Software project management plan voor Schedule-Generator

Matthias Caenepeel Adam Cooman Alexander De Cock
Zjef Van de Poel

28 maart 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

To do

. Lijst met javascripts uitbreiden

. Subtaken en deadlines voor iteratie 2 verwoorden

. Process Improvement plan uitdenken en toevoegen

. Configuration management plan grondiger schrijven en uitdenken.

. Plan voor controle van de eisen

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 ze 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 lay-out 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 lay-out* 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.

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

In de eerste week wordt de initialisatie fase afgewerkt. De belangrijkste deadline hierbij is de afwerking van de eerste versie van het SPMP. Daarnaast kan er ook al met research begonnen worden.

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

De tweede fase krijgt volgens de tijdsschatting ook een week. Hierin wordt meer research gedaan en worden het SDD en SRS geschreven. Het UML diagramma dat gemaakt wordt zal een duidelijke structuur weergeven van de structuur van het programma. Daaruit kunnen de functionaliteiten van de website opgelijst worden en kunnen de grote lijnen voor de werlmethode voor iteratie 1 en 2 bepaald worden.

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

In week 3 wordt er begonnen met het programmeren. De deadline voor het SRS en SDD moet gemakkelijk gehaald worden omdat beide documenten al geschreven zijn in de vorige week. Op de wekelijkse vergadering van week 3 (zie vergadering van 1/3) werden de deadlines voor de volgende week vastgelegd.

Week 3	28/2	**Hoofdtaak:** programmeren
		Deadlines tegen 29 februari ALGEMEEN: SRS 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 brengt de eerste SCRUM meeting. Er wordt een presentatie voorbereidt terwijl ondertussen voortgewerkt wordt aan de verschillende onderdelen van het programma.

Week 5	14/3	**Hoofdtaak:** SCRUM meeting op 16/3, programmeren
		Deadlines tegen 16 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,...

Tijdens de twee weken na de SCRUM worden de documenten herbekeken. Het SPMP, SRS en SDD worden up-to-date gebracht om een goed overzicht te krijgen van de status van het programma. Nu het inloggen van gebruikers op de site werkt wordt overgeschakeld op het weergeven van de gegevens uit de database zoals de kalender en lijsten van lessen.

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.

De eerste iteratie moet afgewerkt zijn tegen het einde van week 8.

Week 8	4/4	**Hoofdtaak:** 1ste iteratie afwerken
		Deadlines tegen 8 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 op 27/4

Week 12	2/5	**Hoofdtaak:** programmeren

Week 13	9/5	**Hoofdtaak:** code afwerken

Week 14	16/5	**Hoofdtaak:** afwerken iteratie 2 + presentaties maken **De tweede iteratie moet af zijn op 20/5**

Week 15	23/5	**Hoofdtaak:** Presentatie geven op 25/5

4.3 Controle plannen

4.3.1 Plan voor controle van de eisen

TODO

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. De minutes van de vergaderingen worden bijgehouden en op de website van het project bijgehouden. Ze worden ook opgelijst in bijlage van dit SPMP.

4.3.3 Kwaliteitscontroleplan

Om de kwaliteit van het product te garanderen werd door de opdrachtgever voorgesteld gebruik te maken van unit test. Dit zou inhouden dat voor elke methode een specifieke test ontwikkeld moet worden. Omdat het projectteam slechts bestaat uit vier groepsleden en het ontwikkelen van unit tests een taak op zich is, werd er besloten om van deze aanpak af te wijken.

Het team stelt voor om elke klasse te onderwerpen aan een gentegreerde tests en pas gebruik te maken van unit tests indien blijkt dat de code faalt op deze gentegreerde test. Daarbij zullen de test zoveel mogelijk worden geautomatiseerd, wat inhoudt dat de test zonder menselijke toezicht kan worden uitgevoerd. De ontwikkeld test zullen worden afgeleverd aan de opdrachtgever als kwaliteitsgarantie in de loop van het project.
Voor het de concrete uitvoering van de tests zal, omdat het merendeel van de code zal worden geschreven in java, gebruik worden gemaakt van JUnit4.92b Dit is een open source testomgeving die toelaat geautomatiseerd tests te schrijven en is standaard reeds aanwezig is in Eclipse. Voor meer informatie over JUnit wordt verwezen naar http://www.junit.org/.
Met de voorop gestelde aanpak wordt de tijd nodig voor het testen aanzienlijk gereduceerd, doordat initieel werkende code vrijgesteld wordt van unit tests, maar wordt de garantie op correctheid natuurlijk ook afgezwakt. Het team gaat er echter van uit dat, rekening houdend met het voorzien aantal manuren, de vooropgestelde werkwijze de beste compromis vormt tussen kwaliteit en kost.

