Software project management plan voor Schedule-Generator

Matthias Caenepeel Adam Cooman Alexander De Cock
Zjef Van de Poel

17 februari 2011 Versie 2.0

Aanpassingsgeschiedenis

. 16/2/2011 versie 0.1: Aanmaak document

. 17/2/2011 versie 0.2: Overzetting naar Tech, toevoeging hoofdstuk 1 en 4

. 19/2/2011 versie 0.3: Toevoeging hoofdstuk 5 en 6

. 20/2/2011 versie 1.0: Verbeteringen doorgevoerd en titels vertaald

. 25/2/2011 versie 1.1: Opmerkingen opdrachtgever in achting genomen en verbeteringen doorgevoerd

. 03/3/2011 versie 1.2: Deadlines toegevoegd om iteratie 1 te verwezenlijken

. 03/3/2011 versie 2.0: Volledige revisie van het document doorgevoerd en grondige aanpassingen gedaan zodat het weer volledig up-to-date is met de huidige stand van het project

Inhoudsopgave

1 Overzicht

1.1 Samenvatting van het project

1.1.1 Doel, bedoeling en objectieven

De volledige (Engelstalige) opdracht kan gevonden worden in het onderdeel 'Bijlagen' van dit software project management plan.

Het doel van het project is de ontwikkeling van een programma dat toelaat om lessenroosters te genereren en weer te geven op een universiteit. Het moet toelaten om data in te voeren (zoals vakken, professoren, lokalen,...) en beperkingen op het plannen van het rooster. Het moet het beste rooster kunnen opstellen. Het moet toelaten om dat rooster manueel aan te passen en te bekijken.

Het programma moet gebruik maken van een website en de data moet opgeslagen worden in een database. Alles moet open-source zijn.

1.1.2 Veronderstellingen en beperkingen

Het project is een opdracht voor het vak 'Software Engineering', gedoceerd door Prof. Ragnhild Verstraeten. Daarom worden de meeste constraints bepaald door de opdracht die gegeven is.
Er is tijd tot eind mei (20/5/2011) om het project af te werken. Een tijdsbestek van 3 maanden. We moeten alles open source programmeren en mogen enkel gebruik maken van open-source onderdelen. Het project moet uitgevoerd worden in een object-georienteerde taal en het programma moet op Wilma kunnen draaien.

1.1.3 Deliverables van het Project

De opdrachtgever gaf ons het volgende schema dat we moeten volgen in verband met de deliverables:

Datum	To Do	delivery media
22/02/2011	indienen SPMP	pdf via Site en mail
08/03/2011	indienen SRD en SDD	pdf via Site en mail
16/03/2011	SCRUM meeting	presentatie
08/04/2011	Einde 1ste iteratie	presentatie + code via svn
27/04/2011	SCRUM meeting	presentatie
20/05/2011	Einde 2de iteratie	code via svn
25/05/2011	presentatie eindresultaat	presentatie

Om de code toegankelijk te maken voor elk groepslid en de opdrachtgever, wordt gebruik gemaakt van een online repository in combinatie met subversion

om de version control in orde te houden. De repository bevindt zich op een server van google code(http://code.google.com/p/schedule-generator/). De link ernaar is te vinden op onze website(http://student.vub.ac.be/ acooman/SE/SE.html). De verslagen zoals het SPMP, SDD en SRC, zullen in pdf beschikbaar gesteld worden op onze website. De exacte inhoud van de SCRUM meetings werd opgegeven door de opdrachtgever. De volgende dingen worden erin getoond en besproken:

- Een demonstratie van de toegevoegde functionaliteiten sinds de vorige iteratie .
- Een analyse van de obstakels en de beslissingen die genomen zijn om ze op te lossen
- Een bespreking van de functionaliteiten die toegevoegd zullen worden in de volgende iteratie
- Een bespreking van de obstakels en risico's die tegengekomen kunnen worden in de volgende iteratie
- Een bespreking van de statistieken zoals werkuren per persoon
- Een bespreking van mogelijke vertragingen en oplossingen om die vertragingen zo klein mogelijk te houden en te voorkomen in de toekomst.

1.1.4 Planning en Budget

Het project wordt in 5 fasen onderverdeeld: Initialisatie, Design, Iteratie 1, Iteratie 2 en tenslotte Terminatie.

