07 09:11:53
RELEASE-NOTES.html	1.43 KB	2017-03-07 09:11:53
commons-collections-3.2.851.jar	578.47 KB	2017-03-07 09:11:53

[Welcome!](#) Guest | [Sign In](#) | [Help](#)

[Get QuickBuild for Free!](#)

search builds
🔍

**JUnit Tests (whole site)**



Passed	39,918 (99%)
Errors	30 (0%)
Failures	46 (0%)
Skipped	247 (1%)

**Cobertura Overview**

Lines Coverage	<div style="width: 30%; background-color: red;"></div> 30%
Branches Coverage	<div style="width: 19%; background-color: red;"></div> 19%
Complexity	2.37
Lines	6,033 detected / 20,073 total
Branches	1,566 detected / 8,249 total

[View Report](#)

**Server JVM Measurements**

Memory Usage
Heap Usage
GC Runs
GC Time
Thread Count

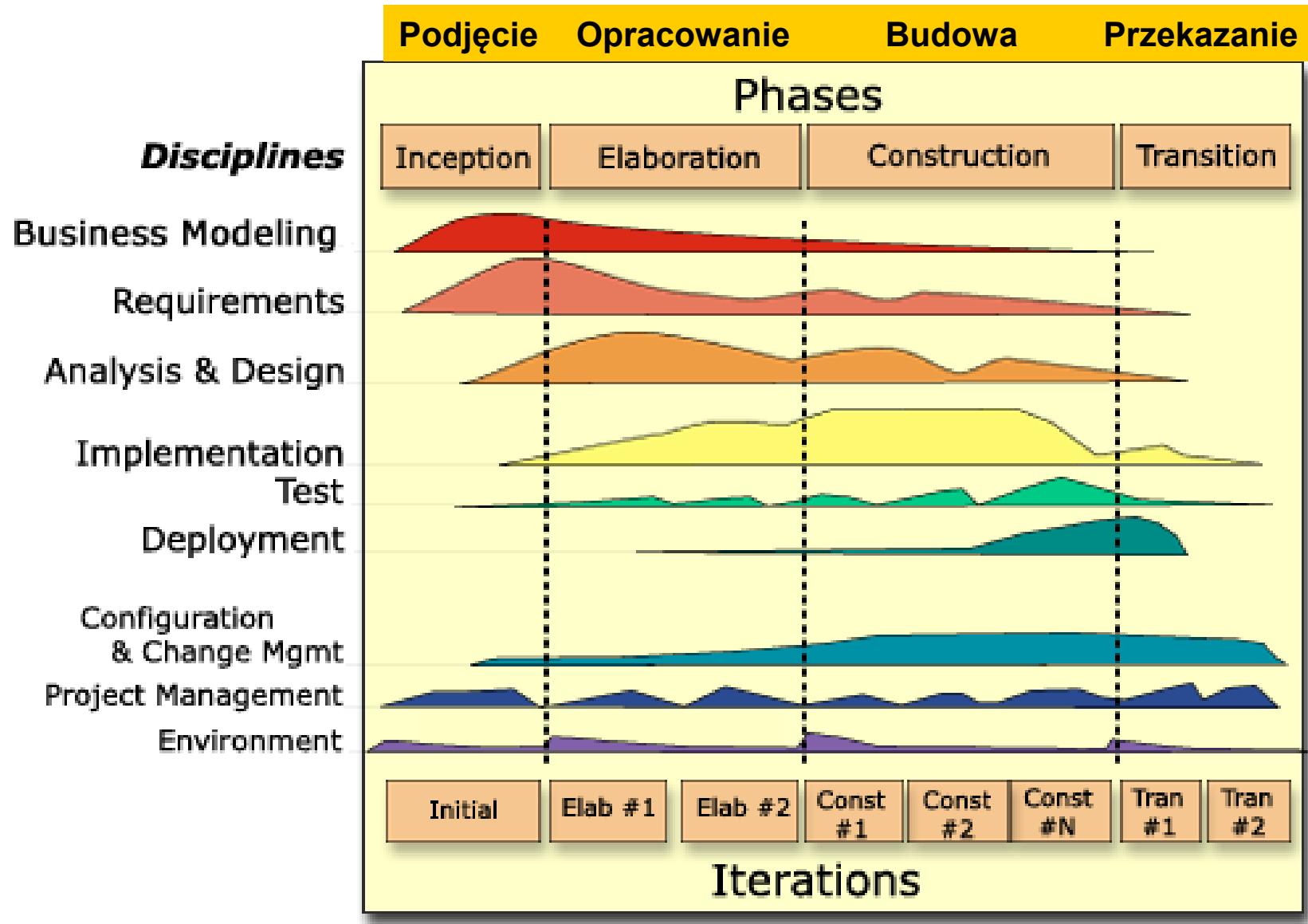
Źródło: QuickBuild Demo, PMEase, <http://demo.pmease.com/>

- 25 -

© WETI PG

# Rational Unified Process

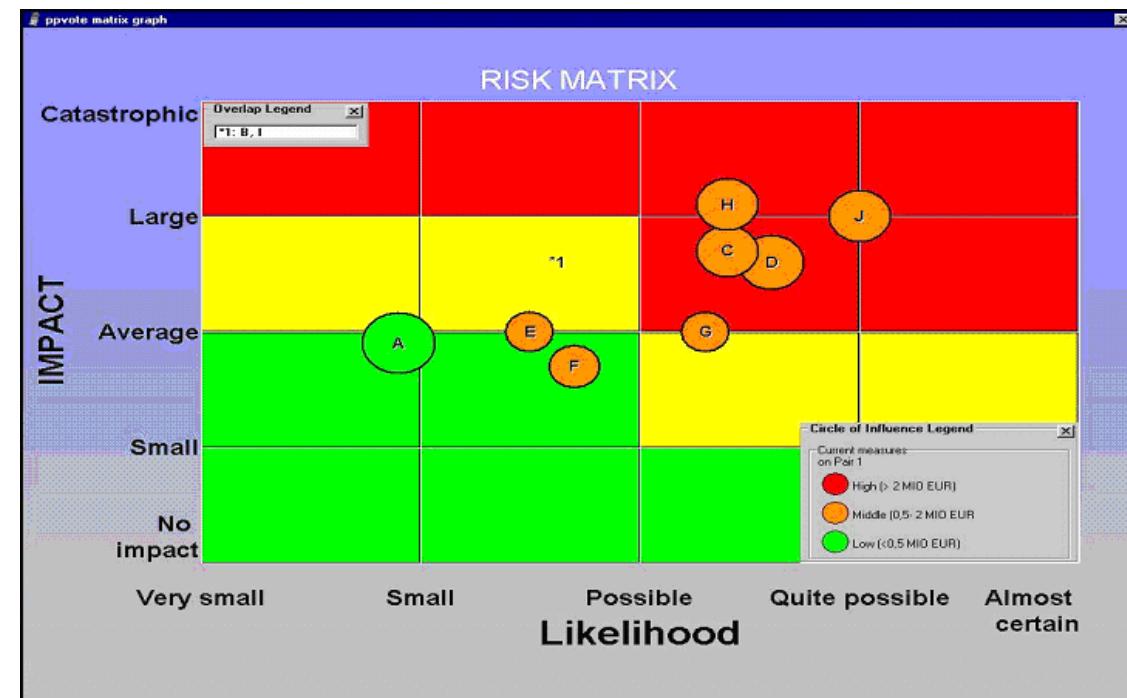
# RUP – widok ogólny



# Proces RUP sterowany ryzykiem

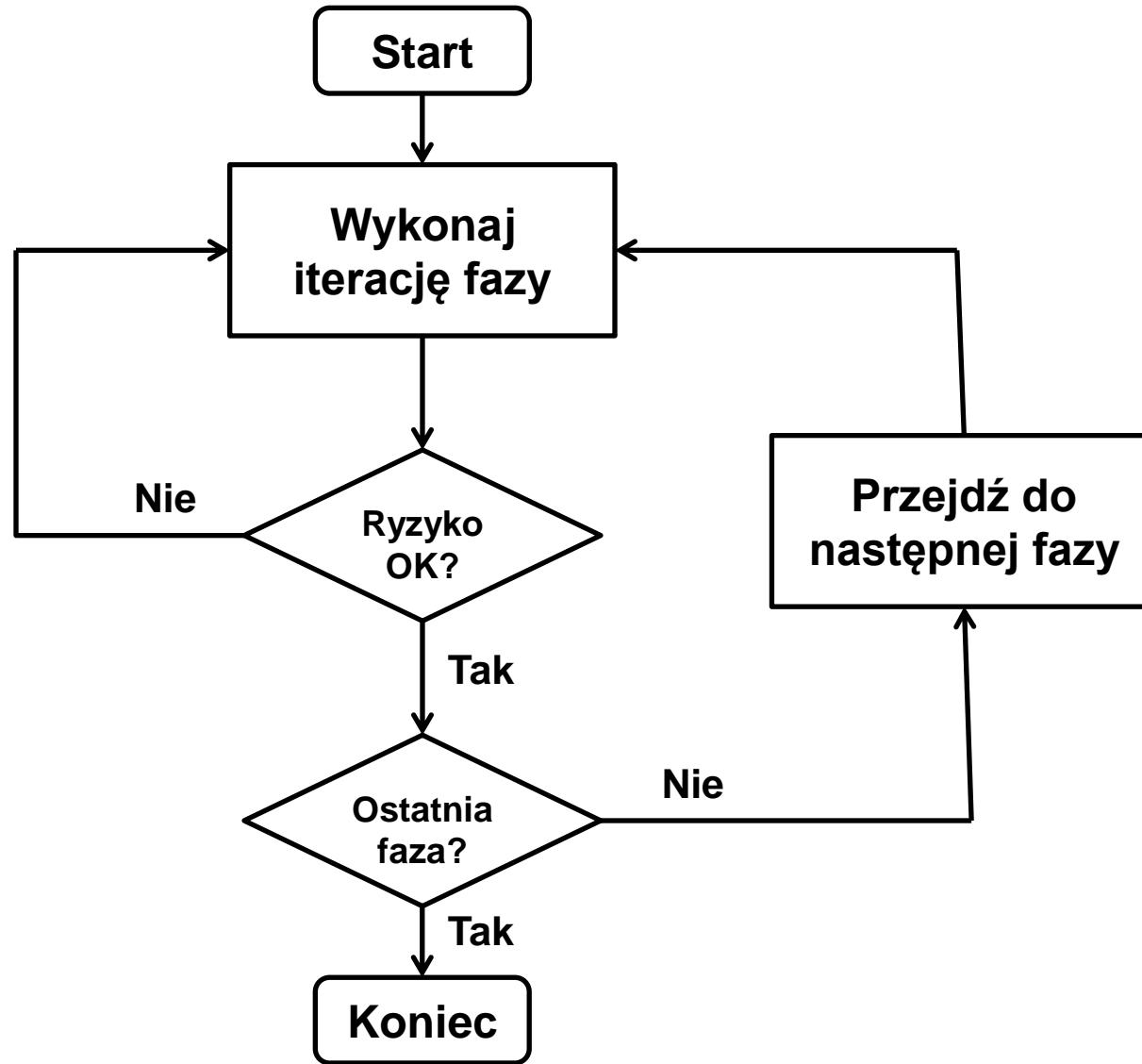
- Ryzyko – możliwość wystąpienia sytuacji o negatywnych skutkach, zmniejszającej sukces projektu/etapu

- Celem każdej fazy jest obniżenie jakiegoś rodzaju ryzyka
- Praca w danej fazie jest tak pomyślana, aby to ryzyko obniżyć



- Faza kończy się, kiedy ryzyko jest akceptowalne

# RUP – iteracyjny proces sterowany ryzykiem



# Fazy – kluczowe hasło i obniżane ryzyko

## ➤ Inception - ZAKRES

obniża ryzyko błędного zakresu produktu (pominięcie istotnych udziałowców i funkcji systemu)

## ➤ Elaboration – ARCHITEKTURA I TECHNOLOGIA

obniża ryzyko błędnego doboru architektury i technologii produktu (rozwiązanie techniczne nie daje produktu wymaganej jakości)

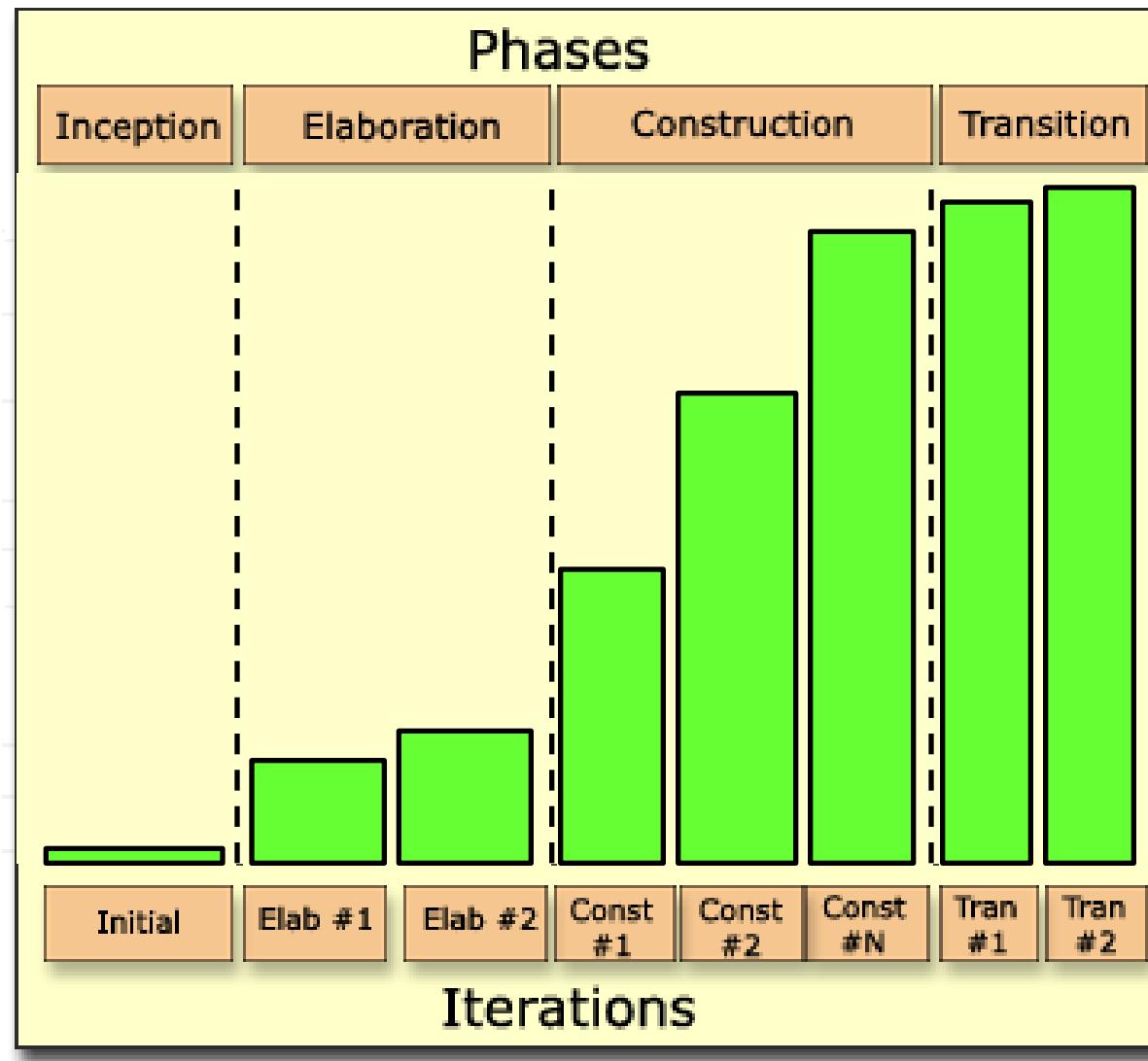
## ➤ Construction - PRODUKCJA

obniża ryzyko przekroczenia terminu i budżetu, niskiej jakości i niewłaściwego zakresu produktu

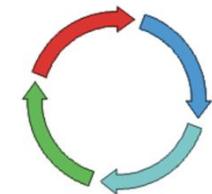
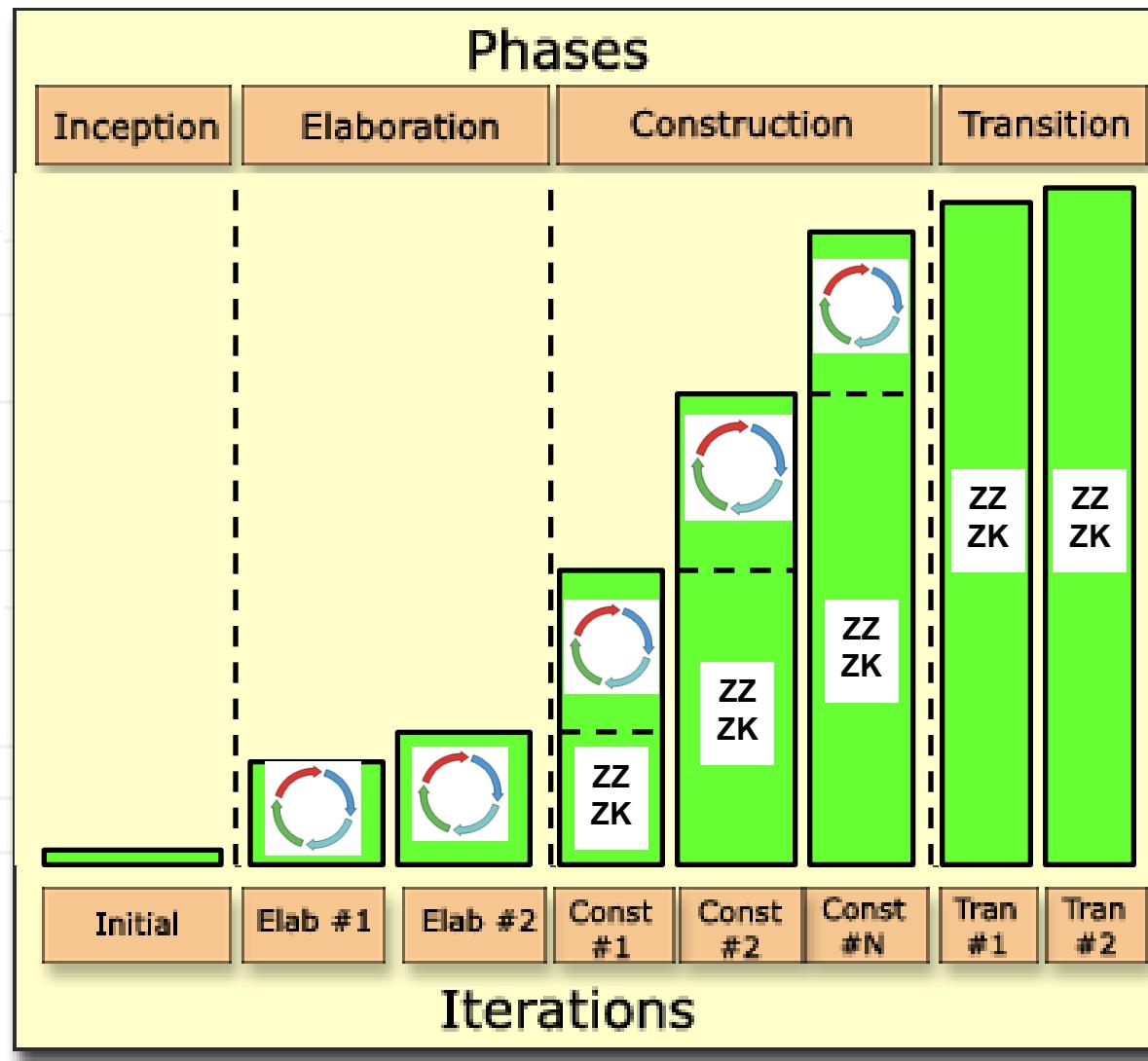
## ➤ Transition - WYDANIE

obniża ryzyko odrzucenia produktu (np. na skutek jego niedopasowania lub niezrozumienia)