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. De minutes van de vergaderingen worden door de Document manager opgesteld en na de vergadering online geplaatst op de website van het project. Hierdoor kunnen de opdrachtgevers nauwgezet opvolgen waar het team mee bezig is. Naast de vergaderingen in teamverband zijn er twee SCRUM meetings opgelegd door de opdrachtgever. Hierin wordt een presentatie gegeven over de gerealiseerde taken (mogelijk met een demonstratie), over de problemen die tegengekomen zijn en over plannen voor de komende weken.

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
- Implementatie
- 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 hoofdzakelijk worden uitgevoerd in de programmeertaal Java. Dit is een veelgebruikt objectgeorinteerde taal ontwikkeld door Sun. Daarin boven is zijn platform onafhankelijk te gebruiken wat een extra voordeel is, aangezien het project zal ontwikkeld worden op Windows terwijl het uiteindelijk bedoeld is voor een Linux server. Door haar grote populariteit beschikt deze taal ook over een groot aantal vrij te gebruiken bibliotheken,omgevingen en APIs.

De website wordt geprogrammeerd in een aantal verschillende talen, zoals de gewoonte is bij websites. De inhoud van de website wordt in HTML 4.0 geschreven. De Layout van het HTML bestand wordt in CSS geschreven en om de beperkingen van HTML op te vangen wordt Javascript gebruikt. Daarnaast wordt ook PHP gebruikt in het weergeven van de kalender.

Voor het aanspreken van de database wordt MySQL gebruikt.

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). 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
VisualParadigma for UML Community Edition	8.0	Programma voor het aanmaken van de UML diagramma's

Om de de teamleden de mogelijkheid te geven om de servlets op hun eigen computers te testen werd Tomcat erop geinstalleerd. Daarvoor is ook de JDK van Java nodig.

Java JDK	1.6.0-24	
Apache Tomcat	7.0.8	Het framework dat op de server geinstalleerd wordt om de servlets te runnen
MySQL Workbench	5.2	Visuele interface voor de MySQL database

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/. De volgende packages en libraries zijn nodig:

Java EE Development Tools	3.2.2	
JST Server Adapters	3.2.2	Laat Eclipse toe om met de geinstalleerde Tomcat server te communiceren
JST Server UI	3.2.2	Extra onderdeel van de Server Adapter
Ganava	1.0.1	Library die toelaat om HTML code te genereren in JAVA
Connector/J	5.1.15	MySQL library
iCal4j	1.0	Library om ICS files aan te maken voor de kalender

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	0.6	Linux terminal voor Windows. gebruikt om met de server te communiceren

Om de website te laten werken wordt gebruik gemaakt van open-source javascripts die gemaakt werden door derden:

Scriptnaam	versie	Beschrijving
Tabber	1.9	Javascript dat de tabbladen genereert op de website
jQuery	1.3.2	JavaScript Library die functies als tabellen aanpassen, zoeken in lijsten en aanmaken van popups toelaat
jQuery.datatables	1.7.6	Javascript voor het aanmaken van aanpasbare tabellen
PHPiCalendar	2.4	PHP script voor de weergaven van de ICS files
slimpicker		Script voor het on-line aanpassen van de kalender

Daarnaast zijn voor elk groepslid nog een browser nodig, een programma om pdf bestanden weer te geven en een zeer simpele text editor zoals notepad.

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 minutes van de teamvergaderingen die beschikbaar zal zijn op de projectwebsite en later zullen worden toegevoegd aan dit document onder de vorm van bijlagen. 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. Bij het documenteren en beschrijven van de code zal zoveel mogelijk gebruik worden gemaakt van de Unified Modeling Language (UML2.0). Voor het opstellen van deze diagrammen word gebruik gemaakt van VisualParadigma for UML 8.0 Community Edition. De tekstdocumenten zoals het SPMP, SDD en SRS zullen worden opgesteld met Latex en afgelverd in pdf formaat.

