Techniki komputerowe w robotyce

WYKŁAD V

Adaptacyjne zarządzanie projektami

Robert Muszyński
KCiR, W12, PWr

– Skład FoilTEX –

Metody prowadzenia projektu

Dążenie do opracowania jednej, uniwersalnej metody idealnej – mrzonka

- **Metodyki ciężkie**
- **Metodyki zwinne** (lekkie, sprawne (*agile methodologies*)) – Adaptacyjne zarządzanie projektami (*Agile Project Management, APD*)

Metodyki zwinne – cechy charakterystyczne:

- zdecydowane ograniczenie liczby dokumentów,
- brak planowania w dłuższej perspektywie czasowej,
- otwartość na zmiany,
- niewielkie zespoły projektowe – zazwyczaj do kilkunastu osób,
- brak wydzielonej fazy projektowej,
- ciągła współpraca z klientem.

Cele i zasady metodyk zwinnych

- elastyczność i adaptacyjność projektowania względem dynamicznie zmieniających się potrzeb i oczekiwań klienta,
- tworzenie wartościowych i innowacyjnych rozwiązań zarówno dla firmy jak i konsumentów na każdym etapie projektowania,
- minimalizacja kosztów m.in. dzięki skróceniu harmonogramów procesu wytwarzania,
- koncentracja na członkach zespołu projektowego, wzrost motywacji wśród pracowników i bezstresowa realizacja projektów,
- ścisła współpraca z klientem – preferowany jest kontakt bezpośredni,
- prostota i samoorganizujące się zespoły,
- satysfakcja klientów dzięki szybkiemu i regularnemu dostarczaniu wartościowego produktu,
- minimalizacja ryzyka.

Programowanie zwinne

(Agile Software Development)

Zaproponowana w 2001r. grupa metodyk wytwarzania oprogramowania opartego o programowanie iteracyjne (model przyrostowy) będących alternatywą dla tradycyjnego podejścia opartego o model kaskadowy.

Przedkłada się w nich:

- jednostki i współdziałania między nimi nad procesy i narzędzia,
- działające oprogramowanie nad szczegółową dokumentację,
- współpracę z klientem nad negocjację umów,
- reagowanie na zmiany nad realizowanie planu.

Założenia Manifestu programowania zwinnego

(Agile Manifesto)

- Osiągnięcie satysfakcji klienta poprzez szybkość wytwarzania
- Działające oprogramowanie jest dostarczane okresowo (raczej tygodniowo niż miesięcznie)
- Podstawową miarą postępu jest działające oprogramowanie
- Późne zmiany w specyfikacji nie mają destrukcyjnego charakteru na proces wytwarzania oprogramowania
- Bliska, dzienna współpraca pomiędzy biznesem a developmentem
- Bezpośredni kontakt – najlepsza forma komunikacji w zespole i poza nim
- Projekty budowane poprzez zmotywowane indywidua, które posiadają pełne zaufanie
- Ciągła uwaga nastawiona na aspekty techniczne oraz dobry projekt
- Prostota
- Samozarządzalność zespołów
- Regularna adaptacja do zmieniających się wymagań

Zespół w programowaniu zwinnym

- Skład zespołów jest wielofunkcyjny oraz samozarządzalny bez zastosowania jakiejkolwiek hierarchii korporacyjnej
- Członkowie zespołu biorą odpowiedzialność za zadania postawione w każdej iteracji
- Członkowie zespołu sami decydują jak osiągnąć postawione cele
- Zasadniczą sprawą jest bezpośrednia komunikacja pomiędzy członkami zespołu minimalizująca potrzebę tworzenia dokumentacji. Jeśli członkowie zespołu są w różnych lokalizacjach to planuje się codzienne kontakty za pośrednictwem dostępnych kanałów komunikacji (wideokonferencja, poczta elektroniczna, itp.)