# RUP – przyrost produktu



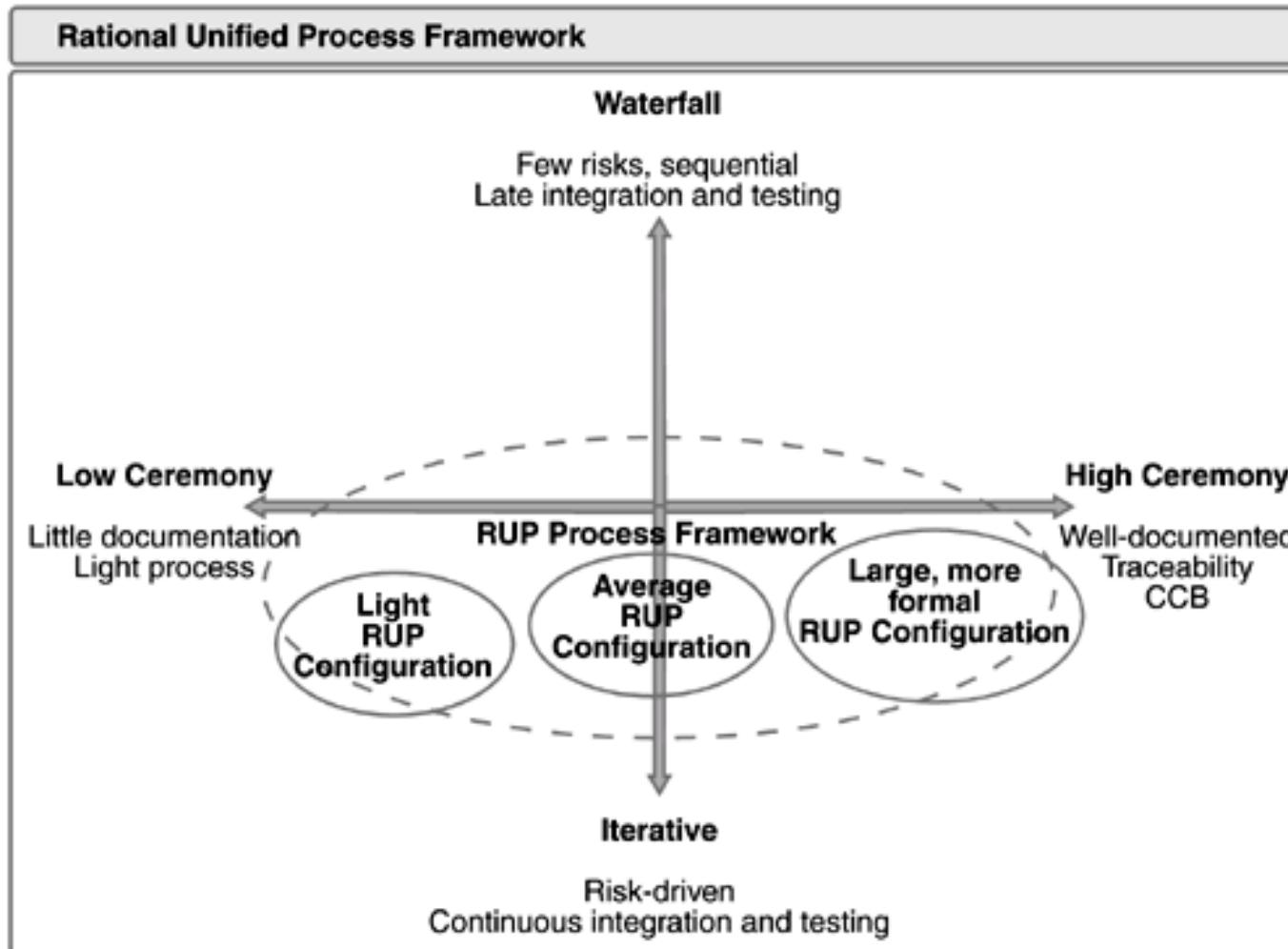
# RUP – praca nad przyrostami produktu



**cykl:**  
wymagania,  
analiza i  
projekt,  
programowanie,  
testowanie

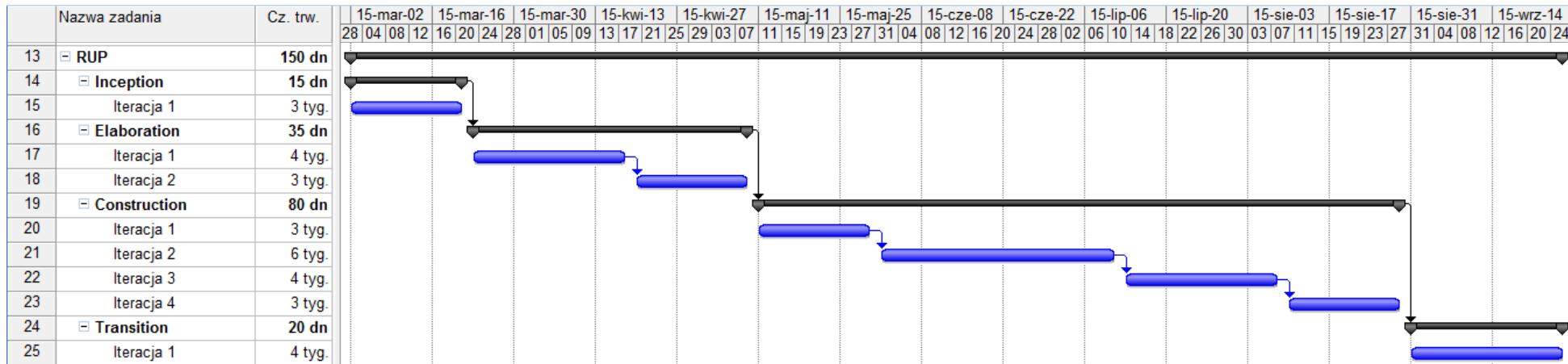
**ZZ –**  
Zarządzanie  
zmianą  
**ZK –**  
Zarządzanie  
konfiguracją

# RUP framework – „metaproces”



Definicja procesu – przycinanie RUP – własne szablony – dopasowanie do potrzeb

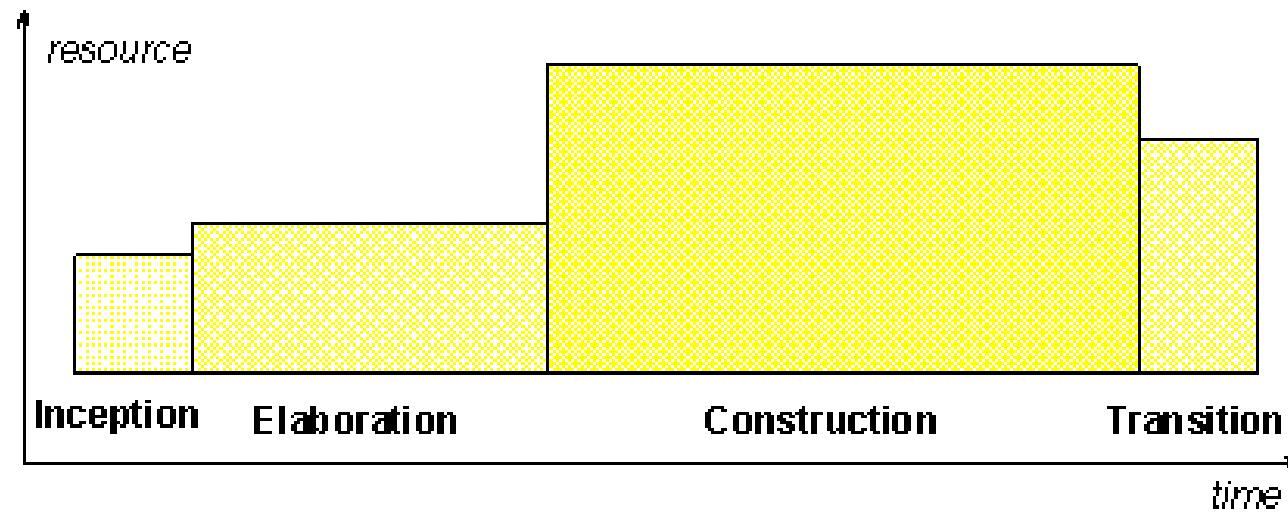
# Proces RUP w czasie



- Różna długość faz
- W każdej fazie iteracje, najczęściej w Construction
- Ogólny Plan Projektu na cały projekt
- Plany Iteracji na początku każdej iteracji

# Proporcje faz RUP

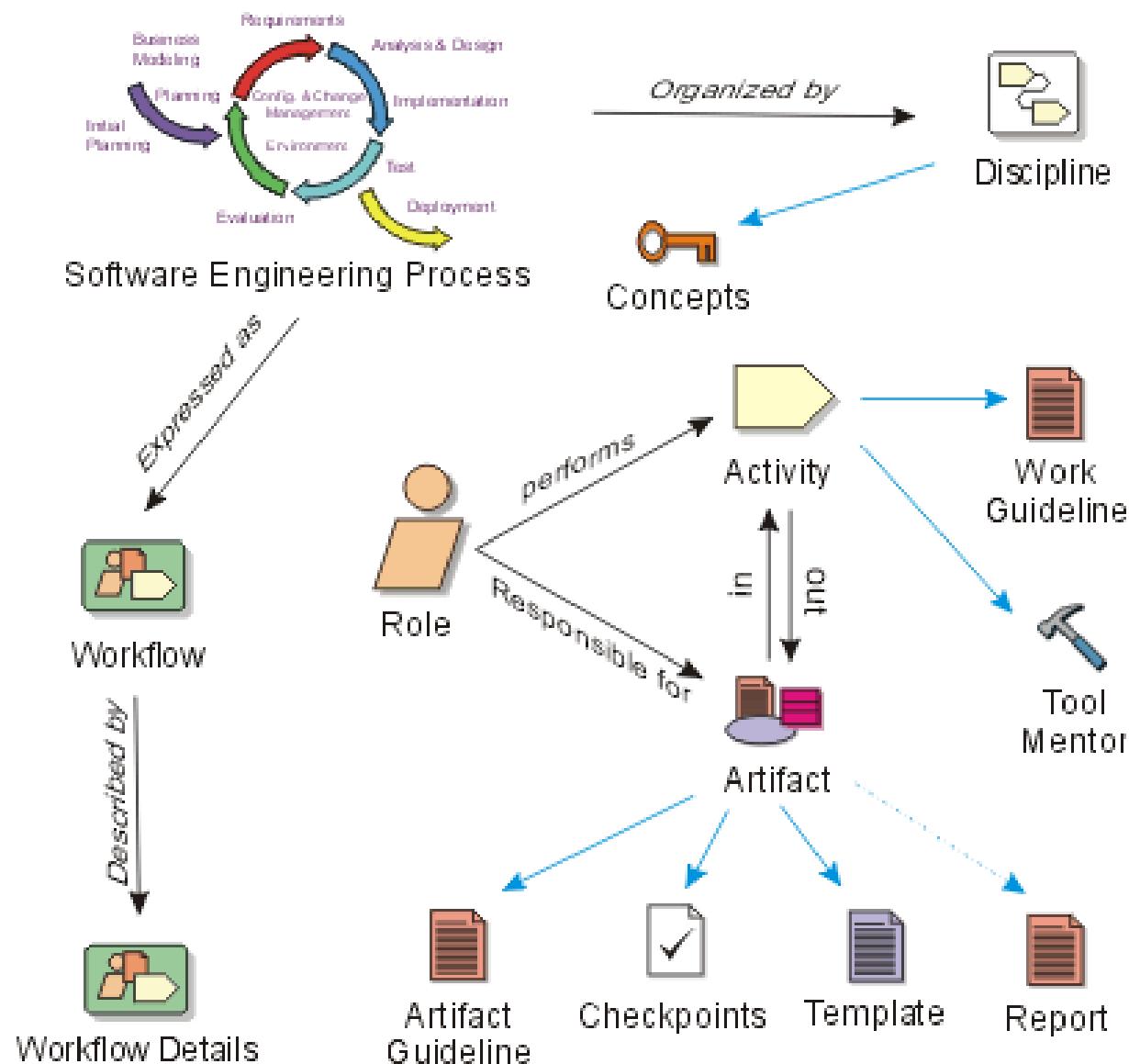
	<u>Inception</u>	<u>Elaboration</u>	<u>Construction</u>	<u>Transition</u>
<b>Nakład</b>	~5 %	20 %	65 %	10%
<b>Czas</b>	10 %	30 %	50 %	10%



# Główne praktyki

- Wytwarzanie iteracyjne
- Zarządzanie wymaganiami
- Używanie architektur komponentowych
- Modelowanie wizualne (UML)
- Ciągła kontrola jakości – w szczególności przez automatyczne testowanie
- Zarządzanie zmianami

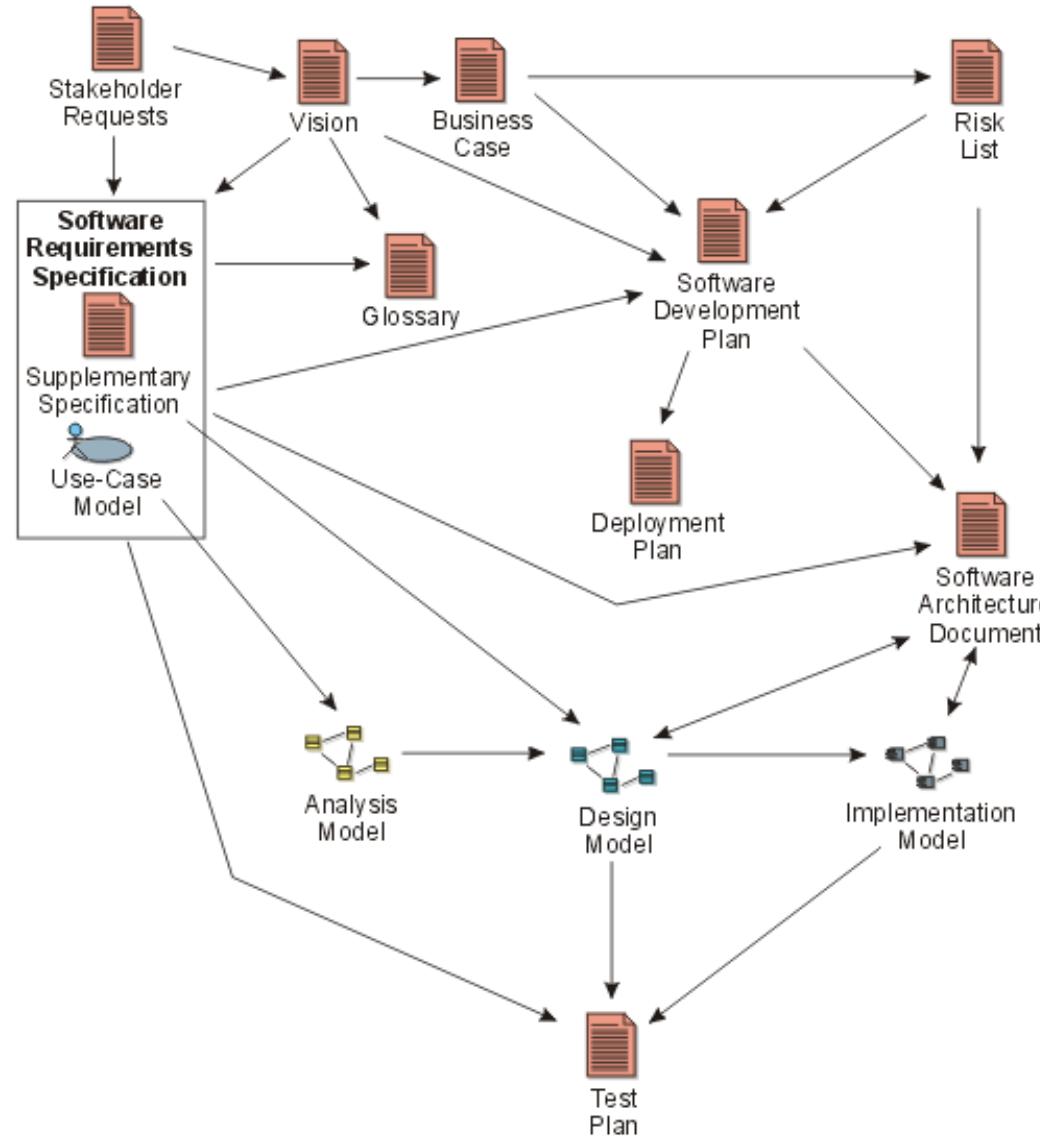
# Składniki RUP



# Najważniejsze role RUP

- **Kierownik projektu**
- **Architekt oprogramowania**
- **Analityk biznesowy**
- **Analityk systemowy**
- **Specyfikujący wymagania**
- **Projektant**
- **Programista, Integrator**
- **Kierownik testów, Analityk testów, Projektant testów, Tester**
- **Redaktor Techniczny, Twórca szkoleń, Grafik**
- **Inżynier procesu**
- **Specjalista ds. narzędzi**
- **Kontrolerzy jakości (ang. reviewers)**

