Process van het maken van een automatische GPU opzichter

Lars Beentjes
Luc de Jonckheere
Gilles Ottervanger
Rob Reijtenbach
Kean Tettelaar
Elgar van der Zande

19 Januari 2017

Teamwork

- Regelmatige meetings
  - Deadlines stellen
  - Gemaakt werk evalueren
  - Ideeën delen
- Whatsapp groep voor snel contact
  - Vragen over elkaars code
  - Vragen over design keuzes
  - Plannen van meetings
- GitHub voor makkelijke samenwerking
  - Overzichtelijk code uitwisselen
  - Issues voor het melden van problemen

Contact met opdrachtgever

- Eerste meeting
  - Iedereen aanwezig
  - Technische requirements bespreken
  - Interactie vastleggen (input/output)
  - Deadlines bespreken
- Latere meetings
  - Kleinere groep mensen
  - Product in huidige staat laten zien
  - Bespreken wat anders moet/beter kan
  - Eerder gemaakte requirements bijstellen

Projectmatig werken

- Combinatie van waterfall en agile
- Waterfall:
  - Requirements vanaf het begin duidelijk
  - Alle feutures werden tegelijk gemaakt (losse modules)
  - Losse stappen in successie (begin)
    - Eerst duidelijk requirements bespreken
    - Met z'n alleen over een goed design gebrainstormd
    - Design implementeren
    - Afwijking...
- Agile:
  - Losse teams (pair programming)
  - Elke interview werkende software
  - Geen groot documentatie document, maar duidelijke code met benodigde comments
  - Nieuw interview na grote verandering
  - Handige/efficiente methode door grootte van het project

Requirement engineering

- Requirements eenduidig; technieken niet erg nodig
- Functionalistische benadering
- **Elicitatie**: interview met opdrachtgever en brainstorming
- **Specificatie**: requirements uitgewerkt en opgeschreven
- **Validatie**: genoteerde requirements naar opdrachtgever gemaild
- **Onderhandelingen**: e-mails met opdrachtgever en interviews
- Weinig sprake van MoSCoW

Design

- Model-View-Controller achtig design patroon
  - Model: GPUMonitor
  - View: GPUViewer
  - Controller: Violation Detector
- Tegelijk ontwikkelen goed mogelijk door design keuze
- Onderdeel kan makkelijk vervangen worden als het niet voldoet
- Geen modeling nodig gehad door kleine codebase en duidelijke code(structuren)
- Code achteraf refactored om “bad smells” te minimaliseren

Kwaliteit

- Opdrachtgever/ontwikkelings manager:
  - Gevraagd probleem opgelost
  - Code makkelijk aan te passen/goed te onderhouden
- Gebruiker:
  - Makkelijk te gebruiken
  - Duidelijke handleiding meegegeven
  - Snel automatiseren van normaal handmatige taken
- Ontwikkelaar:
  - Makkelijk design
  - Logisch design
  - Losse onderdelen, dus makkelijk om delen te hergebruiken

Testen

- Verificatie:
  - Lastig te automatiseren in dit project
  - Weinig mogelijkheden voor input (per module); makkelijk handmatig te testen
  - Modulaire opbouw zorgt voor consistentie in werking
- Validatie:
  - Idle script: kan een process op bepaalde GPUs idle laten draaien
  - Stress test: start process die veel werk uitvoeren