METODYKI WYTWARZANIA

Inżynieria Oprogramowania

Wojciech Starzyk
Wiktor Szyszka
Paweł Socała

AGH

Czym jest metodyka wytwarzania?

Metodyką wytwarzania możemy określić zbiór pojęć, notacji, modeli, języków, technik i sposobów, które zapewniają efektywne i skuteczne tworzenie oprogramowania. Inaczej określany mianem “framework’u” jest podstawą działania zespołów deweloperów w zakresie zarządzania procesami: tworzenia, testowania, wdrażania i utrzymywania oprogramowania.

Elementy metodyki

- **Fazy projektu**
  oraz role jego uczestników
- **Modele**
  tworzone podczas każdej fazy
- **Scenariusze**
  postępowania każdej fazy
- **Reguły przechodzenia**
  pomiędzy fazami
- **Używana notacja**
  pomiędzy fazami
- **Dokumentacja każdej fazy**

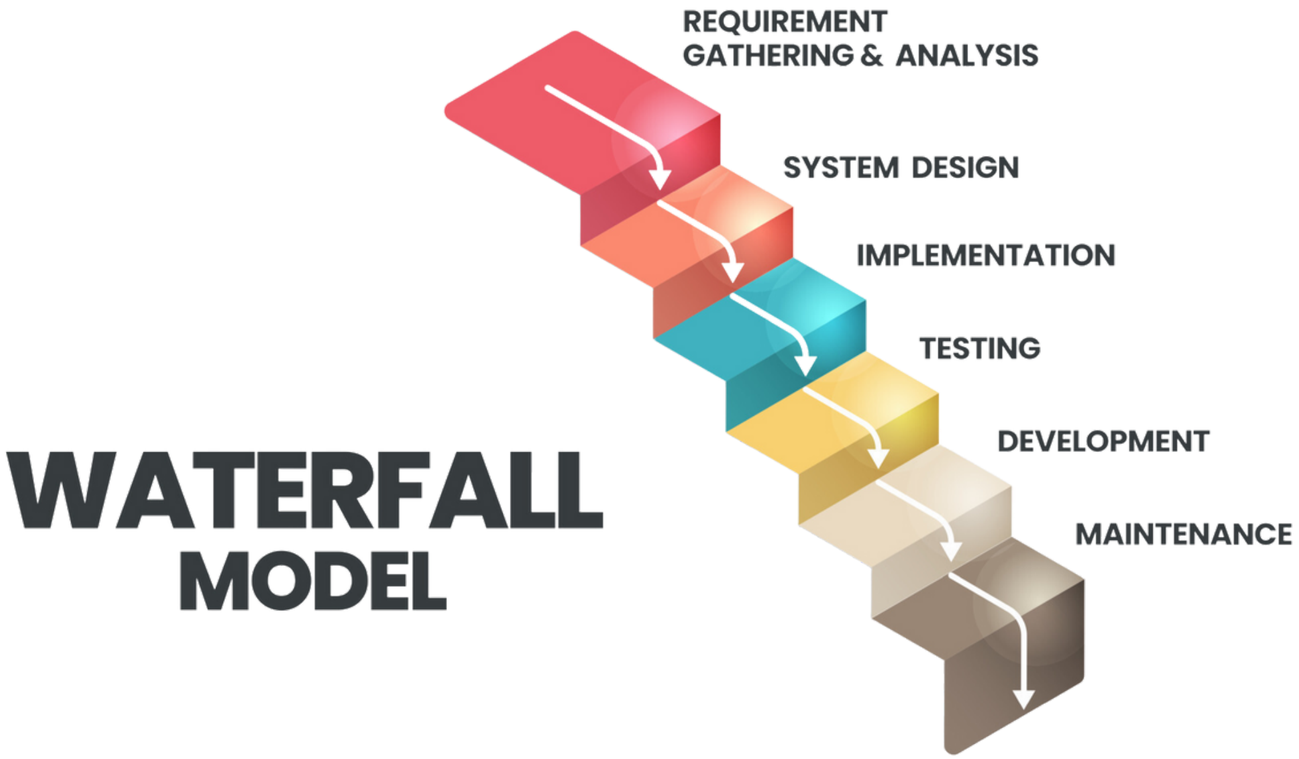
Rodzaje metodyk wytwarzania

Najważniejsze metodyki wytwarzania to

- **Model kaskadowy**
- **Model prototypowy**
- **Model przyrostowy**
- **Programowanie zwinne**
- **Model spiralny**

Model kaskadowy

Waterfall Model



Podstawowy model wytwarzania składający się z następujących kolejno po sobie, odrębnych faz projektowych. Każda z nich jest “kaskadą”, skąd pochodzi nazwa, oraz powszechnie używane zobrazowanie modelu w postaci schodków (bądź wodospadu z ang. “waterfall model”).

