SYSTEMY INFORMATYCZNE

ćwiczenia praktyczne
12.03.2019

Piotr Łukasik

p. 373
email: plukasik@agh.edu.pl / lukasik.pio@gmail.com
www.lukasikpiotr.com

Zakres tematyczny implementacji projektu informatycznego

1. Identyfikacja i definicja
2. Projektowanie i implementacja
3. Zarządzenie
4. Zapewnienie jakości

Klient chce coś, jak to “coś” zrozumieć

Co warto wiedzieć przed rozpoczęciem projektu

- cele projektu,
- zakres
- koszty/czas trwania
- pracownicy
- metodyka projektu
- jakość projektu, dokumentacja

Co jest ważne dla klienta?

1. Czas
2. Koszty

Rozwój oprogramowania jako “Wielki wybuch”

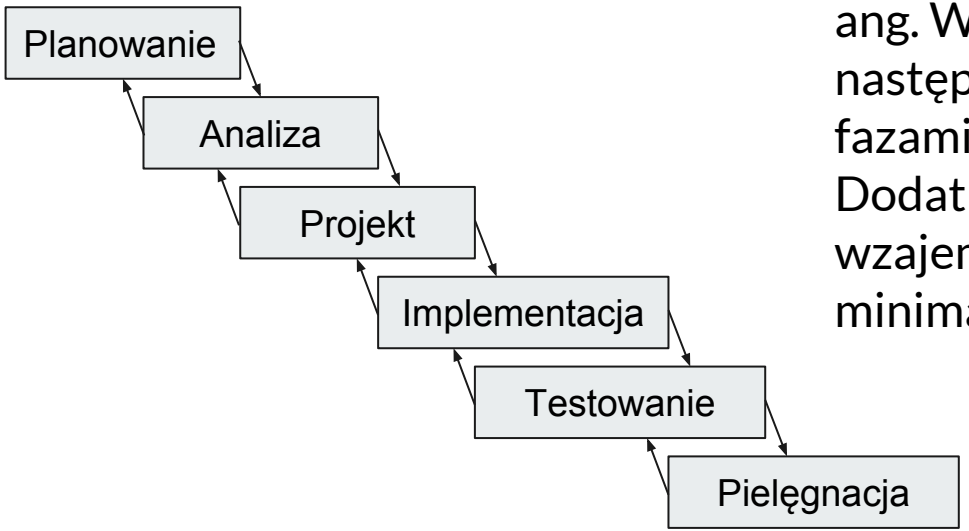
1. Pobieżne zebranie wymagań
2. Tworzenie programu
3. Dostarczenie programu

Efekt?

Fazy projektów i modele konstrukcji systemów oprogramowania

1. Analiza wymagań użytkownika => model logiczny
2. Projektowanie systemu => model fizyczny
3. Implementacja systemu
4. Testowanie i usuwanie błędów
5. Utrzymanie systemu