# Główne artefakty RUP



# Przepływ pracy (workflow)

Dziedzina:  
wymagania

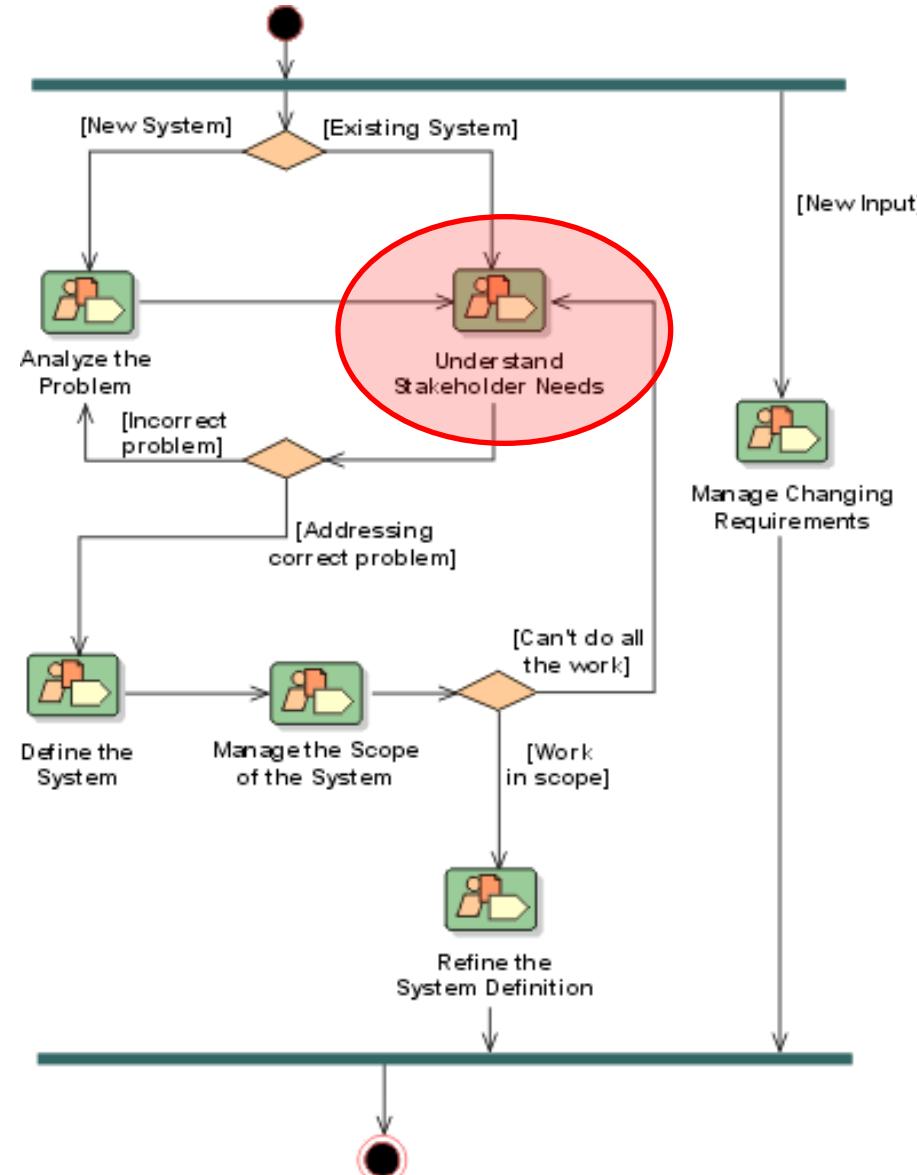
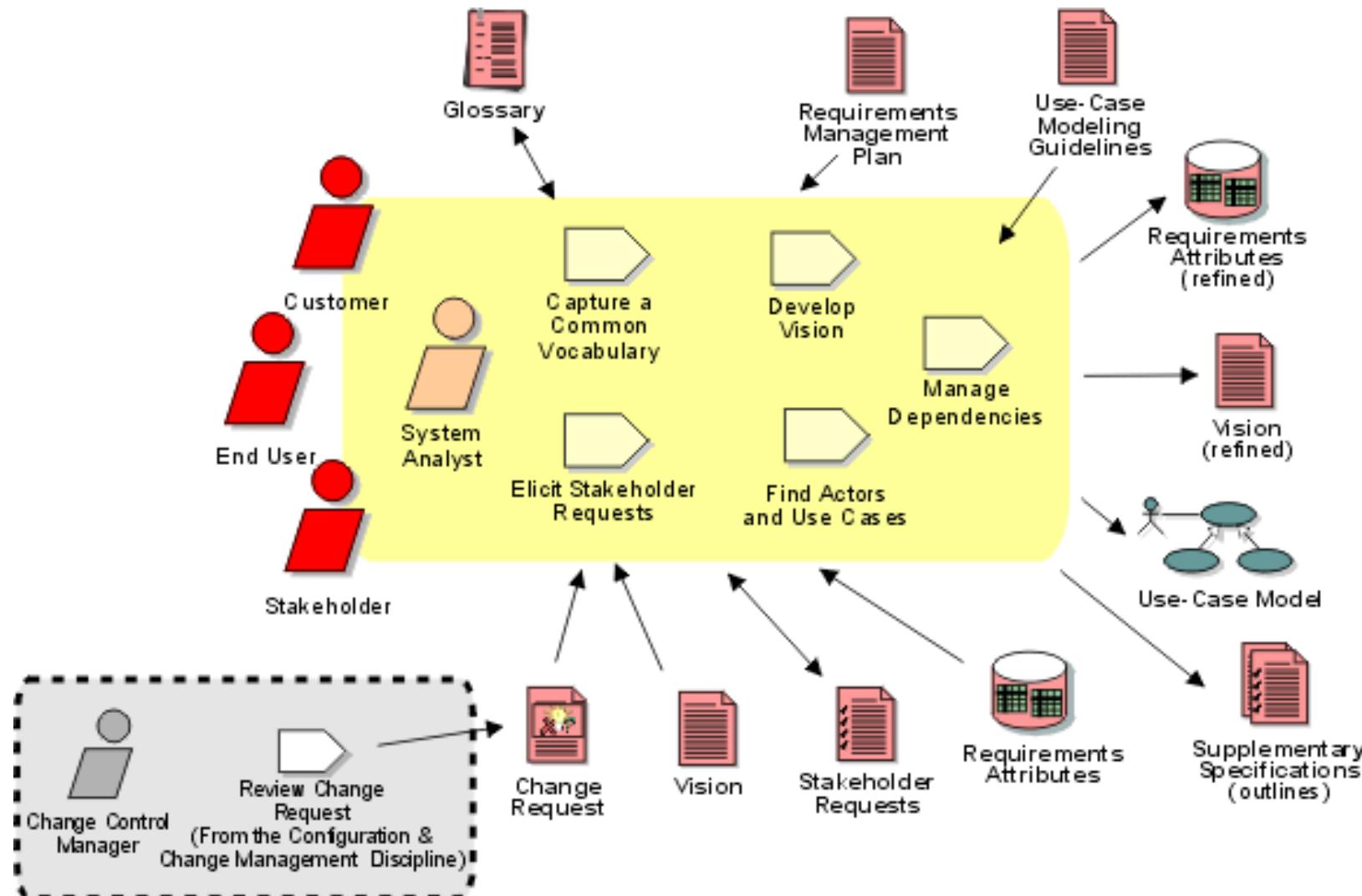


Diagram aktywności UML

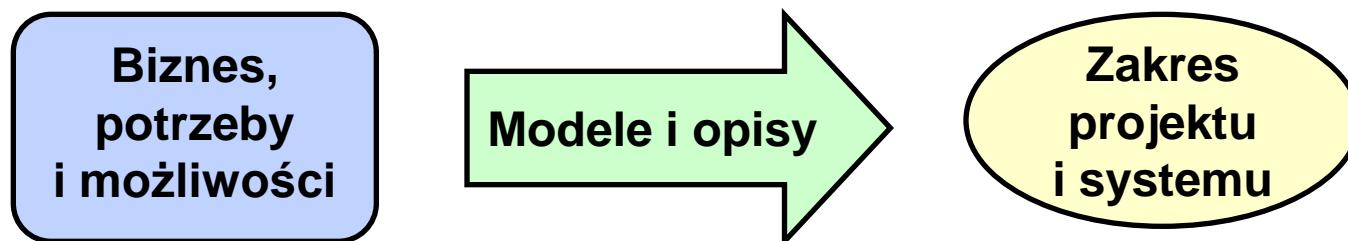
# Uszczegółowienie przepływu (*workflow detail*)



Dziedzina: wymagania, podproces: zrozumienie potrzeb udziałowców

# Faza Inception - cele

- ustalenie zakresu produktu
- określenie głównych scenariuszy i przypadków użycia
- wskazanie potencjalnej architektury systemu
- oszacowanie kosztu i harmonogramu projektu
- ocena ryzyka
- przygotowanie środowiska wspomagającego projektu



# Faza Inception – główne produkty

- Wizja (ang. *Vision*)
- Kontekst biznesowy (ang. *Business Case*)
- Lista zagrożeń (ang. *Risk List*)
- Plan wytwarzania oprogramowania (ang. *Software Development Plan*)
- Proces wytwórczy (ang. *Development Case*)
- Narzędzia
- Słownik pojęć
- Modele biznesowe
- Model przypadków użycia (ang. *Use Case Model*)
- Prototyp/model interfejsu użytkownika
- Struktura informacji UI
- Repozytorium projektu

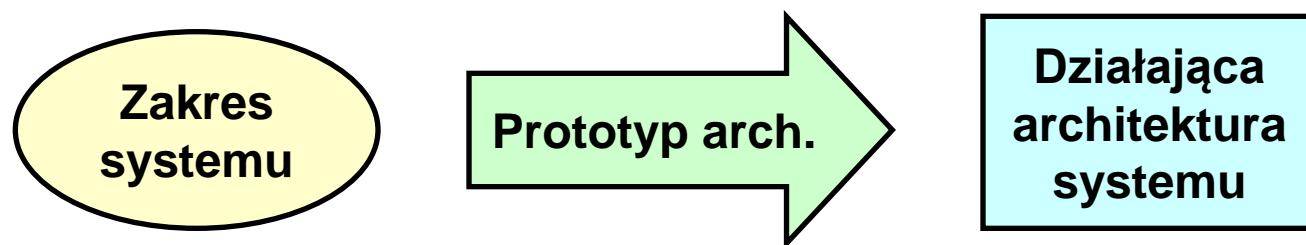
# RUP – Faza Elaboration – cele i wejście

## ➤ Cele fazy Elaboration

- ustabilizowanie planów, wymagań i architektury
- obniżenie zagrożeń związanych z wyborem architektury i technologii
- zatwierdzenie architektury wspierającej zakres systemu
- stworzenie prototypów architektonicznych
- przygotowanie środowiska wspomagającego wytwarzanie

## ➤ Główne artefakty wejściowe z fazy Inception

- Wizja – kandydujące architektury
- Model przypadków użycia – zakres systemu
- Plan wytwarzania oprogramowania – ogólny plan całego projektu



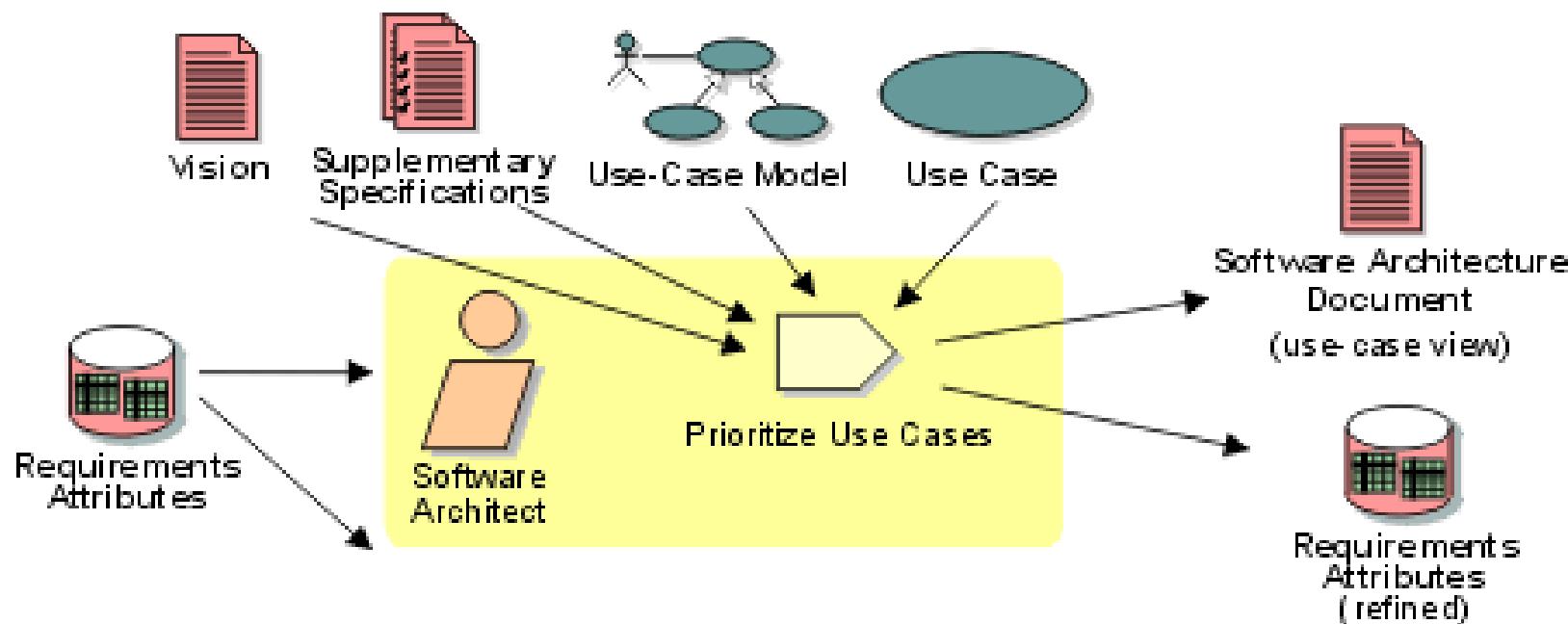
# RUP – Faza Elaboration - produkty

## ➤ Główne artefakty rozwijane w każdej iteracji fazy Elaboration

- Plan iteracji (ang. *Iteration Plan*)
  - Model analityczny (ang. *Analysis Model*)
  - **Definicja architektury oprogramowania (ang. *Software Architecture Document*)**
  - Model projektowy (ang. *Design Model*)
  - Model danych (ang. *Data Model*)
  - Model implementacyjny (ang. *Implementation Model*)
  - Zalecenia do programowania (ang. *Programming Guidelines*)
  - **Prototyp architektoniczny**
  - Architektura automatyzacji testów (ang. *Test Automation Architecture*)
  - Przypadki testowe (ang. *Test Cases*)
  - Skrypty testowe (ang. *Test Scripts*)
- + aktualizacja i rozwój artefaktów z fazy Inception: *Wizja, Model przypadków użycia, Lista zagrożeń, Plan wytwarzania oprogramowania, Proces wytwórczy*

# Wybór wymagań sterujących architekturą (wymagania)

- Architekt oprogramowania biorąc pod uwagę opisy Przypadeków użycia wybiera te, które będą analizowane, projektowane i implementowane w tej iteracji
- Wybrane Przypadki użycia pozwolą dobrać i sprawdzić architekturę systemu
- Kierownik projektu aktualizuje Plan iteracji i Listę zagrożeń



# Wybór wymagań sterujących architekturą

PU \ Cecha	Priorytet klienta	Wydajność	Trwałość	Niezawodność	Ochrona	Ergonomia	Liczba *
PU.1	1	**	*	***	**	***	11
PU.2	1	*	**	**	***	**	10
PU.3	2	**	*	**	**	***	10
PU.4	3	***	**	***	*	**	11
PU.5	3	*	***	**	***	*	10
PU.6	1	**	***	***	**	**	12
PU.7	2	***	**	**	*	***	11
PU.8	1	**	***	***	**	***	13
PU.9	3	***	***	***	*	**	12
PU.10	3	**	**	*	***	***	11
PU.11	2	***	**	**	**	**	11
PU.12	1	**	**	***	**	*	11
PU.13	3	***	**	***	**	**	12

Wybrane przypadki użycia: PU.2, PU.6, PU.8, PU.11

## Wybór wymagań sterujących architekturą (2)

- Wybieramy do prototypu kilka przypadków użycia pozwalających zweryfikować główne wymagania architektoniczne (jakościowe) produktu
- Wybieramy jak najmniejszy zbiór przypadków użycia stanowiący wyzwanie dla architektury i technologii pod względem wszystkich wymagań (jakościowych)
- Priorytet klienta (wartość biznesowa) przypadku użycia ma znaczenie drugorzędne, rozstrzyga przy podobnych wymaganiach architektonicznych
- Prototyp architektoniczny nie musi być używalnym produktem  
- to jest tylko PROTOTYP

# Faza Elaboration - podsumowanie

- Celem fazy Elaboration jest ustalenie architektury oprogramowania, która umożliwi konstrukcję systemu o zadanym zakresie i wymaganiach
- Iteracje fazy Elaboration powtarzane są tak długo, aż architektura zostanie całkowicie ustabilizowana i dalsze wytwarzanie produktu ma już charakter powtarzalny i przewidywalny (produkcja)
- Faza Elaboration kończy się uzyskaniem zatwierdzonego prototypu architektonicznego
- W każdej iteracji ulepszane jest środowisko wytwarzania. Pierwsza iteracja fazy Elaboration może skupiać się na środowisku analizy, projektowania i implementacji, kolejne na środowisku testowania

# RUP – Faza Construction – cele i wejście

## ➤ Cele fazy Construction

- uzyskanie działających wersji produktu
- uzyskanie odpowiedniej jakości produktu
- zakończenie analizy, projektowania, implementacji i testów wszystkich funkcji
- określenie stopnia przygotowania środowiska docelowego
- efektywne wykorzystanie zasobów oraz zrównoleglanie pracy

## ➤ Główne artefakty wejściowe z fazy Elaboration

- Definicja architektury oprogramowania (ang. *Software Architecture Document*)
- Model projektowy (ang. *Design Model*)
- Model danych (ang. *Data Model*)
- Model implementacyjny (ang. *Implementation Model*)
- Prototyp architektoniczny



# RUP – Faza Construction - produkty

## ➤ Główne artefakty rozwijane w każdej iteracji fazy Construction

- Plan iteracji (ang. *Iteration Plan*)
  - Model projektowy (ang. *Design Model*)
  - Model danych (ang. *Data Model*)
  - Model implementacyjny (ang. *Implementation Model*)
  - Komponenty (ang. *Components*)
  - Przypadki testowe (ang. *Test Cases*)
  - Skrypty testowe (ang. *Test Scripts*)
  - Produkt (ang. „*The System*”)
  - Plan wdrożenia (wstępny) (ang. *Deployment Plan*)
  - Materiały szkoleniowe (pierwsza wersja) (ang. *Training Materials*)
- + aktualizacja artefaktów z faz Inception i Elaboration: *Model przypadków użycia, Model analityczny, Zalecenia do programowania, Architektura automatyzacji testów, Plan wytwarzania oprogramowania, Proces wytwórczy, Narzędzia*

# Przebieg fazy Construction - ogólnie

- Kolejne iteracje są planowane z uwzględnieniem priorytetów wymagań i zagrożeń
- W każdej iteracji powstaje przetestowana wewnętrznie wersja produktu
- Identyfikacja wymagań jest zamknięta, nowe wymagania przechodzą przez proces zarządzania zmianami
- Prace projektowe ograniczają się do implementowanych przypadków użycia
- Prace projektowe w kolejnych iteracjach mają mniejszą intensywność i koncentrują się coraz bardziej na aktualizacji projektu na podstawie zmian w wymaganiach
  
- Przebieg wytwarzania produktu analogiczny jak w fazie Elaboration, ale inny jest sposób wyboru zestawu przypadków użycia do kolejnego przyrostu

# RUP – Faza Transition – cele i wejście

## ➤ Cele fazy Transition

- testowanie zewnętrzne
- wdrożenie produktu w środowisku docelowym, w tym konwersja i transfer danych
- szkolenie użytkowników
- pakowanie i przekazanie do sprzedaży (jeżeli produkt na otwarty rynek)
- ocena sukcesu i podsumowanie projektu