Metodyki zwinne – opracowania

- Programowanie ekstremalne (*eXtreme Programming (XP)*)
- Metodyka DevOps
- Metodyka Scrum
- Metodyka Crystal
- Ciągłe doskonalenie (Continuous Improvement – Kaizen)
- Lean Development/Management
- Global Launch Process
- Adaptive Software Development
- Feature Driven Development
- Behavior Driven Development
- (Acceptance) Test Driven Development
- pochodne (Prince II, Rational Unified Process, XPrince)

Programowanie ekstremalne – zalecenia

- **Iteracyjność** Program tworzy się w iteracjach i – co ważniejsze – planuje tylko następną iterację.
- **Nie projektować z góry** Nie można z góry przewidzieć, jaka architektura będzie najlepsza dla danego problemu. Dlatego należy ją tworzyć w miarę rozszerzania programu.
- **Testy podzespołów** Testy podzespołów pisze się zanim w ogóle zacznie się pisać kod – najlepiej na początku iteracji. Potem pisze się kod, który potrafi je wszystkie przejść.
- **Ciągłe modyfikacje architektury** Architektura nie jest czymś, czego nie wolno ruszać. Jeśli modyfikacja architektury ułatwi przejście danej iteracji i nie zepsuje wyników testów uzyskanych na poprzednich iteracjach, należy ją wykonać.
- **Programowanie parami** Programiści piszą w parach: jedna osoba pracuje przy klawiaturze i jest głównym koderem, druga obserwuje pierwszą, zgłasza poprawki, zadaje pytania wyjaśniające. Umożliwia to wyłapanie wielu błędów oraz wzajemną naukę.
- **Stały kontakt z klientem** Specyfikacje są prawie zawsze wieloznaczne, dziurawe i sprzeczne ze sobą. Tak więc należy mieć stały kontakt z tym, dla kogo to oprogramowanie jest tworzone (klient zamawiający program, czy też użytkownicy końcowi). Jeśli kontakt jest dobry, można się nawet obyć bez specyfikacji.

Programowanie ekstremalne – postulaty

- **Planowanie**
  - Twórz w sposób iteracyjny
  - Spraw by każdy wiedział, jak działa każdy moduł
  - Często wypuszczaj wydania
  - Jeśli XP zaczyna zawodzić, zmień je!
- **Projektowanie**
  - Niech projekt będzie prosty
  - Nie dodawaj nadmiernej funkcjonalności
  - Dokonuj refaktoringu wtedy, gdy jest to konieczne
- **Implementacja**
  - Stosuj ustaloną konwencję
  - Najpierw napisz testy, potem właściwy kod
  - Programowanie w parach
  - Często integruj kod
  - Optymalizuj na samym końcu
  - Brak nadgodzin!!!
  - Każdy jest równo odpowiedzialny za kod
- **Testowanie**
  - Każdy fragment kodu, musi mieć odpowiadające mu testy
  - Kod jest umieszczany w repozytorium, dopiero po zaliczeniu wszystkich testów
  - Gdy znajdziesz błąd, stwórz dla niego testy

Metodyka DevOps

- **Dev**elopment and **Op**eration**s** Program tworzy się przy ciągłej komunikacji, współpracy i integracji pomiędzy programistami i specjalistami od eksploatacji – zespolenia rozwoju i eksploatacji. . . i testów
- **Development** – wymaga zmian
- **Operations** – wymaga stabilizacji
- **Dev&Ops** – sposób na minimalizację ryzyka błędów aplikacji produkcyjnych

Rekomendacje (środowisko IT):

- stosowanie programowania zwinnego (lub metod podobnych)
- częste wdrożenia (nawet do kilkunastu dziennie)
- dostępność infrastruktury w chmurze i jej wirtualizacja
- narzędzia automatyzacji i zarządzania konfiguracją w centrum danych

Metodyka DevOps – zasady

- Usprawnienie komunikacji w zespołach projektowych
- Wykorzystanie metodyk zwinnych przy tworzeniu aplikacji
- Automatyzacja procesów testowania i wdrażania na produkcję
- Sprawne wykorzystanie informacji zwrotnej
- Dzielenie się wiedzą: zarówno techniczną (kodem Open Source) jak i biznesową

Metodyka DevOps – proces (∞)

Metodyka DevOps zaleca wykorzystanie różnorodnych zestawów narzędzi (toolchains) z kategorii (odzwierciedlających kluczowe aspekty procesu – wielofunkcyjny tryb pracy):

