IO2 - podział zadań

March 10, 2016

1 Informacje wstępne

Poniżej znajduje się lista zadań przewidzianych w ramach projektu z Inżynierii Oprogramowania 2. Wskutek niezależności większości komponentów, zadania zostały podzielone na oddzielne części. Praca dotyczy implementacji i testowania zarówno komponentów klastra (*Computational Client*, *Communication Server*, *Task Manager*, *Computational Node*), implementacji zadanych algorytmów (*Algorithm Solvers*) jak i ogólnej części *Communication Utils* dostarczającej funkcjonalności dotyczących komunikacji wewnątrz klastra. Zadania zostały podzielone na trzy etapy dane ze specyfikacji, na ich podstawie zostaną przeprowadzone *sprinty*. Dokładny przydział zadań do osób jest ustalany na spotkaniach, informacje na jego temat można znaleźć w notatkach w dokumentacji projektu. Podana została również wycena czasowa poszczególnych zadań.

2 Etap komunikacji

2.1 Communication Utils

- Stworzenie generycznych interfejsów pozwalających komponentom na komunikację po protokole TCP (10h)
- Stworzenie wiadomości wszystkich typów na podstawie schemy (3h)
- Skonstruowanie ogólnej klasy Message, bedącą klasą bazową dla wszystkich wiadomości o strukturze danej ze schemy. (3h)

2.2 Algorithm Solvers

- Stworzenie szkieletowej implementacji TaskSolvera dla *DVRP* - na potrzeby etapu komunikacji zastosować zaślepki w miejscu prawdziwej implementacji algorytmu. (3h)

2.3 Computational Client

- Skonstruowanie szkieletu klas dla projektu (4h)
- Poprawna komunikacja z serwerem bez uwzględnienia awaryjności (3h)
- Poprawna komunikacja z serwerem z uwzględnieniem awaryjności (3h)

2.4 Communication Server

- Skonstruowanie szkieletu klas dla projektu (5h)
- Konfiguracja systemu przez parametry wywołania (3h)
- Implementacja sposobu komunikacji serwer - backupy (4h)
- Implementacja komunikacji serwer - klient (3h)
- Implementacja komunikacji z Task Managerami i węzłami obliczeniowymi (5h)
- Poprawne kolejkowanie wiadomości (3h)
- Uwzględnienie awaryjności (6h)

2.5 Task Manager

- Skonstruowanie szkieletu klas dla projektu (4h)
- Komunikacja z serwerem (4h)
- Zastosowanie zaślepek w kodzie (mocków) mających na celu imitację przyszłego dzielenia problemu przez komponent w celu przetestowania poprawności implementacji komunikacji w systemie (4h)
- Uwzględnienie awaryjności (3h)

2.6 Computational Node

- Skonstruowanie szkieletu klas dla projektu (4h)
- Komunikacja z serwerem (4h)
- Zastosowanie zaślepek w kodzie (mocków) mających na celu imitację obliczeń przez komponent w celu przetestowania poprawności implementacji komunikacji w systemie (4h)
- Uwzględnienie awaryjności (3h)

2.7 Inne

- Globalna konfiguracja systemu (3h)

3 Etap obliczeń (computations)

3.1 Algorithm Solvers

- Stworzenie w pełni działającej implementacji algorytmu dla DVRP (8h)
- Prace nad optymalizacją implementacji TaskSolvera (w tym wielowątkowość, "sztuczki" w wykładniczym algorytmie) (9h)

3.2 Computational Client

- Obsługa przesyłania i odbierania problemu (5h)

3.3 Task Manager

- Poprawne dzielenie problemów (4h)
- Uwzględnienie wielowątkowości (5h)

3.4 Computational Node

- Poprawne obliczenia problemu (4h)
- Uwzględnienie wielowątkowości (5h)

3.5 Inne

- Poprawki wskutek regresji (5h)

4 Etap współpracy (cooperation)

4.1 Computational Client

- Prace nad kompatybilnością (z innymi grupami) (5h)

4.2 Communication Server

- Prace nad kompatybilnością (z innymi grupami) (6h)

4.3 Task Manager

- Prace nad kompatybilnością (z innymi grupami) (4h)

4.4 Computational Node

- Prace nad kompatybilnością (z innymi grupami) (4h)

4.5 Inne

- Poprawki wskutek regresji (5h)

5 Ogólna wycena

Szacowany czas łącznej pracy implementacyjnej nad projektem wynosi 167 godzin. Czas pracy nad zadaniami na osobę, przy założeniu równomiernego nakładu pracy, wynosi około 42 godzin. Dodatkowo trzeba naliczyć pewien czas przeznaczony na spotkania scrumowe (na żywo i wirtualnie) oraz na pracę nad dokumentacją i sporządzanie notatek, który szacujemy na około 25-30 godzin - czyli dodatkowo 6-8 godzin na osobę łącznie.