## ➤ Główne artefakty wejściowe z fazy Construction

- Produkt (ang. „*The System*”)
- Plan wdrożenia (wstępny) (ang. *Deployment Plan*)
- Materiały szkoleniowe (pierwsza wersja) (ang. *Training Materials*)



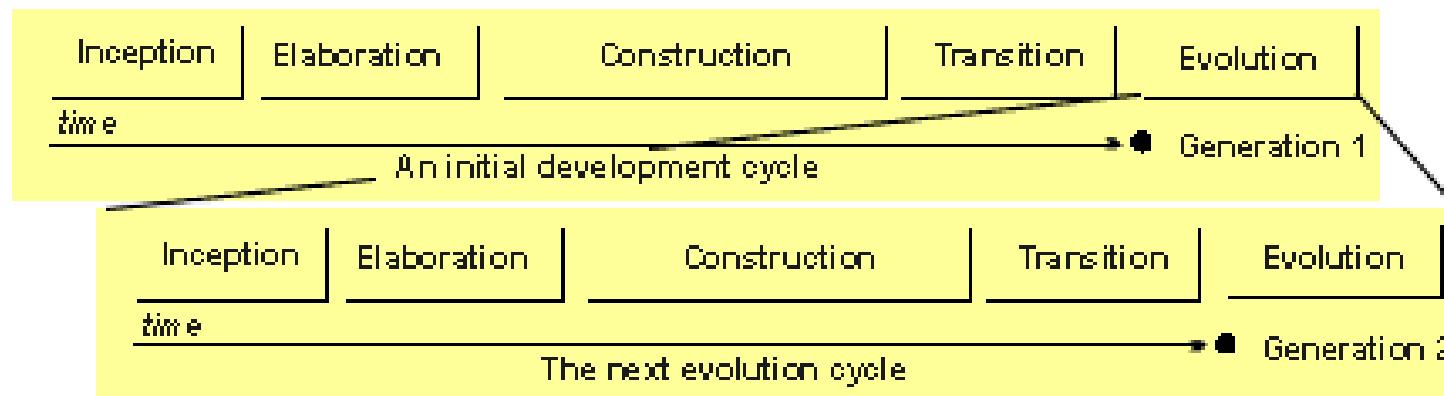
# RUP – Faza Transition - produkty

## ➤ Główne artefakty rozwijane w każdej iteracji fazy Transition

- Plan iteracji (ang. *Iteration Plan*)
  - Plan wdrożenia (ang. *Deployment Plan*)
  - Zestaw wdrożeniowy (ang. *Deployment Unit*)
  - Pakiet produktu (ang. *Product Build*)
  - Informacje o wydaniu (ang. *Release Notes*)
  - Materiały instalacyjne (ang. *Installation Artifacts*)
  - Materiały szkoleniowe (ang. *Training Materials*)
  - Materiały wspomagające użytkowników (ang. *End-User Support Material*)
- + aktualizacja artefaktów z faz *Inception*, *Elaboration* i *Construction*:  
*Model przypadków użycia*, *Model analityczny*, *Definicja architektury oprogramowania*, *Model projektowy*, *Model danych*, *Model Implementacyjny*, *Komponenty*, *Plan wytwarzania oprogramowania*

# Przebieg fazy Transition - ogólnie

- Przekazywanie produktu odbiorcom, zależnie od typu produktu
- Testowanie zewnętrzne i drobne poprawki na bazie wyników tych testów
- Konfiguracja, instalowanie, szkolenia użytkowników
- Usuwanie błędów, w małym stopniu dodawanie nowych funkcji
- Może być wdrażany niekompletny produkt, jednak taki, który daje wartość użytkownikom
- Rozwój produktu i jego kolejne wydania to nowy proces RUP



# Skalowanie metodyk zwinnych

**Katarzyna Łukasiewicz**

*Katedra Inżynierii Oprogramowania*

*Wydział Elektroniki, Telekomunikacji i Informatyki*

*Politechnika Gdańsk*

[katarzyna.lukasiewicz@pg.edu.pl](mailto:katarzyna.lukasiewicz@pg.edu.pl)



# Agenda

- 1. Motywacje**
- 2. Praca w rozproszeniu**
- 3. Metody skalowania**
  1. Scrum of Scrums
  2. Scaled Agile Framework (SAFe)
  3. Disciplined Agile Delivery
  4. Spotify Model
- 4. DevOps**
- 5. Agile beyond software**

**dr inż. Katarzyna Łukasiewicz**

*katarzyna.lukasiewicz@pg.edu.pl*

- **Adiunkt na Wydziale Elektroniki, Telekomunikacji i Informatyki Politechniki Gdańskiej**
- **Project Manager & Agile Coach w firmie Bright Inventions**
- **Zainteresowania naukowe:**
  - Zwinne metodyki wytwarzania oprogramowania (m.in. budowanie postaw „agile mindset“)
  - Bezpieczeństwo systemów informatycznych

# Potencjalne korzyści z wdrożenia metodyk zwinnych

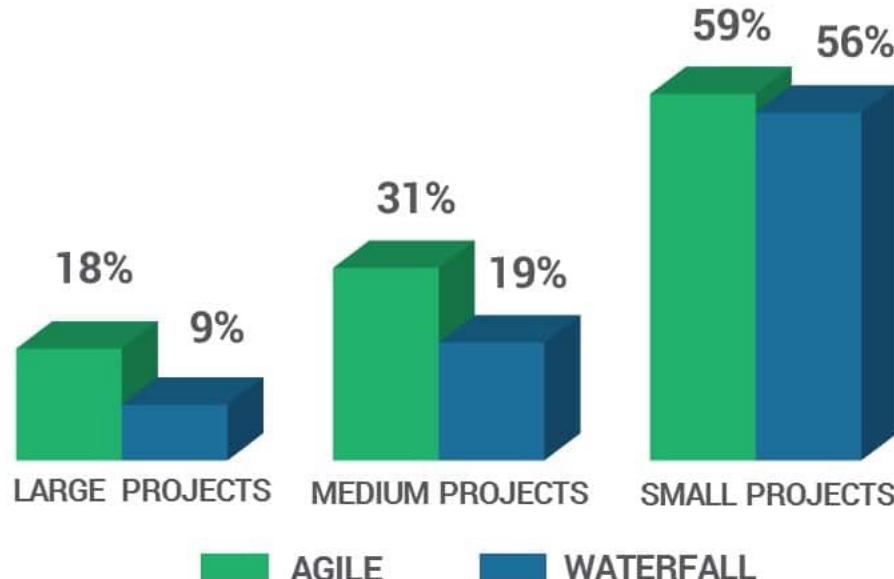
- Lepsze dopasowanie do potrzeb klienta
- Szybsza odpowiedź na zmiany
- Mniejszy koszt wytwarzania
- Poprawa efektywności pracy zespołu

... Czy tylko dla mniejszych, niekrytycznych projektów?

# Agile vs. Waterfall

## PROJECT SUCCESS RATES BY PROJECT SIZE **AGILE VS WATERFALL**

FOR LARGE PROJECTS, AGILE APPROACHES ARE 2X MORE LIKELY TO SUCCEED



Source: Standish Group, Chaos Studies 2013-2017

[WWW.VITALITYCHICAGO.COM](http://WWW.VITALITYCHICAGO.COM)

# Mniej oczywiste sytuacje...

**AGILE TO MAŁE ZESPOŁY PRACUJĄCE W JEDNYM POMIESZCZENIU, ale..**

- **Praca w rozproszeniu?**
- **Duże projekty?**
  - Zespoły kilkudziesięcio-kilkuset osobowe
  - Zakres produktu
- **Duże organizacje?**
  - Rozbudowana struktura
  - Kultura
- **Wymagania względem bezpieczeństwa?**
  - Security (wrażliwe dane, hakerzy itp.)
  - Safety (medycyna, transport itp.)

# Praca w rozproszeniu

- Coraz częściej zespoły pracują w rozproszeniu
- Wg State of Agile Report z 2019 r. 78% zespołów pracujących metodyką zwinną nie pracowało w jednym miejscu
- Powody: głównie koszty,  
ale także umiejętności



- **Komunikacja**
  - Głównie spotkania zdalne
  - Bariera językowa
- **Kultura**
  - Terminy świąt państwowych, dni wolnych
  - Różne postrzeganie organizacji pracy
- **Strefy czasowe**
  - Utrudnienia w komunikacji
  - Dłuższy czas odpowiedzi na zgłoszenia

# Praktyki zwinne wspierające pracę w rozproszeniu

- **Zwinna kultura pracy**
  - nastawienie na współpracę
  - poczucie współodpowiedzialności
  - częsty i mniej formalny kontakt
- **Stosowanie podejścia zwanego w ramach całej organizacji, nie tylko zespołu**
- **Rozszerzona specyfikacja User Stories**
  - W rozproszonych zespołach trudniej działać tylko na podstawie User Stories
  - Dodatkowy opis: kryteria akceptacji, proste diagramy
- **Narzędzia i automatyzacja**
  - Podstawą narzędzia do współdzielenia kodu i konfiguracji
  - Narzędzia do rozszerzonej komunikacji
  - Unifikacja środowiska twórczego

# Duże projekty

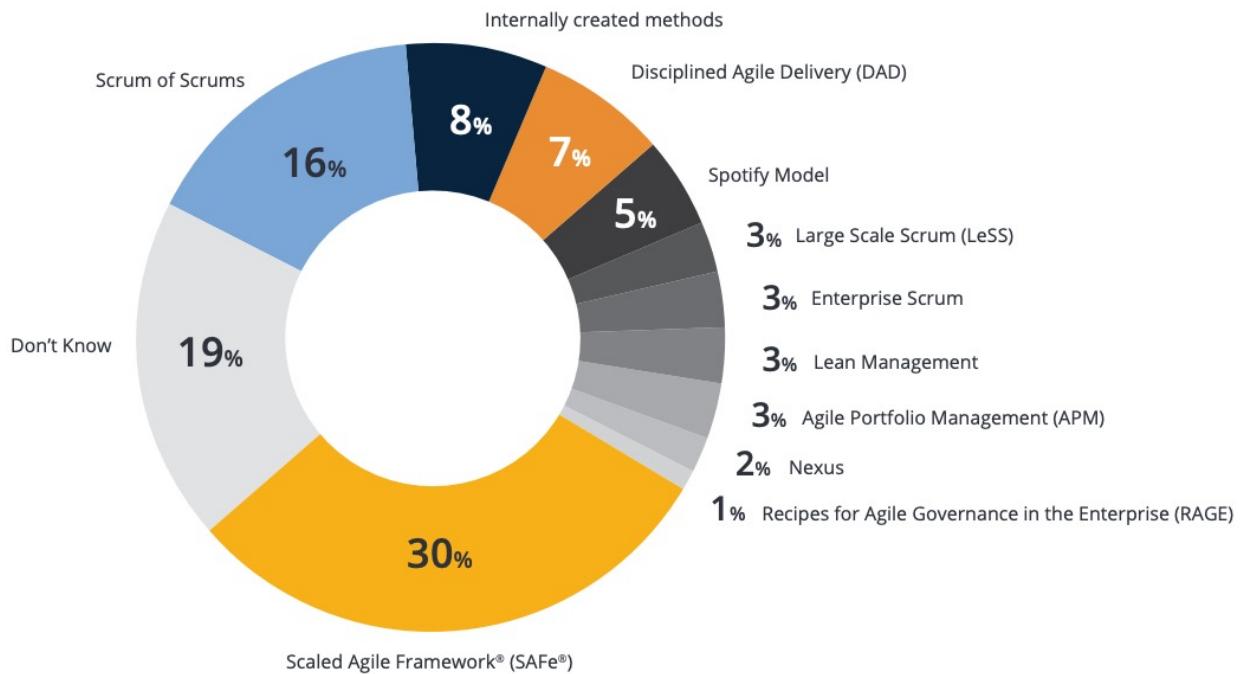
- **Zespoły liczące po kilkaset, kilka tysięcy osób**
- **Złożone systemy**
  - Wiele źródeł wymagań
  - Nowe, nieznane technologie
  - Systemy wbudowane
- **Projekty utrzymujące starsze systemy**
- **Systemy posiadające rozbudowane i złożone funkcjonalności**

# Skalowanie podejścia zwinnego

- Często zespoły deweloperskie już używają praktyk zwinnych – ale jak przenieść to na procesy w całej firmie?
- De-skalowanie organizacji zamiast skalowania podejścia zwinnego
- Więcej autonomii dla zespołów → szybkość, motywacja

# Najpopularniejsze metody skalowania podejścia zwinnego

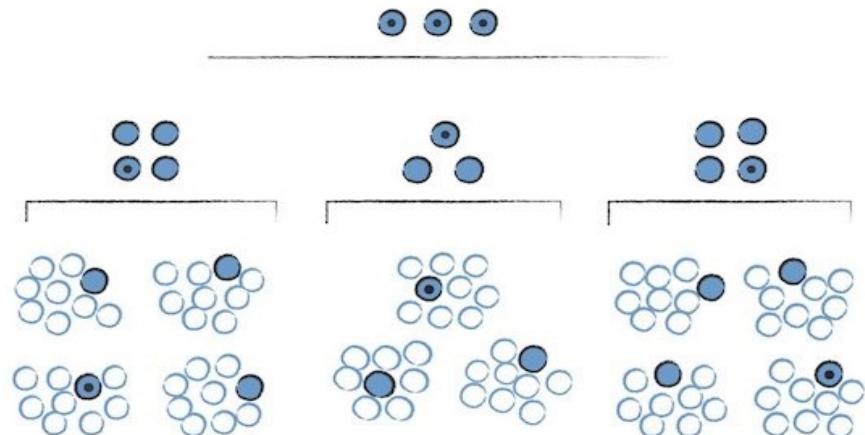
- **Scaled Agile Framework (SAFe)**
- **Scrum of Scrums**
- **Disciplined Agile Delivery (DAD)**
- **Spotify Model**
- **Large-Scale Scrum (LeSS)**
- **Enterprise Scrum**
- **Lean Management**
- ...



Źródło: Raport VersionOne 2019 13<sup>th</sup> Annual State of Agile Report

# Scrum of Scrums

- Metoda pracy dla większych zespołów (**>12 osób**)
- Podział na mniejsze podzespoły, pracujące metodą Scrum
- Każdy zespół:
  - ma wewnętrzne spotkania Scrum
  - deleguje jednego “ambasadora” na spotkania obejmujące cały projekt, tzw. **Scrum of Scrums**



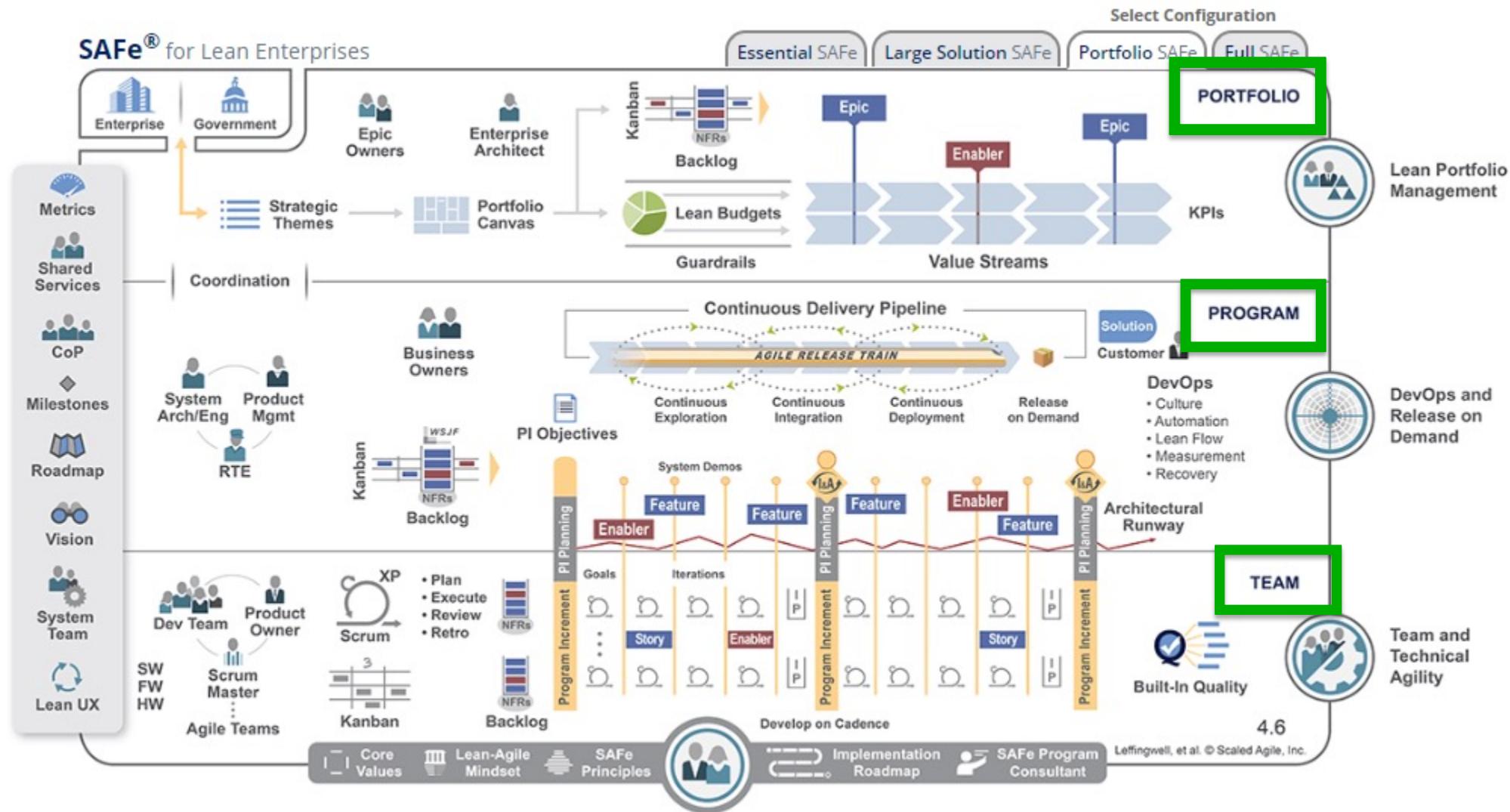
Źródło: Mountain Goat Software

## Pytania na Daily (niekoniecznie) Scrum of Scrums:

- 1. Co twój zespół zrobił od czasu poprzedniego spotkania?**
- 2. Co planujecie wykonać do czasu kolejnego spotkania?**
- 3. Czy twój zespół napotkał jakieś utrudnienia?**
- 4. Czy planujecie działania mające wpływ na pracę innego zespołu?**



- Metoda pozwalająca wdrożyć podejście zwinne (Lean-Agile) na poziomie całej organizacji
- Skalowalna, modułowa
- Obecne wersje 4.6 - 5.0 mają 4 konfiguracje:
  - Essential SAFe (podstawowa wersja, średnie organizacje)
  - Large Solution SAFe (duże organizacje, bez potrzeby poziomu Portfolio)
  - Portfolio SAFe (średnie projekty, zespoły pracujące dość niezależnie)
  - Full SAFe (złożone projekty, setki osób zaangażowanych)