Zalety

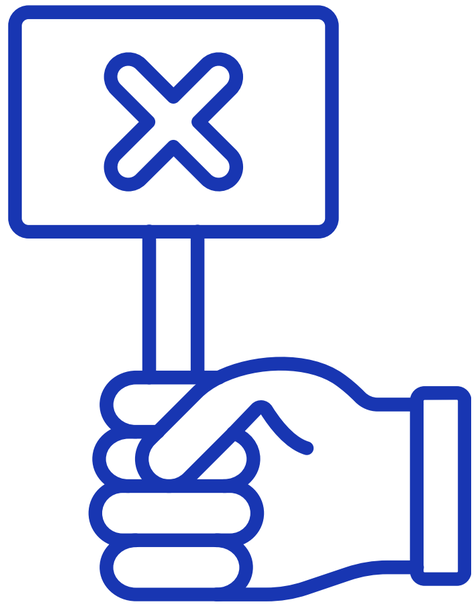
Dobry początek

Wymagania są zazwyczaj dobrze zdefiniowane na początku projektu.

Przyjazność

Struktura jest łatwa do zrozumienia i nadzorowania.

Wady



Specyfikacja

Wymaga pełnej specyfikacji na początku projektu, co może być trudne w niektórych przypadkach.

Dostosowanie

Trudności w dostosowywaniu się do zmieniających się wymagań po rozpoczęciu implementacji.

Przykład

Model kaskadowy

Załóżmy, że firma planuje stworzyć prosty system do zarządzania zadaniami online. Wymagania zostały dokładnie zdefiniowane na początku projektu, a klient nie przewiduje znaczących zmian w trakcie realizacji.

1. Analiza (Requirements)

W tym etapie analizowane są szczegółowe wymagania klienta dotyczące systemu do zarządzania zadaniami. Definiowane są funkcje, interfejsy użytkownika oraz wszelkie inne istotne aspekty systemu.

2. Projektowanie (Design)

Na podstawie zebranych wymagań projektanci tworzą dokładne specyfikacje systemu, w tym struktury danych, architekturę, interfejsy użytkownika i inne elementy projektu.

3. Implementacja (Implementation)

Programiści rozpoczynają pracę nad kodem na podstawie specyfikacji projektowych. Tworzone są poszczególne moduły systemu, które następnie są integrowane w całość.

4. Testowanie (Testing)

Po zakończeniu implementacji, zespół testuje system w celu upewnienia się, że spełnia on wszystkie założone wymagania i działa poprawnie.

5. Wdrożenie (Deployment)

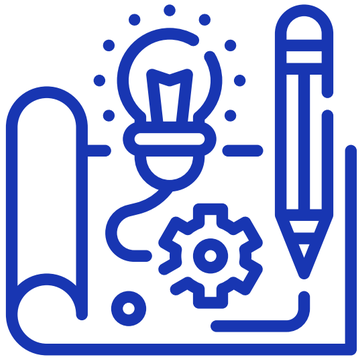
Gdy system zostanie przetestowany i zaakceptowany przez klienta, następuje jego wdrożenie na docelowych środowiskach produkcyjnych.

6. Konserwacja (Maintenance):

Po wdrożeniu, system jest poddawany konserwacji, co obejmuje rozwiązywanie ewentualnych błędów, aktualizacje i wsparcie techniczne.

Model prototypowy

Model prototypowy polega na wczesnym tworzeniu prototypów aplikacji do łatwiejszego zrozumienia wymagań systemu, aby uniknąć kosztów zmian w projekcie w przypadku błędnego ich zrozumienia.



Starodawnie

Rozpisanie lub wizualizacja interfejsów na kartce

Modułowa

Implementacja jedynie kilku modułów

Idea implementacji

Metod działające w większości przypadków w celu pokazanie idei.