Initialisatie fase:

- *Doel:* Verkennen van de opdracht en de nodige kennis vergaren om de opdracht tot een goed einde te brengen.
- *Subtaken:*

  Opzoeken van informatie en software tools
  Opstellen SPMP

Design fase:

- *Doel:* Ontwerpen van structuur van het programma
- *Subtaken:*

  UML design
  Opstellen SRD en SDD

Iteratie I

- *Doel:* Kunnen inloggen op de site en de data die in de database opgeslaan is op de juiste plaatsen kunnen weergeven, lessenroosters kunnen genereren die aan fixed constraints voldoen

- *Subtaken:*

  Klassenstructuur bouwen om nodige eigenschappen weer te geven
  Algoritme maken dat simpel lessenrooster genereert
  Elementen in de database kunnen opslaan en lezen.
  Website maken
  Servlets schrijven die requests van gebruikers verwerken
  Servlets draaien op de server
  Kalender maken om lessenrooster weer te geven

Iteratie II

- *Doel:* Einddoelstellingen halen
- *Subtaken:*

  Te bepalen

Terminatie fase:

- *Doel:* Eindproduct afleveren en voortellen
- *Subtaken:*

  Presentatie maken
  Documentatie afwerken

De deadlines die een onderdeel zijn van de iteratie zullen steeds twee weken op voorhand gepland worden.

1.2 Evolution of the plan

Om het document up-to-date te houden met het project zal er een volledige revisie plaatsvinden na elke SCRUM meeting en op het einde van elke iteratie. Zo'n revisie houdt in dat het document door een teamlid volledig wordt doorgenomen en dat waar nodig stukken aangepast worden of toegevoegd worden.

2 Verwijzingen

Alle documenten kunnen terug gevonden worden op onze website.

- IEEE Standard for Software Project Management Plans; IEEE Std 1058-1998; 8 December 1998; IEEE-SA Standards Board ;

- IEEE Standard for Software Configuration Management Plans; IEEE Std 828-2005; 12 Augustus 2005; IEEE Computer Society ;

- IEEE Standard for Information Technology -Systems Design- Software Design Descriptions; IEEE Ste 1016-2009; 20 Juli 2009; IEEE Computer Society;

- IEEE Recommended Practice for Software Requirements Specifications; IEEE Std 830-1998; 20 Oktober 1998; IEEE Computer Society;

- WE-DINF-6537a Software Engineering Organization of the project; 2010-2011; Vakgroep Computerwetenschappen VUB;

3 Organisatie van het Project

3.1 Interne structuur

Het software development team bestaat uit vier leden. Hierdoor zullen de verschillende onderdelen van het project op individuele basis of in subteams van twee personen plaatsvinden.

Tijdens meetings waarop alle leden aanwezig zijn, worden personen toegewezen aan nieuwe opdrachten, of wordt verslag uitgebracht over een lopende opdracht. Communicatie binnenin een subgroep gebeurt naar believen bijvoorbeeld tijdens een onderlinge meeting of via e-mail.

Om globale controle, management en beheer van alle code en documenten te kunnen waarborgen, wordt van alle personen of groepen vereist dat ze hun ingeleverde of geüpdate code en documenten op een gemeenschappelijke plaats beschikbaar stellen. Hiervoor wordt google-code (http://code.google.com/) gebruikt en svn tortoise (http://tortoisesvn.tigris.org/).

3.2 Rollen en verantwoordelijkheden

Elk teamlid waakt over de kwaliteit en het naleven van deadlines voor de bevoegdheid waar zij eindverantwoordelijkheid over hebben. Verder wordt van ieder lid ook kennis over een specifiek onderwerp verwacht, hetzij door voorkennis, hetzij door training.