# Trzy poziomy:

## 1. Portfolio (*Portfolio level*)

- Zarządzanie zakresem projektu na wyższym poziomie
- Szersze spojrzenie na specyfikacja elementów Backlogu
- Product Management – czuwa nad całością

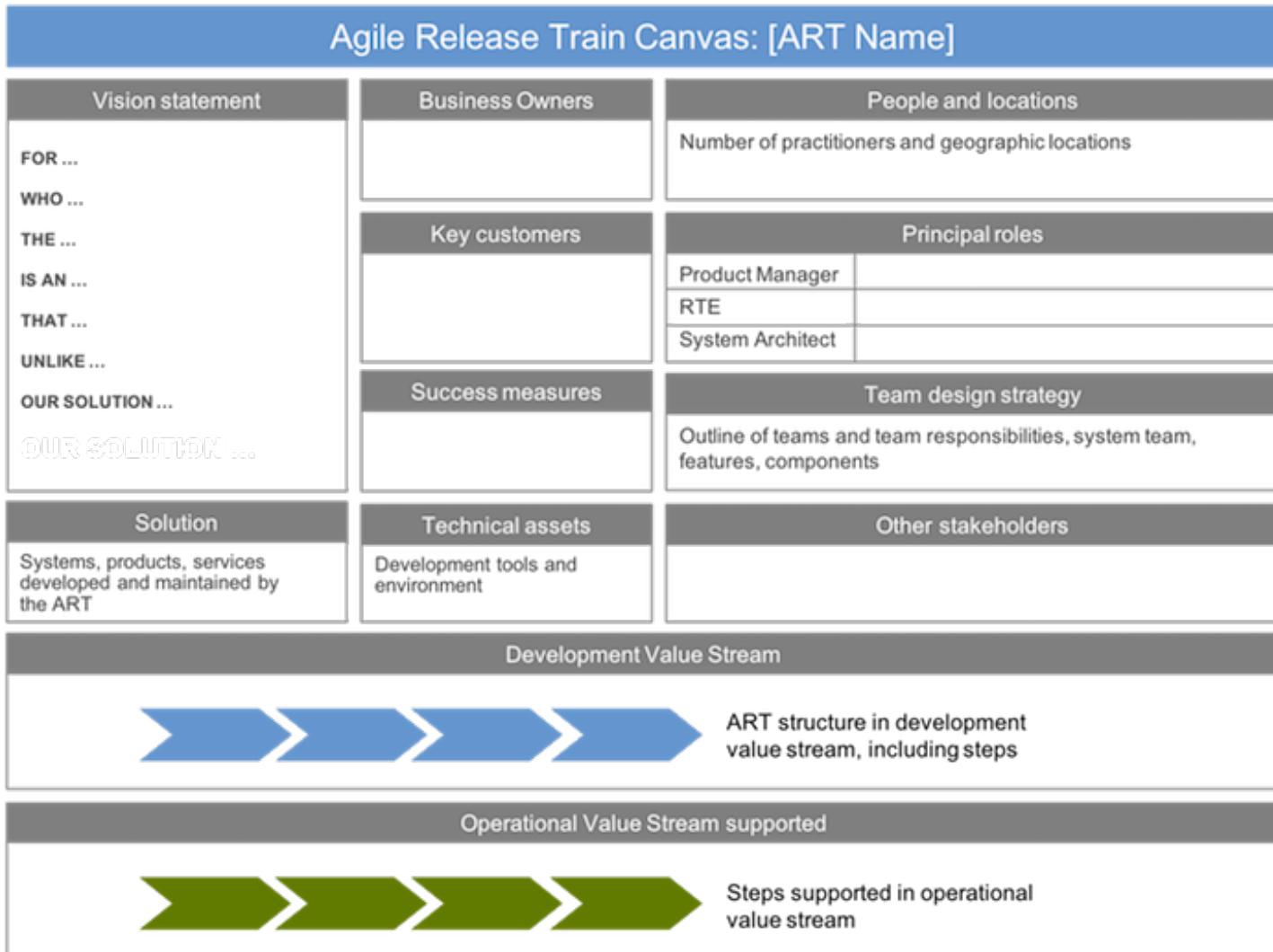
## 2. Program (*Program level*)

- Agile Release Train – wiele mniejszych zespołów pracujących nad elementami tego samego produktu
- Nad pracą zespołu czuwa Release Train Engineer
- Product Increments – większe wydania produktu, skupiające kilka Release Trains, w dłuższych iteracjach np. 10 tygodni

## 3. Zespół (*Team Level*)

- Małe zespoły, dostarczające kolejne wersje produktu w krótkich iteracjach np. 2 tygodnie
- Praca wg. Scrum albo Scrum + Kanban
- Nad pracą zespołu czuwa Scrum Master

# Agile Release Train



# Role w SAFe

- **Na poziomie zespołu:**
  - Scrum Master
  - Właściciel Produktu
  - Zespół Deweloperski
- **Dodatkowo, na poziomie Agile Release Train**
  - Release Train Engineer – jak Scrum Master dla Release Train
  - Product Management – właściciel Program Backlog
  - Architekt Systemu – zapewnia wsparcie w kwestii architektury systemu
  - Klienci, właściciele biznesowi – kluczowi interesariusze Release Train
  - Zespół wsparcia systemowego – wsparcie dot. infrastruktury
  - DevOps – o tym na kolejnych slajdach...

# Portfolio SAFe (5.0)

SAFe® for Lean Enterprises 5.0

Select Configuration

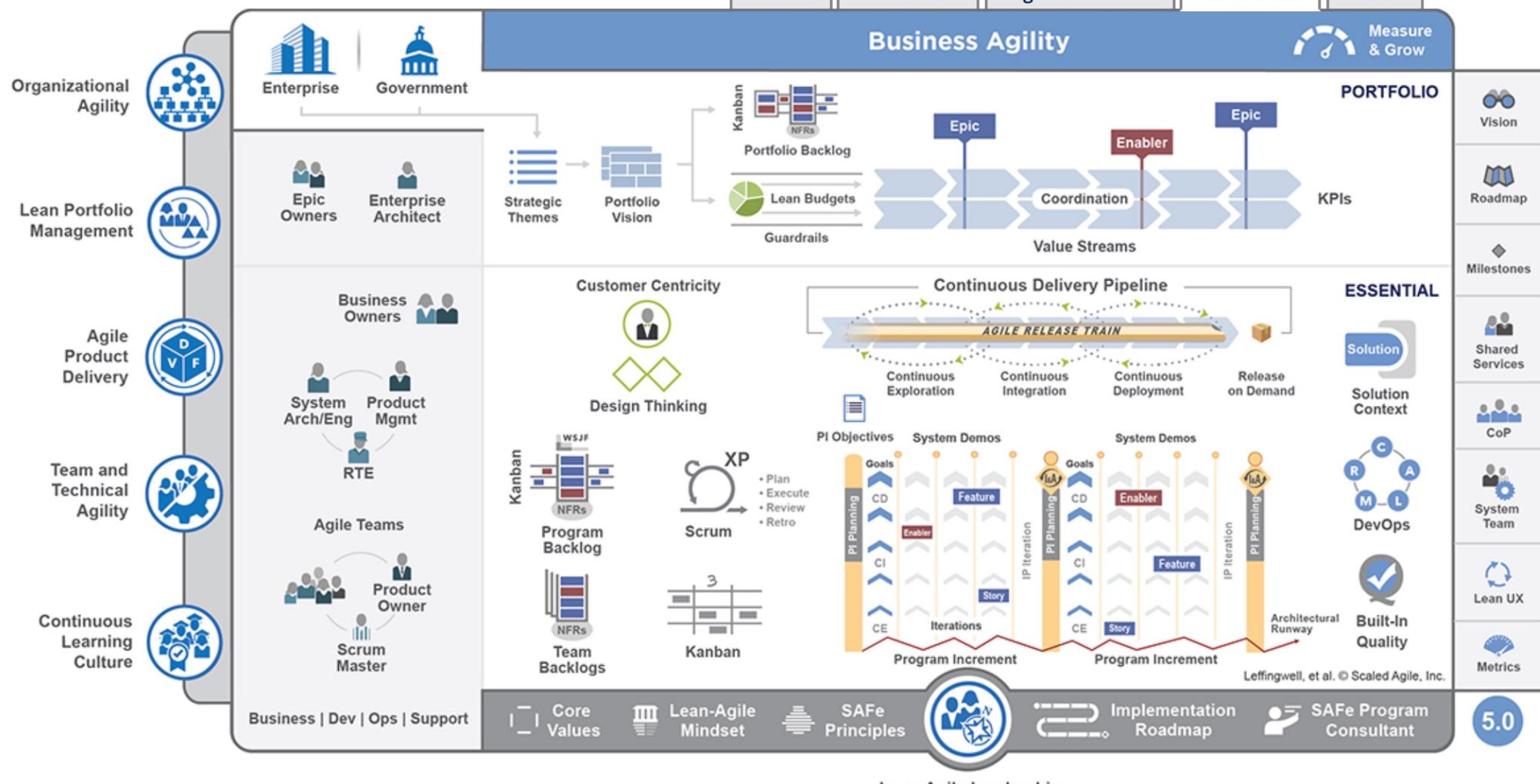
Overview

Essential SAFe

Large Solution SAFe

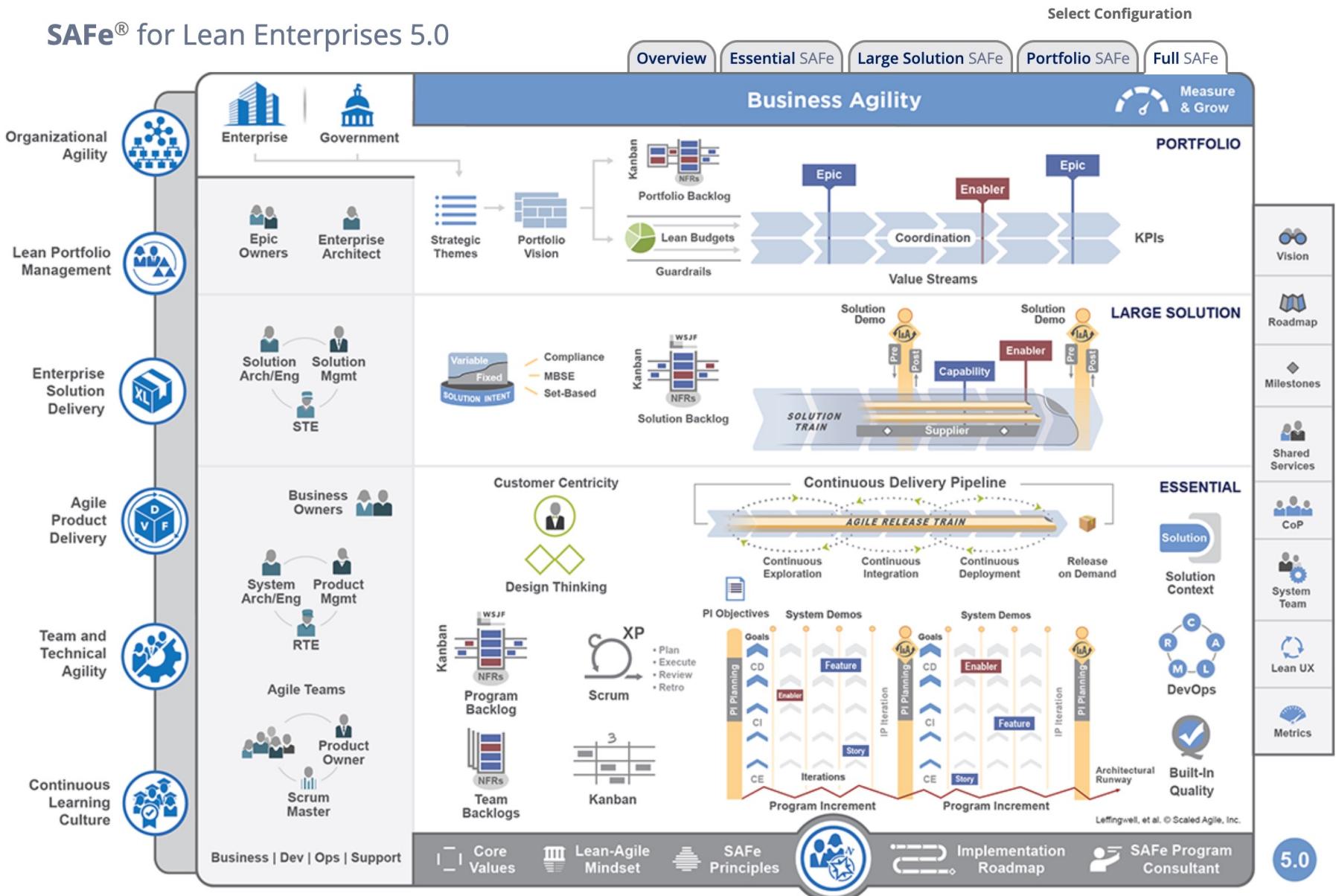
Portfolio SAFe

Full SAFe



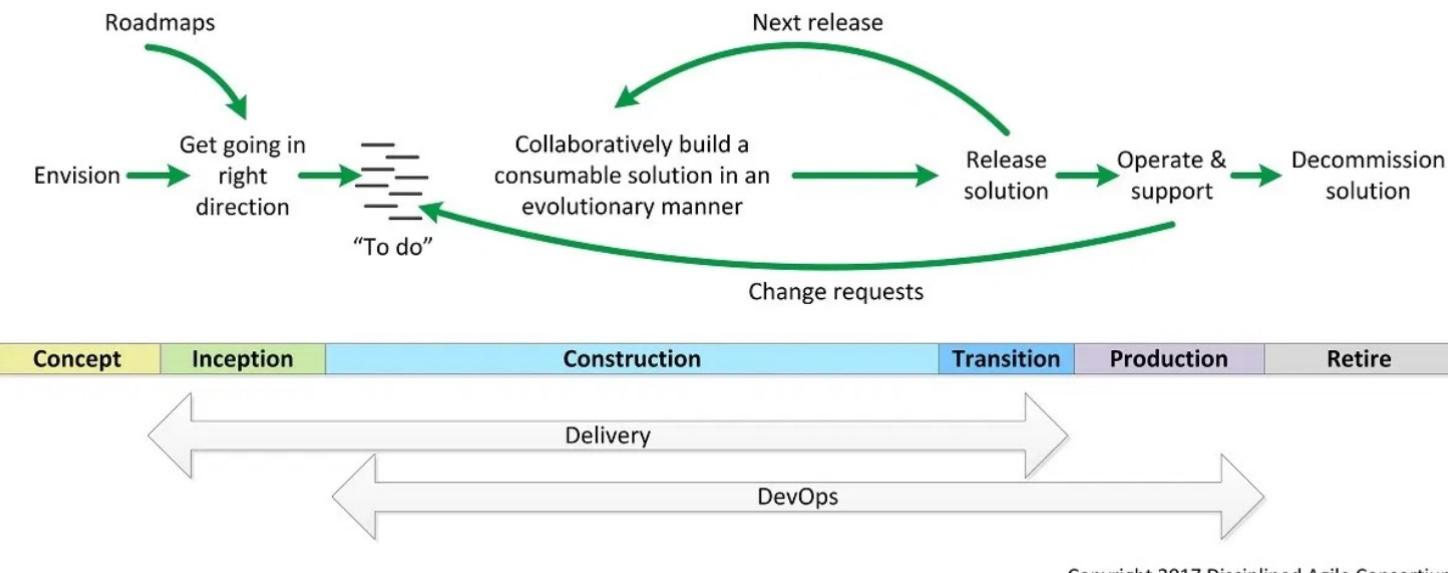
# Full SAFe (5.0)

## SAFe® for Lean Enterprises 5.0



# Disciplined Agile Delivery

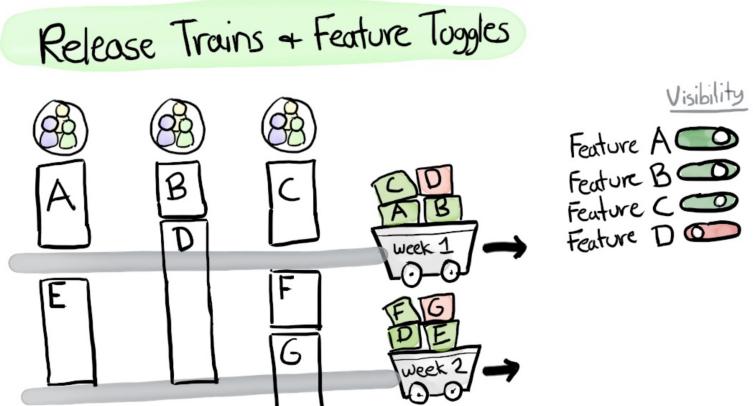
- Podejście hybrydowe, bazujące na innych metodach zwinnych**



- 6 opcji cyklu wytwarzczego (lifecycle): Agile, Lean, Continuous Delivery: Agile, Continuous Delivery: Lean, Exploratory/Lean Startup, Program**
- Role**
  - Podstawowe:** Team Member, Team Lead, Product Owner, Architecture Owner, Stakeholder
  - Wspierające:** Specialist, Independent Tester, Domain Expert, Technical Expert, Integrator

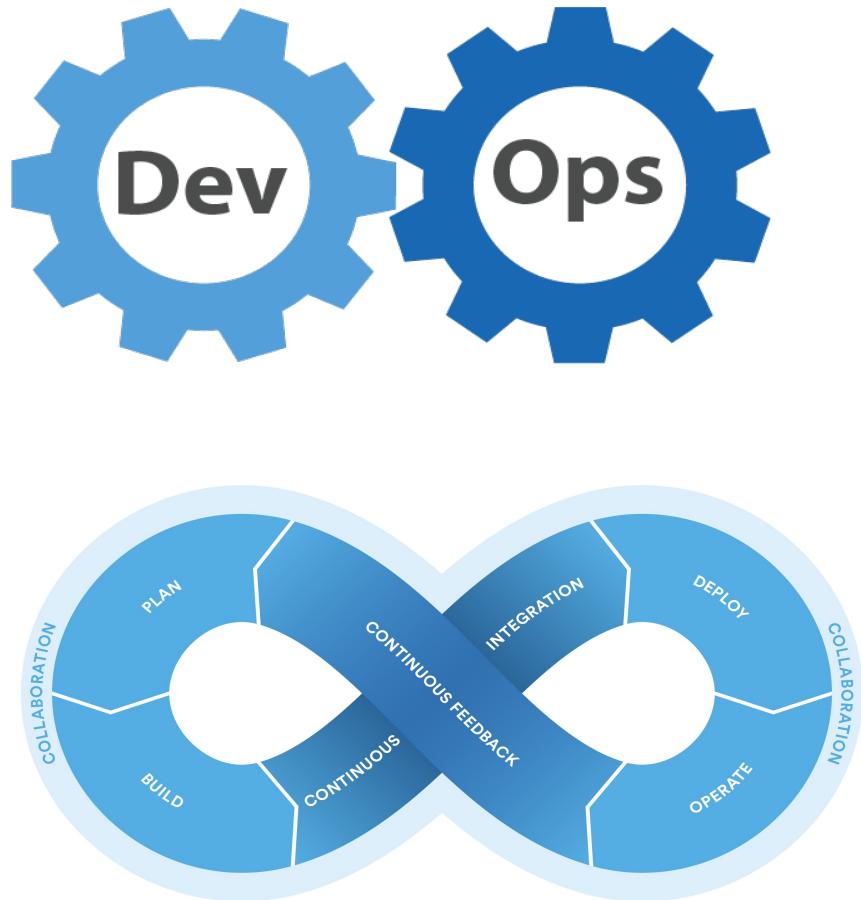
# Spotify model

- Podejście opracowane przez Spotify, od 2012 zyskujące na popularności – formalnie nie jest „metodyką”
- Struktura: *Squads* (zespoły), *Tribes* (grupy zespołów), *Chapter* (specjalści), *Guild* (nieformalny zespół), *Trio* (Design+Product+Team Lead), *Alliance* (3xTrio), *Chief Architect*
- Najważniejsze założenia:
  - Współpraca między zespołami (Guilds)
  - Częste wydania
  - Release Trains + Feature Toggles
  - Zaufanie, wspólnota, motywacja