[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.

7.2 Minutes van de vergaderingen

7.2.1 Vergadering van 22-2-2010

Punten op de agenda:

- Maken UML diagramma
- Bespreken van structuur van programma

Resultaten:

- Beslist om te werken met Eclipse
- Tekenen van structuur die weergeeft welke verschillende variabelen in ons programma aanwezig gaan zijn en in welke structuur ze gelinkt zullen worden.

Eerste deel van de vergadering: 10:15 tot 12:00

Er werd beslist om Eclipse te gebruiken om het diagramma op te stellen
Voordelen: Iedereen heeft het en we werken met minder verschillende programma's
Nadelen: De code kan niet automatisch gegenereerd worden.

Er werd begonnen met het tekenen van het UML diagramma.

Gebruikers van het systeem werden bepaald: Students, Professors, Admins en Guest.
Students en Professors zijn humans met firstName en surName. die naam kan door Guests ingetypt worden om het rooster van andere mensen te kunnen bekijken (zodat de guest niet de accountnaam moet kennen van die persoon.
Omdat de studenten rolnummers hebben en professoren personneelsnummers werden die variabelen niet op dat niveau toegevoegd.
Om de vakken onder te verdelen werd gebruik gemaakt van faclity, program en course. Een faclity bevat een lijst van programs en een program bevat een lijst van courses. Een course bevat een professor
De klasse student bevat dan een lijst programs en een lijst courses om flexibele keuzevakken toe te laten.
Voor de lokalen werd beslist om ze ook aan een gebouw te linken, zodat bij de verdeling van de lokalen rekening gehouden kan worden dat een bepaalde groep best in een bepaald gebouw les krijgt (ingenieurs in gebouw K,...)

Toen probeerden we de aanpassingsrechten voor de verschillende gebruikers toe te voegen aan deze structuur werd gediscussieerd wat de algemene aanpak zou zijn, vooral waar welke gegevens opgeslaan werden en in welke vorm. Als we voor alles een database gebruikten was het schrijven van het klassendiagramma

niet meer zo belangrijk omdat er dan twee totaal verschillendende programma's werken voor het lezen en voor het aanpassen van de databases.
Door tekort aan kennis van MySQL en PHP werd deze discussie op de lange baan geschoven. Omdat het UML diagramma zelf in dat geval een goeie structuur van de databases weergaf werd verder gewerkt.

Tweede deel van de vergadering: 15:15 tot 17:15

- De structuur Course werd uitgebreidt. Er werd een record met informatie aan toegevoegd die nu op opaweb staat. de hoofdprofessor van het vak kan die aanpassen. Het vak werd onderverdeeld in verschillende subCourses. Om theorie en oefeningen op verschillende locaties toe te laten. Een subcourse bevat dan zijn type (WPO, THEORIE, LABO) en het aantal uur dat het onderdeel duurt. Het subCourse krijgt een aparte professor (assistent), een lijst met hardware die nodig is om het vak te geven (beamer, bord,...) en het aantal uur dat dit deel van het vak gegeven wordt. Aan de klasse Room werd dan ook de beschikbare hardware toegevoegd, alsook het aantal studenten dat er kan les krijgen en het gebouw.

- De constraints die kunnen opgegeven worden door de professoren en secretaresses in verband met hun vakken werden opgesomd en genoteerd.

7.2.2 Vergadering van 23-2-2010

Punten op de agenda:

- Opzoeken SRS
- Beginnen SRS
- Taakverdeling voor afwerking SRS
- Opzoeken SDD
- Beginnen SDD
- Taakverdeling voor afwerking SDD

Resultaten:

- SRS begonnen
- SDD begonnen

Deel 1 van de vergadering: 9:00 tot 11:30

De IEEE standaard over SRS documenten werd nagelezen en besproken.
Er werd een LATEX document aangemaakt met de structuur van het SRS erin en de inhoud van de verschillende punten werd besproken en kort samenvat genoteerd. Er werd gekozen om de functionaliteiten in te delen volgens gebruiker en er werd een mooi overzicht gemaakt van de functionaliteiten die we moeten implementeren. Daarna werd opgemerkt dat een flexibelere structuur om gebruikers rechten te geven gewenst was. Er zal per functionaliteit beslist worden of een gebruiker ze heeft of niet, daarom moet de structuur niet meer per gebruiker ingedeeld worden, maar per functionaliteit.
de bestaande structuur bevat wel richtwaarden en verschillende gebruikersklassen die het meest voorkomen. Daarom zullen die structuren in een 'template' opgeslaan zijn zodat sneller machtigingen kunnen gegeven worden, zonder de flexibiliteit van het algemene systeem te verliezen.

