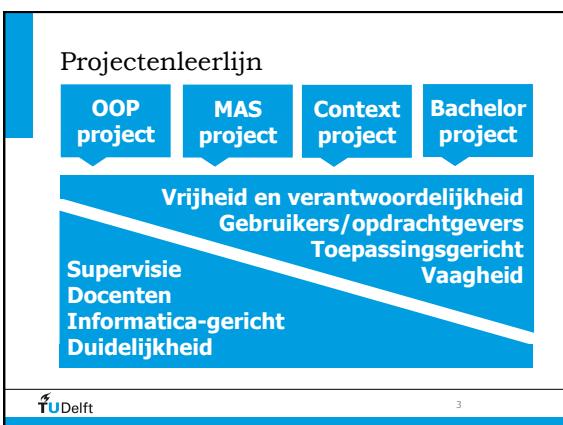




TU Delft
Ghent University of
Technology
Challenge the future



2



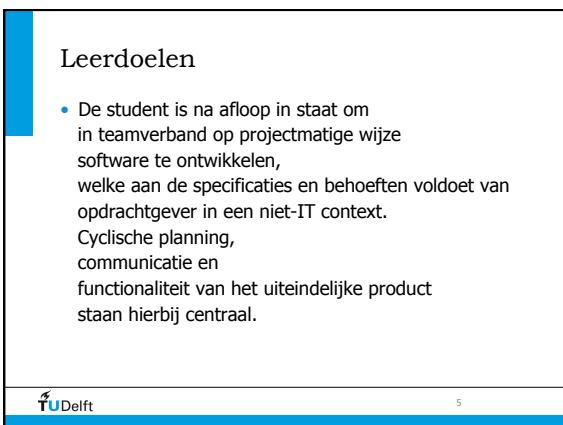
TU Delft

3



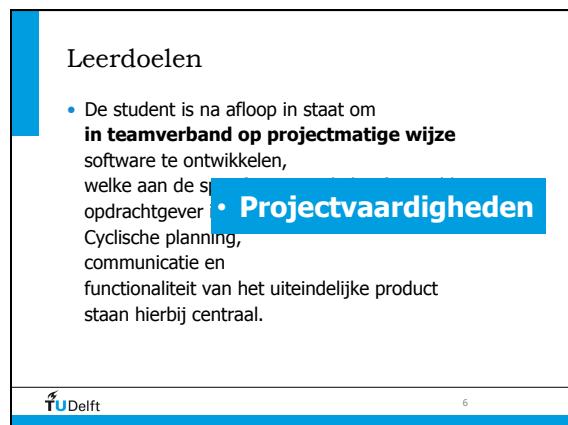
TU Delft

4



TU Delft

5



TU Delft

6

Leerdoelen

- De student is na afloop in teamverband op projectmatige wijze **software te ontwikkelen**, welke aan de specificaties en behoeften voldoet van opdrachtgever in een niet-IT context. Cyclische planning, communicatie en functionaliteit van het uiteindelijke product staan hierbij centraal.



7

• Programmeren

Leerdoelen

- De student is na afloop in teamverband op projectmatige wijze **software te ontwikkelen**, welke aan de specificaties en behoeften voldoet van **een opdrachtgever in een niet-IT context**. Cyclische planning, communicatie en functionaliteit van het uiteindelijke product staan hierbij centraal.



8

• Context seminar • Interviews

Leerdoelen

- De student is na afloop in teamverband op projectmatige wijze **software te ontwikkelen**, welke aan de specificaties en behoeften voldoet van **een opdrachtgever in een niet-IT context**. Cyclische planning, communicatie en functionaliteit van het uiteindelijke product staan hierbij centraal.



9



• Software eng. methoden

Leerdoelen

- De student is na afloop in teamverband op projectmatige wijze **software te ontwikkelen**, welke aan de specificaties en behoeften voldoet van **een opdrachtgever in een niet-IT context**. Cyclische planning, **communicatie** en functionaliteit van het uiteindelijke product staan hierbij centraal.



10

• Mondeling en schriftelijk rapporteren

Leerdoelen

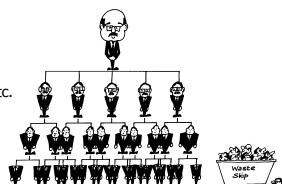
- De student is na afloop in staat om in teamverband op projectmatige wijze **software te ontwikkelen**, welke aan de specificaties en behoeften voldoet van een opdrachtgever in een niet-IT context. Cyclische planning, communicatie en **functionaliteit van het uiteindelijke product** staan hierbij centraal.



