Gunadarma University
http://www.gunadarma.ac.id

COLORING THE GLOBAL FUTURE

Agile Methodology

Agile Methodology

What often happens is that the customer is paralyzed by not knowing what technology could do and the developer is stuck by not knowing what the customer needs to have.

Having a working instance, a prototype, or performing a so called Wizard-of-Oz experiment with a mock-up system.

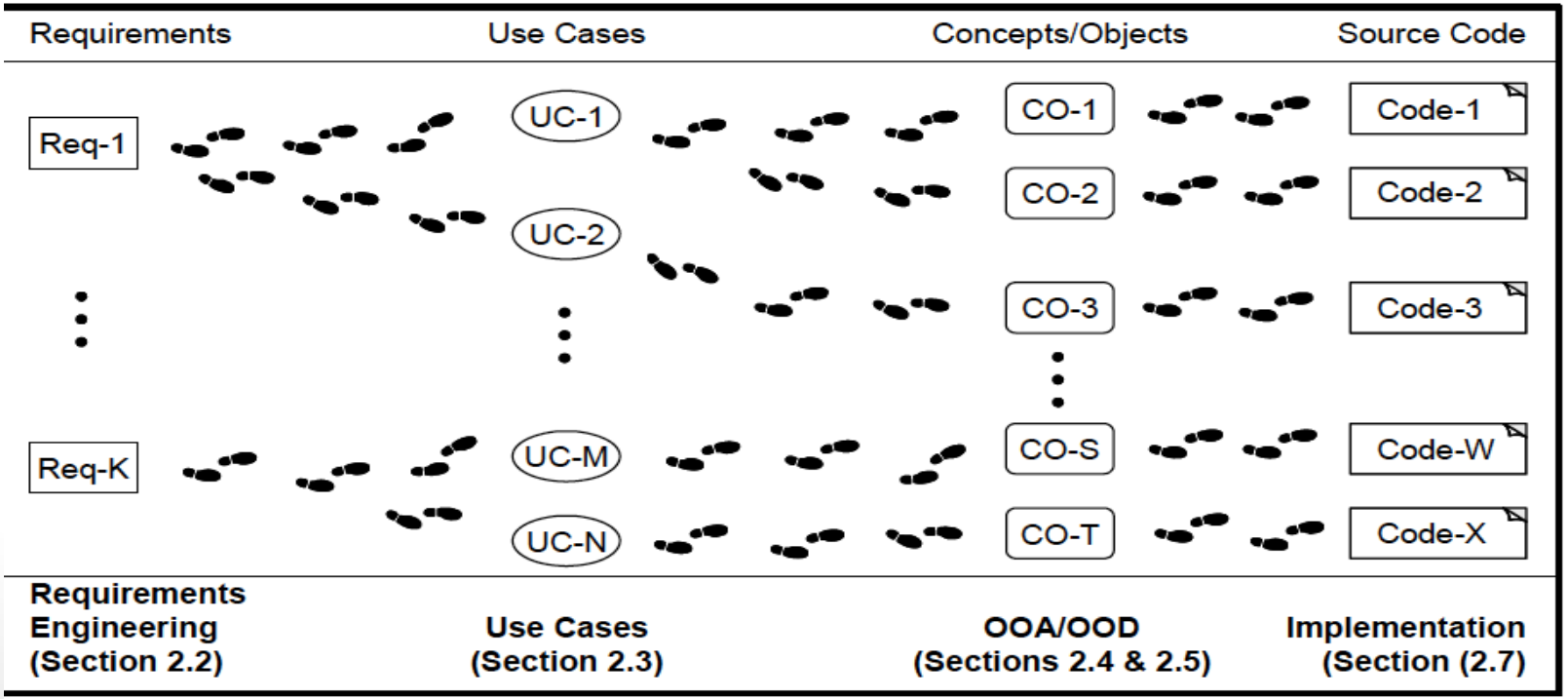
Agile Methodology

- The tradeoff is that writing proper documentation takes time, and it is difficult to maintain the documentation consistent with the code as the project progresses.
- Even greater problem is that the code documents only the result of developer's design decisions, but not the reasoning behind those decisions. Code is a solution to a problem