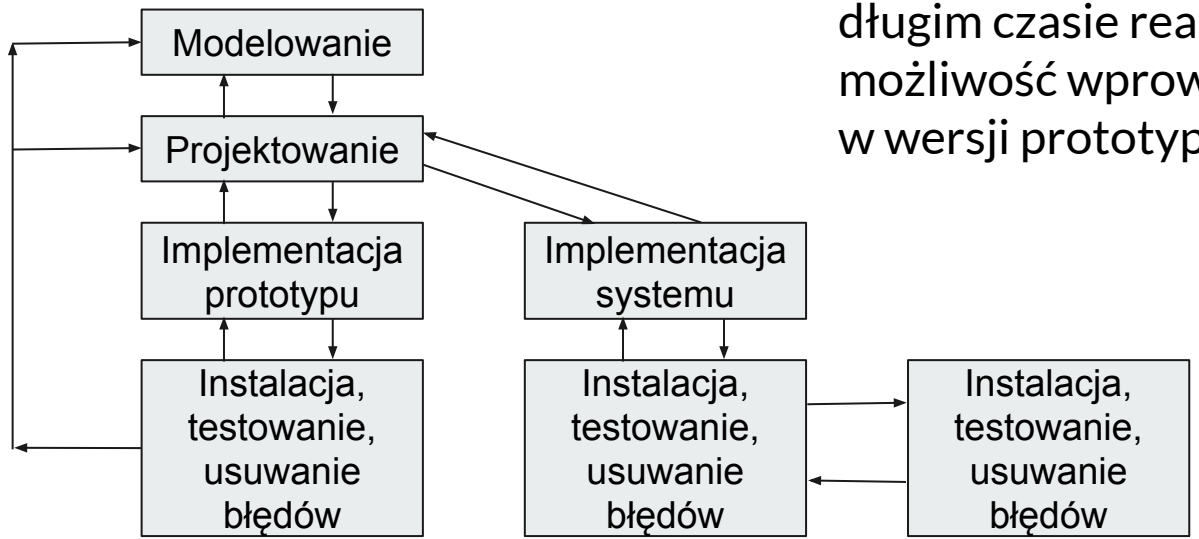
Model kaskadowy



ang. Waterfall cycle model - zwroty następują tylko pomiędzy sąsiednimi fazami konstrukcji systemu. Dodatkowo sąsiednie fazy nakładają się wzajemnie w czasie w stopniu minimalnym.

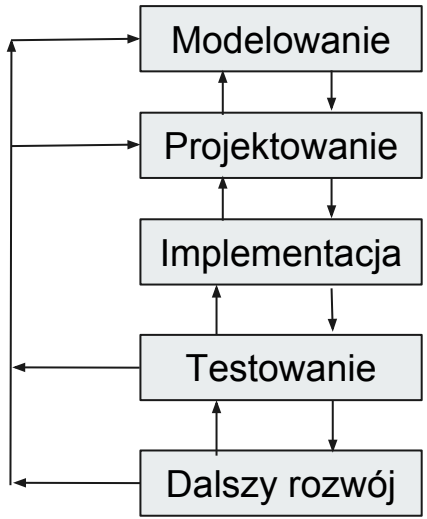
https://pl.wikipedia.org/wiki/Model_kaskadowy

Model ewolucyjny



- dostarczenie niepełnej wersji systemu w krótkim czasie
- preferowany przy projektach o długim czasie realizacji, kiedy jest możliwość wprowadzenia systemu w wersji prototypu

Model iteracyjny



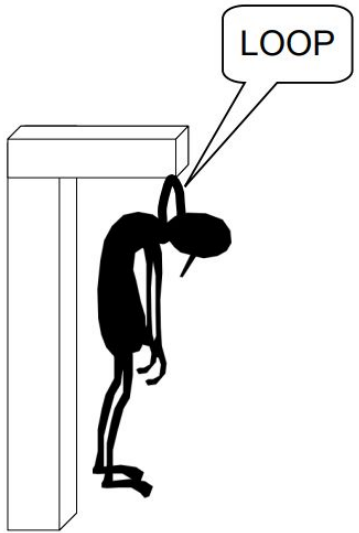
- dostosowany do niepełnych wymagań przy starcie projektu
- pozwala na swobodne przejścia pomiędzy fazami projektu

Jak wybrać?

1. Własny użytek - programowanie odkrywcze
2. Wymaganie dobrze zdefiniowane, mamy doświadczenie w tego typu projektach - model kaskadowy
3. Duża niepewność związana z wytwarzaniem produktu - podejście ewolucyjne
4. Źle określone wymagania, obawy odnośnie powodzenia projektu - prototyp

Dlaczego pojawiają się problemy i niepowodzenia?

- Syndrom LOOP
  - Late
    Późno
  - Over budget
    Przekrocznoy budżet
  - Overtime
    Nadgodziny
  - Poor quality
    Kiepska jakość



http://wazniak.mimuw.edu.pl/images/3/31/Io-12-wyk-bw.pdf

Manifest programowania zwinnego (Agile)

Odkrywamy nowe metody programowania dzięki praktyce w programowaniu i wspieraniu w nim innych.

W wyniku naszej pracy, zaczęliśmy bardziej cenić:

Ludzi i interakcje	od	procesów i narzędzi
Działające oprogramowanie	od	szczegółowej dokumentacji
Współpracę z klientem	od	negocjacji umów
Reagowanie na zmiany	od	realizacji założonego planu.

Oznacza to, że elementy wypisane po prawej są wartościowe, ale większą wartość mają dla nas te, które wypisano po lewej.

