Extreme programming

Główne założenia XP

- **Prostota (Simplicity)**
  - Skupienie się na najprostszych możliwych rozwiązaniach, które działają *tu i teraz*. Zespół unika nadmiernego projektowania i zbędnych funkcji.
- **Komunikacja (Communication)**
  - Dobra komunikacja między członkami zespołu i z klientem jest kluczowa. XP promuje rozmowy twarzą w twarz, wspólne programowanie i otwarte dzielenie się wiedzą.
- **Sprzężenie zwrotne (Feedback)**
  - Częste testy, iteracje i kontakt z klientem pozwalają szybko reagować na błędy i zmiany. Im szybciej zespół dostaje informację zwrotną, tym lepiej.
- **Odwaga (Courage)**
  - Odwaga do refaktoryzacji kodu, do usuwania błędnych rozwiązań, przyznania się do błędu i proponowania zmian, nawet jeśli są trudne.
- **Szacunek (Respect)**
  - Wzajemny szacunek w zespole i wobec klienta. Każdy członek zespołu ma wartość, a wspólne cele są ważniejsze niż indywidualne ambicje.

Praktyki

- **Planowanie:**
  - **Planowanie releasu**
  - **Planowanie iteracji**
- **Małe releasy**
- **Testowanie**
- **Metafora (wspólna wizja)**
- **Prosty projekt**
- **Refaktoryzacja (poprawienie projektu).**

Praktyki

- **Programowanie w parach**
- **Ciągła integracja**
- **Zrównoważony krok**
- **Klient „on-line”**
- **Standardy kodowania**
- **Wspólna własność kodu**

Procesy

- Planowanie (Planning)
- Projektowanie (Designinig)
- Kodowanie (Coding)
- Testowanie (Testing)
- Zarządzanie (Managing)

Planowanie

- Klient uczestniczy w planowaniu
- Planowanie releasu: klient prezentuje programistom co chce mieć (user story). Zespół ustala stopień trudności oraz ryzyka. Plany nie są precyzyjne.
- Planowanie iteracji: dokładniejsze planowanie, na dwa tygodnie. Iteracja kończy się działającą wersją.

Projektowanie

- Pisanie testów jednostkowych
- Projektowanie testów akceptacyjnych
- Nie ma projektowania „na wyrost”
- Nie ma nadmiarowości
- Refaktoryzacja wszędzie tam, gdzie to jest potrzebne.

Kodowanie

- W parach
- Najpierw testy jednostkowe
- Zachowanie standardów kodowania
- Klient zawsze dostępny podczas kodowania
- Ciągła integracja kodu
- Optymalizacja kodu podczas jego tworzenia
- Kodowanie funkcjonalności według określonych priorytetów
- Funkcjonalność jest tworzona osobno i jest pod koniec dnia integrowana
- Programiści nie pracują więcej niż 40h na tydzień