Decisive Methodological Factors

- Software quality can be greatly improved by paying attention to factors such as traceability, testing, measurement, and security.
- Traceability refers to the property of a software artifact, such as a use case or a class, of being traceable to the original requirement or rationale that motivated its existence
- must be maintained across the lifecycle

Traceability Work



Testing

- Test-Driven Development (TDD) is that every step in the development process
- must start with a plan of how to verify that the result meets some goal. The developer should not
- create a software artifact (such as a system requirement, a UML diagram, or source code) unless he has a plan of how it will be tested.

Testing

- The testing process is not simply confined to coding. Testing the system design with walkthroughs and other design review techniques is very helpful.
- Agile TDD methodology prescribes to make progress just enough to pass a test and avoid detailed analysis. When a problem is discovered, fix it. This approach may not be universally appropriate, e.g., for mission critical applications.

Testing

- Agile TDD claims to improve the code, and detect design brittleness and lack of focus. It may well do that, but that is not the main purpose of testing, which is to test the correctness, not quality of software

Measurement

- Metrics and measurement are relatively rarely used, particularly for assessing software product quality.
- Agile methods have emphasized using metrics for project estimation, to track progress and plan the future iterations and deliverables.
- Software product metrics are intended to assess program quality, not its correctness (which is assessed by testing and verification).
- Metrics do not uncover errors; they uncover poor design.

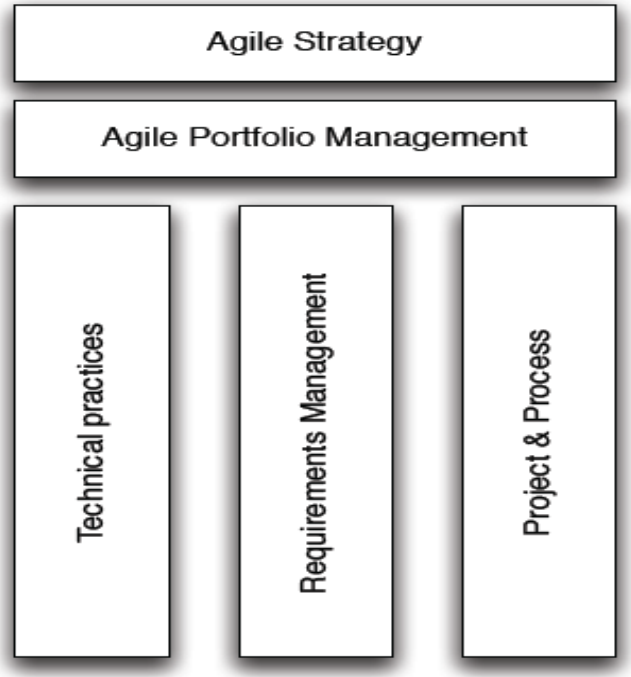
Security

- There are two kinds of technology-based security threats in software systems.
- One arises because of bad software, where the attacker exploits software defects.
- The other arises because of network interconnectedness, when the attacker exploits other infected systems to poison the traffic to or from targeted computers.

Security

- Security risk management focuses on minimizing design flaws (architectural and design-level problems) and code bugs (simple implementation errors in program code).
- Identifying security flaws is more difficult than looking for bugs, because it requires deep understanding of the business context and software architecture and design. We work to avoid design flaws while building secure software systems.
- Techniques include risk analysis, abuse cases (trying to misuse the system while thinking like an attacker), and code quality auditing.

Three pillars of Agile supporting high Agility



These three pillars provide the operations base on which organizations can push to portfolio and strategic Agile. (See (Kelly, 2010b) for more about Agile at the portfolio and strategy level.)