Opmerking over Veiligheid van de site
Om verschillende accounts verschillende functionaliteiten te geven (en om te controleren of ze gemachtigd zijn om te doen wat ze naar de server sturen) werd een systeem uitgewerk met een code.
Als de gebruiker zich aanmeldt, wordt een code gegenereerd voor die gebruiker (random nummer) die lokaal bijgehouden wordt door de gebruiker.
Bij elk commando van de gebruiker wordt de code meegegeven.
Op de server wordt een tijdelijke lijst bijgehouden die de code aan accounts verbindt. (en inlogtijd,... om ze na een tijd weg te kunnen smijten als de gebruiker niet letterlijk uitlogt).
Bij elk commando van de gebruiker wordt de code in de lijst opgezocht, de

rechten van de bijhorende gebruiker gecontroleerd en al dan niet kan het commando uitgevoerd worden.

Deel 2 van de vergadering: 15:00 tot 17:00

De IEEE standaard over SDD documenten werd nagelezen en besproken.
Er werd een LATEX document aangemaakt met de structuur van het SDD erin en de inhoud van de verschillende punten werd besproken en kort samengevat genoteerd. De verschillende taken paragrafen werden onder de leden van de groep verdeeld, zodat iedereen de nodige research kan doen thuis en de langere volledige tekst schrijven.

> De taakverdeling en deadline voor het afwerken van het SRS en het SDD werd vastgelegd:
>
> De tekst van het SRS moet af zijn en naar Adam doorgestuurd worden donderdagavond 24/2
>
> De tekst van het SDD moet af zijn en naar Adam doorgestuurd worden zondagavond 27/2

De documenten zullen dan afgewerkt worden (layout + nalezen) zodat ze tegen dinsdag 1/3 online staan om de deadline van de opdrachtgever te halen.

7.2.3 Vergadering van 24-2-2011

Punten op de agenda:

- Servlets laten werken op Eclipse
- Algemene afspraken voor Eclipse

Resultaten:

- Project voor ons programma aangemaakt
- Nodige packages in eclipse bepaald

Deel 1 van de vergadering: 12:30 tot 16:00

Om inzicht te verwerven in het gebruik van Tomcat en servlets werd geprobeerd om een servlet uit te voeren. We kozen een voorbeeldservlet van het internet om uit te voeren.
Na lang onderzoek werd bepaald dat de volgende Eclipse uitbreidingen nodig zijn en werd een servlet uitgevoerd.
Ieder lid van het team moet de volgende uitbreidingen van Eclipse hebben:

Eclipse Java EE Development Tools

JST Server Adapters

JST Server UI

Daarnaast zijn nog andere programma's nodig:

Java JDK

Apache Tomcat

Er werd beslist om de Code Style van Zjef te gebruiken. Het is een beter versie van de standaard Eclipse Code Style.
Het invoeren in Eclipse gaat als volgt: Window - Preferences - Java - Code Style - Formatter - Import - ZVdP.xml
Alexander begon dan met het aanmaken van de mappenstructuur van ons project en plaatste ze in de subversion map.
De e-mail met feedback van de opdrachtgever werd ontvangen en Matthias voerde de verbeteringen door in het SPMP door.
Zjef Deed research naar JUnit testing

7.2.4 Vergadering van 1-3-2011

Punten op de agenda:

- XML file interface en Database interface uitleggen
- Doelstellingen eerste iteratie bespreken
- Taakverdeling eerste iteratie bespreken
- Deadlines opstellen
- JUnit bespreken

Resultaten:

- Inhoud voor de komende taken voor iteratie 1 opgesteld
- Deadlines voor de komende taken voor iteratie 1 opgesteld