Testowanie

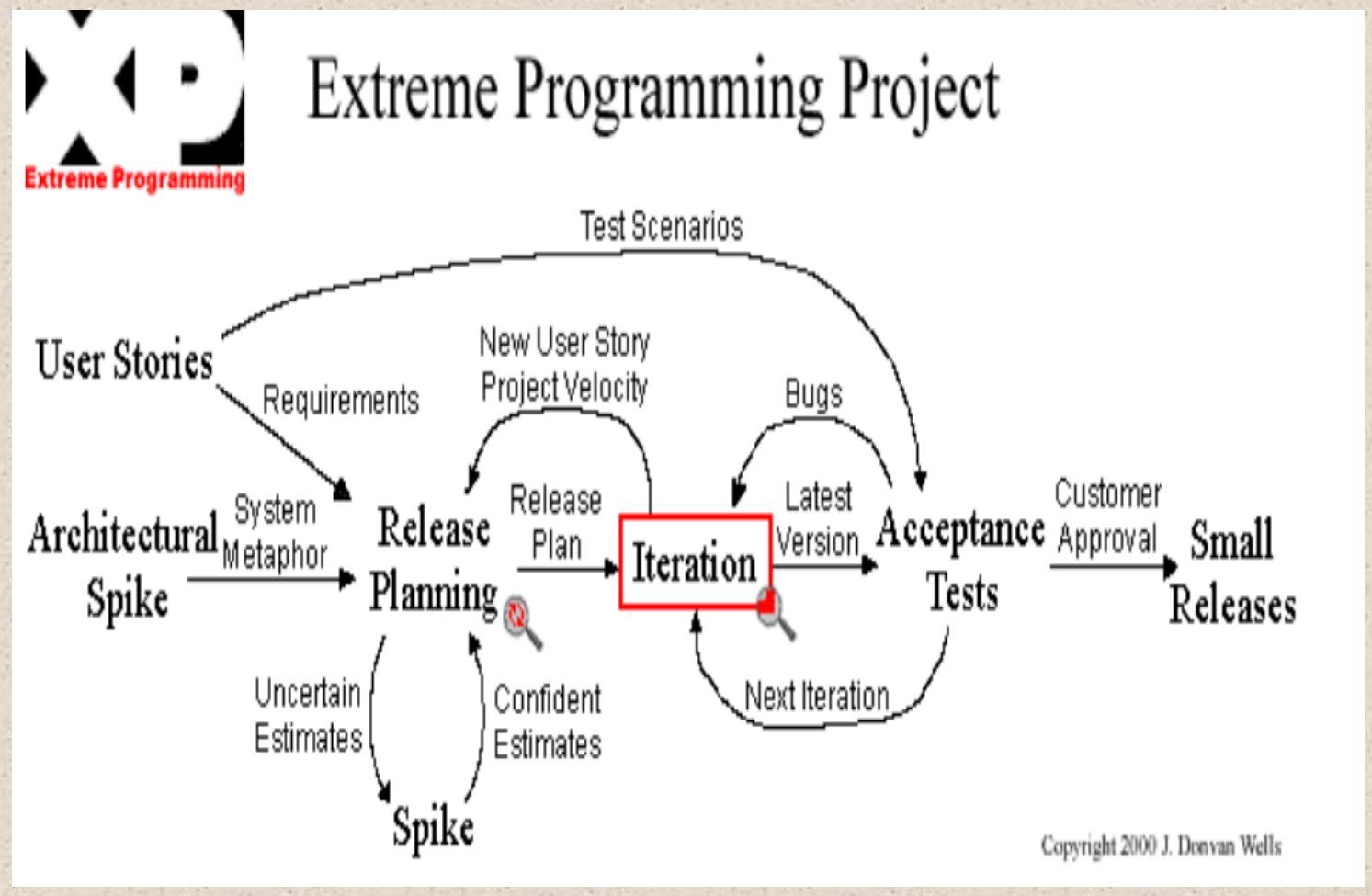
- Testy jednostkowe
  - Dla każdej funkcjonalności
  - Przed wypuszczeniem releasa powinny wszystkie testy jednostkowe przechodzić.
- Testy funkcjonalne
  - Sprowadzenie czy jest cała zakładana funkcjonalność
  - Nabranie pewności, że rozwój produktu idzie w dobrą stronę

Kluczowe cechy testowania

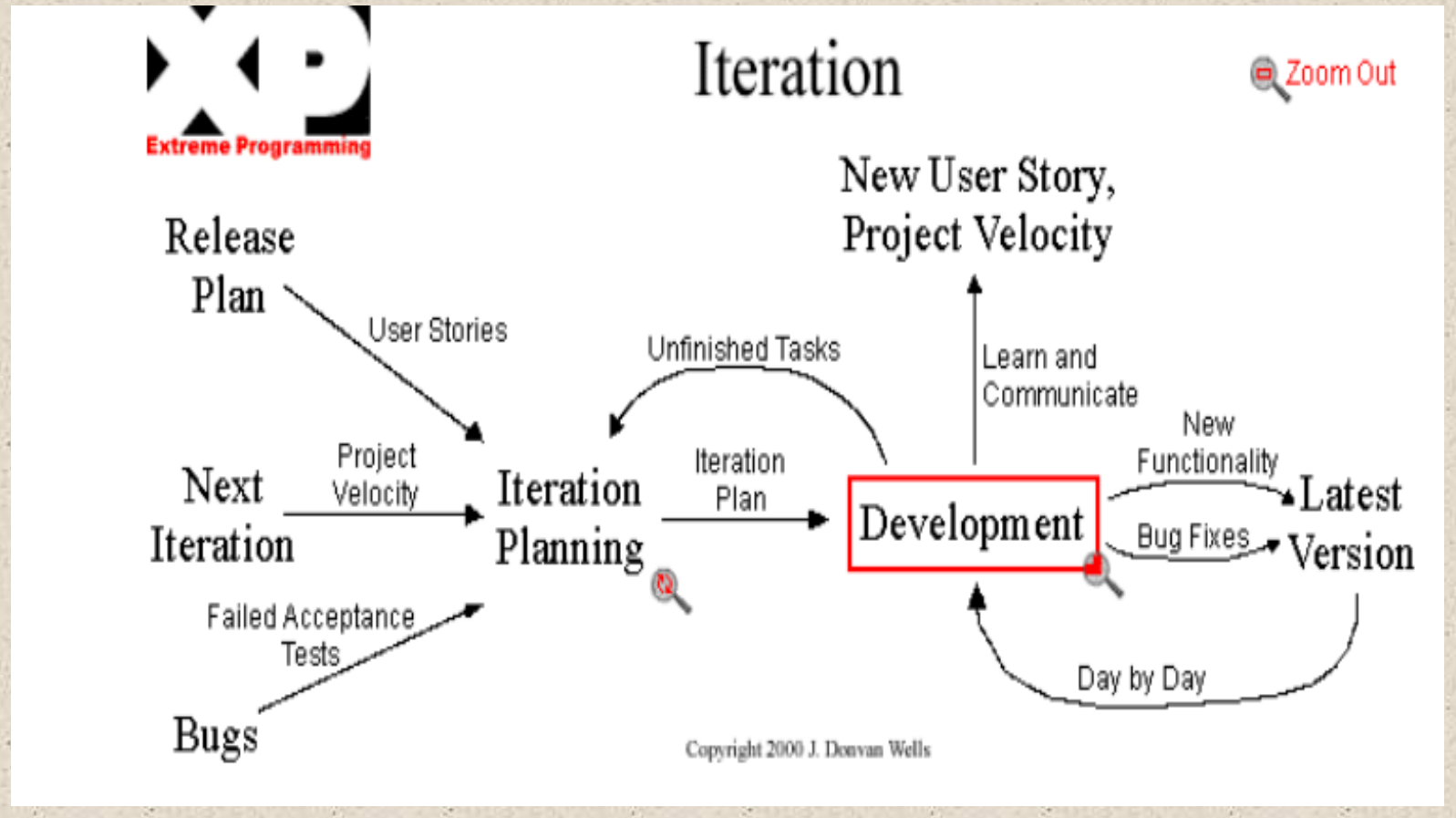
- Nadzorowane przez klienta - klient powinien rozumieć testy;
- Powtarzalność - programiści powinni tworzyć testy po dodaniu nowej funkcjonalności;
- Automatyczność – powinny być automatycznie uruchamiane;
- Publiczne – powinny być publicznie dostępne, klient powinien widzieć postęp w tworzonym oprogramowaniu.

