

Software project management plan voor Schedule-Generator

Matthias Caenepeel Adam Cooman Alexander De Cock
Zjef Van de Poel

17 februari 2011 Versie 1.0

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: Verberingen doorgevoerd en titels vertaald

Inhoudsopgave

1	Overzicht	4
1.1	Samenvatting van het project	4
1.1.1	Doel, bedoeling en objectieven	4
1.1.2	Veronderstellingen en beperkingen	4
1.1.3	Deliverables van het Project	4
1.1.4	Planning en Budget	6
2	Verwijzingen	7
3	Organisatie van het Project	8
3.1	Interne structuur	8
3.2	Rollen en verantwoordelijkheden	8
4	Bestuurlijke proces plannen	9
4.1	Start-up plan	9
4.1.1	Schattingsplan	9
4.1.2	Personeelsplan	9
4.1.3	Plan voor het bekomen van middelen	9
4.1.4	Personeelstrainingsplan	9
4.2	Werkplan	10
4.2.1	Werkactiviteiten	10
4.2.2	Plan voor het controleren van de planningen	10
4.3	Controle plannen	10
4.3.1	Plan voor controle van de eisen	10
4.3.2	Planningscontroleplan	10
4.3.3	Kwaliteitscontroleplan	10
4.3.4	Rapporteringsplan	11
5	Technische plannen	12
5.1	Proces model	12
5.2	Methode hulpmiddelen en technieken	13
5.3	Productaanvaarding plan	13
6	Ondersteunende processplannen	14
6.1	Configuration management plan	14
6.2	Verificatie en validatieplan	14
6.3	Documentatieplan	14
6.4	Oorspronkelijke opdracht	14

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-georiënteerde taal en het programma moet op Wilma kunnen draaien.

Aangezien de aard van het project, werd er beslist dat de database met gebruikers van de software compatibel moet zijn met de bestaande database van de VUB.

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. De link ernaar is te vinden op onze website. 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

Nr.	Datum	Activiteit
1	14/2	Research: Programmeertalen (Java) Sitetalen (XHTML, CSS, Javascript) Server (Tomcat), Database (MySQL)
2	21/2	Research: Hetzelfde bestaande structuren zoeken die we kunnen gebruiken. Opstellen algemene structuur programma in UML diagramma Deadlines maken voor aparte onderdelen
3	28/2	programmeren, SRD en SDD afwerken
4	7/3	programmeren
5	14/3	SCRUM: UML diagramma met algemene structuur plan voor eerste iteratie
6	21/3	programmeren
7	28/3	programmeren
8	4/4	eerste iteratie af: Databases werken. Elementaire interface voor database
9	11/4	paasvakantie: programmeren
10	18/4	paasvakantie: programmeren
11	25/4	SCRUM
12	2/5	programmeren
13	9/5	programmeren, code af
14	16/5	grondig debuggen + presentaties voorbereiden
15	23/5	tweede iteratie af: volledig project werkt

Na het opstellen van de structuur van het project in een grondig UML schema zal de planning verfijnd worden en zullen specifieke deadlines opgesteld worden voor verschillende onderdelen van het programma.

2 Verwijzingen

- IEEE Standard for Software Project Management Plans; IEEE Std 1058-1998; 8 December 1998; IEEE-SA Standards Board ; Te vinden op pointcarre
- IEEE Standard for Software Configuration Management Plans; IEEE Std 828-2005; 12 Augustus 2005; IEEE Computer Society ; Te vinden op pointcarre
- IEEE Standard for Information Technology -Systems Design- Software Design Descriptions; IEEE Ste 1016-2009; 20 Juli 2009; IEEE Computer Society; Te vinden op pointcarre
- IEEE Recommended Proctice for Software Requirements Specifications; IEEE Std 830-1998; 20 Oktober 1998; IEEE Computer Society; Te vinden op pointcarre
- WE-DINF-6537a Software Engineering Organisation of the project; 2010-2011; Vakgroep Computerwetenschappen VUB; Te vinden op pointcarre

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.

Adam: Webmaster, document manager

Alexander: Configuration manager, UML specialist, Server interspace specialist

Matthias: Team leader, Algoritme specialist

Zjef: Code implemetation leader (controleert code en houdt alles bij elkaar), Database specialist

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 (zie ook 1.1.4. Scheduling and Budget Summary), waarbij er elke week een onderdeel van het project moet afgemaakt worden.

Op het einde van week 2, zal er een UML klassendiagramma van het volledig project beschikbaar zijn. Aan de hand hiervan zal de voorlopige planning herherbekeken, meer uitgediept en indien nodig aangepast worden.

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.

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.

4.2.2 Plan voor het controleren van de planningen

Voorlopig is er nog maar een zeer algemeen tijdsschema opgesteld (zie ook 4.1.1. Estimation Plan en 4.1.4. Scheduling and Budget Summary). Naarmate het project vordert zal men dit ook verder uitdiepen. Men heeft wel al rekening gehouden met verlofdagen en andere projecten die ook moeten volbracht worden in dezelfde periode als externe factoren.

4.3 Controle plannen

4.3.1 Plan voor controle van de eisen

Om een goed beeld te krijgen van de vorderingen van het project en de kwaliteit van het geleverde werk, zullen de teamleden elke week vergaderen. Iedereen vertelt wat hij die week bereikt heeft en welke problemen hij daarbij mogelijk gehad heeft. Het is ook de bedoeling dat iedereen een beeld heeft van elk onderdeel van het project; dit lijkt haalbaar omdat het team slecht uit vier leden bestaat (egoless programming). Een extra voordeel is hierbij dat iedereen controle heeft over wat de anderen doen en zo kunnen fouten vermeden of opgespoord worden.

4.3.2 Planningscontroleplan

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.

4.3.3 Kwaliteitscontroleplan

Men zal steeds kijken of de ontwikkelde code voldoet aan de door het team op voorhand vastgelegde vereisten. 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 wekelijks 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 Eclipse. 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 projectdocumenten en -bestanden wordt gebruikt gemaakt van een GoogleCode account en Subversion (SVN).

5.3 Productaanvaarding plan

Enkel werkende en foutloze code zal worden afgeleverd aan het einde van het project mits goedkeuring van elk van de groepsleden. Verdere evaluatie van het project wordt volledig bepaald 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 projectdocumenten 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 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 ondervorm 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.

6.4 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 efficient automatic scheduling is also important, but even more important is a possibility to manually re-schedule or pre-schedule 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.