Kreatory

Szybsze stworzenie interfejsów

Zalety

Łatwość edycji

Prototyp jest łatwy do zmiany. Zwiększa zrozumienie programistów co do potrzeb klienta

Przejrzystość

Pozwala klientowi zobaczyć jak mniej więcej system będzie wyglądał.

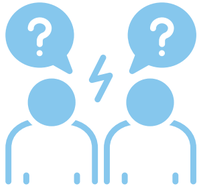
Szybkość przystosowania

W zależności od rodzaju prototypu, może pozwalać rozpocząć szkolenie obsługi systemu po stronie klienta.

Wady

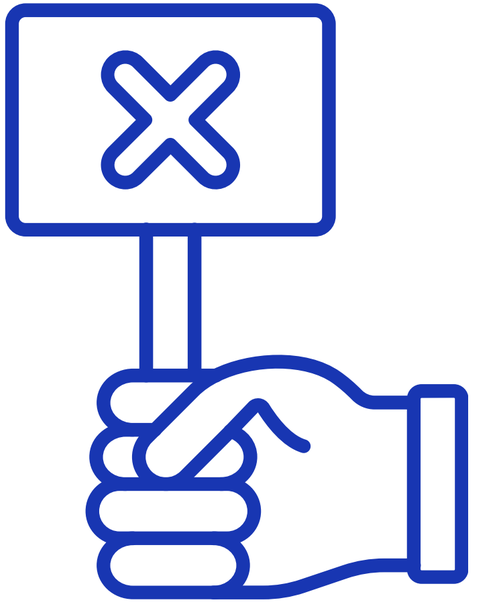
Pieniądze

Wysoki koszt budowy systemu



Podekscytowanie

Możliwość nieporozumień z klientem (klient widzi prawie gotowy produkt, który w rzeczywistości jest dopiero w początkowej fazie rozwoju).



Przykład

Model prototypowy

Załóżmy, że firma planuje stworzyć prosty system do zarządzania zadaniami online. Wymagania zostały dokładnie zdefiniowane na początku projektu, a klient nie przewiduje znaczących zmian w trakcie realizacji.

1.Analiza wstępna (Initial Analysis)

Na tym etapie przeprowadzana jest rozmowa z klientem, aby zrozumieć ogólne wymagania i cele projektu. Może to obejmować dyskusję na temat funkcji, interfejsu użytkownika i innych aspektów aplikacji.

2. Tworzenie prototypu (Prototype Creation)

Na podstawie zebranych informacji, zespół tworzy wstępny, działający prototyp aplikacji. Prototyp ten może być prostym modelem interaktywnym lub mockupem, który prezentuje wygląd i funkcjonalności aplikacji.

3. Prezentacja klientowi (Client Presentation)

Klientowi zostaje zaprezentowany prototyp, który umożliwia mu zobaczenie, jak aplikacja może działać w praktyce. Klient ma możliwość przetestowania funkcji i wyrażenia swoich opinii oraz sugestii.

4. Feedback i dostosowania (Feedback and Adjustments)

Na podstawie opinii i sugestii klienta, zespół dokonuje odpowiednich dostosowań prototypu. Mogą to być zmiany w wyglądzie, funkcjonalnościach lub interfejsie użytkownika.

