AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE
AGH UNIVERSITY OF SCIENCE
AND TECHNOLOGY

Stosowanie Agile – przykłady

Piotr Boryło

Zespół i konsultanci

- Niewystarczająca liczba specjalistów względem potrzeb zespołów
- Zespół konsultantów
  - Specjalistyczna wiedza
  - Niezwiązani z żadnym projektem
  - Rozwiązują konkretne problemy
  - Szkolą i dzielą się wiedzą

Zespół i konsultanci

Table 3-1 Roles, Benefits, and Downsides

	NATURAL FIT	BENEFITS	DOWNSIDES
TEAM CONSULTANT	• SOFTWARE ARCHITECTS • DESIGNERS AND UI • TECHNICAL WRITERS • DEEP TECHNICAL EXPERTS • DEVELOPMENT MANAGERS • SOFTWARE LEADS	• FOCUS ON ONE CRAFT • REMAIN LONE WOLF • ACHIEVE SATISFACTION OF HELPING OTHERS LEARN YOUR SPECIALITY • BECOME A LEADER IN ONE SPECIALTY • MANAGE ONE'S OWN COMMITMENTS	• MAY NEVER SEE THE FRUITS OF LABOR IN FINISHED PROJECT • NOT MUCH OPPORTUNITY TO LEARN NEW SKILLS • MUST PROVIDE GOOD SERVICE OR MAY QUICKLY BECOME OVERHEAD
CORE TEAM MEMBER	• MULTI-TALENTED PROGRAMMERS OR TESTERS • INDIVIDUALS WHO WANT TO GROW THEIR SKILLS • PEOPLE WHO LIKE WORKING ON ONE PROJECT AT A TIME	• CAN FOCUS ON ONE PROJECT THROUGHOUT LIFECYCLE • LEARN NEW SKILLS AS PART OF CROSS-FUNCTIONAL TEAM • HELP MAKE OTHERS BETTER BY TEACHING THEM A NEW APPROACH • ALLOWS A PERSON TO GROW PROFESSIONALLY AND TECHNICALLY	• MUST BE ABLE TO WORK WELL AS A TEAM MEMBER • NOT A GOOD FIT FOR PRIMA DONNAS

https://www.timecockpit.com/blog/2013/06/25/Six-Reasons-for-Time-Tracking-in-Agile-Projects

Zespół i konsultanci

- Macierz kompetencji
  - Uzupełnianie braków przez konsultantów
- Schemat współpracy zespołu i konsultantów
  - XP
  - Udział w spotkaniach

Przewidywanie szybkości

- Problematyczne szczególnie dla nowych zespołów
- Możliwości:
  - Wykorzystaj dane historyczne
  - Ślepa estymacja
  - Zacznij działać i określ szybkość w trakcie

Wykorzystaj dane historyczne

- Zalety
  - Jakikolwiek punkt wyjścia
  - Doświadczenie innych zespołów
- Wady
  - Każdy zespół i projekt jest inny
  - Nie mamy swoich danych
  - Nowa technologia i nowy zespół

Ślepa estymacja

- Zalety
  - Jakikolwiek punkt wyjścia, szczególnie jeśli mamy doświadczenie w temacie
  - Kierownictwo to lubi
- Wady
  - To tylko założenia, mogą ograniczać
  - Fałszywe poczucie bezpieczeństwa
  - Fałszywy punkt odniesienia