Deel 1 van de vergadering: 14:00 tot 16:20

Zjef begon met de demonstratie van zijn interface voor het communiceren met de xml bestanden en voor het communiceren met de database. De beschrijving van die interfaces kan gevonden worden in het SDD.
Matthias Had tijdens het weekend de klassestructuur opgesteld die in het SDD beschreven wordt onder 'Logical'. Deze werd overlopen en aangepast waar nodig. Omdat matthias met MAC werkt en het veel problemen gaf om de juiste software werkende te krijgen op MAC (vooral svn) werd een windows laptop gezocht voor hem. Deze werd hem officieel overhandigd, geformatteerd en geinstalleerd nadat een klein probleem met de voeding gerepareerd was.

De lijst met noodzakelijke functionaliteiten werd opgesteld om te realiseren voor de eerste iteratie. Ze werden uit het SRS gehaald.

Volledig identificatie (3.2.4 van het SRS)

Aanmelden, aanmelden als gast, afmelden

Opvragen van gegevens (3.2.5 van het SRS)

Opvragen faculteit, programma, vak, student, docent

lessenrooster op programma, student, docent

Beheren van vakken (3.2.6)

Aanmaken, verwijderen, wijzigen als beheerder, koppelen (aan student, aan docent), onderverdelen in programma's

Beheren van programma (3.2.7 van SRS)

Aanmaken, verwijderen, wijzigen, koppelen als beheerder

Beheren van accounts (3.2.10 van SRS)

Aanmaken, verwijderen, wijzigen

Overige

Aanmaken van software databasestructuur via SQL

Daarnaast werden de mogelijke fucntionaliteiten in verband met het lessenrooster ook opgenomen in de lijst omdat daarmee de andere functionaliteiten getest kunnen worden. We moeten dus alle functionaliteiten implementeren die toelaten om handmatig een lessenrooster op te stellen.

Beheren van beperkingen (3.2.11 van SRS)

Tijdsbeperkingen aanmaken, verwijderen wijzigen

Beheren van lessenroosters

Van al deze noodzakelijke functionaliteiten werden de volgende vooropgesteld voor iteratie 1:

Volledig identificatie (3.2.4 van het SRS)

Aanmelden, aanmelden als gast, afmelden

Overige

Aanmaken van software databasestructuur via SQL

Opvragen van gegevens (3.2.5 van het SRS)

Opvragen faculteit, programma, vak, student, docent

lessenrooster op programma, student, docent

De andere functionaliteiten zijn nodig om deze punten te realiseren, maar de aanpassingen zullen op de server zelf gebeuren in de servlet, niet via de site. Om de lessenrooster editor te maken is er keuze tussen een JAVA applet of javascript. Er werd beslist om bestaande oplossingen te zoeken en dan deze keuze te maken.
De volgende deadlines werden vastgelegd (allemaal nog in 2011):

Tegen 2 maart: ZJEF: Schatting voor tijd die nodig is voor het afwerken van de database interface

Tegen 8 maart: ALEX: Tomcat op de server installeren en testen. Account klasse definieren. UML klassendiagramma aanpassen.

Tegen 8 maart: ADAM: Basislayout site afwerken (tabellen, locaties,... geen grafische elementen)

Tegen 8 maart: ADAM en ALEX: inlogscherm in HTML en servlet laten overeen komen

Tegen 8 maart: ZJEF: Kalenderstructuur opmaken in JAVA.

Tegen 8 maart: ZJEF: JUnit tutorial uitleggen aan de rest van het team en regels opstellen waaraan de testmethodes moeten voldoen

Tegen 8 maart: MATTHIAS: Datastructuur aan JAVA zijde afwerken (zie design viewpoint 2: Logical van het SDD) en constraints oplijsten + vertalen naar JAVA

Tegen 17 maart: SCRUM MATTHIAS: werking algoritme bepalen en presentatie voorbereiden over UML klassendiagramma (SDD viewpoint 2)

Tegen 17 maart: SCRUM ADAM en ALEXANDER: Inloggen op site met echte connectie met server + presentatie erover (beveiliging, servlets, tomcat,... (SDD Viewpoint 3))

Tegen 17 maart: SCRUM ZJEF: Database interface afgewerkt en geeft presentatie over database

Tegen 17 maart: SCRUM presentatie maken over toekomstige plannen, fouten, al de rest voor de SCRUM

Tegen 1 april: ITER1 MATTHIAS: Lessenrooster kunnen opstellen dat aan de fixed constraints voldoet.