Teamlid	verantwoordelijkheid
Matthias	Team Leader, Algoritme specialist
Adam	Webmaster, Document manager
Alexander	Configuration manager, UML specialist, Server specialist
Zjef	Code implemetation leader, Database specialist

Deze verdeling zal gerbuikt worden om de verschillende taken aan teamleden toe te kennen. In de Initialisatie fase, Design fase en Terminatie fase wordt door heel het team ongeveer dezelfde taak uitgevoerd. Het is enkel bij de beide implementatie fases (iteratie 1 en 2) dat de taakverdeling zal doorgevoerd worden:

Iteratie I

- *Doel:* Kunnen inloggen op de site en de data die in de database opgeslaan is op de juiste plaatsen kunnen weergeven, lessenroosters kunnen genereren die aan fixed constraints voldoen
- *Subtaken:*

  Klassenstructuur bouwen om nodige eigenschappen weer te geven
  Omdat de klassenstructuur in eerste instantie gebruikt wordt door het algoritme om het lessenrooster op te stellen zal Matthias deze taak uitvoeren.

Algoritme maken dat simpel lessenrooster genereert
De algoritme specialist Matthias voert deze taak uit.

Elementen in de database kunnen opslaan en lezen.
Duidelijk een taak voor de database specialist Zjef.

Website maken
De webmaster Adam zal deze taak op zich nemen

Servlets schrijven die requests van gebruikers verwerken
Deze subtaak vereist een combinatie van server kennis en website kennis. Daarom zal hier aan gewerkt worden door Adam en Alexander.

Servlets draaien op de server
Deze taak vereist kennis van de server en wordt daarom aan Alexander gegeven

Kalender maken om lessenrooster weer te geven
Deze taak vereist zowel kennis van de database als van de servlets. daarom zal de door Zjef, Adam en Alexander uitgevoerd worden.

Iteratie II

- *Doel:* Einddoelstellingen halen
- *Subtaken:*

  Te bepalen

4 Bestuurlijke proces plannen

4.1 Start-up plan

4.1.1 Schattingsplan

Allereerst heeft men een overzicht gemaakt van de taken die moeten volbracht worden opdat het project slaagt. Aan de hand van deze informatie, heeft men dan ingeschat hoeveel tijd er zal moeten gespendeerd worden aan het project. Voorlopig is men uitgegaan van een weekindeling, waarbij er elke week een onderdeel van het project moet afgemaakt worden.

Initialisatie fase:

- *Doel:* Verkennen van de opdracht en de nodige kennis vergaren om de opdracht tot een goed einde te brengen.
- *Subtaken:*

  Opzoeken van informatie en software tools
  Opstellen SPMP
  Website maken en hosten om projectvooruitgang op te slaan
  Opzetten van subversion account voor synchronisatie van bestanden

- **tijdsschatting:** De deadline voor het SPMP op 22 februari bepaald het einde van deze fase, daarom werd er 1 week voor uitgetrokken.

Design fase:

- *Doel:* Ontwerpen van structuur van het programma
- *Subtaken:*

  UML design
  Opstellen SRD en SDD

- **tijdsschatting:** De deadline voor het maken van het SRS en het SDD liggen op 8 maart. Omdat in deze documenten het design van het programma vastgelegd wordt heeft het team dus 2 weken tijd om het design af te werken. Er werd geschat dat er maar 1 week nodig is om dit design te verwezenlijken. zodat er vroeger met het programmeren begonnen kon worden.

Iteratie I

- *Doel:* Kunnen inloggen op de site en de data die in de database opgeslaan is op de juiste plaatsen kunnen weergeven, lessenroosters kunnen genereren die aan fixed constraints voldoen
- *Subtaken:*

  Klassenstructuur bouwen om nodige eigenschappen weer te geven
  Algoritme maken dat simpel lessenrooster genereert
  Elementen in de database kunnen opslaan en lezen.