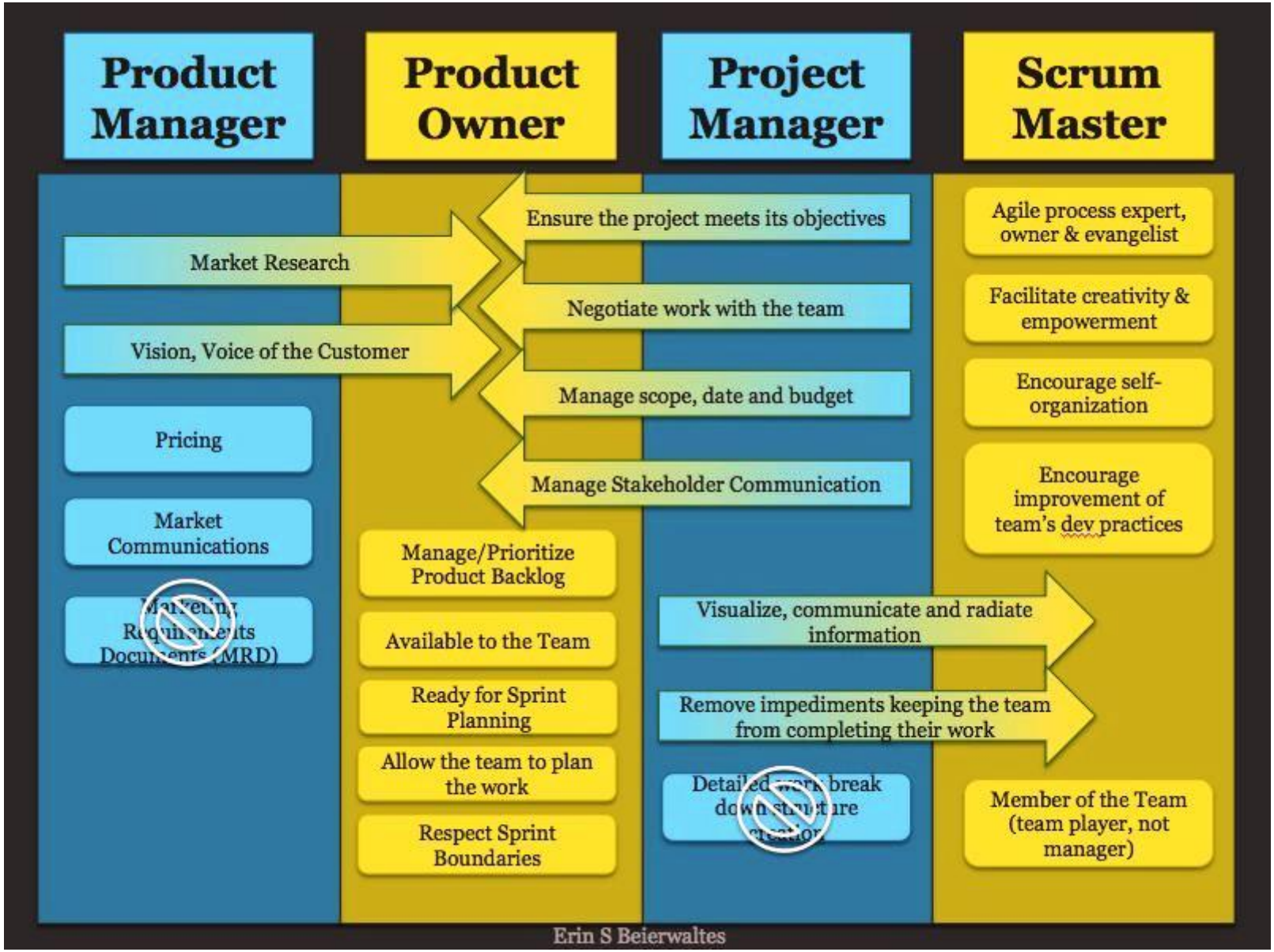
Określi szybkość działając

- Zalety
  - Wartościowe i trafne oszacowania
  - Bazujemy na faktach i ocenie postępów
- Wady
  - Kierownictwo się nie zgodzi
  - Minimum 3 iteracje potrzebne na start
  - Ciągły monitoring, szczególnie dla nowego zespołu (rosnąca prędkość)

Przewidywanie szybkości

- Istniejący zespół i znana technologia: historia, działaj, zgadnij
- Istniejący zespół i nowa technologia: działaj, historia, zgadnij
- Nowy zespół i znana technologia: działaj, zgadnij, historia
- Nowy zespół i nowa technologia: działaj, zgadnij, historia

Problem wielu ról jednej osoby



https://www.timecockpit.com/blog/2013/06/25/Six-Reasons-for-Time-Tracking-in-Agile-Projects

Problem wielu ról jednej osoby

- Czasochłonne
- Konflikt interesów
  - PO wymaga
  - SM chroni
- Problemy z implementacją, jako PO decyduję o usunięciu zadań
- Członek zespołu skupiony na zadaniu traci szerszy punkt widzenia