Źródło: H. Kniberg, labs.spotify.com

- DevOps = software  
**DEVelopment + IT OPerationS**
- Współpraca między zespołami deweloperskimi, a specjalistami operacji IT i zapewniania jakości (Quality Assurance)



Źródło grafiki: [Atlassian.com](https://atlassian.com)

- **Cele:**
  - Krótsze cykle wytwarzcze
  - Częstsze wydania
  - Lepsza jakość
- **Przeniesienie idei zaczerpniętych z podejścia zwinnego na poziom organizacji**
- **Główne zmiana sposobu myślenia i kultury oraz propozycje rozwiązań wspierających taką zmianę**
- **Dla dużych organizacji ważny element wprowadzania podejścia zwinnego**

# DevOps – jak?

- **Współpraca i komunikacja między departamentami:**  
współdzielenie odpowiedzialności, wspólny język i perspektywa
- **Automatyzacja procesów:** infrastruktura, przepływ danych, testowanie, ... wszystko co tylko można!
- **Mniejsze, częste wydania**
- **Współzielona konfiguracja i środowisko**

- Podstawa – użycie narzędzi np. TeamCity, Puppet, Jira, Jenkins, Docker, GitHub
- Ważne praktyki:
  - Automatyzacja testów
  - Zarządzanie konfiguracją
  - Continuous integration
  - Continuous deployment
  - Business Intelligence

# Jak skalować z sukcesem metodyki zwinne

- 1. Spójne i konsekwentne podejście do procesu i praktyk**
- 2. Zastosowanie jednolitych narzędzi we wszystkich zespołach**
- 3. Zatrudnienie konsultantów lub trenerów metodyk zwinnych**
- 4. Wprowadzenie roli czuwającego nad całością Właściciela Projektu (*Executive Sponsor*)**
- 5. Stworzenie doradczego zespołu wsparcia dla wdrażanego podejścia zwanego**

**Wg. Raportu VersionOne 2016 10<sup>th</sup> Annual State of Agile Report**

- **Manifest Agile wprost określa, że dotyczy oprogramowania...**  
**“Odkrywamy nowe metody programowania dzięki praktyce w programowaniu ...”**
- **... ale żeby spełnić założenia manifestu w pełni, potrzebna jest współpraca poza działami IT**
- **Horyzontalne wdrażanie agile vs wertykalne**

# Creative economy

- Firmy takie jak Apple, Google, Zara
- Główny cel – zadowolić klientów, użytkowników
- Zmiana ideowa na wyższych szczeblach kierownictwa
- Transparentność, proces ciągłego udoskonalania

# Podsumowanie

- 1. Agile mindset – “zwinny” sposób myślenia**
- 2. Uczciwe i sumienne stosowanie zaplanowanych praktyk**
- 3. Ludzie są podstawą**
- 4. Motywacja wewnętrzna bardziej niż zewnętrzna**
- 5. Współodpowiedzialność i dzielenie kodu**



# Realizacja Projektu Informatycznego



***Jakub Miler***

*Katedra Inżynierii Oprogramowania  
Wydział Elektroniki, Telekomunikacji i Informatyki  
Politechnika Gdańskia  
jakubm@eti.pg.edu.pl*

*Materiały pomocnicze do wykładu na Wydziale ETI Politechniki Gdańskiej.  
Wykorzystanie materiałów w innym celu i ich rozpowszechnianie bez zgody WETI PG zabronione.*

# Wykład 7

## Dobór i adaptacja metodyki

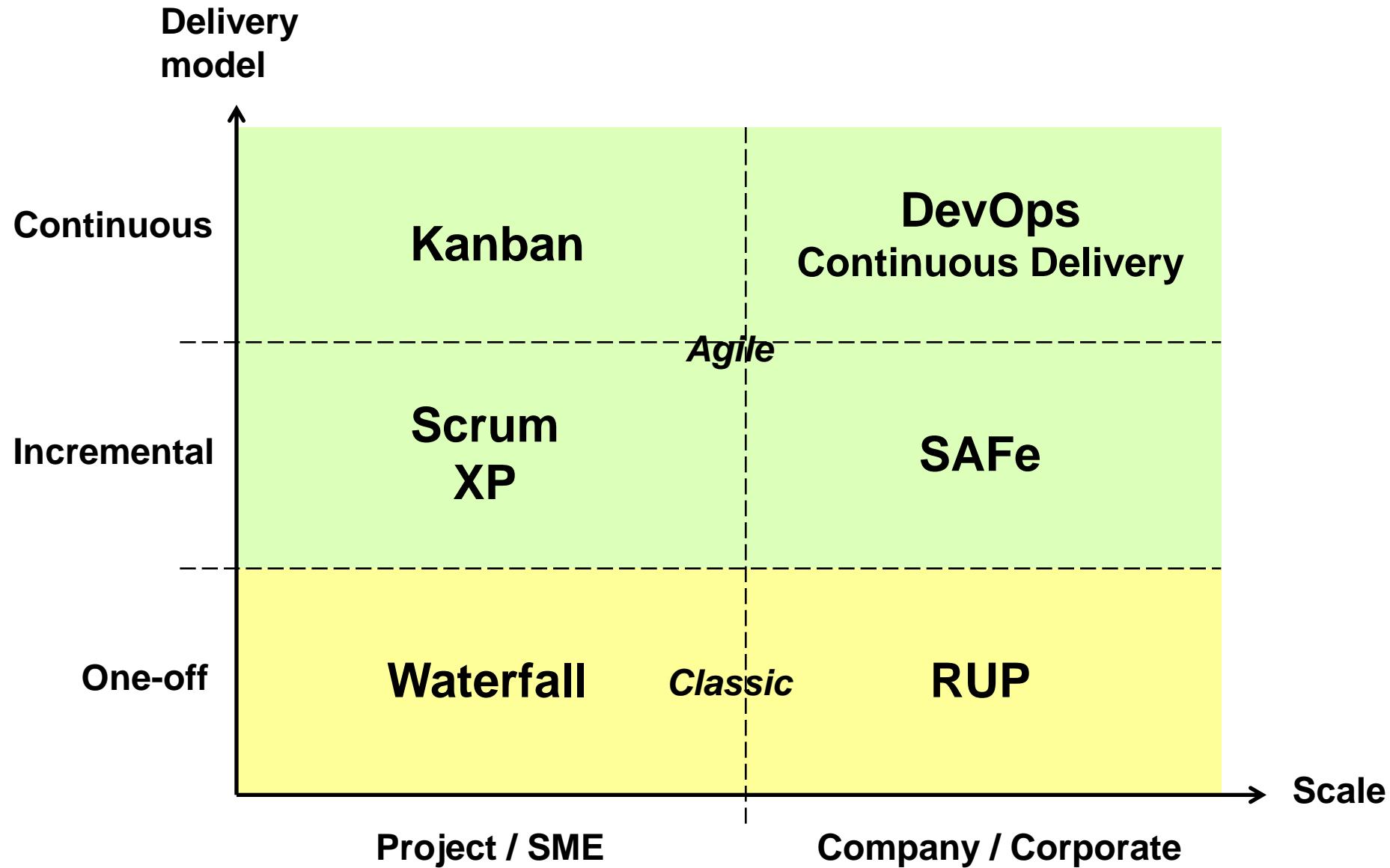


# Idea doboru i adaptacji metodyki



- Jakie przyjąć kryteria doboru metodyki?
- Dobieramy metodykę do projektu czy do firmy?
- Dobieramy metodykę do projektu czy projekt do metodyki?
- Jakie wprowadzić adaptacje, żeby zachować istotę metodyki?

# Development methodologies



# Cechy projektów sterujące doborem metodyki agile vs klasyczne/kaskadowe

- Główne cele projektu
- Środowisko, otoczenie biznesowe
- Komunikacja w projekcie
- Podejście do definiowania wymagań
- Podejście do wytwarzania
- Klient
- Kultura, mindset

Zaadaptowane na podstawie:

B. Boehm, R. Turner,  
*Balancing Agility and Discipline, A Guide for the Perplexed*,  
Addison-Wesley, 2004

# „Własne podwórka” metodyk\*

Cecha	Metodyki zwinne	Metodyki klasyczne
<b>Zastosowanie</b>		
Główne cele	sztywna wartość, reakcja na zmiany	przewidywalność, stabilność, wysoka wiarygodność
Rozmiar	mniejsze zespoły i projekty	większe zespoły i projekty
Środowisko	niestabilne, szybkozmienne, skupione na projekcie	stabilne, wolnozmienne, skupione na projekcie i organizacji
<b>Zarządzanie</b>		
Relacja z klientem	wyznaczony klient na miejscu, skupia się na priorytetach wydań	spotkania z klientem gdy potrzeba, dostarcza ustaleń (np. do specyfikacji)
Planowanie i nadzorowanie	plany zaszyte w opisie produktu, nadzór jakościowy	jawnie dokumentowane plany, nadzór ilościowy
Komunikacja	niejawną, współdzielona wiedza	jawną, udokumentowaną wiedzą

\* Okoliczności, w których metodyki dają sporą szansę na sukces projektu

# „Własne podwórka” metodyk (2)

Cecha	Metodyki zwinne	Metodyki klasyczne
<b>Techniczne</b>		
Wymagania	nieformalne historyjki i przypadki testowe z priorytetami, nieprzewidywalne zmiany	sformalizowane wymagania, proces, dojrzałość, interfejsy, jakość, przewidywalna ewolucja
Wytwarzanie	prosty projekt, krótkie przyrosty, refaktoryzacja z założenia możliwa i tania	szczegółowy projekt, dłuższe przyrosty, refaktoryzacja zakładana jako droga
Testowanie	wykanywalne testy definiują wymagania	udokumentowane plany i procedury testowania
<b>Osoby</b>		
Klient	wyznaczone, dostępne na miejscu osoby klasy CRACK	osoby klasy CRACK, nie muszą być stale dostępne
Wykonawcy	co najmniej 30% pełnoetatowych ekspertów poziomu 2 i 3, bez osób poziomu 1B i -1	50% osób poziomu 3 na początku projektu, potem 10%; może być do 30% osób poziomu 1B, bez osób poz. -1
Kultura	zadowolenie i moc poprzez wiele stopni swobody (sklonność do chaosu)	zadowolenie i moc poprzez zestaw zasad i procedur (sklonność do porządku)

# Główne cele projektu

## ➤ Główne cele metodyk zwinnych:

- szybkie dostarczenie wartości biznesowej
- elastyczna reakcja na zmiany

## ➤ Cechy praktyczne metodyk zwinnych:

- zbudujmy szybko i sprawdźmy, ile jest warte
- unikamy dużych nakładów podjętych na podstawie błędnych założeń
- nastawienie na reakcję na zmiany niż ich uwzględnianie w planie
- „złośliwy” klient może doprowadzić do bardzo chaotycznego projektu

## ➤ Główne cele metodyk klasycznych:

- przewidywalność
- stabilność
- wysoka wiarygodność produktu

## ➤ Cechy praktyczne metodyk klasycznych:

- stosowanie planów, specyfikacji, standaryzacji, pomiarów, nadzorowania
- sprawdzają się w stabilnych projektach, generują dużo pracy na utrzymanie definicji procesu przy zmiennych warunkach
- są wymagane do certyfikacji procesów

# Otoczenie biznesowe

## ➤ Otoczenie dla metodyk zwinnych

- szybko zmienne, w zmiennych warunkach biznesowych
- gdzie liczy się „produkt w rękach”, a mniej utrzymanie czy integracja
- innowacyjne produkty na nowym rynku, MVP

## ➤ Otoczenie dla metodyk klasycznych

- wymagania są w dużej części znane na początku (w tym pozyskane przez prototypowanie) i pozostają dość stabilne
- potrzeba szerokiego zakresu prac: produkcja, organizacja, biznes
- system ma wpływ na wiele osób na różnych stanowiskach ( wielu udziałowców)
- sytuacje przejściowe związane z wycofywaniem systemów spadkowych
- słabo podzielne wymagania (nie da się dostarczać produktu w częściach)

# Komunikacja w projekcie

## ➤ Metodyki zwinne

- bazują na niejawnej rozproszonej wiedzy w zespole
- podstawą wiedzy jest doświadczenie osób z zespołu
- wiedza o projekcie jest zbierana w czasie spotkań planistycznych i przeglądowych
- wiedza jest dzielona poprzez tworzenie nowych par i zespołów
- bazują na częstej, osobistej i dwukierunkowej komunikacji
- w większych zespołach utrzymywanie spójności wiedzy jest utrudnione (dla N osób jest  $N*(N-1)/2$  ścieżek komunikacji osoba-osoba)
- również zawierają elementy jawnie zapisanej wiedzy (np. przypadki testowe, kod)
- tworzona jest tylko niezbędną dokumentacją - „skalowanie w górę”

## ➤ Metodyki klasyczne

- bazują na jawnie zapisanej wiedzy
- komunikacja jest głównie jednokierunkowa (opisy procesów, raporty z postępu prac)
- również zawierają komunikację bezpośrednią
- tworzona jest dokumentacja wymagana przez proces z eliminowaniem („przycinaniem”) niepotrzebnych elementów – „skalowanie w dół”
- mniej doświadczony zespół będzie stosował całą metodykę (proces), ponieważ nie będzie umiał przyciąć go do potrzeb projektu, realizując nadmiarowe praktyki

# Podejście do definiowania wymagań

## ➤ W metodykach zwinnych

- wymagania są reprezentowane jako elastyczne, nieformalne historyjki
- szybkie iteracje pozwalają ustalać zmiany wymagań
- wymagania o najwyższym priorytecie do włączenia do iteracji są ustalane wspólnie przez programistów i klienta
- wymagania jakościowe postrzegane są jako kolejna cecha oprogramowania

## ➤ W metodykach klasycznych

- wymagania są formalnie zatwierdzane, kompletne, spójne, śladowe i weryfikowalne
- często specyfikacje powstają w innym zespole niż oprogramowanie
- by zmniejszyć problem dezaktualizujących się wymagań stosuje się cykle spiralne (oparte na ryzyku)
- lepiej specyfikuje się wymagania jakościowe np. niezawodność, wydajność, skalowalność; postrzegane są jako złożone cechy mające szeroki wpływ na system

# Podejście do wytwarzania

## ➤ Metodyki zwinne

- promują „prosty projekt” – taki który przechodzi testy dla danego wydania i nie posiada cech nadmiarowych z punktu widzenia zakresu tego wydania
- projekt powinien być utrzymywany w prostocie – upraszczany przez refaktoryzację
- założenie 1: koszt zmian w projekcie pozostaje mały wraz z upływem czasu
- założenie 2: scenariusz zastosowania zmienia się tak szybko, że nadmiarowe rozwiązania projektowe nie będą użyte
- doświadczenie pokazuje, że założenie 1 jest w praktyce nie do spełnienia
- refaktoryzacja ma ograniczenia; są wymagania, które łamią architekturę np. skalowalność, wydajność, wielojęzyczność, tolerowanie awarii, duży rozmiar danych
- założenie 2 realizowane jest w postaci zasady YAGNI; nie jest ona uzasadniona, gdy można sensownie przewidzieć przyszłe potrzeby

## ➤ Metodyki klasyczne

- używają planowania i architektur by przygotować się na zmiany wymagań
- umożliwia to m.in. reużycie komponentów w linii produktów
- duża część pracy wiąże się z definicją wydolnej architektury w początkowym etapie projektu (ang. *Big Design Up Front – BDUF*)
- przy bardzo zmiennych wymaganiach wydatek na dobrą definicję architektury rozłoży projekt

# Klient

## ➤ W metodykach zwinnych

- zaufanie klienta budowane jest przez działający system i uczestnictwo klienta
- konieczna jest wysoka „jakość” reprezentanta klienta
- reprezentant klienta powinien być: chętny do współpracy, reprezentatywny, uprawniony, zaangażowany i mieć odpowiednią wiedzę (ang. *Collaborative, Representative, Authorized, Committed, Knowledgeable – CRACK*)
- brak chęci współpracy wprowadza konflikty, frustrację i obniża morale zespołu
- niereprezentatywność prowadzi do niewłaściwych produktów
- brak uprawnienia powoduje opóźnienia przy zatwierdzaniu decyzji lub nieuprawnione decyzje
- brak zaangażowania skutkuje unikaniem swojej roli w procesie
- brak wiedzy skutkuje opóźnieniami, niewłaściwym produktem lub oboma naraz

## ➤ W metodykach klasycznych

- podstawą relacji z klientem są umowy i specyfikacje
- reprezentanci klienta klasy CRACK mogą być angażowani czasowo w prace projektu
- głównym wyzwaniem jest uniknięcie ścisłego, biurokratycznego nadzoru nad projektem, gdzie zgodność z planem ważniejsza jest od rezultatów projektu

# Kultura i mindset

## ➤ Metodyki zwinne

- oferują „wiele stopni swobody”
- klasyczne środowisko „zakładu rzemieślniczego” – osoby ufają w umiejętności i pracę innych oraz starają się nie zawieść zaufania innych
- wymagają zwinnego sposobu myślenia – agile mindset