Website maken

Servlets schrijven die requests van gebruikers verwerken

Servlets draaien op de server

Kalender maken om lessenrooster weer te geven

- **tijdsschatting:** Het einde van iteratie 1 ligt ook vast. Door voor de twee vorige fases van het project elk een week te voorzien, zijn er 6 weken beschikbaar om de iteratie af te werken. Het Algoritme kan redelijk los van de rest van het programma ontwikkeld worden. De grootste bottleneck in het begin is het aanspreken van de database. Omdat het schrijven van de interface geen simpele materie is zullen na de eerste week dummy-methoden beschikbaar zijn die de rest van het team al kan gebruiken. De eigenlijke implementatie ervan houdt dan de rest van het team niet te veel tegen. Er wordt wel verwacht dat ze na de tweede week klaar is. De klassenstructuur opstellen krijgt ook een grote prioriteit, omdat de gegevens allemaal op de website moeten kunnen weergegeven worden. Daarom wordt verwacht dat deze ook al na de tweede week klaar is. De website layout krijgt ook prioriteit omdat de servlets deze layout moeten gebruiken. De basisfuncties die nodig zijn om de doelstellingen van iteratie 1 te halen kunnen op zich apart geimplementeerd worden, maar ze hebben de lay-out nodig om in de website geintegreerd te worden. Deze taak krijgt daarom ook een week de tijd. De mogelijkheid om de servlets op de server te kunnen draaien zijn niet prioritair omdat de werking van de server gesimuleerd kan worden op de computers van de teamleden. tegen het einde van iteratie 1 moet deze functionaliteit wel afgewerkt zijn.

Iteratie II

- *Doel:* Einddoelstellingen halen
- *Subtaken:*

  Te bepalen

- **tijdsschatting:**

Terminatie fase:

- *Doel:* Eindproduct afleveren en voortellen
- *Subtaken:*

  Presentatie maken

  Documentatie afwerken

- **tijdsschatting:**

De software en de middelen die men nu denkt nodig te hebben, zijn voorlopig beschikbaar. Er zullen dus geen bijkomende kosten zijn.

4.1.2 Personeelsplan

Gedurende het project zal er nood zijn aan kennis over Java, het ontwerpen van sites, databasestructuren en webcontainers (zoals Tomcat).

Het beschikbare team bestaat uit vier leden, die ervaring hebben met Java en het ontwerpen van sites. Het is de bedoeling dat ze zich gedurende de eerste twee weken van het project zullen bezighouden met het vergaren van kennis over de andere topics. Na deze fase van het project zullen de taken verder en in meer detail verdeeld worden over de teamleden.

Aangezien er maar 4 teamleden zijn, gaat men uit van egoless programming. De bedoeling is dat iedereen zich met alles een beetje bezighoudt, op die manier is er geen nood aan een hiërarchische structuur en kan men elkaar beter controleren.

4.1.3 Plan voor het bekomen van middelen

Er zal enkel gewerkt worden met open source software. Gedurende de eerste twee weken zullen de teamleden deze software verzamelen, zodat men na deze eerste fase zich geen zorgen meer hoeft te maken over het vergaren van software. Er werd beslist om alles in windows te ontwikkelen. de teamleden die geen windows computer hadden werden voorzien van het nodige materieel.

De programma's die gebruikt worden door het team zijn de volgende:

Voor de aanmaak van de documenten:

TeXnicCenter	1.0	Omgeving voor het aanmaken van de documenten in LaTeX
MikTeX	2.8	Programma voor het compileren van de LaTeX bestanden

De programmeeromgeving die gebruikt wordt is Eclipse. Er moeten de volgende packages en libraries aan toegevoegd worden:

Java EE Development Tools	3.2.2	
JST Server Adapters	3.2.2	
JST Server UI	3.2.2	
Ganava	1.0.1	Library die toelaat om HTML code te genereren in JAVA
MySQL		
Java JDK	1.6.0-24	
Apache Tomcat	7.0.8	Het framework dat op de server geinstalleerd wordt om de servlets te runnen

Om de verschillende mappen te synchroniseren met het web worden de volgende programma's gebruikt:

TortoiseSVN	1.6.12	Programma gebruikt om de bestanden met de groep te delen en te synchroniseren
FileZilla	3.3.5.1	FTP client om met de server te communiceren en om de website online te zetten
Putty		

Daarnaast zijn voor elk groepslid nog een browser nodig en een programma om pdf bestanden weer te geven.
Om de website te laten werken wordt gebruik gemaakt van open-source javascripts die gemaakt werden door derden:

Scriptnaam	versie	Beschrijving
Tabber		Javascript dat de tabbladen genereert op de website

4.1.4 Personeelstrainingsplan

Het is de bedoeling dat het team zelf aan de nodige informatie komt. Er wordt ook onderling overlegd zodat ze elkaar kunnen helpen bij het vergaren van bepaalde vaardigheden of kennis die vereist is.