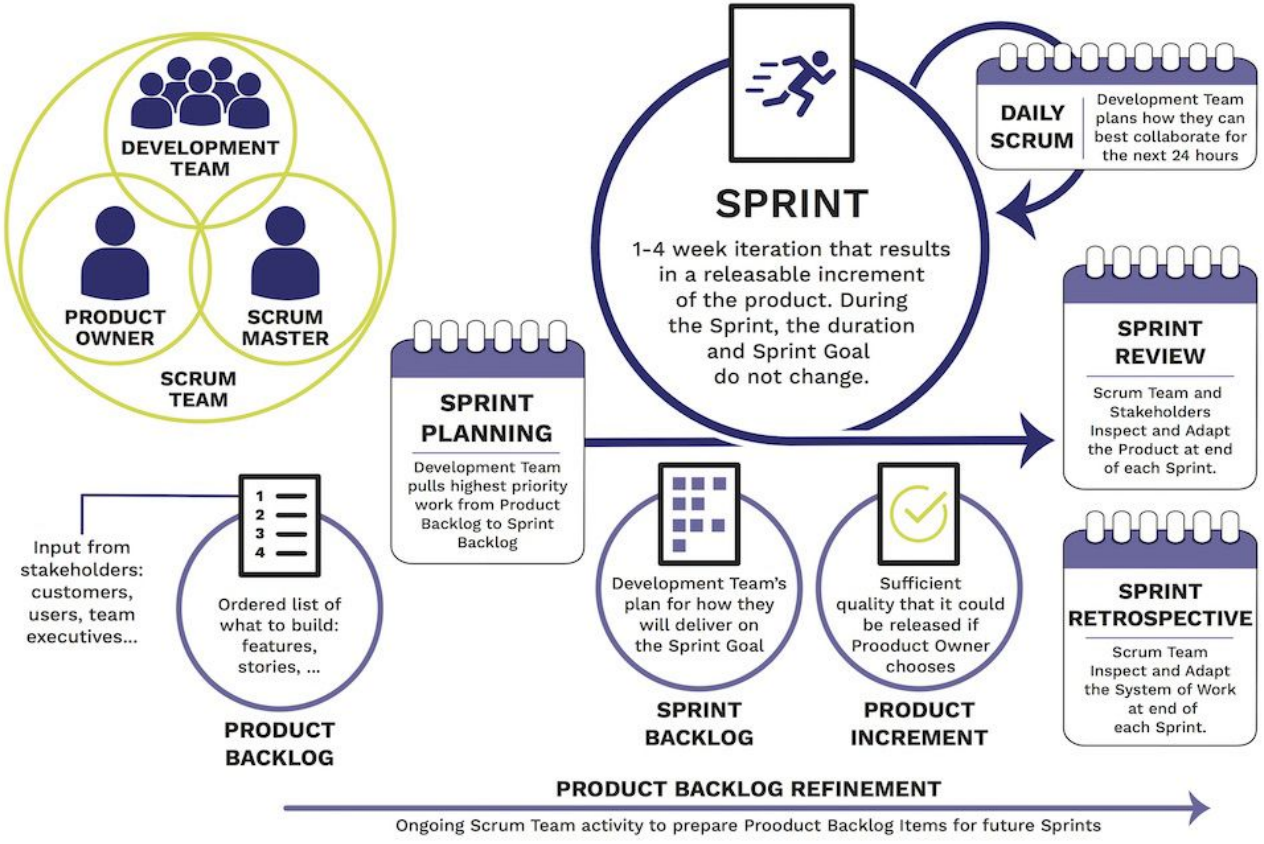
Agile Software Development

(Lekkie metodyki wytwarzania oprogramowania)

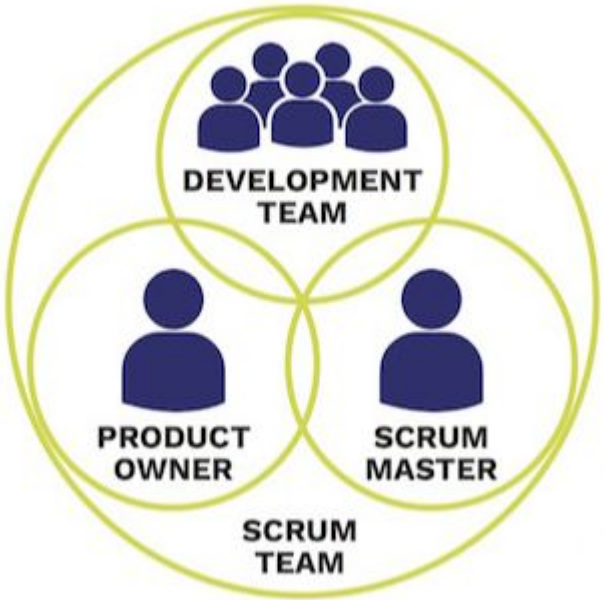
1. **Programowanie ekstremalne (XP)** - wydajne tworzenie małych i średnich “projektów wysokiego ryzyka”.
2. **Feature Driven Development (FDD)** - w krótkich iteracjach dostarczane są wersje produktu zawierających wybrany zestaw cech.
3. **Test-driven development (TDD)** - pisanie testów do funkcjonalności, która nie została jeszcze napisana.