Tegen 1 april: ITER1 ADAM en ALEX en ZJEF: Inloggen en Opvragen van gegevens afgewerkt (SRS 3.2.5)

Adam zal zich bezig houden met het maken van het Gant diagramma van deze deadlines om overzicht te krijgen en te voldoen aan de wensen van de heilige opdrachtgever.

7.2.5 Vergadering van 8-3-2011

Punten op de agenda:

- Bespreking deadlines en demonstratie van code voor de deadline
- Deadlines voor volgende week SCRUM herbekijken en aanpassen waar nodig
- maken van gestandaardiseerde workspace

Resultaten:

Deel 1 van de vergadering: 15:20 tot

De deadlines die er waren tegen vandaag waren de volgende:

Tegen 8 maart ALEX: Tomcat op de server installeren en testen. Account klasse definieren. UML klassendiagramma aanpassen.

Tomcat is geinstalleerd en draait, maar we kunnen er nog niet aan omdat de link die we moeten gebruiken om een servlet aan te spreken nog niet bekend is.
UML diagramma's zijn verbeterd
Account klasse is nog niet gedefinieerd

Tegen 8 maart ADAM: Basislayout site afwerken (tabellen, locaties,... geen grafische elementen)

basislayout van site is afgewerkt (zie site bij test) en in een reeks XML bestanden gezet met oog op vertaling en dergelijke.
hetzelfde moet nog gedaan worden met de CSS bestanden
er moet nog een XML editor gevonden worden om de klant toe te laten die bestanden aan te passen

Tegen 8 maart ADAM en ALEX: inlogscherm in HTML en servlet laten overeen komen

Er

Tegen 8 maart ZJEF: Kalenderstructuur opmaken in JAVA.

iCalendar standaard gebruikt om kalender op te slaan. Er zijn interfaces gevonden om calenders aan te maken en uit te lezen naar JAVA.
iCal4j-1.0-rc3 te vinden op svn
Er moet nog een iCalendar viewer gevonden worden om het van de JAVA code om te zetten naar HTML/javascript (eventueel applet) op de site

Tegen 8 maart ZJEF: JUnit tutorial uitleggen aan de rest van het team en regels opstellen waaraan de testmethodes moeten voldoen

Tegen 8 maart MATTHIAS: Datastructuur aan JAVA zijde afwerken (zie design viewpoint 2: Logical van het SDD) en constraints oplijsten + vertalen naar JAVA

ongeveer gebeurd, paar details aangepast.
algoritme wordt via course opgebouwd
bevat: naam, studentcounter, hoursWPO, hourHOC neededhardware (per WPO en HOC), educator zit in vak
de opmerking werd gemaakt dat de educators ook een lijst van vakken moet bevatten om zijn rooster op ge vragen
de keuze voor WPO en HOC moest uitbreidbaar zijn. Het moet aangepast worden
Elk vak moet ook een lijst hebben van programmas die het vlak volgt
Extra Zjef paste de XML file interface aan om links te parsen en HTML code toe te voegen. (zie version history op svn)
Hij paste ook de database interface aan om het uitlezen aan te passen (zie version history op svn)

De deadlines voor volgende week en voor de SCRUM meeting zijn de volgende:

Tegen 17 maart SCRUM MATTHIAS: werking algoritme bepalen en presentatie voorbereiden over UML klassendiagramma (SDD viewpoint 2)

Tegen 17 maart SCRUM ADAM en ALEXANDER: Inloggen op site met echte connectie met server + presentatie erover (beveiliging, servlets, tomcat,... (SDD Viewpoint 3))

Tegen 17 maart SCRUM ZJEF: Database interface afgewerkt en geeft presentatie over database (alsook andere interfaces die hij ondertussen al geschreven heeft)

Tegen 17 maart SCRUM presentatie maken over toekomstige plannen, fouten, al de rest voor de SCRUM

de outline van de presentatie voor de SCRUM meeting werd vastgelegd:

algemene inleiding	MATTHIAS
voorstellen van team + functie	MATTHIAS
overloping van de opdracht	MATTHIAS
inhoud van de presentatie	MATTHIAS
klassenstructuur, UML diagramma tonen	MATTHIAS
Algoritme	MATTHIAS
Database structuur interface	ZJEF
XML structuur interface	ZJEF
iCalendar, interface	ZJEF
demonstratie van servlet	ALEXANDER
uitleggen beveiliging	ALEXANDER
toekomstplannen voor volgende iteratie	ADAM
afscheid	ADAM

Er was nog onzekerheid over de duur van de presentatie en daarvoor werd professor Ragnhild gecontacteerd.