Wdrażanie technik inżynierskich

- Dlaczego nie TDD
  - Za dużo roboty z wyprzedzeniem i bezwartościowego kodu
  - Błędy wrzucamy do kolejnej iteracji
  - Mamy porządny debuger

Wdrażanie technik inżynierskich

- Dlaczego nie *continous integration*
  - U mnie działa! Szkoda czasu na synchronizację
- Formatowanie kodu
  - Piszę po swojemu, tak jest najszybciej
  - Szkoda czasu na zastanawianie się jak powinienem sformatować kod skoro działa

Wdrażanie technik inżynierskich

- Dlaczego nie programowanie w parach
  - Nie ma czasu na zabawę, musimy utrzymać prędkość
  - Email, facebook, pogoda, telefon…

Wdrażanie technik inżynierskich

- Wykazać, że pokonując barierę wejścia przyspieszenie będzie takie, że zwrot inwestycji czasu nastąpi po X iteracjach
- Dodatkowe zalety:
  - Większa przewidywalność czasowa
  - Lepsza jakość
  - Mniej nerwów
  - Kod jest wspólną własnością

Wdrażanie technik inżynierskich

- Dobre praktyki
  - Wdrażaj techniki stopniowo ale konsekwentnie
  - Wdrażanie jako elementy *backlog*
  - Spraw żeby zespół zobaczył potrzebę a nie czuł się zmuszony
  - Szkolenia zewnętrzne lub wewnętrzne

Warianty programowania w parach

- Bezładne programowanie w parach (Promiscuous Pair Programming)
  - Podział czasu na bloki
  - Rotacja pomiędzy zespołami
  - Primary i secondary

Warianty programowania w parach

- Micro-pairing
  - A: pisze test, który nie przechodzi
  - B: pisze kod, test przechodzi
  - B: pisze kolejny test
  - A: pisze kod, test przechodzi
  - A: pisze kolejny test
  - …

Organizacja dnia

- Elastyczne godziny pracy
  - Poranny jogging vs. wieczorne układanie lego z dziećmi
- Wspólna praca
  - Nauka
  - Współpraca
  - Poczucie wspólnoty i budowanie więzi

Organizacja dnia

- Cały zespół podaje godziny
  - Początek dnia pracy
  - Lunch
  - Koniec dnia pracy
- Znajdź część wspólną przed i po lunchu

Organizacja dnia

- Satysfakcjonująca?
  - Jeśli nie, poszukaj kompromisów, np. wspólny lunch
- Dodatkowe utrudnienia
  - Różne strefy czasowe
  - Praca w niepełnym wymiarze
  - Inne obowiązki w pracy i poza nią

Zarządzanie defektami

- Czy funkcjonalność może być traktowana jako gotowa jeśli zawiera defekt?
- Różne podejścia:
  - Przed wdrożeniem dwie iteracje poprawiania błędów i stabilizacji?
  - Dwa dni na koniec iteracji?

Zarządzanie defektami

- Dobra praktyka:
  - Priorytety defektów (2-4 poziomy)
  - Definicja priorytetów
  - Błędy krytyczne poprawiamy natychmiast (zasada 1:10:100)
    - Zasada jednej godziny
  - Istotne do *backlog*

Pilne zgłoszenia

- Use Case:
  - Zespół utrzymuje równolegle system
  - Inne zespoły nie mają takiej wiedzy
  - Zmienne tempo i dostępność zasobów
- Podejścia
  - Dedykowany czas
  - Dedykowany zespół

Sprint Review

Jaki jest najprostszy sposób na sprawdzenie w trakcie *Sprint Review* czy system i wdrożone funkcjonalności działają zgodnie z oczekiwaniami interesariuszy?

Sprint Review

- Pozwolić im przetestować to co powstało
  - Czy działa tak jak oczekiwali
  - Czy coś można poprawić
  - Czy przewidzieliśmy wszystkie schematy użycia
  - Dobre *Review* inicjuje działania w kolejnej iteracji
  - Budowanie wzajemnego zrozumienia

Sprint Review

- Problemy i niejasności
  - Czy slajdy zawsze są złym rozwiązaniem
  - Zakres: cel, co obiecaliśmy, co zrobiliśmy, czego nie zrobiliśmy, najważniejsze decyzje, wskaźniki, demonstracja
  - Rób notatki i proś o zatwierdzenie

Retrospektywa

- Nie ma czasu na retrospektywę gdy do zrobienia jest prawdziwa praca
- Narzekanie
- To zespół musi chcieć i wdrażać usprawnienia
- Ktoś z doświadczenia może opowiedzieć o ciekawym rozwiązaniu?