THE SCRUM FRAMEWORK AT A GLANCE

AGILEFORALL



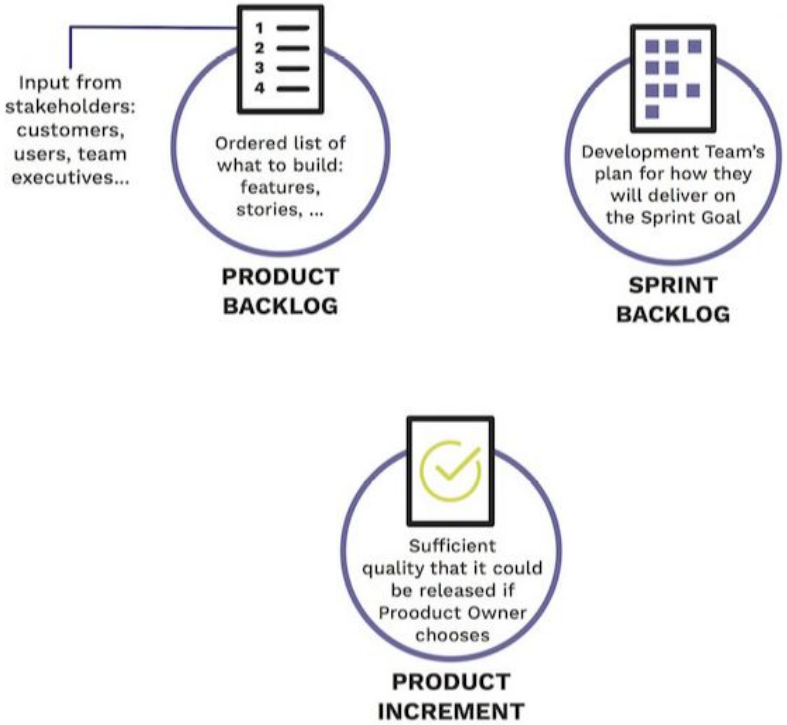
https://agileforall.com/resources/introduction-to-agile/



Role w Scrum

- **Product Owner** odpowiada za podejmowanie decyzji co do rozwoju produktu i wykorzystaniu czasu Development Team tak, żeby zbudować jak największą wartość.
- **Development Team** odpowiada za planowanie, organizację i wykonanie pracy.
- **Scrum Master** odpowiada za wprowadzenie Scrum w życie, zrozumienie i przestrzeganie zasad frameworka. Wspiera zespół poprzez facylitację i usuwanie przeszkód (ang. impediments).

https://www.qagile.pl/scrum/scrum-opis/



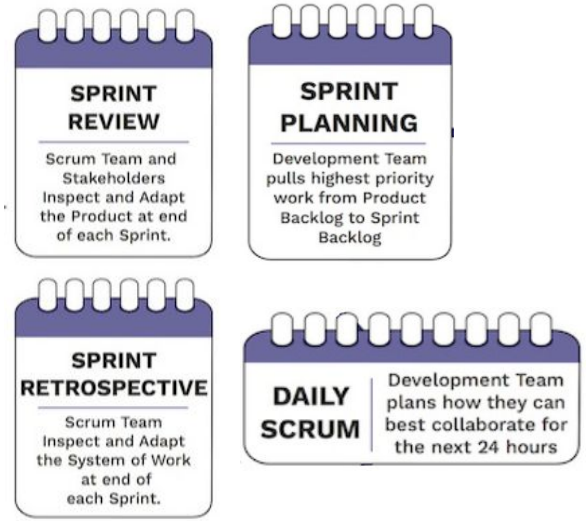
Artefakty w Scrum

- **Product Backlog** — lista rzeczy do zrobienia dla produktu, jedyne miejsce, z którego Development Team bierze pracę. Product Backlog powinien być zawsze aktualny, żeby jasno pokazywać ile jeszcze jest rzeczy do zrobienia dla produktu;
- **Sprint Backlog** — lista rzeczy do zrobienia w tym Sprincie. Sprint Backlog powinien być zawsze aktualny, żeby jasno pokazywać jaki jest postęp pracy.
- **Product Increment** — ukończony, przetestowany i zintegrowany przyrost funkcjonalności i właściwości produktu, który pokazuje co jest faktycznie zrobione i jak to wygląda.