De belangrijkste programmeertalen die elk teamlid zal moeten aanleren zijn de volgende:

. Adam: Java, Javascript, XHTML, CSS, PHP

. Alexander: Tomcat

. Matthias: Java, Tomcat, MySQL, Javascript

. Zjef: MySQL

4.2 Werkplan

4.2.1 Werkactiviteiten

Voorlopig heeft men hier nog geen goed beeld over kunnen vormen. Dit zal men aanpassen en bekijken na de eerste fase van het project. Dan beschikt men ook over het UML klassendiagramma die het overzicht van de work activities en hun onderlinge relatie zal verduidelijken.

Week 1	14/2	**Hoofdtaak:** Research
		Programmeertalen (Java) Sitetalen (XHTML, CSS, Javascript) Server (Tomcat), Database (MySQL)

Week 2	21/2	**Hoofdtaak:** Research
		bestaande structuren zoeken die we kunnen gebruiken. Opstellen algemene structuur programma in UML diagramma Deadlines maken voor aparte onderdelen
Week 3	28/2	**Hoofdtaak:** programmeren
		Deadlines tegen 29 februari ALGEMEEN: SRD en SDD afwerken
Week 4	7/3	**Hoofdtaak:** programmeren
		Deadlines tegen 8 maart ZJEF: Kalenderstructuur opmaken in JAVA. ADAM: Basislayout site afwerken ADAM en ALEXANDER: inlogscherm in HTML en servlet ALEXANDER: Tomcat op de server installeren en testen ALEXANDER: UML klassendiagramma aanpassen. MATTHIAS: Datastructuur aan JAVA zijde afwerken MATTHIAS: constraints oplijsten + vertalen naar JAVA
Week 5	14/3	**Hoofdtaak:** SCRUM meeting op 17/3, programmeren
		Deadlines tegen 17 maart ZJEF: Database interface afgewerkt, presentatie over database ADAM en ALEXANDER: Inloggen op site met server presentatie erover maken (beveiliging, servlets, tomcat,...) MATTHIAS: werking algoritme bepalen presentatie voorbereiden over algoritme en UML klassendiagramma ALGEMEEN: presentatie maken over toekomstige plannen, fouten,...
Week 6	21/3	**Hoofdtaak:** programmeren
Week 7	28/3	**Hoofdtaak:** programmeren
		Deadlines tegen 30 maart MATTHIAS: Manier om constraints weer te geven implementeren ALEXANDER en ADAM: HTMLBuilder integreren in bestaande servlet. gegevens van de database op de site kunnen weergeven. ZJEF: Mogelijkheid tot aanpassen van de kalender invoegen. Database interface grondig documenteren in SDD ADAM: SPMP nalezen en up to date brengen. ALEXANDER: SRS nalezen en up to date brengen. ZJEF: SDD nalezen en up to date brengen.

Week 8	4/4	**Hoofdtaak:** 1ste iteratie afwerken
		Deadlines tegen 1 april MATTHIAS: Lessenrooster opstellen met fixed constraints ADAM ALEXANDER en ZJEF: Opvragen van gegevens afgewerkt
Week 9	11/4	**Hoofdtaak:** paasvakantie: programmeren
		Deadlines tegen 1 april Te bepalen
Week 10	18/4	**Hoofdtaak:** paasvakantie: programmeren
Week 11	25/4	**Hoofdtaak:** SCRUM
Week 12	2/5	**Hoofdtaak:** programmeren
Week 13	9/5	**Hoofdtaak:** code afwerken
Week 14	16/5	**Hoofdtaak:** debuggen + presentaties maken
Week 15	23/5	**Hoofdtaak:** 2de iteratie

4.2.2 Plan voor het controleren van de planningen

Dit soort plan wordt niet nodig geacht, aangezien dat het team slechts uit 4 leden bestaat die elkaar elke werkdag ontmoeten.

4.3 Controle plannen

4.3.1 Plan voor controle van de eisen

Aangezien de SRS nog niet helemaal is afgerond, heeft men nog geen beeld van de eisen waaraan de software moet voldoen. Daarom kan men dus nog geen metrieken bepalen.

4.3.2 Plannigscontroleplan

Er zal ook elke week tijdens vergaderingen gekeken worden of men heeft volbracht wat gepland is. Indien dit niet het geval is, zal men de oorzaak hiervan onderzoeken en kijken of de planning voor de toekomst nog wel realistisch en of ook deze niet herbekeken moet worden. De data van de vergaderingen liggen nog niet vast, wel is het zeker dat er elke week minstens een vergadering is.