Retrospektywa

- Pomysły
  - Wynieść krzesła
  - Pilnować tempa i agendy
  - Tablice: co poszło dobrze, co poprawić
  - Pogrupować
  - Priorytetyzacja (każdy ma 100$)
  - Wybierz problemy do rozwiązania i przypisz im opiekuna

Codzienny Scrum, codzienne problemy

- Spóźnienia
- Głębokie rozważanie tematów
- Nieuwaga
- Zbyt długie i niejasne komunikaty
  - 2 dni robiłem jakieś testy i będę je kontynuował
- *Stand-up* dłuższy niż 15 min
- Pomysły na przeciwdziałanie?

Codzienny Scrum, codzienne problemy

- 15 min po przybyciu ostatniej osoby
- Kary: skarbonka na piwo, wypieki, pompki
- Tylko trzy pytania, nic więcej i tempo
- Każdy powinien mieć przygotowane odp.
- Maskotka dająca głos, każdy może zgłosić odejście od tematu
- Cierpliwość dla nowych zespołów

Powiększanie zespołu

» Ciąg przyczynowo-skutkowy:

Problemy z dostarczenie projektu na czas
-> Powiększmy zespół ->
Wdrożenie nowej osoby zajmuje czas ->
Projekt zwalnia ->
Problem jest jeszcze większy

» Pomysły na rozwiązanie?

Powiększanie zespołu

- Kilka sesji programowania w parach
- Przygotować test dotyczący system, pytania na które nie będzie znał odpowiedzi sprawią, że będzie się uczył najważniejszych spraw
- Stany zespołu: formowanie, burzenie, normowanie, działanie
- Wymuszenie burzenia żeby wzbudzić poprawę

Wychodzenie z katastrofy

- Przykład: Scrum Master przeniesiony do innego zespołu w połowie ważnego sprintu
- Możliwości:
  - Usuń przeszkody
  - Poproś o pomoc
  - Ogranicz zakres i cel iteracji
  - Odwołaj iterację

Zrównoważone tempo

- Tuż przed terminem zespół pracuje po 60 godzin tygodniowo
- Później wydajność znacznie spada, potrzebują odpoczynku
- Nie są w stanie działać w równym tempie bo interesariusze nie współpracują
- Pomysły na rozwiązanie?

Zrównoważone tempo

- Pokazać interesariuszom, że problem jest także po ich stronie (może być bolesne)
- Rozwiązanie długoterminowe:
  - Zwolnić do komfortowego tempa
  - Wdrażać techniki sukcesywnego zwiększania tempa (krótsze iteracje)
  - Monitorować proces i wnosić poprawki (burndown)

Działający kod po każdej iteracji

- Co w przypadku złożonego systemu?
  - Np. system do obsługi zleceń i monitorowania przesyłek
  - Rozbudowane bazy danych
  - Bezpieczeństwo
  - Integralność danych
  - Interfejs użytkownika
- Czy można pokazać cokolwiek?

Działający kod po każdej iteracji

- Pokazać podstawową funkcjonalność: złożenie zamówienia (od początku do końca)
- Inne elementy systemu jedynie pozorowane (mock)
- Np. baza danych = plik
- Ważna jest dla nas informacja zwrotna

Działający kod po każdej iteracji

- Krytyczne kwestie
  - Skalowalność
  - Rozpocznij pracę od najbardziej ryzykownego elementu
  - Rozbudowa i ciągłe usprawnianie do chwili gdy w pełni spełnisz oczekiwania interesariuszy
  - Kompletne scenariusze

Jak zespół spędza czas

- Standardowo na wykresach są zadania związane z iteracją
- Naciski z góry na lepsze wyniki
- Warto pokazać dodatkowe zadania:
  - Obowiązkowe spotkania
  - Estymacja przyszłych zadań
  - Dług techniczny
  - Inne

Jak zespół spędza czas

- Rozwiązania:
  - Redukcja spotkań lub ich długości
  - Estymacja zadań związanych z estymacją przyszłych zadań
  - Wpisywanie zadań technicznych do *backlog*