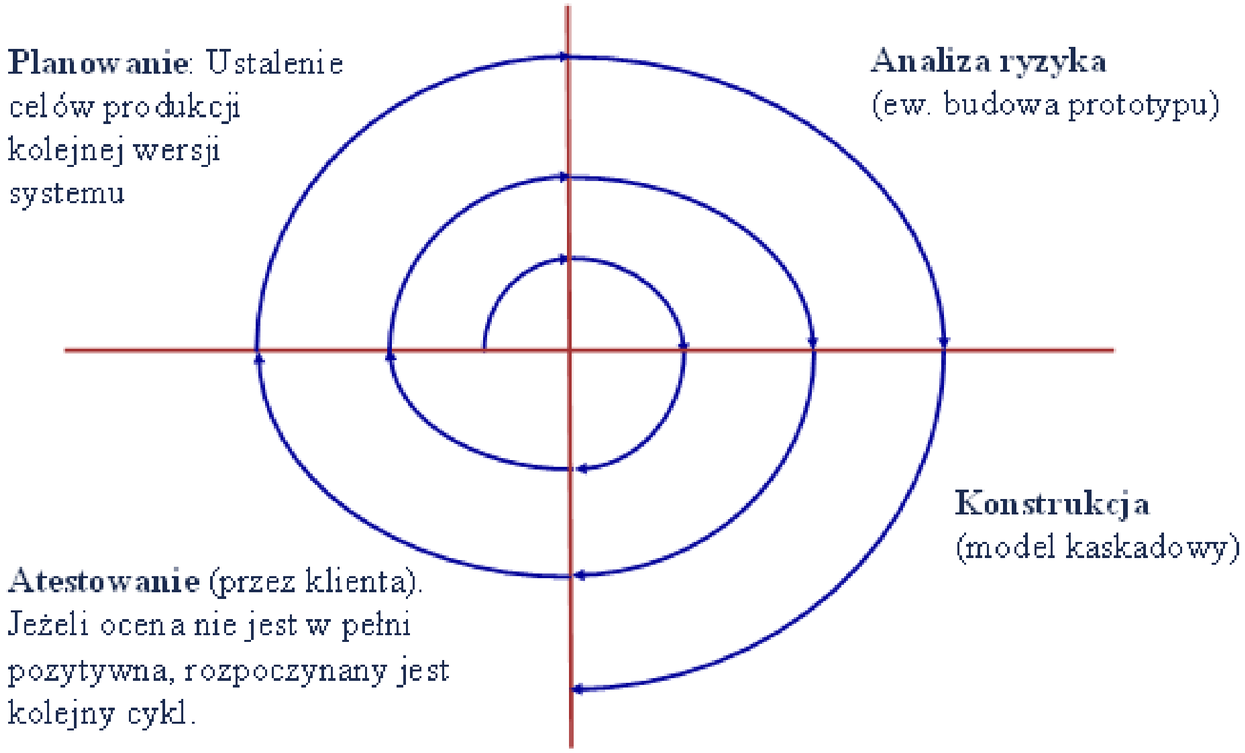
5. Iteracje (Iterations)

Proces tworzenia prototypów i prezentowania ich klientowi może być powtarzany przez kilka iteracji, aż klient będzie zadowolony z efektu końcowego.

6. Finalizacja i implementacja (Finalization and Implementation)

Po uzyskaniu akceptacji klienta, prototyp zostaje rozwinięty w pełnowartościową aplikację. Kolejne etapy, takie jak implementacja, testowanie, wdrożenie i konserwacja, mogą być realizowane w oparciu o bardziej tradycyjne metodyki, np. kaskadową.

Model Spiralny



Proces tworzenia przypomina spiralę, będącą podzieloną na cztery części. Projekty wykonywane są w sposób cykliczny, gdzie cztery części spirali są powtarzane. Stosowane w dużych projektach, które wymagają ciągłego rozwoju i doskonalenia. Główną cechą takiego podejścia jest duża analiza ryzyka.

Zalety

Łatwość edycji

Duża elastyczność i możliwość adaptacji

Dogłębna analiza

Bardzo mocny nacisk na analizę i wykrywanie potencjalnego ryzyka

Aktywny udział klienta

Mocne zaangażowanie ze strony klienta

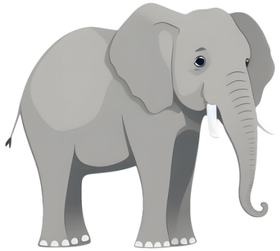
Wady

Pieniądze

Duża złożoność i koszt

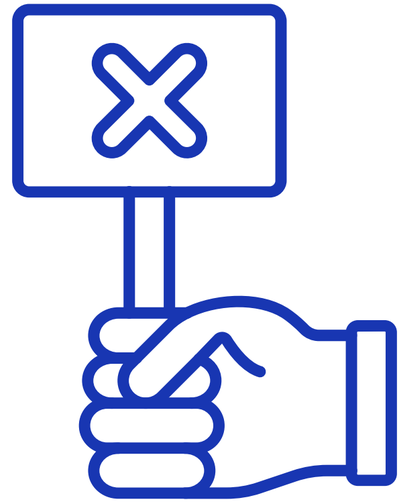
Czasochłonny

Potencjalne opóźnianie projektu przez nadmierne skupienie na wykrywaniu ryzyka



Megaprojekty

Odpowiedni tylko dla dużych projektów



Przykład

Model spiralny

Załóżmy, że firma planuje stworzyć prosty system do zarządzania zadaniami online. Wymagania zostały dokładnie zdefiniowane na początku projektu, a klient nie przewiduje znaczących zmian w trakcie realizacji.

