6. Techniki szybkiego wytwarzania aplikacji (RAD)

Cel technik RAD (Rapid Application Developpement)

- Tworzenie aplikacji w krótkim czasie, przy niskich kosztach
- Założenia wstępne technik RAD:
  - Niemożliwe jest jednoznaczne okrslenie wymagań użytkowników
  - Projektowanie, kodowanie programów i testowanie są procesami iteracyjnymi,
  - Rozwój aplikacji nie jest nigdy definitywnie zakończony
  - Warunkiem budowy efektywnej aplikacji jest upoważnienie projektantów do podejmowania decyzji określających funkcjonowanie

Przyczyny rozwoju technik RAD

- Niepewność działania przedsiębiorstwa (zmienność firmy i otoczenia)
- Niepewność procesu tworzenia aplikacji (ogromne możliwości wyboru wyboru składników do struktury systemu)

Główne cechy technik RAD

- Szybka realizacja aplikacji
- Prototypowanie rozwiązań i wczesne ich testowanie przez użytkowników
- Używanie gotowych komponentów oprogramowania
- Ściśle przestrzegane harmonogramy
- Zastosowanie narzędzi wspomagajacych tworzenie aplikacji (CASE)

Przykłady technik RAD

- Metoda DSDM (Dynamic System Developpement Method)
- Extreme Programming
- Agile Programming
- Technika Bold for Delphi
- …

Extreme programming

Jako jedna z technik RAD

Niedogodności tradycyjnych metod tworzenia oprogramowania

- Wdrażanie dopiero po wykonaniu całego oprogramowania
- Obowiązek wykonania kompletnego projektu zakresu funkcjonalnego i projektu interfejsu – przed rozpoczęciem kodowania programów.
- Uwzględnianie w systemie zbyt wielu funkcji, bardzo rzadko wykorzystywanych
- Obszerność założeń kompleksowych metodologii programowania, utrudnia ich stosowanie w praktyce programowania

Wstęp

- Nowa metoda – opracowana na początku lat 90-ych
- Kent Beck i Ward Cunningham
- Najważniejszy - człowiek

Elementy usprawniające tworzenie oprogramowania – wykorzystane w metodzie „Extreme Programming”

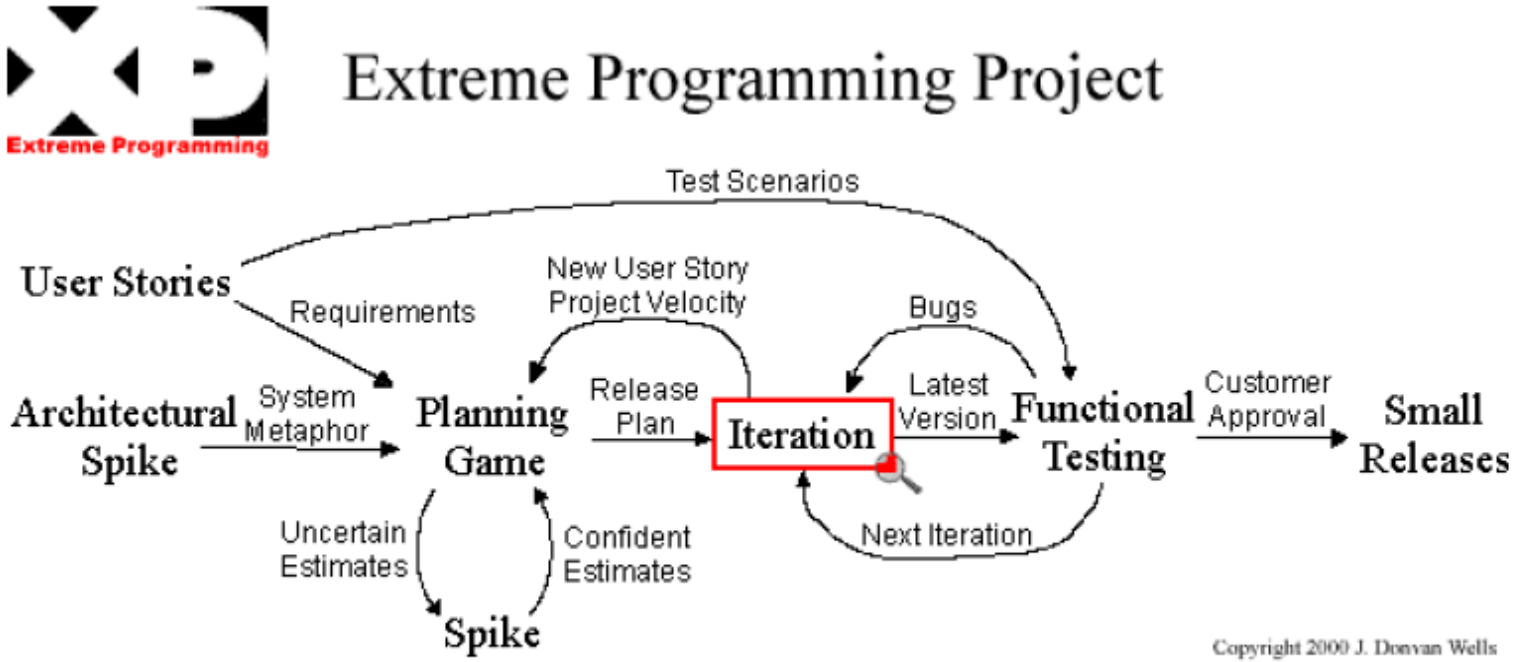
- Usprawnienie komunikacji wewnątrz zespołu
- Usprawnienie komunikacji zespołu programistów z użytkownikami
- Dążenie do maksymalnej prostoty
- Otwarta postawa wobec zmian (refaktoryzacja kodu)
- Zapewnienie sprzężenia zwrotnego z użytkownikami (np. Udział klientów w testowaniu modułów programowych