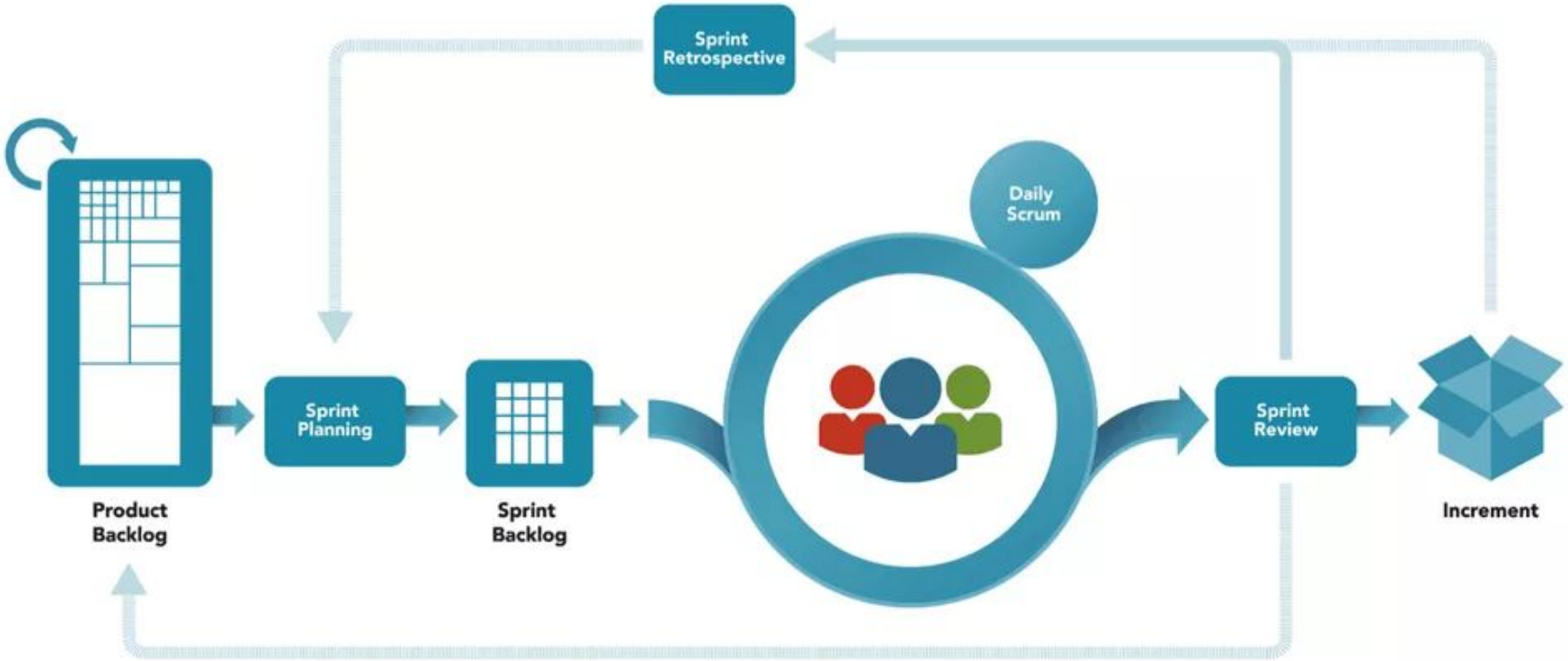
https://www.qagile.pl/scrum/scrum-opis/

Wydarzenia w Scrum

- **Sprint** — iteracja nie dłuższa niż jeden miesiąc, w ramach której zawarte są pozostałe wydarzenia;
- **Sprint Planning** — na początku każdego Sprintu patrzymy co jest najbardziej wartościową rzeczą do zrobienia przez Development Team i prognozujemy jak będzie wyglądała praca w tej iteracji;
- **Daily Scrum** — codziennie Development Team spotyka się, żeby sprawdzić stan pracy i wspólnie zdecydować co należy dalej zrobić;
- **Sprint Review** — pod koniec Sprintu sprawdzamy jak wygląda przyrost produktu i zbieramy feedback, żeby interesariusze mieli jasny obraz faktycznego postępu i dostosować wymagania do ich oczekiwań;
- **Sprint Retrospective** — na koniec Sprintu zatrzymujemy się na chwilę, przyglądamy się procesowi, narzędziom, interakcjom, żeby zdecydować jak usprawnić proces;



https://www.qagile.pl/scrum/scrum-opis/



https://www.qagile.pl/scrum/scrum-opis/

Ważne pojęcia w Scrum

- Burn-down Chart
- Burn-up Chart
- Daily Scrum
- Definition of Done
- Product Backlog
- Product Backlog refinement
- Scrum Board
- Scrum Master
- Sprint
- Sprint Goal
- Sprint Planning
- Sprint Retrospective
- Stakeholder
- Velocity

Jak teoria ma się do praktyki