1. Planowanie

W pierwszej spirali zespół i klient identyfikują główne aspekty projektu, w tym wymagania, cele biznesowe oraz ewentualne ryzyka związane z projektem.

2. Faza analizy ryzyka

Na podstawie analizy ryzyka, zespół i klient opracowują plan dla pierwszej iteracji. Wybierają najważniejsze funkcjonalności do implementacji w pierwszej fazie projektu.

3. Implementacja

W tej fazie zespół skupia się na implementacji wybranych funkcjonalności związanych z podstawową obsługą zadań w aplikacji. Mogą to być takie elementy jak tworzenie, edycja i usuwanie zadań.

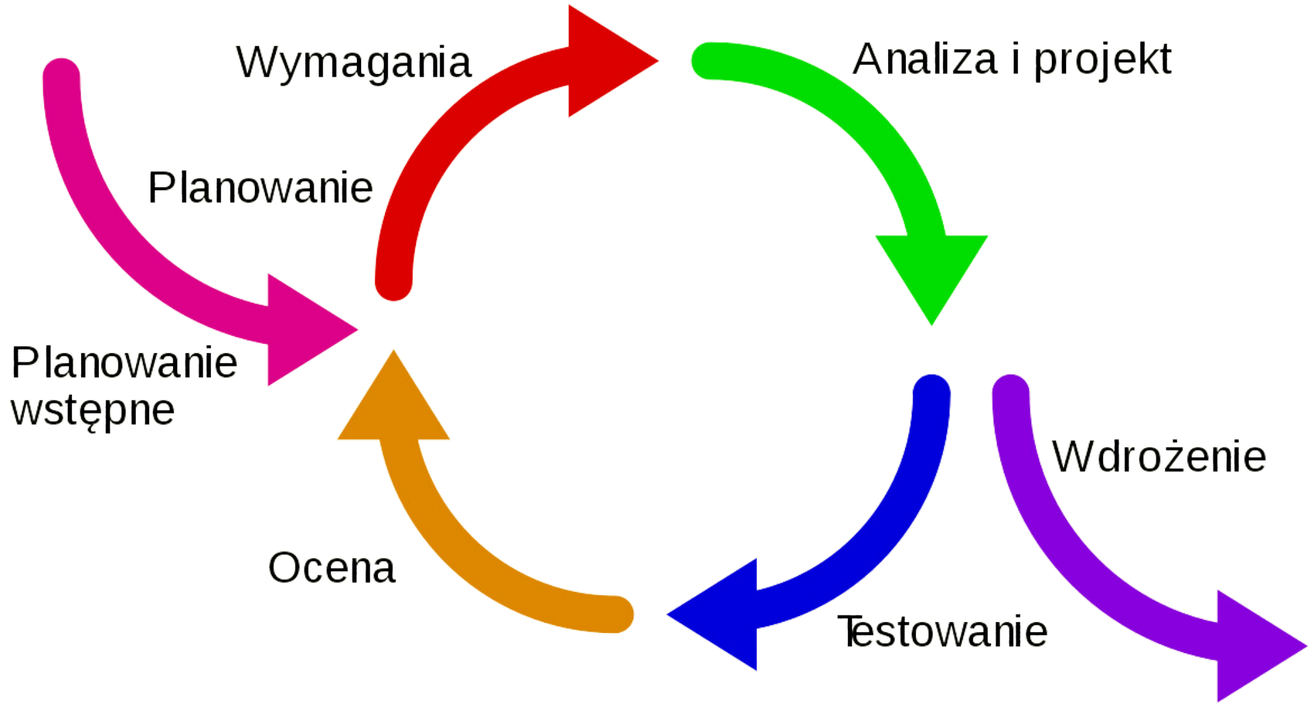
4. Testowanie

Po zakończeniu implementacji, zespół przeprowadza testy, aby upewnić się, że funkcjonalności działają poprawnie.

5. Analiza wyników i planowanie kolejnej spirali

Na podstawie wyników testów oraz opinii klienta, zespół analizuje, co poszło dobrze a co można poprawić. Następnie planują kolejną spiralę, wybierając kolejne funkcjonalności do implementacji.