4.3.3 Kwaliteitscontroleplan

Men zal steeds kijken of de ontwikkelde code voldoet aan de door het team op voorhand vastgelegde vereisten deze zijn terug te vinden in de SRS. Indien dit niet het geval is, zal men dit toch proberen te bekomen. Aangezien iedereen elkaar controleert en evalueert, zal dit de kwaliteit ten goede komen.

4.3.4 Rapporteringsplan

Er zal elke vergadering door iedereen mondeling verslag worden uitgebracht; waarbij elk teamlid vertelt wat de doelstellingen waren en of deze al dan niet bereikt zijn. Bovendien zal er steeds de mogelijkheid zijn voor andere groepsleden om vragen te stellen.

5 Technische plannen

5.1 Proces model

Tijdens het project zal onderstaande planning worden opgevolgd. Concrete invulling van elke van de processtappen zal slechts gebeuren op korte termijn bij het begin van elke fase. Het verloop van de processtappen zal worden gedocumenteerd aan de hand van een logboek beschikbaar op de projectwebsite.

Initialisatie fase:

- In ontvangst name van de projectbeschrijving
- Groepsoverleg
- Opzoeken van informatie en software tools
- Opstellen SPMP

Design fase:

- Groepsoverleg
- UML design
- Opstellen SRD en SDD

Implementatie fase:

Iteratie I

- Doelstellingen formuleren
- Taakverdeling
- Implementatie
- SCRUM bijeenkomst I
- Implementatie
- Revisie

Iteratie II

- Doelstellingen formuleren
- Taakverdeling
- Implementatie
- SCRUM bijeenkomst II
- Revisie

Terminatie fase:

- Groepsoverleg
- Eindproduct afleveren
- Presentatie resultaten
- Einde project

5.2 Methode hulpmiddelen en technieken

Tijdens het project zal de Agile methodologie worden gevolgd. Dit houdt in dat concrete doelstelling (en de stappen nodig om deze te bereiken) steeds op korte termijn zullen worden gedefinieerd. Een snelle evaluatie van de verwezenlijkte resultaten is dus noodzakelijk. Dit wordt gegarandeerd door wekelijks overleg tussen de groepsleden en externe feedback afkomstig van de geplande scrum meetings. Het project zal worden uitgevoerd in de programmeertaal Java. Als werkomgeving voor het schrijven en documenteren van de code wordt gekozen voor Eclips. Voor meer informatie wordt doorverwezen naar http://www.java.com/ en http://www.eclipse.org/. Bij het documenteren en beschrijven van de code zal zoveel mogelijk beroep worden gedaan op Unified Modeling Language (UML). Voor meer informatie over deze standaard http://www.uml.org/. Voor het beheren van de project documenten en -bestanden wordt gebruikt gemaakt van een GoogleCode account en Subversion (SVN).

5.3 Productaanvaarding plan

Enkel werkende code zal worden afgeleverd aan het einde van het project mits goedkeuring van elk van de groepsleden. Verdere evaluatie van het project wordt volledig bepaal door de opdrachtgever, Prof. Ragnhild Verstraeten.

6 Ondersteunende processplannen

6.1 Configuration management plan

Voor het beheren van de projectdocumenten en -bestanden wordt gebruikt gemaakt van een GoogleCode account en Subversion (SVN). Alle groepsleden hebben volledige toegang tot deze account en dus ook tot alle project documenten en -bestanden. Elk groepslid zal verantwoordelijk worden gesteld voor het beheer van zijn bijdrage tot het project. Daarbij zal aan het einde van elke projectfase[1] een back-up worden gemaakt van alle bestanden om eventueel falen van de Google server het hoofd te kunnen bieden.

6.2 Verificatie en validatieplan

Wekelijks zal de groep samenkomen om de ontwikkelingen van het project te bespreken, elkaars werk te controleren en eventuele problemen op te lossen. Anderzijds zullen tijdens de scrum samenkomsten ook externen commentaar kunnen geven op het project verloop.

6.3 Documentatieplan

Tijdens het project zullen onderstaande documenten zeker worden afgeleverd.