- Kodowanie (Code) – narzędzia do rozwój i przegląd kodu, kontroli wersji, scalanie kodu
- Budowanie (Build) – narzędzia do ciągłej integracji, śledzenia statusu kompilacji
- Testowanie (Test) – narzędzia do określanie wydajności na podstawie wyników testów
- Przygotowanie pakietów (Package) – repozytoria, przygotowanie aplikacji do wdrożenia
- Udostępnianie wydań (Release) – narzędzia do zarządzania zmianami, automatyzacji i zatwierdzania wydań
- Konfigurowanie (Configure) – narzędzia do zarządzania i konfiguracji infrastrukury
- Monitorowanie (Monitor) – narzędzia do monitorowania wydajności aplikacji, opinii użytkownika końcowego

Metodyka DevOps – narzędzia

- **Monit** – monitorowanie i korekcja błędów (watchdog)
- **Collectd/Collectl** – monitorowanie pracy komputera
- **Nagios (& Icinga)** – monitorowanie usług i zasobów sieciowych
- **Consul** – zestawianie i konfiguracja serwisów
- **Elastic Stack (& Kibana)** – zarządzanie danymi (i ich wizualizacja)
- **Jenkins** – wykonywanie zadań (ciągła integracja)
- **Docker** – wdrażanie i uruchamianie aplikacji
- **Ansible** – automatyzacja zadań i zarządzania konfiguracją
- **Git (GitHub)** – system kontroli wersji

Więcej na `https://devops.com/9-open-source-devops-tools-love/`

Metodyka Scrum

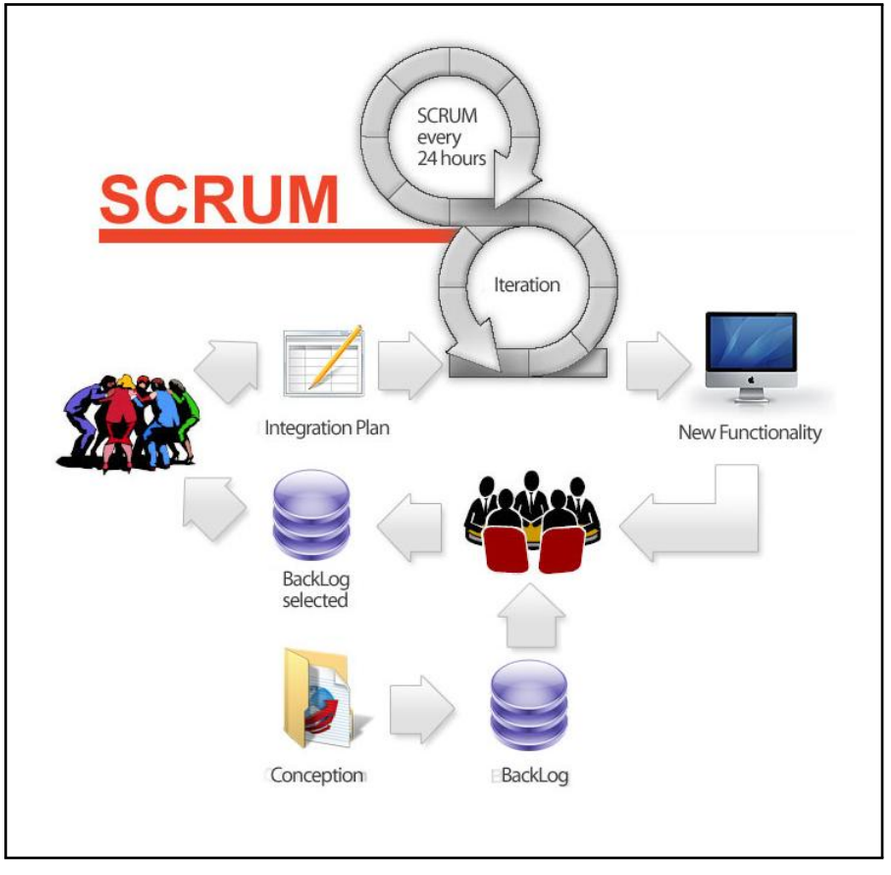
- Iteracyjne podejście do zadania wytworzenia produktu – sekwencja mini-przedsięwzięć zwanych iteracjami (sprintami)
- Inkrementalne wytwarzanie produktu – funkcjonalność produktu rośnie poprzez nowe własności, dodawane kolejno podczas każdej iteracji
- Samoorganizujące się zespoły
- Role uczestników projektu:
  - właściciel produktu (*product owner*),
  - mistrz scrum (*scrum master*),
  - członek zespołu (*team member*)
- Zestaw narzędzi oraz technik:
  - wykres malejący (*burn-down chart*),
  - wykaz prac (zbiór wymagań) produktu (*product backlog*),
  - wykaz prac sprintu (*sprint backlog*),
  - spotkania planowania sprintu (*sprint planning meetings*),
  - spotkania przeglądu sprintu (*sprint review meetings*),
  - codzienne zebrania (*brief meetings*)