Zarządzanie

- Dedykowany open space dla zespołu
- Odpowiednia oraz mierzalna prędkość prowadzenia projektu
- Stand-up meeting (tak jak w SCRUM)
- Częste zmiany przypisywania osób do zadań
- Zmiana reguł XP gdy nie pasują do projektu.



Źródło: http://www.extremeprogramming.org/rules.html



Źródło: http://www.extremeprogramming.org/rules/iterative.html

Kwestie kontrowersyjne

- Brak dokładnej specyfikacji.
- Konieczna stała dostępność przedstawiciela klienta.
- Wspólna "własność" kodu - każdy może zmieniać dowolny fragment systemu.

XP vs Scrum- różnice

- Scrum nie zezwala na zmiany wewnątrz sprintu. W XP jeśli nie rozpoczęła się praca nad daną funkcjonalnością, może być ona wymieniona na coś innego.
- W XP klient ustala priorytet zadań. W scrumie właściciel produktu ustala product backlog a zespół ustala kolejność realizacji.
- Scrum nie ustala żadnych praktyk inżynierskich, XP natomiast: test-driven development, programowanie w parach, testy automatyczne, refaktoryzacja, itd.
- Scrum daje odpowiedzialność Scrum masterowi, w XP jest powszechna własność kodu, która umożliwia każdemu modyfikację kodu.

XP vs Scrum- różnice

- XP: ciągłą walidacja produktu, Scrum: testowanie na koniec sprintu;
- XP: 40 godzinny tydzień pracy, Scrum: nie ma takiego ograniczenia;
- XP: programowanie w parach, Scrum: nie ma takiego pojęcia

XP vs Scrum- podobieństwa

- Uwzględniają nieprzewidywalność specyfikacji;
- Kładą nacisk na wyprodukowanie produktu w krótkim czasie;
- Kładą nacisk na komunikację pomiędzy członkami zespołu.

Manifest Agile

- Osiągnięcie satysfakcji klienta poprzez szybkość wytwarzania oprogramowania,
- Działające oprogramowanie jest dostarczane okresowo (raczej tygodniowo niż miesięcznie),
- Podstawową miarą postępu jest działające oprogramowanie,
- Późne zmiany w specyfikacji nie mają destrukcyjnego wpływu na proces wytwarzania oprogramowania,
- Bliska, codzienna współpraca pomiędzy biznesem a deweloperem,
- Bezpośredni kontakt jako najlepsza forma komunikacji w zespole i poza nim,
- Ciągła uwaga nastawiona na aspekty techniczne oraz dobry projekt (design),
- Prostota,
- Samozarządzalność zespołów,
- Regularna adaptacja do zmieniających się wymagań.

Materiały

- https://ronjeffries.com/xprog/what-is-extreme-programming/
- http://www.extremeprogramming.org/rules.html