Jak zespół spędza czas

- Podsumowując:
  - Na wykresach prezentujemy różne typy zadań
  - Pełna transparentność, jeśli spłacamy dług technologiczny to nie wstydźmy się do tego przyznać
  - Monitorujmy proporcje pomiędzy typami zadań

Dokumentacja

- Scrum czyli zero dokumentacji?
- Dokumentacja na początku projektu
  - Nie wiemy jak potoczą się sprawy
  - Marnotrawimy czas i zasoby
  - Ogólna
  - Najważniejsze historie użytkowników

Dokumentacja

- Dokumentacja na końcu
  - Brak czasu
  - „Pełna” wiedza, dużo czasu wpłynęło od wykonania pracy
- *Document as we go*
  - Aktualizacja dokumentacji możliwa w trybie ciągłym
  - Jak przekonać interesariuszy

Dokumentacja

- Podejmij decyzje o sposobie prowadzenia dokumentacji
- Trzymaj się tego planu
- Potwierdź z interesariuszami
- Zainwestuj w automatyzację
  - Procesu dokumentacji
  - Innych procesów => mniej dokumentacji

Problem ogromnego *backlogu*

- Tło:
  - 20 interesariuszy
  - 300 historii
  - Wszystko należy estymować i nadać priorytety
- Pierwsze kroki:
  - Wydrukuj kartki z historiami
  - Znajdź naprawdę dużą ścianę ☺

Problem ogromnego *backlogu*

- Pierwsze spotkanie tylko z zespołem
  - Podział historii na „duże i małe” (przyklejone po prawej i lewej)
  - Dalsze grupowanie historii ze względu na punkty
  - Oznaczenie historii, które budzą wątpliwości

Problem ogromnego *backlogu*

- Drugie spotkanie, zaproszenie interesariuszy
- Drugi wymiar podziału (interesariusze):
  - Na górze historie o dużym priorytecie
  - Na dole historie o małym priorytecie
- Oznaczanie problematycznych historii
- Ostatni krok to podzielić duże historie o najwyższym priorytecie

Problem ogromnego *backlogu*

- Wynik
  - Lewy górny róg, małe historie o dużym priorytecie, pierwsze do zrobienia, ryzyko: mogą okazać się większe
  - Prawy górny róg, duże historie o dużym priorytecie, kolejne do zrobienia, ale należy je szybko przeanalizować

Problem ogromnego *backlogu*

- Wynik
  - Lewy dolny róg, małe historie o małym priorytecie, mogą znikać w trakcie
  - Prawy dolny róg, duże historie o małym priorytecie, odległa przyszłość
- Walidacja na koniec
- *Backlog grooming* to proces ciągły

Problem ogromnego *backlogu*

- Wskazówki
  - Ustal ramy czasowe
  - Kontrowersyjne historie odkładaj na koniec
  - Bądź przygotowany, że powstaną kolejne historie
  - Przypomnij, że wszystko się zmieni

Jak przekonać klienta

- Będziesz płacił w małych transzach tylko za działające oprogramowanie
- Unikać stwierdzeń, że Scrum rozwiązuje wszystkie problemy, później się to zemści
- Wyjaśnij klientowi szczegóły działania
- Okres wypowiedzenia o długości X iteracji

Struktura kontraktu

- Kontrakt wstępny
  - Ustal zakresy
    - Określ typ użytkowników
    - Napisz historie użytkowników
    - Dokonaj estymacji
  - Ustal koszt i czas wykonania

Struktura kontraktu

- Kontrakt wykonawczy
  - Zmiany
- Klient musi być aktywny
- Określone okno czasowe na akceptację
- Nadawanie priorytetów w trybie ciągłym
- Zaufanie

Typowe problemy i rozwiązania

Problem	Rozwiązanie
Eksperci dokonują estymacji	Poker
Nie wszyscy czują się właścicielami kodu	Programowanie w parach
Zbyt dużo zadań przypisanych jednocześnie	Odgórne ograniczenie do jednego lub dwóch
Zbyt duże zadania	Skróć długość iteracji
Nieznany status zadań	Użyj tablicy

Dziękuję za uwagę!

Piotr Boryło