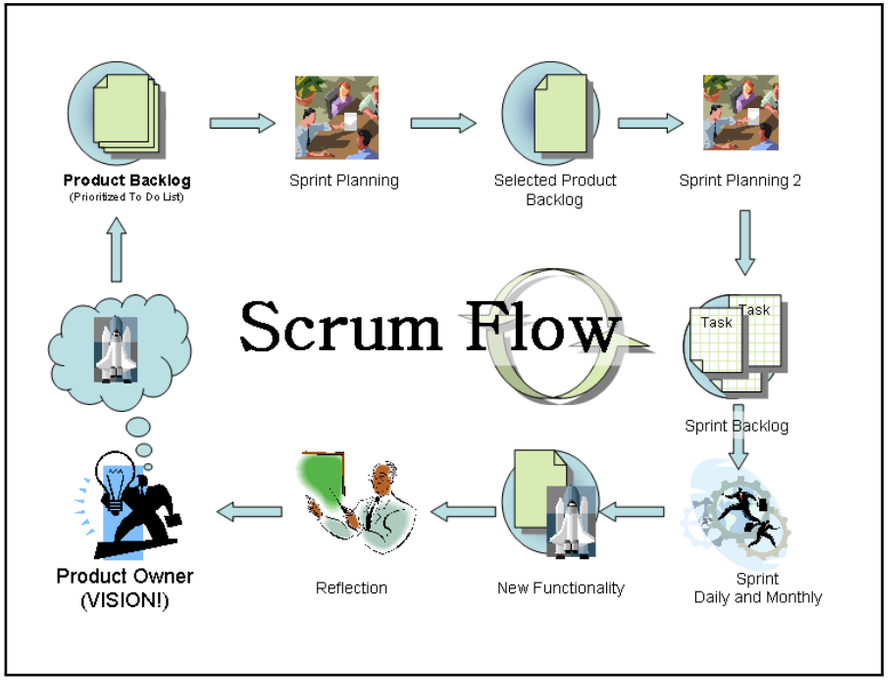
Metodyka Scrum – role

- **Właściciel produktu** – określa początkowy wykaz prac produktu, plan edycji produktu (hierarchię), budżet
- **Mistrz scrum** – odpowiada za egzekwowanie praktyk i zasad Scrum, nadzoruje sposób wykorzystania metodyki, „ekranuje” zespół i usuwa przeszkody
- **Zespół scrum** – buduje produkt, jest „międzyfunkcyjny” i „samoorganizujący się”: obejmuje całą wiedzę niezbędną dla dostarczenia w każdym Sprincie produktu będącego potencjalnym wynikiem sprintu oraz posiada bardzo duży stopień autonomii i odpowiedzialności

Metodyka Scrum – role

Metodyka Scrum – cykl





Metodyka Scrum – sprint

Sprint to pojedyncza iteracja, o sugerowanym czasie trwania równym trzydzieści dni – ważne by wszystkie iteracje miały tę samą długość.

- **Spotkanie planowania sprintu** – zajęcie jednodniowe (do ośmiu godzin)
  - czterogodzinna sesja planowania – powstaje wykaz prac sprintu: lista czynności o najwyższym priorytecie możliwych do wykonania w pojedynczej iteracji wybrana z wykazu prac produktu, oraz cel sprintu: korzyść, jaką zmiany w produkcie muszą dostarczyć (z udziałem właściciela produktu)
  - podział wykazu prac sprintu na zadania – jednostki pracy szacowane na cztery do sześciu godzin – do wykonania w celu dostarczenia wymaganej funkcjonalności; lista zadań niekoniecznie musi być kompletna (bez udziału właściciela produktu, który jednakże powinien być dostępny)

Przestrzeganie ograniczeń czasowych: jeśli upłynął czas przeznaczony dla danej czynności, to musi ona zostać zakończona – należy raczej usunąć zadanie z listy zadań niż dopuszczać do pracy po godzinach; przy codziennym wybieraniu zadań członkowie zespołu powinni brać zadania „których nie wiedzą jak zrobić” – będą się dzięki temu doskonalić i nie popadną w rutynę