Model Przyrostowy



Model przyrostowy stosowany jest w przypadkach kiedy projekt można podzielić na mniejsze części („przyrosty”). Każdy taki fragment jest kompletną i funkcjonalną częścią aplikacji, który może być rozwijany niezależnie od reszty fragmentów

Zalety

Niekompletność

Brak konieczności zdefiniowania z góry całości wymagań

Niezależność

Wczesne wykorzystanie przez klienta fragmentów systemu (funkcjonalności)

Aktywny udział klienta

Mocne zaangażowanie ze strony klienta

Wady

Pieniądze

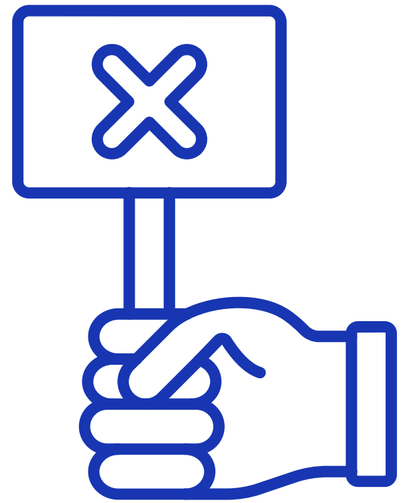
Dodatkowy koszt związany z niezależną realizacją fragmentów systemu

Dodatkowa praca

Konieczność implementacji szkieletów (interfejs zgodny z docelowym systemem) – dodatkowy nakład pracy (koszt), ryzyko niewykrycia błędów w fazie testowania

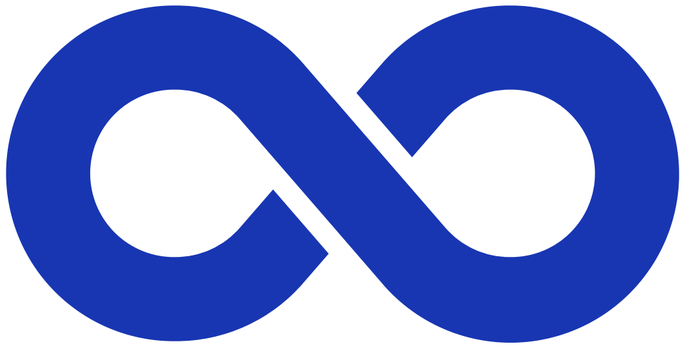
Nie niezależność

Potencjalne trudności z wycinaniem podzbioru funkcji w pełni niezależnych



Przykład

Model przyrostowy



1. Pierwszy inkrement

W pierwszym inkremencie zespół skupia się na podstawowej funkcjonalności, która obejmuje rejestrację użytkowników, logowanie, tworzenie nowych zadań i ich podstawowe zarządzanie (np. dodawanie, usuwanie, edytowanie).

2. Prezentacja i opinie klienta

Po zakończeniu pierwszego inkrementu, klient zostaje zaproszony do przetestowania i oceny aplikacji. Klient może wyrazić opinie na temat interfejsu użytkownika, wydajności oraz funkcjonalności.

3. Drugi inkrement

Na podstawie opinii klienta oraz nowych wymagań, zespół decyduje, że w kolejnym inkremencie zostanie dodana funkcjonalność przypisywania zadań do konkretnych użytkowników oraz możliwość ustawiania priorytetów i terminów realizacji zadań.

4. Prezentacja i opinie klienta (etap drugi)

Klient ponownie zostaje zaproszony do przetestowania aplikacji z nowymi funkcjonalnościami i wyrażenia opinii na temat ich działania i użyteczności.

5. Trzeci inkrement

W kolejnym inkremencie zespół postanawia zaimplementować funkcje powiadomień o zadaniach oraz możliwość filtrowania i sortowania zadań.

6. Prezentacja i opinie klienta (etap trzeci)

Klient ponownie ocenia aplikację i zgłasza swoje opinie oraz sugestie dotyczące ostatnich dodanych funkcjonalności.