11

Rollen

- Coördinatoren
 - regelen de gang van zaken
- Gebruikers
 - vertellen hun *user stories*
 - beoordelen het product
 - spreek je af en toe
- Contextdocenten
 - organiseren seminar, beoordelen documenten etc.
 - spreek je regelmatig
- Studentassistenten
 - geven feedback en advies, geen opdrachten
 - spreek je wekelijks
- Software engineers: doen het werk...



12

Team (software engineering)

- Coordinator: Andy Zaidman
- Teaching assistants:
 - Michiel van Dam (Computer Games)
 - Tiago Espinha (Health Informatics)
 - Pieter Hameete (Crisis Management)
 - Daniele Romano (Cultural Heritage)
 - Wouter Willems (Programming Life)



13

Werken aan de opdracht

- In teams van 5 studenten
 - Zorg ervoor dat ieder teamlid bijdraagt aan het success van het project!
- Thuis & aan de Driebbelweg
 - Q3: Wednesday + Friday, 13:45 - 17:30, PC DW 0.010 (8 hours/week)
 - Q4: Mon+Thu+Fri, 8:45 – 12:30, various rooms (12 hours / week)
- Belasting
 - 7 ECTS = 196 hours = 12.5 hours/week (16 weeks)



14

Globale agenda

- Q3 (Design het software systeem, start implementatie)
 - week 1-3 Requirements analysis
 - week 5 Architectural design
 - week 6 Test and implementatie plan
 - week 6-7 1ste iteratie van de implementatie
- Q4 (Verder gaan en afwerken van de implementatie)
 - week 1-2 2de iteratie, 1ste korte demo
 - week 3-4 3de iteratie, 2de korte demo
 - week 5-6 4de iteratie, 3de korte demo
 - week 7-8 5de en laatste iteratie, 4de korte demo
 - week 9 Groupspresentaties



15

Documenten

- In het Nederlands (English is ok as well)
- Voor de analyse en design documenten
 - Bekijk het materiaal van de Software Engineering cursus
 - Documenting requirements, the architecture, test and implementation plan
 - Zorg ervoor dat figuren en tabellen een caption hebben
 - Zorg ervoor dat figuren en tabellen een anker in de tekst hebben (bv., Plaatje 1 toont)
 - Zorg ervoor dat nieuwe documenten consistent zijn met vorige
 - Voeg "back-links" toe naar informatie in vorige documenten, bv. het nummer van een requirement.
- Meer info in de projectwijzer



16

Ontwikkelproces

- Volg een iteratief process
 - 1-2 weeks per iteration
- Voor elke iteratie
 - Kies de features die je wil implementeren
 - De meest belangrijke features eerst
 - Leg de taken vast, typisch ~1-5 per feature
 - Schat de effort per taak in en wijs toe aan een ontwikkelaar
 - Dan, en enkel dan beginnen met programmeren *en* testen
 - Aan het einde: check welke taken klaar zijn en welke meer werk nodig
 - Bekijk welke taken terugkomen in de volgende iteratie
- Plan is een actief document, telt mee voor je eindcijfer... doe je in de online Planbox tool



17

Voorbeeld plan voor iteratie 5

ST4 Planning					
Group:	4	Sprint:	5		
Feature	Taken	Verwacht	Tijd	Ontwikkelaar	Status
Routes beheren door koerier	gebruikersinterface	8 uur	9 uur	Erik	voltooid
	datalaag uitbreiden voor routes (o.a. Kruistabel)	4 uur	4 uur	Jan	voltooid
Inlogfunctionaliteit	gebruikersbeheer	10 uur	11 uur	Wilson	voltooid
	datalaag uitbreiden voor gebruikers	2 uur	2 uur	Erwin	voltooid
Overig	openstaande problemen bugtracker	2 uur	3 uur	Quinten	voltooid
	verificatie van stores	5 uur	4 uur	Jan	voltooid
tests uitbreiden (library splitsen, nieuwe tests)	tests uitbreiden (library splitsen, nieuwe tests)	8 uur	5 uur	Erwin	voltooid
	migratie van de gemaakte branch	4 uur	4 uur	Jan	voltooid
	uitwerken optimalisatieproblemen	4 uur	2 uur	Wilson	voltooid,
	consistente error handling in huidige code	8 uur		Wilson	niet voltooid
herschrijven handlers mht koeriers->booktrader	debuggen koerierapplicatie	3 uur	Erik	voltooid	
	3 uur	Quinten	voltooid		
	4 uur	Erwin	voltooid		
gebruik timeouts en laadbalken verbeterd	4 uur	Quinten	voltooid		
	1 uur	Erwin	voltooid		