- **Zebrania codzienne** – organizowane o tej samej porze (najlepiej przed rozpoczęciem pracy), w tym samym miejscu, co najwyżej pięćdziesięciominutowe spotkanie (najlepiej na stojąco).
  - otwarte dla wszystkich, jednak głos zabierać mogą tylko członkowie zespołu (obecność obowiązkowa)
  - służy jedynie synchronizowaniu działań (a nie rozwiązywaniu problemów)
  - każdy członek zespołu odpowiada na pytania:
    - Co robiłeś wczoraj?
    - Co będziesz robił dziś?
    - Co ci stoi na przeszkodzie?
- **Spotkanie przeglądu sprintu** – na koniec sprintu zespół przedstawia właścicielowi produktu, co osiągnięto w ostatniej iteracji; po tym właściciel produktu określa, czy cel sprintu został osiągnięty (maksymalnie czterogodzinne)
- **Spotkanie retrospektywne sprintu** – po spotkaniu przeglądu sprintu zespół i mistrz scrum rozmawiają o tym, które zadania i czynności zostały wykonane należycie i jak można usprawnić następny sprint (maksymalnie trzygodzinne, bez właściciela produktu)

- **Nadzwyczajne zakończenie sprintu** – mistrz scrum lub właściciel projektu ma prawo przerwania sprintu jeśli
  - cel sprintu okazał się już nieistotny,
  - występują szczególnie uciążliwe problemy z realizacją postawionych zadań.

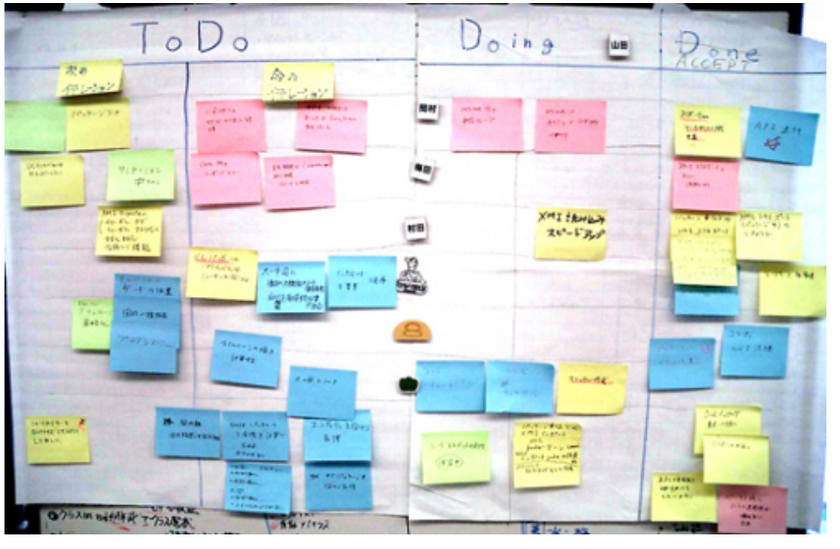
  W razie potrzeby zostaje podjęta akcja naprawienia problemu, po czym organizuje się nowy sprint, najpewniej z nowym wykazem prac i nowym celem.