Programowanie zwinne (Scrum)

- **Planowanie produktu**
- **Planowanie sprintu (oraz sprint backlog)**
- **Development**
- **Daily Scrum**
- **Przegląd i retrospekcja sprintu**
- **Dostarczenie**
- **Aktualizacja planów produktu**

Programowanie zwinne jest modelem opartym na modelu przyrostowym. Główną różnicą jest nacisk na elastyczność (zwinność) pracy w zespole, w związku z częstą zmiennością wymagań klienta. Jednym z rodzajów programowania zwinnego jest Scrum, w którym iteracje nazywają się sprintami i trwają maksymalnie miesiąc. Występują tez (najczęściej) dzienne “Scrumy”, czyli spotkania omawiające postępy projektu.

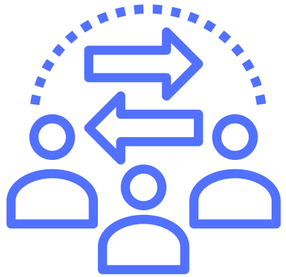
Zalety

Łatwość edycji

Duża elastyczność i możliwość adaptacji

Testy, testy jeszcze raz testy

Regularne testowanie zwiększa jakość aplikacji końcowej



Komunikacja

Dobra komunikacja w zespole

Transparentność procesu

Wszyscy uczestnicy mają dostęp do informacji na temat postępów prac i wyników retrospekcji

Wady

Klient

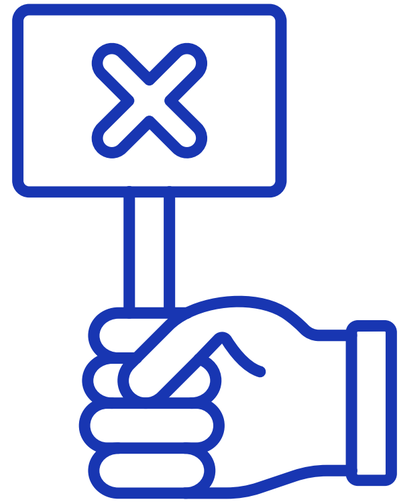
Potrzebne zaangażowanie klienta

Dodatkowa praca

Trudny do wdrożenia w mało elastycznych organizacjach. Ryzyko przeciążenia zespołu przez zbyt ambitne planowanie sprintu

Zaangażowanie innych

Wymagana duża dyscyplina i zgranie w zespole



Przykład

Programowanie zwinne (Scrum)

1. Tworzenie Product Backlogu

Na tym etapie przeprowadzana jest rozmowa z klientem, aby zrozumieć ogólne wymagania i cele projektu. Może to obejmować dyskusję na temat funkcji, interfejsu użytkownika i innych aspektów aplikacji.

2. Sprint Planning

Zespół i klient razem planują pierwszy sprint. Wybierają funkcjonalności z Product Backlogu, które zostaną zaimplementowane w ciągu najbliższych dwóch tygodni (typowy okres trwania sprintu w Scrum).

3. Implementacja w pierwszym sprincie

W trakcie pierwszego sprintu zespół skupia się na implementacji wybranych funkcjonalności związanych z podstawową obsługą zadań w aplikacji. Może to obejmować tworzenie, edycję i usuwanie zadań, a także podstawową funkcjonalność związaną z ich przypisywaniem i zarządzaniem.

4. Daily Scrum

Codziennie, w trakcie krótkiego spotkania (Daily Scrum), członkowie zespołu omawiają postępy, wyzwania i plany na kolejne dni.

5. Sprint Review

Po zakończeniu pierwszego sprintu, zespół prezentuje wykonaną pracę klientowi, który ma możliwość przetestowania i wyrażenia opinii na temat nowych funkcjonalności.

6. Retrospektywa sprintu

Po Sprint Review odbywa się retrospektywa sprintu, na której zespół analizuje, co poszło dobrze i co można poprawić na przyszłość.

7. Planowanie kolejnego sprintu

Na podstawie opinii klienta i wniosków z retrospektywy, zespół razem z klientem planuje kolejny sprint, wybierając nowe funkcjonalności do implementacji.

8. Kontynuacja iteracji

Proces powtarza się w każdym kolejnym sprincie, co umożliwia stopniowe rozbudowywanie aplikacji o nowe funkcjonalności i dostarczanie wartości klientowi na bieżąco.

Model dla naszego projektu

Naszym zdaniem byłby to model prototypowy ze względu na możliwość wykonywania mniejszych prototypów do różnych funkcji aplikacji, aby upewnić się czy całość będzie działała odpowiednio.

Dziękujemy