7.2.6 Vergadering van 22-3-2011

Punten op de agenda:

- Updaten van Adam na zijn week ziekte
- Bespreken van de doorgevoerde veranderingen: Database, Algoritme, kalender, session tracking, HTML builder,...
- Opstellen nieuwe deadlines en taken voor de komende weken
- Bespreken moeilijkheden van configuratie management voor de verschillende versies van de interfaces

Resultaten:

- Nieuwe deadlines opgesteld
- Site inhoud grondiger gespecifieerd

Minutes 12:30 tot 14:00 en van 16:00 tot 18:00

Bespreken van de vorige taken:

Algoritme: Er werd beslist om CHOCO Java library te gebruiken voor het oplossen van het kalenderprobleem.

Database: Er zijn een paar bugfixes doorgevoerd. We zitten nu op versie 1.2

Kalender: PHP iCalendar is ingevoerd in een tabblad van de site. Er kunnen parameters meegegeven worden in het iFrame om aanpassingen door te voeren

Session Tracking: De session tracking werkt en de gebruikers worden automatisch na 5 minuten uitgelogd

HTML builder: De HTML builder werkt, maar moet nog uitgebreid en gedocumenteerd worden. Hij moet nu in de bestaande servlet met session tracking geintegreerd worden

Nieuwe deadlines tegen woensdag 30/3

MATTHIAS: Degelijke tijd/locatie/subcourse structuur opstellen en daarnaast een methode schrijven die toelaat om de subcourse op plaatsen en tijdstippen te plaatsen zodat aan een constraint voldoen is.

ALEXANDER en ADAM: HTMLBuilder integreren in bestaande servlet.
Verschillende methoden implementeren die de 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. Aanpassen van stukken delegeren.

ALEXANDER: SRS nalezen en up to date brengen. Aanpassen van stukken delegeren.

ZJEF: SDD nalezen en up to date brengen. Aanpassen van stukken delegeren. Bespreken van de tabbladen en functies van de site

Er werd beslist om de breedte van de site van 600 pixels te veranderen naar 1024 pixels.
Dan werden de verschillende tabbladen nog is opgesomd en de functies die erin moeten gespecifieerd, zodat Adam en Alexander eraan kunnen beginnen werken.

Login tab
login form
log in as guest
Als gast
Search tab:
Naam kiezen van student of prof of schuiver met faculteiten en dan schuiver met programma's Rooster bekijken
Als student
Account tab:
Wachtwoord veranderen
Mijn cursussen Programma wordt weergegeven en de cursussen die daarin zitten worden getoond en Lijst van cursussen die hij als keuzevakken heeft (met de mogelijkheid om ze te verbergen)
Voeg vak toe
Voeg programma toe
(voor educators) edit course
Lessenrooster tab:
toont persoonlijk lessenrooster
Voor educators
Availability tab
lijst van constraints die de professor opgegeven heeft (edit, add, delete)
Voor Admins
Building management tab
Lijst van buildings met daarin een lijst van lokalen
per builing: add, edit, delete
per lokaal: add, edit delete
Courses management tab
toont lijst van programs en courses die erin zitten en dan de subcourses
add/delete/edit program knop
add/delete/edit course knop
add/delete/edit subcourse knop

Facultaire kalender tab (ingeven van de algemene niet-educator gebonden constraints)
Lijst met constraints weergeven
add/edit/delete constraint
Kalender editor tab
toont kalender en kan kiezen welke vakken er in getoond kunnen worden
kan vakken van programmas weergeven en aparte vakken toevoegen aan de lijst die weergegeven wordt
Accounts management tab (voor de admins)
Lijst van accounts weergeven, bij elke account: delete of edit
add account
search account
Bereken rooster
Start het uitrekenen van lessenrooster