Inżyniera Oprogramowania 2

Klaster Obliczeniowy

Dokumentacja wstępna

Michał Berent, Marcin Chudy, Maciej Lewinski, Mateusz Jabłoński

Politechnika Warszawska
Wydział Matematyki i Nauk Informacyjnych

Akceptacja specyfikacji

Akceptujemy obecną formę specyfikacji z dnia 3 marca 2016 r.

Metodologia pracy

Zastosowaną metodologią pracy będzie Extreme Programming (XP) dostosowane do rozmiaru zespołu oraz specyfiki tworzonego projektu. Stosowane przez nas praktyki pochodzące z programowania ekstremalnego to:

- Test Driven Development – zaczynamy od pisania testów do funkcjonalności, po czym ją implementujemy
- Continuous Design – brak całościowego planowania z góry, dopuszczalne ciągłe zmiany w architekturze projektu
- Iteracyjność - projekt będzie tworzony w iteracjach (odpowiadającym kolejnym etapom oddawania projektu)
- Możliwie prosta architektura – przy planowaniu rozpatrywana będzie najprostsza architektura konieczna do osiągnięcia rezultatu (zgodna z zasadami programowania obiektowego oraz otwarta na modyfikacje), z możliwością jej rozbudowy w przyszłości, jeśli zajdzie taka potrzeba
- Częsta i ciągła refaktoryzacja kodu – zmiany w kodzie poprawiające jego jakość, usuwanie redundantnych fragmentów są dozwolone w każdym momencie (każdy członek zespołu jest uprawniony do zmian we wszystkich fragmentach kodu)

Rezygnujemy z niektórych z cech programowania ekstremalnego, takich jak programowanie w parach (ze względu na problemy logistyczne oraz zbyt małą ilość członków zespołu) oraz ciągłej komunikacji z klientem (ze względu na specyfikę projektu).

Git workflow

Na gałęzi `master` będzie utrzymywana aktualna stabilna wersja systemu. Będzie ona uaktualniana na koniec każdej iteracji (kolejne wersje systemu będą ponadto oznaczone za pomocą tagów). Na gałęzi `develop` będzie utrzymywana aktualna developerska wersja systemu (nie powinna ona zawierać kodu pochodzącego z niedokończonych zadań). Dla każdego zadania odpowiedzialny za nie członek zespołu będzie pracował na dedykowanej gałęzi (tzw. *feature branch).* Po zakończeniu pracy gałęzie te będą dołączane do gałęzi `develop`. W momencie dołączania gałęzi kod musi przechodzić wszystkie dotychczasowe testy. Krytyczne poprawki błędów mogą być dokonywane bezpośrednio na gałęzi `develop`. Opisy commitów powinny być krótkie i zwięzłe, ale jednocześnie dobrze opisywać, jakie zmiany wprowadzają.

Technologie

Projekt zostanie stworzony z wykorzystaniem technologii .NET w wersji 4.6 oraz języka C# 6.0. Do tworzenia testów jednostkowych zostanie wykorzystany framework xUnit.net.

Spotkania zespołu

Spotkania planowania iteracji

Przed rozpoczęciem każdej iteracji planujemy organizować spotkania na których nastąpi wybór oraz analiza koniecznych do zrealizowania zadań. Każde z zadań będzie miało przydzielony priorytet (niski, normalny lub wysoki) oraz osobę odpowiedzialną za wykonanie.

Stand-up meetings

Nie planujemy codziennych spotkań zespołu ze względu na problemy czasowe oraz logistyczne. Krótkie spotkania tego typu będziemy organizować w przypadku nieoczekiwanych problemów (np.

trudnością z ukończeniem jednego z zadań lub koniecznością przeprowadzenia dyskusji na temat zmian w architekturze projektu).

Zadania

Praca, zgodnie z zasadami programowania ekstremalnego, została w miarę możliwości podzielona na możliwie małe zadania. Szczegółowy plan oraz przydział zadań do poszczególnych członków zespołu będzie tworzony podczas spotkania planowania iteracji.

Osoba odpowiedzialna za każde z zadań jest również odpowiedzialna za zapewnienie odpowiednich testów jednostkowych.

Priorytet zadań jest oznaczony literą (H – wysoki, N – normalny, L – niski).

Przydział zadania jest oznaczony inicjałami (MB - Michał Berent, MC – Marcin Chudy, MJ – Mateusz Jabłoński, ML – Maciej Lewinski)

Komunikacja

- Stworzenie podstawowej struktury solucji (dodanie projektów dla komponentów i projektów testowych) H MC (zrobione)
- Obsługa pliku konfiguracyjnego w serwerze H MJ
- Obsługa pliku konfiguracyjnego w pozostałych komponentach H MB
- Obsługa parametrów z linii poleceń L ML
- Stworzenie klas reprezentujących wiadomości H MJ
- Mechanizm przesyłania wiadomości H MC
- Obsługa awarii węzła N MJ
- Dodanie trybu backup dla serwera L MC
- Obsługa awarii serwera N ML
- Stworzenie mocków umożliwiających testowanie przepływu przykładowych wiadomości pomiędzy komponentami H ML
- Testy integracyjne komunikacji między komponentami H MB
- Dodanie trybu verbose L MB

Obliczenia

- Zaprojektowanie algorytmu do rozwiązania DVRP
- Stworzenie mechanizmu pluginów (dla różnych Task Solverów)
- Implementacja algorytmu do rozwiązania DVRP
- Równoległa obsługa wielu węzłów przez serwer

Współpraca

- Testy systemowe z innymi grupami
- Optymalizacja algorytmu

- Optymalizacja komunikacji