Project Software-ontwikkeling en Professionele Vaardigheden

Vereisten en richtlijnen voor het analyseverslag

Universiteit Hasselt

Academiejaar 2014-2015

1. Doel

De eerste weken van het project worden besteed aan het maken van een analyse: onderzoek naar de nodige algoritmen en datastructuren met behulp van cursussen, boeken en het Internet. Het resultaat van deze analyse wordt beschreven in het analyseverslag. Hiernaast bevat het verslag ook een planning en een taakverdeling.
Hierdoor worden jullie gestimuleerd om, nog vóór er een regel broncode wordt geschreven, na te denken over de belangrijkste/meest complexe onderdelen van het project, waaronder datastructuren en algoritmen, OO ontwerp, een gedetailleerde planning en taakverdeling.
Bovendien krijgen de begeleiders en opdrachtgever zo al vroeg een duidelijk beeld van de geplande werkwijze, zodat er indien nodig tijdig kan bijgestuurd worden.

2. Inhoud

Onderstaande onderdelen verwachten we in het analyseverslag.

Titelpagina

Maak een duidelijk voorblad waarop minstens informatie zoals groepsnummer, namen van de teamleden, onderwerp, en academiejaar vermeld worden.

Diepgaande beschrijving van het onderwerp

Leg uit wat het einddoel is van het project. Beschrijf grondig de interpretatie van de opgave. Vermeld ook welke features van je applicatie je als prioritair beschouwt, en welke als extra's voorzien worden indien er tijd overblijft. Bespreek wat de noden van de opdrachtgever(s) zijn, en welke algemene features je project moet ondersteunen om aan die noden te voldoen. De gedetailleerde beschrijving van hoe je die features zal gaan vertalen naar een goed gedefinieerd informatica-probleem, gebeurt in de rest van dit analyseverslag. Leg alle begrippen die je gebruikt voldoende uit. Dit is de basis waarop de rest van het project zal verder gewerkt worden.

Bestaande software

In deze sectie wordt een lijst gegeven van bestaande software die een gelijkaardige functionaliteit heeft. Geef aan wat je hieruit wel/niet meeneemt in je eigen project en waarom. Wees volledig!

Geraadpleegde bronnen / literatuur

In deze sectie wordt aangegeven welke implementaties en technieken er reeds bestaan om

het beschouwde probleem op te lossen. Voorbeelden hiervan zijn papers of boeken die de nodige algoritmes of technieken beschrijven, en internetbronnen die je geraadpleegd hebt.

Evaluatiecriteria

In deze sectie geven jullie een lijst van features en criteria waaraan het eindproduct getoetst moet worden. Deze lijst zal belangrijk zijn om tijdens het schrijven van de code te controleren of je alle gevraagde functionaliteit geïmplementeerd hebt.

Zorg ervoor dat de lijst volledig is, en dat de criteria concreet en meetbaar zijn. Voorbeelden van concrete criteria zijn:

- "Het uitvoeren van actie A neemt maximaal 100 milliseconden in beslag."
- "Het algoritme moet rekening houden met deze en deze constraints."
- "Verkeerde invoer van de gebruiker wordt correct opgevangen en een toepasselijke foutboodschap wordt getoond."

Analyse

In dit deel wordt de eigenlijke analyse van het project besproken. Hierin moet zeker aan bod komen:

- Beschrijving en verantwoording van de gekozen algoritmes
  Complexe algoritmes of onderdelen ervan worden volledig toegelicht, uitgewerkt en in pseudo-code beschreven om de eigenlijke implementatie later te vergemakkelijken; alternatieven worden besproken.

- Beschrijving en verantwoording van de keuze van ADT's
  Wat zijn de belangrijke/complexe ADTs die in je applicatie nodig zijn? Hoe ga je ze implementeren? Verwijs ook naar low-level beschrijvingen; bv. *std::map*.
  Elke keuze moet ook voldoende gemotiveerd worden; bv. w*aarom gebruik je een boomstructuur? Waarom is structuur X efficiënter dan structuur Y?*

  Merk op dat de keuze vaak gekoppeld is aan specifieke algoritmes. Je mag ervoor kiezen om de ADTs bij de algoritmen zelf toe te lichten, of hieraan een aparte sectie te wijden.

- Klassendiagram en OO ontwerp
  Een klassendiagram kan je in eerste instantie afleiden uit de beschrijving van het onderwerp en later verder uitbreiden en aanpassen. Geef hier de globale structuur van je applicatie aan: welke klassen worden geïmplementeerd en – heel belangrijk – welke onderlinge relaties hebben ze. Licht de belangrijke zaken uit het diagram ook kort toe.

