Opracowanie		
MPiSO	25 IV 2019	Kolokwium 1

Spis treści

1 Zagadnienia

1. Typy relacji pomiędzy zadaniami w harmonogramie + przykłady z projektów informatycznych
2. Cykle życia oprogramowania - opis + różnice pomiędzy poszczególnymi cyklami

2 Źródła

Opracowanie na podstawie:

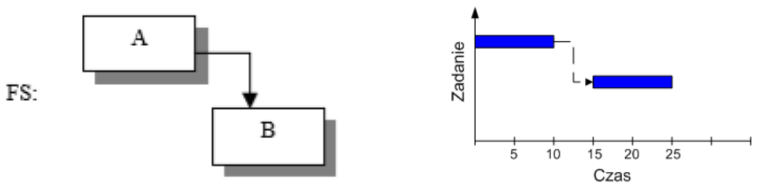
- Zarządzanie projektem informatycznym, Kazimierz Frączkowski -> doku wiki
- `https://www.pmbypm.com/finish-to-start-relationship/`
- `https://www.pmbypm.com/start-to-start-relationship/`
- `https://www.pmbypm.com/finish-to-finish-relationship/`
- `https://www.pmbypm.com/start-to-finish-relationship/`
- `https://existek.com/blog/sdlc-models/`

3 Opracowanie zagadnień

3.1. Typy relacji

3.1.1. Finish to start

Koniec – Start (ang. Finish-to-Start FS) – zadanie B nie może rozpocząć się przed ukończeniem zadania A. Oznaczenie na harmonogramie:



(a) (b)

Rysunek 1: Finish to start

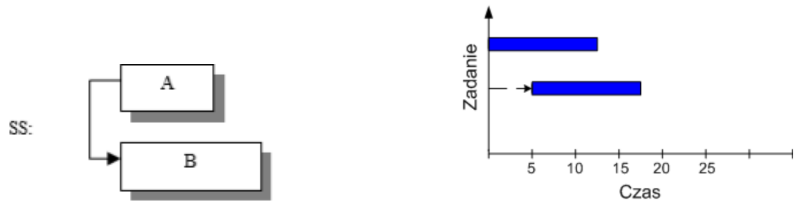
Przykłady z projektu informatycznego:

- Projekt interfejsu / prototyp interfejsu
- Napisanie user guide / wydrukowanie user guide
- Zebranie wymagań / podpisanie umowy dotyczącej wymagań

- Podaj definicję, graficzną reprezentację oraz 3 przykłady relacji Finish-to-Start.

3.1.2. Start to start

Start – Start (ang. Start-to-Start SS) – zadanie B nie może rozpocząć się przed rozpoczęciem zadania A. Oznaczenie na harmonogramie:



(a) (b)

Rysunek 2: Start to start

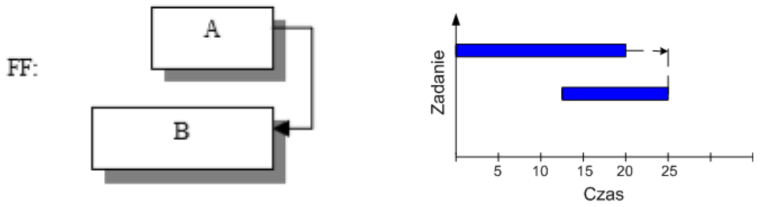
Przykłady z projektu informatycznego:

- Pisanie dokumentu HTML / pisanie pliku CSS
- Uzyskanie danych z REST API / obróbka danych z REST API
- Pisanie kodu / Pisanie dokumentacji

- Podaj definicję, graficzną reprezentację oraz 3 przykłady relacji Start-to-Start.

3.1.3. Finish to finish

Koniec – Koniec (ang. Finish-to-Finish FF) – zadanie B nie może zakończyć się dopóki nie zakończy się zadanie A.
Oznaczenie na harmonogramie:



(a) (b)

Rysunek 3: Finish to finish

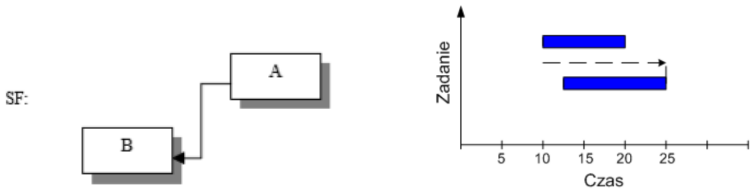
Przykłady z projektu informatycznego:

- Napisanie kodu dla modułu X / testowanie jednostkowe modułu X
- Lutowanie połączeń / Testowanie połączeń
- Wdrożenie systemu / Zarządzanie projektem

- Podaj definicję, graficzną reprezentację oraz 3 przykłady relacji Finish-to-finish.

3.1.4. Start to finish

Start – Koniec (ang. Start-to-Finish SF) – zadanie B nie może zakończyć się dopóki nie rozpocznie się zadanie A.



(a) (b)

Rysunek 4: Start to finish

Przykłady z projektu informatycznego:

- Wypuszczenie nowej wersji systemu / Likwidacja starego systemu
- Publiczna reklama systemu / Opracowanie strategii marketingowej
- Dostosowanie do potrzeb niepełnosprawnych / Otrzymanie certyfikatu bezpieczeństwa

- Podaj definicję, graficzną reprezentację oraz 3 przykłady relacji Start-to-finish.