XP – reguły

XP określa reguły dotyczące 4 kroków tworzenia oprogramowania:

- *Planowania*
- *Projektowania*
- *Kodowania*
- *Testowania*

Cykl realizacji aplikacji wg metody XP



Cykl Extreme Programming – „zdjecie" ze strony www.extremeprogramming.org/map/project.html

Planowanie

- Wykaz wymagań użytkownika,
- terminarz,
- często nowe wersje,
- mierzalna prędkość projektu,
- iteracje,
- przegrupowywanie osób między różnymi pracami,
- rozpoczynanie dnia spotkaniem zespołu,
- dopasowywanie XP do sytuacji.

"Planning Game"

Badanie		
Podejście	Użytkownik	Programista
Cel: co system ma robić? Metody: pisanie i przewidywanie wymagań Rezultat: Zestaw szacunkowych wymagań	Pisze wymagania Podział wymagań Odrzucenie wymagań	Oszacowanie czasu jeżeli >3 tygodnie; użytkownik podzieli wymagania

"Planning Game"

Planowanie	
Podejście	**Akcje**
Cel: następne, lepsze wydanie **Metody**: Powtarzanie akcji od 1 do 4 **Rezultat**: lista wymagań które powinny być uwzględnione w następnym wydaniu	**1.Użytkownik**: Sortowanie wymagań od najważniejszych do najmniej ważnych **2.Programista**: Sortowanie wymagań według ryzyka **3.Programista**: Ustalenie szybkości **4.Użytkownik**: Wybór zakresu wymagań dla następnego wydania

"Planning Game"

Sterowanie
Podejście
Cel: ***Zarządzanie projektem w toku realizacji***   Metody**: Akcja – Reakcja**   Rezultat**: Użytkownik zmienia wymagania w trakcie realizacji tak aby osiągnąć najlepszy efekt.**

"Planning Game"

Sterowanie
Ruch, Akcja, Reakcja
Ruch: ***Iteracja;***   **Akcja: Start iteracji;**   **Reakcja: użytkownik podaje jedno wymaganie, a programista je implementuje**

"Planning Game"

Sterowanie
Ruch, Akcja, Reakcja
Ruch: ***Ponowne ustalenie szybkości realizacji modułu;***   **Akcja: Programista uznaje że szybkość jest zła;**   **Reakcja: Programista ponownie szacuje szybkość a użytkownik dostosowywuje liczbę wymagań tak aby zmieścić się w dacie wydania**

"Planning Game"

Sterowanie
Ruch, Akcja, Reakcja
Ruch: ***Nowe wymaganie;*** **Akcja: Użytkownik podaje nowe, wartościowe wymaganie;** **Reakcja:** **Programista szacuje wymaganie, użytkownik usuwa niekompletne poprzednie punkty planu i dodaje nowe wymaganie**

"Planning Game"

Sterowanie
Ruch, Akcja, Reakcja
Ruch: *Ponowne szacowanie czasu realizacji*; **Akcja: Programista uznaje że plan jest trudny do wykonania;** **Reakcja: Programista ponownie szacuje szybkość i wszystkie wymagania, użytkownik ustala nowe spojrzenie na plan.**

2. Projektowanie

- Prostota,
- Przenośność systemu,
- Używanie karty CRC (KLASA – ODPOWIEDZIALNOŚĆ – KOOPERACJA)
- Rozwiązania prototypowe,
- Dodawanie funkcjonalności we właściwym czasie,
- Przekształcanie programu w miarę potrzeb.

3. Kodowanie

- Klient jest zawsze dostępny,
- Ustalony standard,
- Moduły testujące - najpierw,
- Pary programistów,
- Tylko jedna para dokonuje integracji kodu,
- Częste integrowanie,
- Wspólna własność,
- Optymalizacja na końcu,
- Przestrzeganie terminów.

4. Testowanie

- Cały kod ma moduły testujące,
- Przed wypuszczeniem – kod przechodzi wszystkie testy,
- Do każdej znalezionej pluskwy tworzone są testy,
- Testy akceptacyjne są często uruchamiane, a wyniki podawane do wiadomości.

Zalety

- Zmniejszenie ceny programu
- Elastyczność
- Dużo mniejsza ilość dokumentacji
- Rozproszony proces programowania
- Zadania wykonywane w krótkich przedziałach czasowych
- Brak własności kodu

Wady

- XP w czystej postaci nie nadaje się do tworzenia dużych projektów informatycznych
- Wymaga bardzo doświadczonych programistów
- Żadna osoba nie jest w stanie powiedzieć dokładnie, jakie są zasady działania programu

Bibliografia

- Beck K. : Wydajne programowanie, MIKOM 2001.