Zwinne metodyki wytwarzania oprogramowania

1. Wstęp

Podczas analizy różnych zwinnych metodyk wytwarzania oprogramowania, szukaliśmy takiej która sprawdzi się w małym dwuosobowym zespole. Jednocześnie nie będzie wprowadzała znacznego obciążenia związanego z dużą liczbą generowanych artefaktów.

Rozważane metodyki:
AgileUP - Z metodyki korzystamy na projekcie z PSI. W przypadku naszego projektu zbyt duży nacisk kładzie na modelowanie

SCRUM - Z metodyki SCRUM korzystaliśmy już podczas pracy inżynierskiej. W dwuosobowym zespole ciężko wyróżnić role scrum-mastera.

XP - Przestrzeganie wszystkich założeń metodyki przy naszej pracy nie byłoby możliwe. Podczas statrup-u większe korzyści widać ze zrównoleglenia procesu, a niżeli pracy dwóch osób nad jednym zadaniem.

Kanban - Prosta w realizacji. W łatwy sposób wizualizuje wykonywane zadania. Może dać zadowalające efekty w małym zespole.

Dlatego postanowiliśmy wybrać metodykę Kanban.

2. Backlog
   - 2.1 Prototypy interfejsów
   - 2.2 Konfiguracja serwera CI
   - 2.3 Automatyczne testy logiki aplikacji
   - 2.4 Implementacja logiki aplikacji
   - 2.5 Implementacja interface'u użytkownika
   - 2.6 Dokumentacja

3. Opis Metodyki

System Kanban został zapoczątkowany w latach 40 przez koncern Toyouta. Pierwotnie w koncernie używano różnych pojemników zawierających części aby jasno komunikować, które materiały mają być uzupełniane na bieżąco. W idealnej sytuacji materiały są dostarczane na linie dokładnie w momencie ich użycia, co eliminuje konieczność ich magazynowania.

Idee te można przenieść także do procesu wytwarzania oprogramowania. Podstawowym założeniem metodyki jest prioretetyzacja i kolejkowanie zadań. Zadania, które należy wykonać zapisuję się na pojedynczych karteczkach po czym trafiają one do jednej z kolumn “To do”, “In progress”, “Done”. Ilość zadań w stanie “In progress” jest ograniczona. Czas wykonywania zadań jest monitorowany celem likwidacji wąskich gardeł.

Tablica Kanban prezentuje, wizualizuje zadania czekające na wykonanie, aktualnie wykonywane i wykonane

Opis Etapów:

1. TODO - korzystając z backlogu wybierana są najistotniejsze zadania, które powinny być zrealizowane w przeciągu najbliższych tygodni. Zadania mają przydzielony priorytej: LOW, MEDIUM HIGH. Zadania zostają przypisane do osób. Powinny być one możliwie elementarne. Do zadania jest estymowany przewidywany czas jego realizacji. Podczas wykonywania zadania programista loguje rzeczywisty czas jego wykonania, co pozwala poprawić przyszłe planowanie.

2. In Progress - jeżeli zadanie jest aktualnie wykonywane przez programistę przechodzi do stanu In Progress. Maksymalna liczba aktualnie wykonywanych zadań jest ograniczona przez liczbę wykonawców do dwóch

3. Jeśli zadanie jest wykonane przechodzi do stanu Done. Po tym jest sprawdzane przez drugiego członka zespołu. A na zajęciach przez prowadzącego. Jeżeli pojawią się błędy związane z zadaniem do poprawy, zostają one zapisane w zadaniu i zadanie trafia do etapu TODO. Poprawa błędów ma priorytet wyższy niż implementacja nowej funkcjionalności.

Wyróżnione role w zespole:

- Właściciel projektu - zleca zadania, kontroluje wytwarzane artefakty
- Programiści - wykonują zadania, aktualizują przepływ zadań na tablicy

4. Wybrane narzędzia:

   4.1 Do wizualizacji metodyki oraz planowania zostanie użyta wirtualna tablica: https://kanbanflow.com/
   4.2 Do implementacji zostanie użyte środowisko R-studio
   4.3 Do projektowania mockapów zostanie użyte narzędzie pencil
   4.4 Do tworzenia dokumentacji, narzędzia pakietu office, google docs
   4.5 Repozytorium kodu - bitbucket.org
   4.6 Hosting aplikacji http://www.shinyapps.io/
   4.7 Serwer CI - jenkins działający w chmurze openshift (O ile uda się go połączyć z serwerem aplikacji)

5. Architektura aplikacji

Wybrany framework definiuje w dużym stopniu architekturę aplikacji.
Interfejs użytkownika musi być opisany w pliku ui.R. Obsługa interfejsu użytkownika jest realizowana przez plik serwer.R.

Ponadto chcemy wydzielić logikę aplikacji do osobnych modułów dla których będą wykonywane testy jednostkowe w RUnit zgodnie z zasadą TDD - test, code, refactore.