## ➤ W metodykach klasycznych

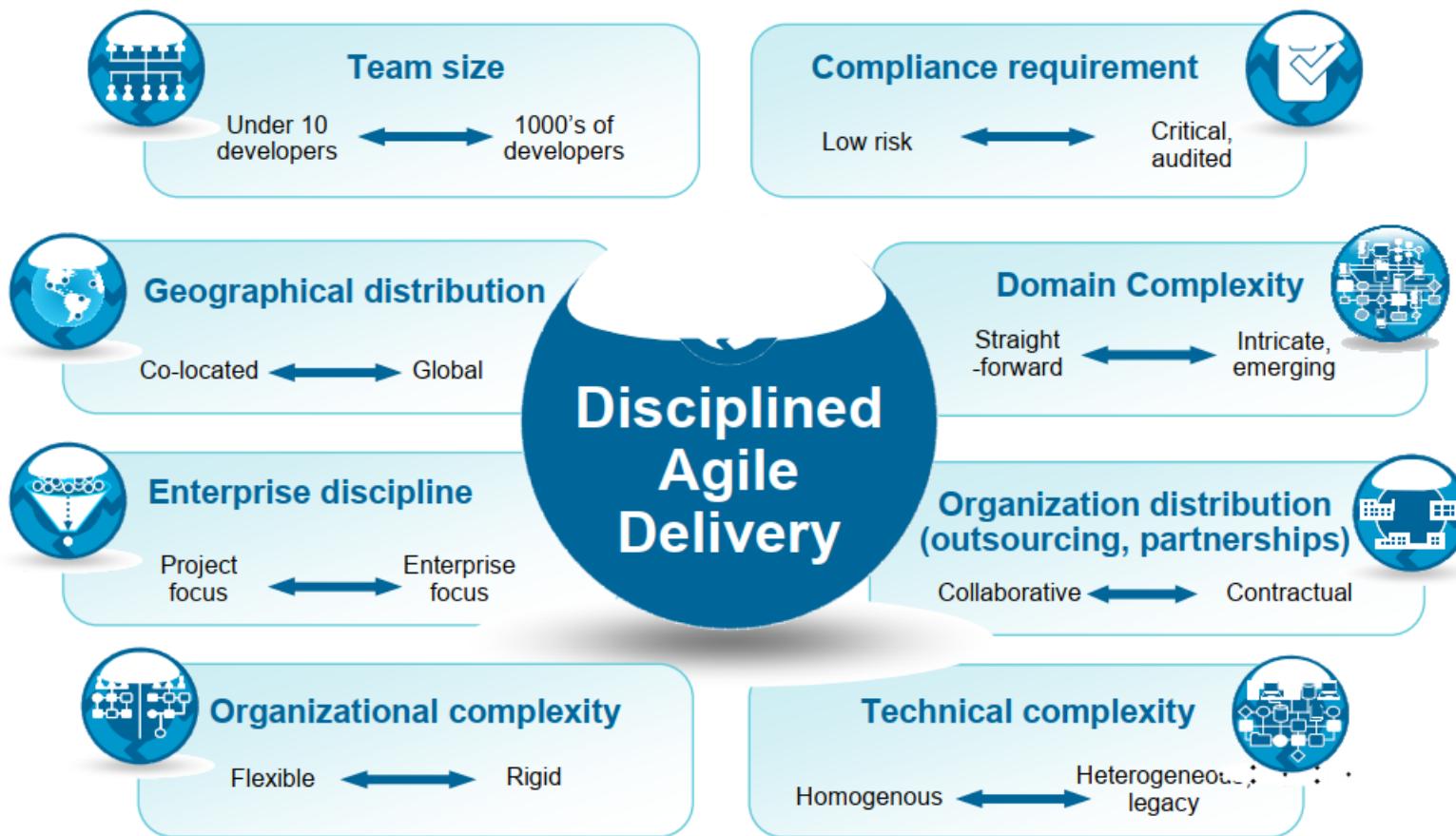
- ludzie czują się dobrze, gdy mają jasne zasady i procedury
- środowisko „linii produkcyjnej” – zadania każdej osoby są ścisłe zdefiniowane, inne osoby oczekują, że te zadania zostaną wykonane zgodnie ze specyfikacją i nie muszą wnikać w strukturę tych zadań
- w ekstremalnej postaci przybiera formę biurokratyczną
- bazują na sztywnym sposobie myślenia – fixed mindset

# Skalowanie metod zwinnych

## by Scott Ambler

### Agile Scaling Factors

IBM



Źródło: Scott W. Ambler, Agile Scaling Model, IBM Rational, 2011,  
<https://www.agilealliance.org/wp-content/uploads/2016/01/Agile-Scaling-Model.pdf>

© 2011 IBM Corporation

# Kryteria Scotta Amblera

## Geographical distribution.

What happens when the team is distributed within a building or across continents?

## Team size.

Mainstream agile processes work well for small teams (10-15), but what if the team is fifty people? One hundred people? One thousand people?

## Compliance requirement.

What if regulatory issues – such as Sarbanes Oxley, ISO 9000, or FDA CFR 21 – are applicable?

# Kryteria Scotta Amblera (2)

## **Domain complexity.**

What if the problem domain is intricate ( such as bio-chemical process monitoring or air traffic control), or is changing quickly (such as financial derivatives trading or electronic security assurance). More complex domains require greater exploration and experimentation, including but not limited to prototyping, modeling, and simulation.

**Organization distribution.** Sometimes a project team includes members from different divisions, different partner companies, or from external services firms.

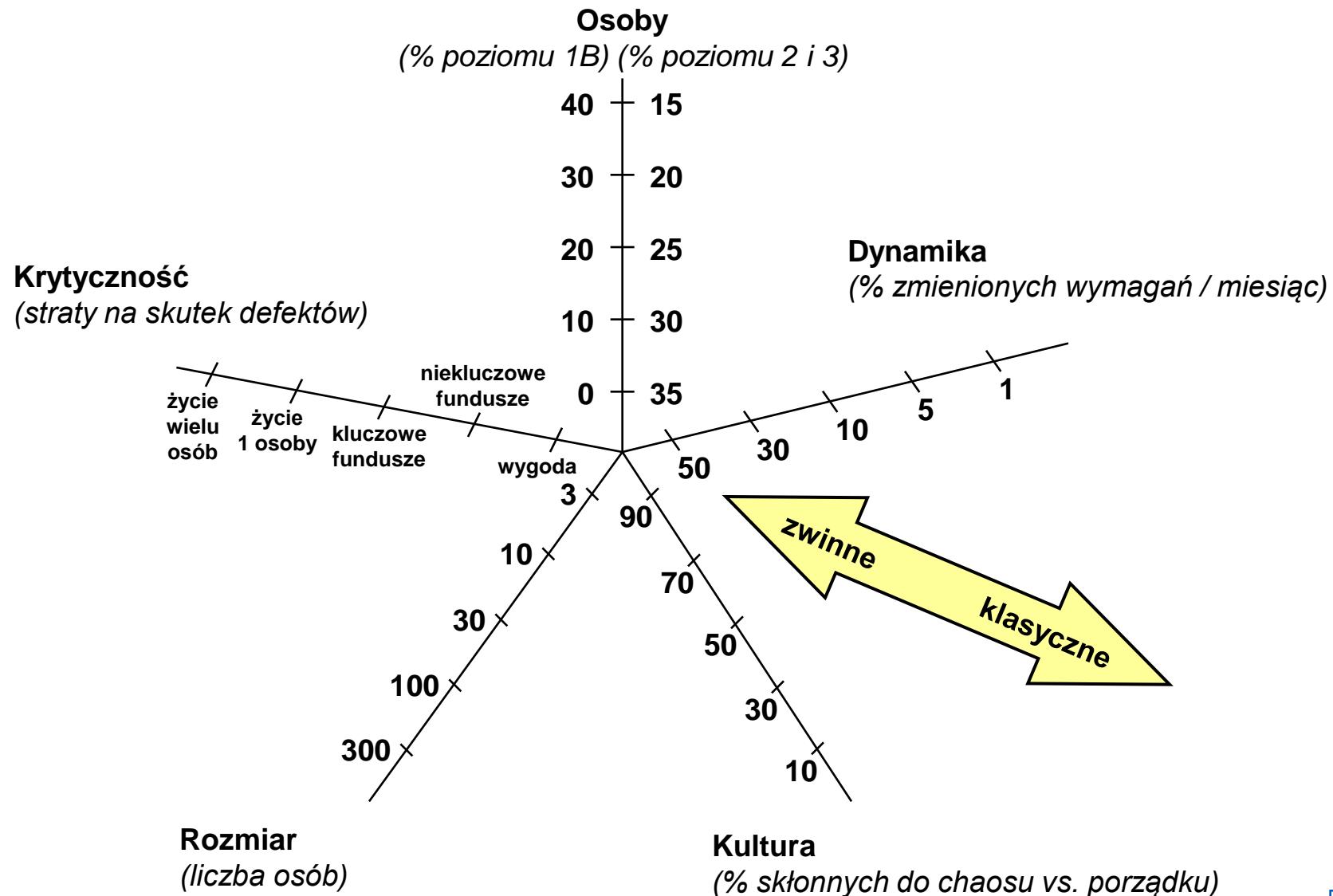
**Technical complexity.** Working with legacy systems, multiple platforms, or blending disparate technologies can add layers of technical complexity to a solution. Sometimes the nature of the problem is very complex in its own right.

# Kryteria Scotta Amblera (3)

**Organizational complexity.** The existing organizational structure and culture may reflect traditional values, increasing the complexity of adopting and scaling agile strategies. Different subgroups within the organization may have different visions as to how they should work. Individually, the strategies can be quite effective, but as a whole they simply don't work together effectively.

**Enterprise discipline.** Organizations want to leverage common infrastructure platforms to lower cost, reduce time to market, and to improve consistency. They need effective enterprise architecture, enterprise business modeling, strategic reuse, and portfolio management disciplines. These disciplines must work in concert with, and better yet enhance, the disciplined agile delivery processes.

# 5 głównych kryteriów zwinne vs klasyczne wg Boehma i Turnera



# Kryteria wyboru metodyki - opis

## ➤ Rozmiar

- metodyki zwinne: małe produkty i zespoły, trudności ze skalowaniem w górę
- metodyki klasyczne: przeznaczone do dużych produktów i zespołów, trudne do „przykrojenia” do potrzeb małych projektów

## ➤ Krytyczność

- metodyki zwinne: niesprawdzone w zastosowaniach związanych z bezpieczeństwem
- metodyki klasyczne: przeznaczone do zastosowań krytycznych (mission-critical)

## ➤ Dynamika

- metodyki zwinne: szybkozmienne otoczenie, dodatkowa praca przy małych zmianach
- metodyki klasyczne: stabilne otoczenie, dodatkowa praca przy częstych zmianach

## ➤ Osoby

- metodyki zwinne: wymagają ciągłej obecności osób na poziomie 2 i 3, osoby 1B ryzykowne
- metodyki klasyczne: wymagają dużej liczby osób poziomu 2 i 3 na początku, później mniej, włączając osoby poziomu 1B

## ➤ Kultura

- metodyki zwinne: skłonność do chaosu
- metodyki klasyczne: skłonność do porządku

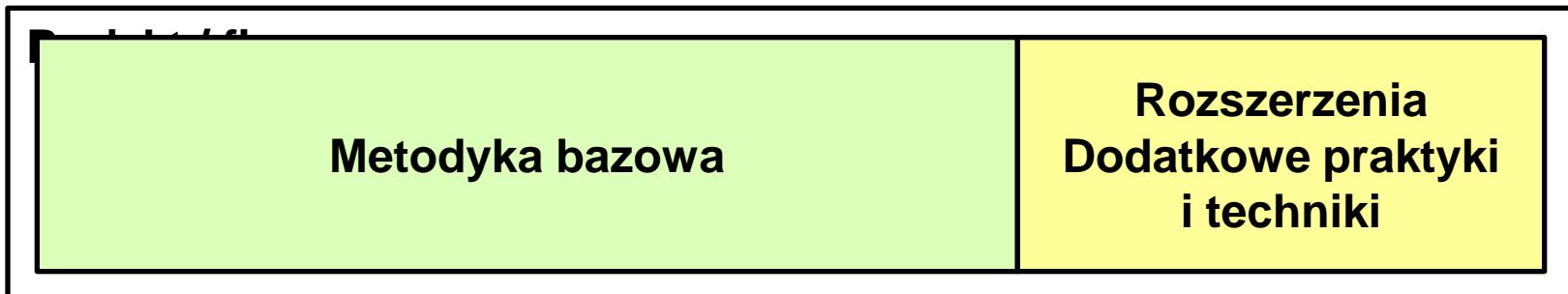
# Świadomość metodyczna

## ➤ Poziomy świadomości i głębi rozumienia metodyki wg Boehma i Turnera

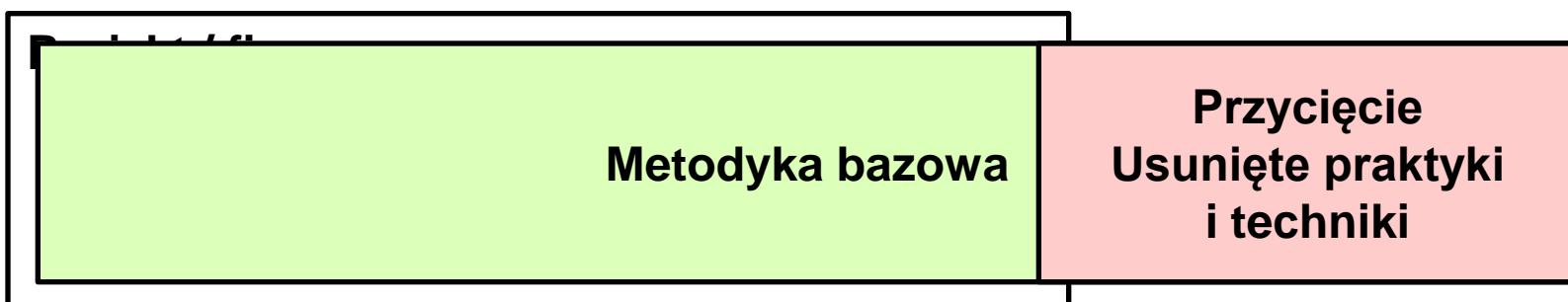
- 3 – osoba potrafi przedefiniować metodykę (złamać zasady) w nowej nieznanej sytuacji**
- 2 – osoba potrafi przyciąć metodykę i dopasować ją do znanej sytuacji**
- 1A – osoba potrafi wykonywać twórcze kroki metodyki (np. stosowanie wzorców, złożona refaktoryzacja, integracja COTS)**
- 1B – osoba potrafi wykonywać odtwórcze kroki metodyki (np. kodowanie metody, prosta refaktoryzacja, stosowanie standardów kodowania, wykonywanie testów)**
- 1 – osoba może mieć umiejętności techniczne, ale nie potrafi lub nie chce współpracować w zespole i postępować zgodnie z ustaloną metodyką**

# Adaptacja metodyki

- Metodyka bazowa jest „za mała” do potrzeb projektu / firmy
  - należy rozszerzyć metodykę - dodać nowe praktyki / techniki



- Metodyka bazowa jest „za duża” do potrzeb projektu / firmy
  - należy przyciąć metodykę – usunąć zbędne praktyki / techniki



**Uwaga na destrukcyjne adaptacje niszczące fundamenty metodyki!**

# Adaptacje Scruma w firmach informatycznych

Adaptacja	Źródło	Kategorie przyczyn	Filary	Element Scruma
Pośredniczący Właściciel Produktu	Literatura	Cechy Zespołu	Przejrzystość	Rola
Lokalny Scrum Master	Literatura	Cechy Zespołu	Inspekcja, Adaptacja	Rola
Planowanie dwóch kolejnych Sprintów	Literatura	Cechy projektu	Przejrzystość, inspekcja	Zdarzenie
Wydłużanie Sprintów	Literatura	Cechy Zespołu	Inspekcja	Zdarzenie
Podział planowania Sprintu	Literatura	Cechy Zespołu	Inspekcja, adaptacja	Zdarzenie
Historyjka dewelopera	Literatura i Firma C	Cechy projektu	Przejrzystość	Artefakt
Właściciel historyjki	Firma A	Cechy Zespołu i projektu	Przejrzystość	Rola
Zewnętrzny deweloper	Firma B	Cechy organizacji	Adaptacja	Rola
Zarządzanie Backlogiem Produktu przez Zespół Deweloperski	Firma B	Cechy zespołu	Przejrzystość, inspekcja i adaptacja	Rola
Backlog Sprintu w formie diagram Gantta	Firma D	Cechy projektu	Przejrzystość	Artefakt
Elementy Backlogu, jako część przyrostu	Firma A i Firma C	Cechy Zespołu i projektu	Przejrzystość	Artefakt
Czas pracy jak kryterium akceptacyjne	Firma A	Cechy projektu	Inspekcja	Artefakt
Testy akceptacyjne jako część Przyrostu	Firma B	Cechy projektu	Inspekcja	Artefakt
Elementy Backlogu spoza prognozowanego Przyrostu w Backlogu Sprintu	Firma A	Cechy projektu	Przejrzystość	Artefakt
Minimalna pojemność poświęcona na podnoszenie jakości produktu	Firma C	Cechy Zespołu i projektu	Nie dotyczy	Inny
Zespół Scrumowy Scrum Masterów	Firma D	Cechy projektu	Nie dotyczy	Inny

# Problemy scrumowych zespołów rozproszonych (1)

<b>Id</b>	<b>Problem</b>	<b>Rozwiązanie</b>	<b>Ocena</b>
P1	Problem z komunikacją spowodowany brakiem kontaktu twarzą w twarz	Narzędzia dodatkowe podczas rozmów (shared desktop, interactive whiteboard).	4,14
		Używanie narzędzi do rozmów oraz video konferencji (Skype, Lync, HipChat).	4,09
P2	Problem wynikający z dużej różnicy czasu pomiędzy miejscami, w których znajduje się zespół	Obarczanie rozproszeniem cały zespół – jedni zostają później, ale za to drudzy przychodzą wcześniej.	3,40
P3	Problem z jednolitym stanem wiedzy wszystkich członków zespołu	Gdy odległość nie jest zbyt duża, spotkania 2-3 razy w tygodniu na żywo.	4,12
		Całodzienna komunikacja całego zespołu, stworzenie „wirtualnego pokoju”, rozwiązywanie problemów przez wszystkich członków zespołu, aby każdy wiedział, jak doszło się do rozwiązania.	3,37
P4	Problem braku zaangażowania podczas wirtualnej rozmowy	Interakcyjne prowadzenie Scruma/rozmów przez team lidera. Zadając pytanie, co ktoś o tym myśli, zmusza do bycia na bieżąco w rozmowie.	4,07
		Prowadzenie rozmów wirtualnych z obrazem. Widać, gdy robi się coś innego, dlatego każdy stara się uczestniczyć czynnie w spotkaniu.	3,72
P5	Problem z nieprzekazywaniem wszystkich istotnych informacji	Zapisywanie najważniejszych rzeczy do przekazania i prowadzenie rozmów zgodnie z listą.	4,15
		Oznaczanie osób, które mogą być zainteresowane poruszonym tematem zarówno w dokumentach lub mailu, lepszy przepływ informacji.	3,93

Źródło: Elżbieta Wierzbicka, *Wspomaganie zastosowania Scruma w zespołach rozproszonych*, praca dyplomowa magisterska, opiekun J. Miler, WETI, PG, 2016