  Verantwoord bondig de belangrijkste keuzes die gemaakt werden betreffende het OO ontwerp, aan de hand van je opgedane kennis uit de vakken Object-georiënteerd programmeren I en Object-georiënteerd programmeren II. Zorg voor een goede scheiding tussen je grafische user interface en de applicatielogica (Model-View), pas de GRASP patronen toe om verantwoordelijkheden te verdelen tussen de verschillende klassen (denk o.a. aan low coupling, high cohesion), en hou rekening met de SOLID principes (Single Responsibility Principle, Open/Closed Principle, …).

  Beperk je op dit moment tot de meest belangrijke onderdelen van de applicatie. Beperk je bijvoorbeeld voor het GUI gedeelte enkel tot de koppeling tussen de GUI en applicatielogica, en laat de specifieke details van de GUI klassen weg. Vermeld wel zeker

klassen die deel uitmaken van de eerder beschreven algoritmen.

- Data en databaseschema (indien van toepassing)
  In applicaties die data opslaan, verwerken en/of (interactief) weergeven dient een inventaris gemaakt te worden van welke data er precies nodig is, wat er met die data dient te gebeuren (welke operaties dient de applicatie aan te bieden, in welke stappen wordt data verwerkt) en hoe de data intern opgeslagen gaat worden. Dit kan bijvoorbeeld gebeuren in een SQL database, of met behulp van speciale data-objecten. Vermeld deze in je verslag, en geef aan op welke manier je beslist hebt over de structuur waarin deze data zal worden bewaard.

- Bestand (indien van toepassing)
  Indien je applicatie gegevens van bestanden zal inlezen of naar bestanden zal wegschrijven, leg je best op voorhand al een beschrijving van de bestandsstructuur vast. Indien er reeds bestaande formaten voor het probleem bestaan, ga dan na of het mogelijk is om dit formaat ook te gebruiken.

Mockups

Geef een initieel beeld van hoe de grafische interface van de applicatie eruit zal zien met behulp van niet-interactieve paper mockups. Maak hiervoor gebruik van de opgedane kennis en vaardigheden uit het vak Humane en Sociale Aspecten van Informatica (o.a. het toepassen van ontwerpprincipes en richtlijnen). Let erop dat de mockups concrete voorbeeldinhoud bevatten (dus geen placeholders zoals "lorem ipsum" tekst of lege afbeeldingen). Het is voldoende om de gemaakte schetsen als ingescande afbeeldingen toe te voegen aan het analyseverslag.

Taakverdeling en planning

Op basis van de analyse is het duidelijk welke onderdelen ontworpen moeten worden. Het is belangrijk om de taken op voorhand te verdelen en hier ook een planning aan te koppelen.

Stel een aantal mijlpalen voorop die stap voor stap tot een afgewerkte oplossing leiden. Stel hierbij de juiste prioriteiten! Welke vereisten vormen de kern van de applicatie? Welke afhankelijkheden zijn er?
Zorg ervoor dat iedereen een evenwichtig takenpakket heeft gedurende de implementatie van het project. Een taakverdeling waarbij één persoon de volledige GUI implementeert en een andere persoon de overige code is niet toegelaten. Koppel ook deadlines aan mijlpalen en geef elk groepslid voor elke mijlpaal een aantal duidelijk afgelijnde taken.

Maak de planning en taakverdeling zo gedetailleerd mogelijk! Het is onvoldoende te vermelden dat "teamlid A tijdens het eerste trimester werkt aan de zoekboom". Schrijf dat "teamlid A van dag X1 tot dag Y1 werkt aan het invoegen van nodes, van X2 tot Y2 aan het testen van het verwijderen van nodes, ...".

Let op: zorg ervoor dat de planning realistisch is; laat voldoende ruimte voor onderdelen die later bijkomend onderzoek kunnen vereisen, en voorzie steeds voldoende tijd om grondig te testen.

Bibliografie

Een lijst van geraadpleegde werken. Zorg voor duidelijke referenties in de tekst.

Log

Zoals reeds aangegeven in de algemene opgave is het verplicht om gebruik te maken van de versioning server. Bundel de commit messages, samen met een algemeen overzicht van wanneer welk werk geleverd werd, in een aparte sectie van je analyseverslag.

Let op: zorg steeds voor een gepaste *commit message*, zodat de commit log uiteindelijk als echt logboek kan dienen.