Implementatie

- Tests zijn een must!
 - **xUnit tests** voor de central functionaliteit
 - Indien beschikbaar voor de programmeertaal: **test coverage**
- IDE
 - Vrije keuze
- Programmeertaal
 - Hangt af van context
- Versiebeheer:
 - Vrije keuze (straks ook een primer Git)
- Zorg ervoor dat alle teamleden aan de code kunnen bijdragen



19

Implementatie (2)

- Plannen met Planbox
 - Alle teamleden hebben toegang (of krijg je binnenkort van je SE TA)
 - Julie SW TA kan jullie voortgang mee volgen



20

Altijd een werkende versie van je systeem

- Korte demo's
 - Om de 2 weken
 - Exacte data worden nog bekend gemaakt
 - Laat de meest recent features zien
 - Informeel (geen slides, enkel draaiend systeem + testen)
 - 5-10 minuten



21

Score

- Score is gebaseerd op
 - Rapporten (kwaliteit, compleetheid, originaliteit, presentatie)
 - Proces, stijl van werken (volgen van een process - planning, reflectie)
 - De applicatie (functionaliteit, originaliteit)
 - Korte demos
 - Implementatie
 - Moet draaien en moet getest zijn
 - Source code wordt geëvalueerd door SIG
 - Presentatie & demo
 - Score is voor de groep
 - Soms wordt hier een uitzondering op gemaakt ("speciaal geval")



22

Elke week onmoeting met TA's

- Bereid elke meeting voor
- Laat de voortgang zien
- Leg ideeen, oplossingen en problemen voor
- Krijg feedback op de rapporten en de implementatie
 - Integreer de feedback in de finale versies



23

Communicatie

- Eerstelijns: teaching assistant
- Tweedelijns: context docent



24

Succes!

- Dick de Ridder
- Andy Zaidman
- Context Project 2012-2013 coordinatoren



Extra information on deliverables

Product vision

= description of how the stakeholders get value

- A simple way to generate a product vision is to use the *elevator statement* template
- FOR (customer) WHO (statement of need) THE (product name) IS A (type of product) THAT (has this compelling reason to buy/use). UNLIKE (competitive products) OUR PRODUCT (is differentiated in these ways).

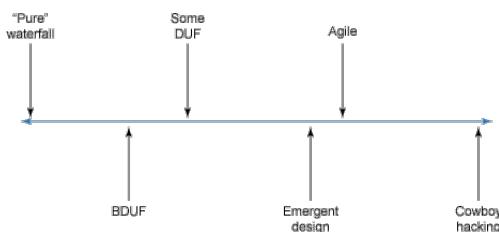
Product vision example

FOR book lovers WHO LIKE TO discuss and share experiences with like minded-people mybookshelf.com IS AN internet community THAT allows people to create their own virtual bookshelf and discuss their opinions and experiences. UNLIKE traditional discussion forums, OUR SERVICE provides tailored features to their needs.

Product vision

- Extra information can be found on p.292 and further of “Essential Scrum” (available online @ Safari Books)

Emergent Architecture



Planned versus emergent

- **Planned architecture** = design time artifact, something that is envisioned, designed and then implemented, is mainly static
- **Emergent architecture** = architecture-as-is, emerges from efforts and intentions of many people, changes along the way

Emergent architecture guidelines

- Think of **high-level components, deployment**
- Later on, the interfaces between components, sub-systems become important
- Details can be filled in later on during each sprint → living document
- Not a detailed UML class diagram, just everything you need to work together in a team

Backlog

- A (prioritized) list of items, typically user stories
 - Features
 - Defects
 - Technical work
 - Knowledge acquisition

User story

A user story describes functionality of a system that will be valuable to a **Non Development Team (NDT)** **stakeholder** of a system or software. User stories are composed of three aspects:

- a **written description or short title** of the story used as a token for planning and as a reminder to have conversations
- **conversations** about the story that serve to flesh out the details of the story
- **acceptance tests** that convey and document details and that can be used to determine when a story is complete

User story (example 1)

A company can pay for a job posting with a credit card.

Note: Accept Visa, MasterCard, and American Express.
Consider Discover.

User story (example 2)

A company can pay for a job posting with a credit card.

Note: Will we accept Discover cards?

Note for UI: Don't have a field for card type (it can be derived from first two digits on the card).

Tasks

- Individual steps that a team performs in order to complete a backlog item
- Defined to be small
 - A few hours to a day (or 2) of work, typically 4 – 16 hours

Tasks

- Examples
 - Create database
 - Writing business logic
 - Write UI for feature
 - Write tests

When is something “done”?

- If it can be shipped to the stakeholder