Sprint – komentarze

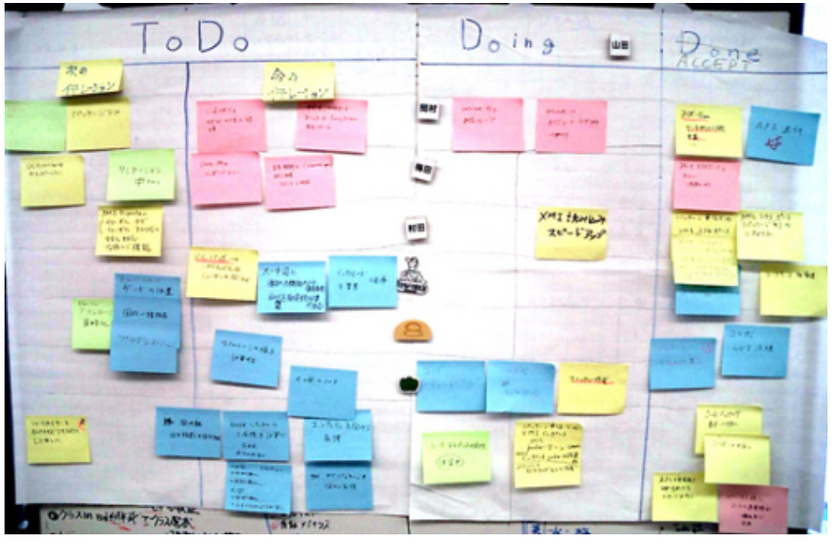
- **Zarządzanie zespołem** – właściciel produktu oraz mistrz scrum powinni pohamować swe zapędy do rządzenia zespołem, nawet jeśli członkowie zespołu tego sobie życzą. Zespół powinien sam wypracować sobie równowagę, bez wpływów z zewnątrz. Każda zewnętrzna pomoc jedynie utrudnia zespołowi samoorganizowanie się.
- **Modyfikacja wykazu prac sprintu** – niedopuszczalna! Jeśli nowe cechy są tak ważne, że dodanie ich jest nieodzowne, należy zarządzić nadzwyczajne zakończenie sprintu.

Metodyka Scrum – Tablica zadań

Story	To Do	In Process	To Verify	Done
As a user, I... 8 points	Code the... 9; Test the... 8; Code the... 2; Code the... 8; Test the... 8; Test the... 4	Code the... DC 4; Test the... SC 8	Test the... SC 6	Code the...; Test the...; Test the...; Test the... SC 6
As a user, I... 5 points	Code the... 8; Test the... 8; Code the... 4; Code the... 6	Code the... DC 8		Test the...; Test the...; Test the... SC 6



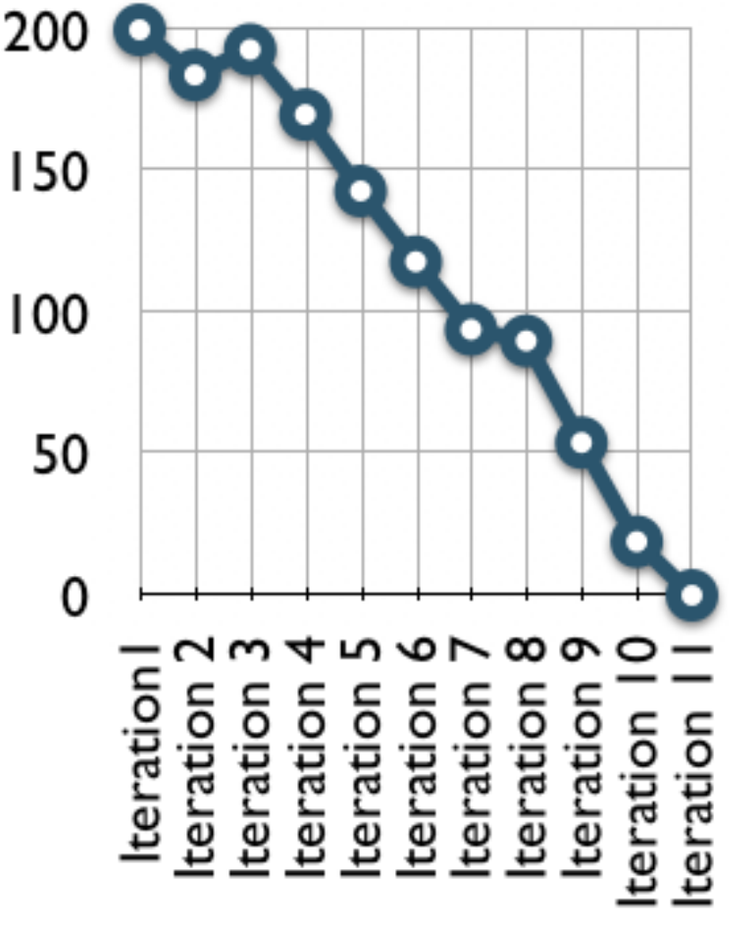
Metodyka Scrum – Tablica zadań



Metodyka Scrum – Tablica zadań

Metodyka Scrum – Wykresy wygaszania

Wygaszanie sprintu



Wygaszanie wypuszczenia

Ciągłe doskonalenie (Kaizen)

Kaizen – japońskie słowo o podwójnym znaczeniu:

- kaizen (w japońskim pisane znakami kanji) oznacza poprawę, polepszenie, zmianę na lepsze (Kai – zmiana, Zen – dobrze),
- kaizen (w japońskim pisane pismem sylabicznym katakaną) oznacza japońską filozofię biznesową (sposób postępowania) ustawicznego polepszania, poprawiania procesu zarządzania i produkcji na wszystkich jego szczeblach, z uwzględnieniem m.in. technik biznesu „just-in-time”.