# Problemy scrumowych zespołów rozproszonych (2)

<b>Id</b>	<b>Problem</b>	<b>Rozwiążanie</b>	<b>Ocena</b>
P6	Problem ze zrozumieniem siebie nawzajem	Otwarta komunikacja członków zespołu; nie bać się powiedzieć, że czegoś się nie rozumie.	4,50
		Poznanie człowieka na żywo, wizyty w zespołach między sobą co najmniej raz do roku, wyjazdy integracyjne.	4,36
P7	Nieszanowanie dni wolnych przez członków zespołów z innych krajów	Przygotowywanie się wcześniej poprzez sprawdzenie świąt oraz dni wolnych w danym kraju przed ustawieniem spotkań.	4,23
		Stworzenie kalendarza, gdzie każdy z członków zespołu zaznacza swoje święta i dni wolne.	4,22
P8	Problem z różnym zaangażowaniem oraz jakością pracy członków zespołu wynikający z charakteru	Wybranie odpowiedniego team leadera dbającego o pilnowanie zaangażowania i pomaganie osobom, które mają je mniejsze niż reszta.	4,11
		Dawanie różnych zadań, stawianie nowych wyzwań, ale pod ciągłą obserwacją team leadera.	3,50
P9	Problem ze zmniejszonym zaufaniem do innych członków zespołu	Uprzejmość, miłe podejście do ludzi, budowanie dobrych relacji.	4,43
		Wyjazdy oraz poznanie osobiste całego zespołu - budowa zaufania.	4,32
P10	Problem ze spotkaniami z odbiorcą produktu	Częste wyjazdy i spotykanie się z klientem na żywo.	3,98
		Tworzenie dokumentów możliwych do edycji zarówno przez team leadera, jak i odbiorcę produktu.	3,56
P11	Problem związany z różnicą produktywnością członków zespołu	Czujność team leadera na konflikty w zespole powodujące brak zaufania między członkami zespołu i zmniejszanie ich wydajności.	4,38
		Codzienny kontakt z przełożonym i/lub częścią zespołu.	3,77

Źródło: Elżbieta Wierzbicka, *Wspomaganie zastosowania Scruma w zespołach rozproszonych*, praca dyplomowa magisterska, opiekun J. Miler, WETI, PG, 2016

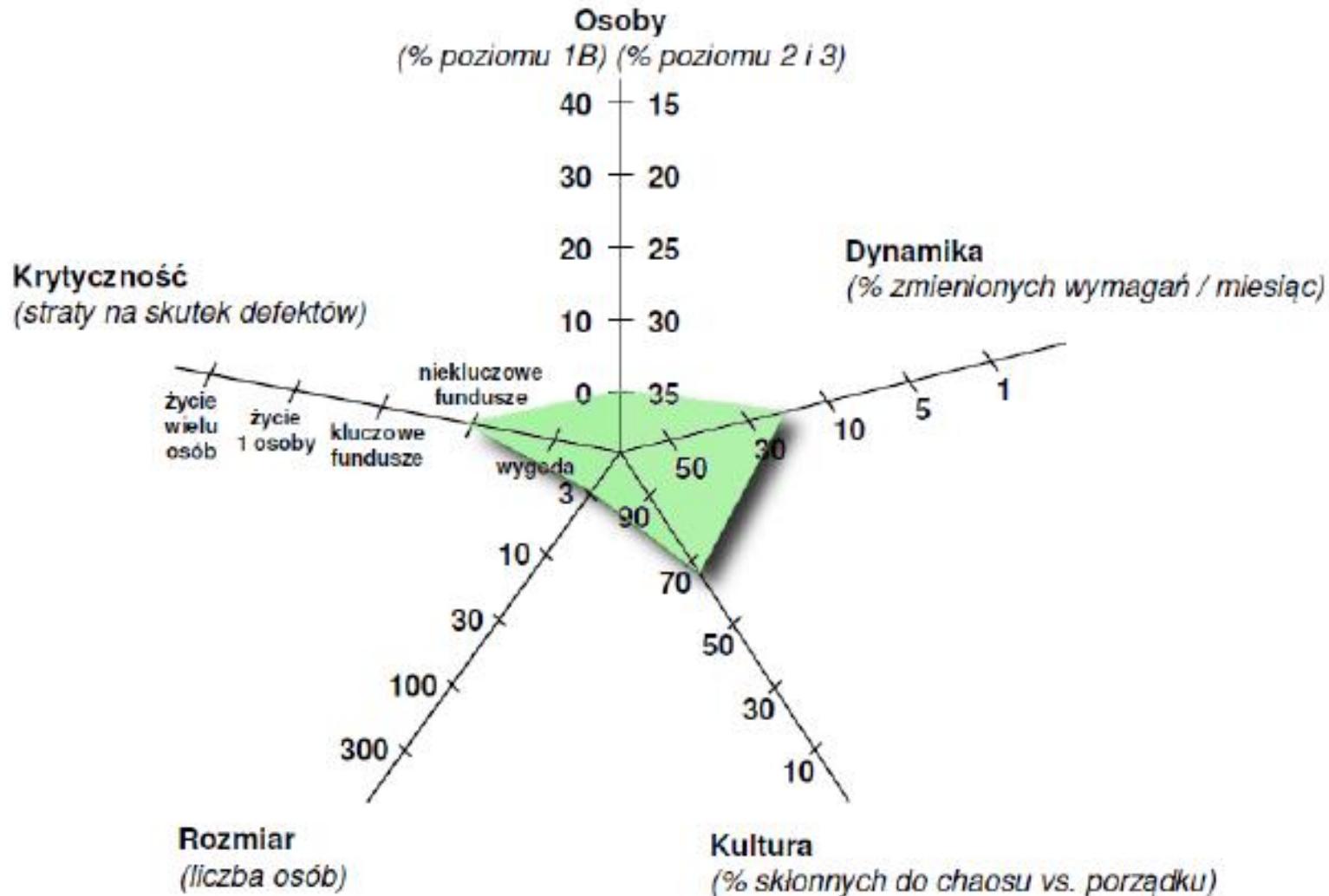
# Przykłady doboru metodyki

**Szkoła nauki jazdy – sukces zwinności**  
**RiskGuide 3.0 – Scrum z problemami**

# Szkoła nauki jazdy - zwinność

- **Projekt dyplomowy inżynierski pt.**  
*„System wspomagający komunikację w szkole nauki jazdy”*
- **Czas realizacji całego projektu dyplomowego: kwiecień 2012 – grudzień 2012**
- **Projekt realizowany zgodnie z metodyką Scrum, zespół 3-osobowy**
- **Klient/konsultant – jedna z trójmiejskich szkół jazdy**
- **Przebieg projektu**
  - kwiecień – czerwiec:
    - spotkania, formowanie zespołu, budowa infrastruktury projektu
  - lipiec – grudzień:
    - 9 sprintów (lipiec – wrzesień: 2 tygodniowe, październik – grudzień: 3 tygodniowe)
- **16 spotkań udokumentowanych notatkami, brak codziennych spotkań Scrum**

# Uzasadnienie doboru metodyki (1)



**Źródło:** M. Błaszkowski, M. Dargacz, K. Żmijewski, System wspomagający komunikację w szkole nauki jazdy. Projekt dyplomowy inżynierski, Katedra Inżynierii Oprogramowania, Politechnika Gdańskia, 2012

# Uzasadnienie doboru metodyki (2)

Kryteria wyboru:

- Osoby - uważamy, że nie ma w zespole osoby, która nie potrafiłaby przyciąć metodyki do swoich potrzeb. Kryterium to silnie wskazuje na metodyki zwinne.
- Dynamika - nie jesteśmy jeszcze w stanie przewidzieć jaka może być procentowa ilość zmienionych wymagań na miesiąc. Przyjęliśmy jednak założenie, że jesteśmy w stanie obsłużyć do 20 procent takich wymagań.
- Kultura - w zespole są 2 osoby skłonne do chaosu i jedna preferująca porządek.
- Rozmiar - nasz zespół składa się jedynie z 3 osób, a projekt będzie realizowany przez ponad 6 miesięcy.
- Krytyczność - błędy w oprogramowaniu będą mogły spowodować co najwyżej utratę nie-kluczowych funduszy klienta.

# RiskGuide 3.0 – Scrum z problemami

- **Projekt grupowy pt. „Internetowe narzędzie wspomagające zarządzanie ryzykiem w projekcie informatycznym”**
- **Czas realizacji: marzec 2010 – luty 2011**
- **Zespół projektu**
  - Właściciel produktu – Jakub Miler
  - Zespół deweloperski – 3 studentów 8 i 9 sem. studiów jednolitych magisterskich
- **Cechy projektu**
  - Istniejący sprawdzony produkt RiskGuide 2
  - Cel: nowa generacja narzędzia w nowych technologiach
  - Założenie technologii: .NET/MS SQL lub Java/PostgreSQL
  - Bardzo kompetentny i zaangażowany klient
  - Zespół deweloperski znał się wcześniej, częściowo pracował razem w projektach
  - Wysokie wymagania co do ochrony danych, ergonomii, elastyczności wspieranego procesu
  - Oczekiwany produkt o potencjale komercyjnym
  - Założony termin wdrożenia – koniec stycznia 2011

# Przebieg projektu – sem. 8

## ➤ marzec 2010

- dyskusje wizji systemu i technologii – propozycja Zespołu deweloperskiego: PHP (z PHPTal)
- udostępnienie wersji testowej RiskGuide'a 2
- wstępne zdefiniowanie zakresu projektu

## ➤ kwiecień 2010

- dwa obszerne spotkania Właściciela produktu z Zespołem deweloperskim, wyjaśnienie wymagań
- budowa infrastruktury projektu (np. Acunote)
- opracowanie *product backlog*
- określenie przez Właściciela produktu priorytetów, oszacowanie złożoności przez Zespół
- spotkanie w celu zaplanowania pierwszego sprintu

## ➤ maj, czerwiec, wrzesień 2010: Sprinty 1, 2 i 3

- sprinty 3-4 tygodniowe
- sprint 1 i 2: ok. 1/4 kluczowej funkcjonalności, bez żadnego projektu UI (zasada YAGNI)
- sprint 3: projekt i częściowa implementacja ochrony dostępu

# Przebieg projektu – sem. 9

## ➤ październik 2010: Sprinty 4 i 5

- sprinty 2 tygodniowe
- sprint 4: pewien przyrost w zakresie ochrony danych
- Cel na sprint 5: implementacja docelowego interfejsu użytkownika („skórki”)
- **Właściciel produktu orientuje się, że zespołowi brakuje SPECYFIKACJI i sam opracowuje dokładną (ale tylko tekstową) specyfikację priorytetów funkcji na interfejsie użytkownika, danych (atrybutów), formatek (raportów, kolumn, sortowania)**
- sprint 5: pierwsza wersja interfejsu użytkownika – mało udana

## ➤ listopad 2010 – grudzień 2010: Sprinty 6, 7, 8, 9

- sprinty 2 tygodniowe
- implementacja 1-3 nowych funkcji na sprint
- wiele poprawek istniejących funkcji
- powolna, stopniowa poprawa interfejsu użytkownika dla już opracowanych ekranów

## ➤ styczeń 2011 – luty 2011: Sprint 10, 11, 12, 13, 14

- sprinty maksymalnie 2 tygodniowe
- sprinty 10 i 11: implementacja pozostałych kluczowych funkcji produktu
- sprinty 12, 13, 14: poprawki, poprawki, poprawki

# RiskGuide 3.0 - produkt

## RiskGuide<sup>3</sup> Risk Management Tool

Strona główna Moje konto (Jan Nowak) Moje projekty EN Wyloguj się

RiskGuide > Przeglądanie listy projektów >>

Przeglądy Raporty Zespół

Identyfikacja Szacowanie Analiza Raport

Projekt: Przykładowy projekt  
 Przegląd: Przykładowy przegląd  
 Raport:

+ Nowy projekt

Data	Projekt	Maksymalne oszacowanie	Domyślny?
Sortuj	Sortuj	Sortuj	Sortuj
Mon, 31 Jan 2011	<b>Przykładowy projekt</b> Jan Nowak Opis przykładowego projektu Pokaż/ukryj kategorie ryzyk <b>[+]</b>	10	<input checked="" type="checkbox"/>
Mon, 31 Jan 2011	<b>Kolejny przykładowy projekt</b> Jan Nowak Opis tegoż projektu. Pokaż/ukryj kategorie ryzyk <b>[+]</b>	6	<input type="checkbox"/>

Zespół Projektowy KIO © 2010  
 Risk Guide ver 3.0

# RiskGuide 3.0 – sukces czy porażka?

## ➤ Rezultat projektu

- pokrycie większości wymagań kluczowych, bardzo niewielu ponadto
- wdrożenie opóźnione o ok. 2 miesiące
- użyta technologia to PHP (z PHPTal), baza danych MySQL – utrzymanie?
- brak testów akceptacyjnych – czy rzeczywiście działa?
- brak dokumentacji projektowej (jedynie podręczniki użytkownika i administratora)

## ➤ Problemy

- chaotyczne zaangażowanie jednego z członków Zespołu (głównego projektanta i programisty interfejsu użytkownika)
- osobne repozytorium jednej z osób – problemy ze scalaniem kodu
- nieprosty projekt, szczególnie interfejsu użytkownika – konieczne długotrwałe poprawki

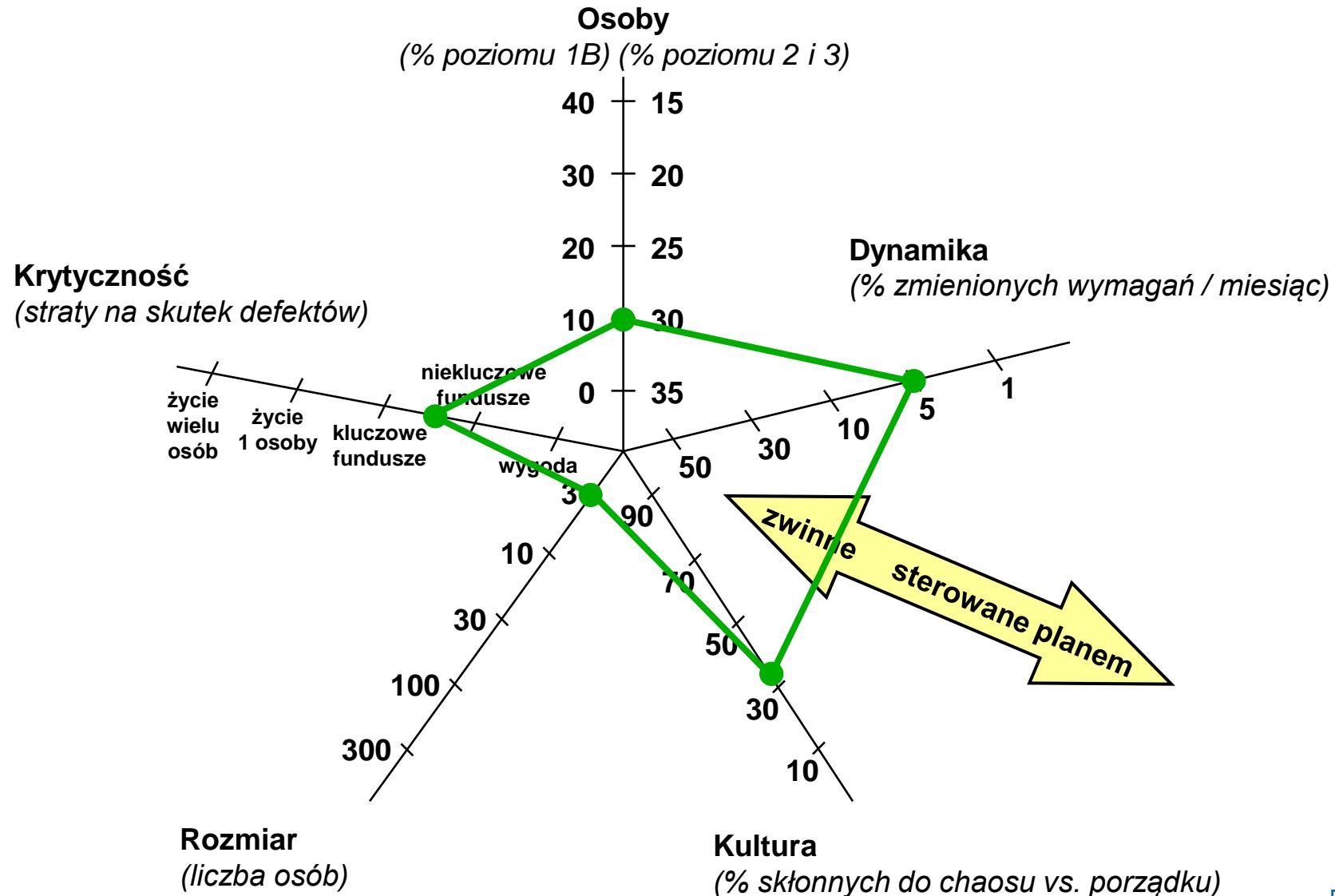
# RiskGuide 3.0 – dobór metodyki

Cecha	Metodyki zwinne	Metodyki sterowane planem
<b>Zastosowanie</b>		
Główne cele	szbka wartość, reakcja na zmiany	przewidywalność, stabilność, wysoka wiarygodność
Rozmiar	mniejsze zespoły i projekty	większe zespoły i projekty
Środowisko	niestabilne, szybkozmienne, skupione na projekcie	stabilne, wolnozmienne, skupione na projekcie i organizacji
<b>Zarządzanie</b>		
Relacja z klientem	wyznaczony klient na miejscu, skupia się na priorytetach wydań	spotkania z klientem gdy potrzeba, dostarcza ustaleń (np. do specyfikacji)
Planowanie i nadzorowanie	plany zaszyte w opisie produktu, nadzór jakościowy	jawnie dokumentowane plany, nadzór ilościowy
Komunikacja	niejawną, współdzielona wiedza	jawną, udokumentowaną wiedzą

# RiskGuide 3.0 – dobór metodyki (2)

Cecha	Metodyki zwinne	Metodyki sterowane planem
<b>Techniczne</b>		
Wymagania	nieformalne historyjki i przypadki testowe z priorytetami, nieprzewidywalne zmiany	sformalizowane wymagania, proces, dojrzałość, interfejsy, jakość, przewidywalna ewolucja
Wytwarzanie	prosty projekt, krótkie przyrosty, refaktoryzacja z założenia możliwa i tania	szczegółowy projekt, dłuższe przyrosty, refaktoryzacja zakładana jako droga
Testowanie	wykanywalne testy definiują wymagania	udokumentowane plany i procedury testowania
<b>Osoby</b>		
Klient	wyznaczone, dostępne na miejscu osoby klasy CRACK	osoby klasy CRACK, nie muszą być stale dostępne
Wykonawcy	co najmniej 30% pełnoetatowych ekspertów poziomu 2 i 3, bez osób poziomu 1B i -1	50% osób poziomu 3 na początku projektu, potem 10%; może być do 30% osób poziomu 1B, bez osób poz. -1
Kultura	zadowolenie i moc poprzez wiele stopni swobody (skłonność do chaosu)	zadowolenie i moc poprzez zestaw zasad i procedur (skłonność do porządku)

# RiskGuide 3.0 – dobór metodyki (3)



# RiskGuide 3.0 – wnioski

- **Scrum nie był dobrą decyzją**
  - były stabilne wymagania
  - zespół nie w pełni znał technologię
  - występowały duże problemy komunikacyjne w zespole
- **Przy stabilnych wymaganiach konieczna jest szczegółowa specyfikacja na początku projektu**
- **Wspólna infrastruktura to absolutna podstawa współpracy w zespole**
- **Zasada prostego projektu i YAGNI wymaga dobrej znajomości technologii**
- **Lepsze są krótsze iteracje**