3.2. Cykle życia oprogramowania

Software Development Life Cycle (SDLC), czyli cykl życia oprogramowania to model określający fazy tworzenia i funkcjonowania oprogramowania, a także ich przebieg. SDLC powinno być odpowiednio dobrane do wykonywanego projektu.
Do najbardziej znanych SDLC należą:

- Waterfall model (kaskadowy)
- V-shaped model
- Evolutionary model (ewolucyjny)
- Spiral Method (spiralny)
- Iterative and Incremental (Iteracyjny i przyrostowy)
- Agile development

3.2.1. Fazy tworzenia oprogramowania

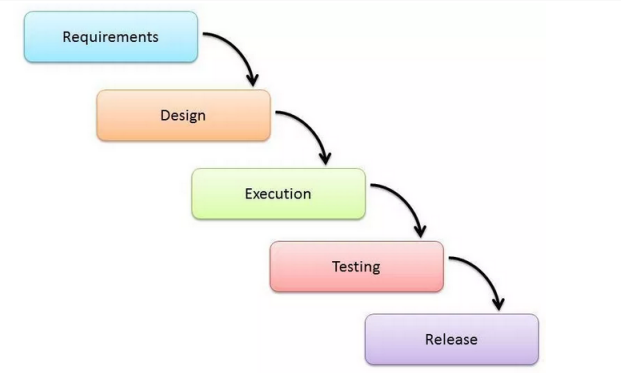
Niezależnie od wyboru SDLC w procesie tworzenia oprogramowania obecnych jest kilka faz. Znajomość tych faz pozwala na lepsze zrozumienie każdego SDLC oraz różnic pomiędzy nimi.

- Określenie wymagań
- Projektowanie
- Implementacja
- Testowanie
- Wdrożenie

3.2.2. Waterfall model

Definicja - Waterfall model

W tym modelu każda z faz tworzenia oprogramowania zaczyna się dopiero wtedy gdy poprzednia została zakończona.



Rysunek 5: Model kaskadowy

Wady:

- Trudność wprowadzania zmian w projekcie
- Kosztowny i wymagający dużo czasu

Zalety:

- Łatwy do wyjaśnienia
- Pozwala dokładnie zaplanować projekt

- Opisz model kaskadowy, podaj dwie wady i zalety.

3.2.3. Evolutionary model

Definicja - Evolutionary model

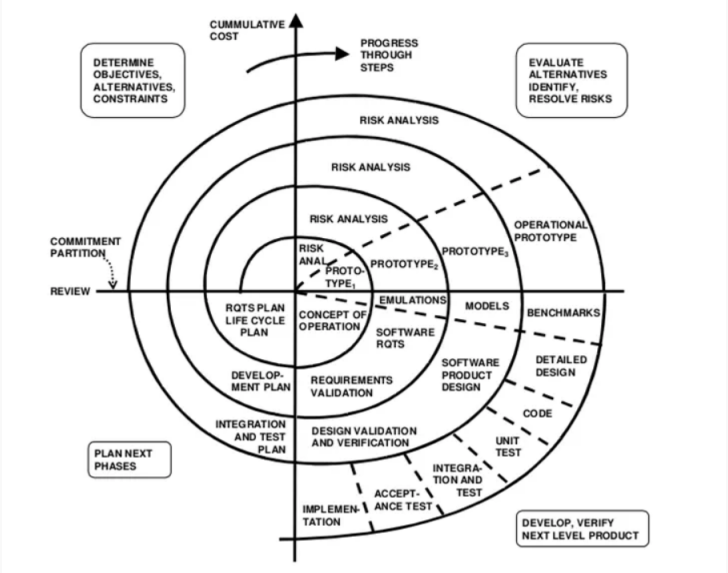
Celem modelu ewolucyjnego jest poprawienie modelu kaskadowego poprzez rezygnację ze ścisłego, liniowego następstwa faz
Pozostawia się te same czynności, ale pozwala na powroty, z pewnych faz do innych faz poprzedzających
Tym samym umożliwia się adaptowanie do zmian w wymaganiach i korygowanie popełnionych błędów (oba zjawiska występują w niemal wszystkich praktycznie wykonywanych projektach – stąd model ewolucyjny jest bardziej realistyczny od kaskadowego)

- Czym różni się model ewolucyjny od kaskadowego ?

3.2.4. Spiral model

Definicja - Spiral model

W tym modelu oprogramowanie przechodzi iteracyjnie przez cztery fazy: rozwój oprogramowania, przewidywanie ryzyka, planowanie, weryfikacja. W odróżnieniu od innych modeli duży nacisk kładziony jest na szacowanie ryzyka.



Rysunek 6: Model spiralny

Wady:

- Wymagana umiejętność szacowania ryzyka
- Kosztowny i wymagający dużo czasu

Zalety:

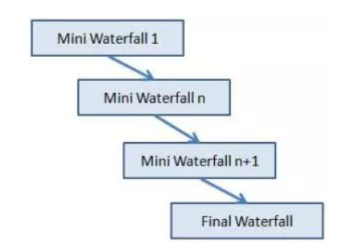
- Szansa realnej estymacji (czasu, kosztów itp.) ze względu na wczesną eliminację ryzyka

- Opisz model spiralny, podaj wady i zalety.

3.2.5. Incremental model

Definicja - Model przyrostowy

Jest to model iteracyjny, który polega na realizacji środkowych (planowanie raz na początku i deployment raz na końcu) faz modelu kaskadowego, ale w krótszych odstępach czasowych.



Rysunek 7: Model przyrostowy

Wady:

- Wymaga dużego zaangażowania klienta
- Sztywno zdefiniowana kolej wykonywania procesów

Zalety:

- Możliwość wprowadzania zmian w projekcie
- Łatwa identyfikacja błędów i ich naprawa

- Opisz model przyrostowy, podaj dwie wady i zalety.