Skuteczne przez stosowanie w uporządkowany sposób:

- zarządzania przez jakość – Total Quality Management (TQM)
- dostaw dokładnie na czas – Just-in-time (JIT)
- całkowitego produktywnego utrzymania ruchu maszyn – Total Productive Maintenance (TPM)
- rozwijania strategii – Policy Deployment (Hoshin Kanri)
- systemu sugestii – Staff Suggestion System (Kaizen Teian)
- pracy w małych grupach – Small Group Activity (SGA)

7 zasad Lean Software Development

- Eliminacja marnotrawstwa
- Wzmocnienie pozyskiwania wiedzy
- Podejmowanie decyzji najpóźniej jak to możliwe
- Wdrażanie najwcześniej jak to możliwe
- Respektowanie zespołu
- Tworzenie jakości i spójności
- Spojrzenie na całość

Global Launch Process

Całościowe, ustrukturyzowane i metodyczne podejście do procesu projektowania i wdrożenia

Cele główne

- Bezpieczne wdrożenia (NM)
- Jakość (GCA, DRR)
- Terminowość (bramki Go / No go & SORP)
- Wydajność (JPH, Akceleracja)
- Budżet (€)

Projektowe DNA

- Continuous Improvement Teamwork
- PDCA (Plan-Do-Check-Act – Zaplanuj-Wykonaj-Sprawdź-Popraw; cykl Deminga)
- Lessons Learned
- Kaizen

Global Launch Process

Główne procesy

- „Uruchomienie” zespołów
- Planowanie
- Szkolenie
- Przygotowanie fabryki
- Budowa w fabryce
- Walidacja

Global Launch Process

Inne procesy

- Instalacja maszyn
- Remonty kapitalne
- Szybkie wdrożenia inżynieryjne
- Projekty R&D
- Uruchamianie nowych działów
- Projekty środowiskowe (oczka wodne, domki dla pszczół itp.)
- Wydarzenia socjalne (pikniki rodzinne itp.)
- Strefy komfortu (przyciąganie pracowników)
- Projekty IT (serwerownie itp.)

Global Launch Process

Zagadnienia

- Proces produkcji versus projekt
- Określenie „Punktu bez powrotu”
- Symulacja konsekwencji
- Czwarta rewolucja przemysłowa – Przemysł 4.0 (Industry 4.0)

Metodyka XPrince *(eXtreme PRogramming IN Controlled Environments)*

Własności:

- jest zwinna,
- posiada mechanizmy kontroli,
- zachowuje odpowiedni poziom dokumentacji technicznej,
- ma prostą i efektywną strukturę organizacyjną,
- jest przejrzysta dla wyższej kadry zarządczej,
- wykorzystuje dobre praktyki programistyczne,
  - zarządzanie wersjami,
  - ciągła integracja,
  - testy jednostkowe,
  - testy akceptacyjne,
  - implementacja kierowana testami (TDD – Test Driven Development),
  - opracowanie rozwiązań próbnych (spike solution).

Metodyki zwinne – porównanie

- Metodyki zwinne ze względu na swą różnorodność są nieporównywalne
- Metodyka Scrum dotyczy strony zarządzania projektem, pozostawiając inżynierskie sprawy – projektowanie, kodowanie, zarządzanie konfiguracją, testowanie itd. – samoorganizującemu się zespołowi
- Programowanie ekstremalne dotyczy spraw inżynierskich i nie dostarcza żadnych specyficznych narzędzi do zarządzania projektem
- Lean Development dotyczy zarządzania projektem ale w znacznie szerszym ujęciu niż metodyka Scrum – z punktu widzenia organizacji całego przedsiębiorstwa
- W praktyce często stosuje się wspomniane metodyki razem, istnieją także metodyki oparte na ich połączeniu

Metodyki zwinne (by dilbert)

© Scott Adams, Inc./Dist. by UFS, Inc.