- Software Project Management Plans (SPMP)
- Software Design Descriptions (SDD)
- Software Requirements Specifications (SRS)

Voor het opstellen van deze documenten wordt steeds gebruik gemaakt van de IEEE standaards vermeld in de referenties van dit document. Vervolgens zal ook het projectverloop worden gedocumenteerd onder vorm van een logboek dat beschikbaar zal zijn op de projectwebsite en later zal worden toegevoegd aan dit document onder de vorm van een bijlage. Elke groepslid is verantwoordelijk voor het documenteren van zijn bijdragen tot het project. Elk onderdeel van de documentatie zal door alle groepsleden worden nagelezen en gecontroleerd.

[1] Zie 5.1 Procesmodel voor de indeling van de fases.

7 Bijlagen

7.1 Oorspronkelijke opdracht

abstract
The program MyCourses provides as optimal as possible a plan for scheduling courses. Every university is faced each year with the same problem : How to schedule a large number of courses in an optimal way while fulfilling a number of constraints, such as available lecture rooms, limited availability of lecturers, students' selections of the courses, and similar. MyCourses should be implemented as an interactive program that (i) enables entering data, such as courses, the faculty members, the available facilities and some constraints related to the course scheduling, (ii) calculates and proposes a scheduling for courses, (iii) makes it possible to manually update the proposed schedule, but keeping track of the consistent scheduling, and (iv) provides a presentation of scheduled courses.

introduction
Course scheduling is a tedious and error-prone task when done manually or semi-manually. For this reason a program that can automatically produce course scheduling with given requirements and constraints is very important. The goal of this project is to develop a course scheduler, MyCourses.
MyCourses will make it possible to enter data and requirements in a simple way using a webbased interface, calculate and propose a schedule, enable manual updates, and finally present the schedule for the selected courses. Since different people (students, lecturers, program planners, etc.) will use the program its user-friendliness is crucial. An effcient automatic scheduling is also important, but even more important is a possibility to manually re-schedule or preschedule some courses or course elements (like lectures, labs, etc.).
The project includes requirements solicitation, requirements specification, design and the implementation. The program should be implemented as a distributed web-based, application, and data should be stored in a database. Since the program is aimed for universities, it is expected that FLOSS (Free/Libre and Open Source Software) will be used.

Functional Requirements
MyCourses is described by several scenarios (taken from the assignment at http://score-contest.org/2011/Projects.php)

1. Entering programs and courses
   A program administrator who is responsible for management of the programs at the university defines programs. She identifies the program, its running period (starting and ending year),

and a program manager who will have the overall responsibility for the program. Typically when defining a new program, the same program from the previous year would be copied, with some data changed afterwards. A program manager, when enabled, can enter all details about the program : Which courses it includes, which of these courses are obligatory and which optional, etc. The courses may already exist, and in that case she creates a new course instance with a given period of its execution, who is the main lecturer ("examiner"), and some additional general information about the course. If the course is new, then the program manager creates it first, and then creates a new instance of it.
The main lecturer defines the details about the course instance he is assigned for: Which are other people from the teaching staff involved in the course, which are the course elements (lectures, tutorials, labs, projects...), the way of possible course execution (a number of lectures and other elements per week, preferred days, expected number of students, and similar). He may wish to (smart) copy all data from a previous instance of the course.

2. Entering resources
   An administrator, or a program administrator enters data about different resources: The available lecture rooms and laboratories in which the courses (or particular elements) will take place, and some other elements such as data about number of available places, or availability of the room is entered.

3. Scheduling
   Several users can run a (semi)automatic scheduling process that provides a schedule proposal for a course or a set of courses (e.g., the entire program or selected courses): the days and time, and places should be scheduled. The scheduler is not necessarily an automatic solver but it allows some manual predefinition of the schedule, and manual changes after the proposal is created. The main lecturer can run the scheduler for his course, and mark it as a course schedule proposal. The scheduler shows if some conflicts occur. The program manager can verify the scheduling in combination with other courses in the program. The program manager can modify the scheduling if necessary and then freeze it (i.e. make it official).

4. Presentation
   Different users can use the program to present data. Examples of presentations: Availability and utilization of the facilities ; Schedule of a particular course; Schedule for a faculty member; A schedule for a student (after she defines in which courses is she enrolled), and similar.