Some overarching principles

1. **Business value focused**: requirements are a means to an end. The overall objective is to deliver business value
2. **Goal directed projects** : With a goal clear requirements can be discovered by working backwards. Requirements are the things that will move the organization from where it is today towards its goal.
3. **Customer/End User involvement**: those who will actually use the end product need to have a voice in how it is built, and need to have early sight of what is being created

Some overarching principles

4. Iterative: in common with the rest of Agile requirements require an iterative approach.

5. Just in time : Since requirements are an ongoing process there is simply no need to create a store so we adopt a just-in-time principle instead

6. Dialogue over document: communication of requirements is primarily a dialogue rather than an exhaustive document.

7. Analysis not synthesis : The process of building something is primarily synthesis. No amount of analysis will create synthesis, the individuals who are best suited to analysts are usually different to those who are best at synthesis.

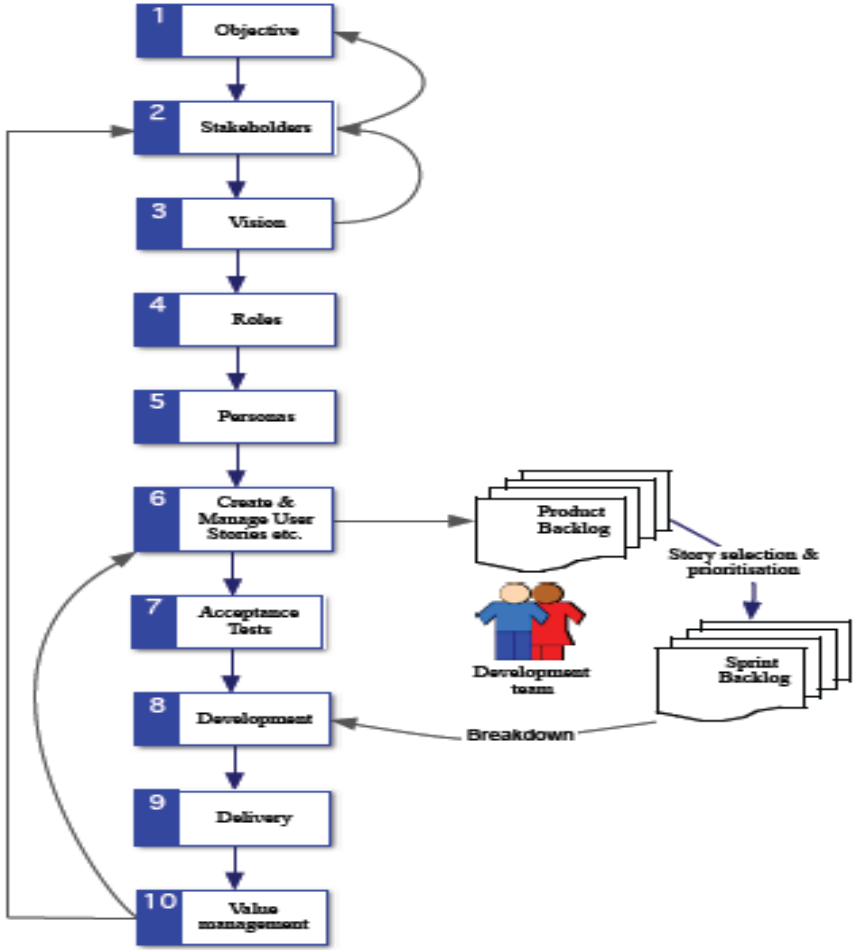
Who manages requirements?

- The subject of just who is responsible for managing the requirements is worth an article in itself. One of the main reason for IT project failure has been lack of user involvement.
- Agile's answer to this was to involve the customer, make them central to the development process.

Who manages requirements?

- For example, Microsoft Word has several million customers. While these customers may be segmented in various groups (Home, Business, Education, etc.) something needs to be done to understand competing needs, and priorities still need to be decided.

- In short, the "end user" as requirements gatherer and decider model - whether the XP or Scrum version has problems. What is needed is a requirements professional who can take on these issues and proxy for the final customer/users.



10 Step Model Overview

10-Step model overview

1. Objective: the objective is given from outside the model - usually from higher up the management chain. → project started
2. Stakeholders: stakeholders are those people, and groups of people, who have some interest in the work being undertaken. internal stakeholders and external stakeholders
3. Vision: The vision both expands on the objective and answers the objective. If the objective specifies a problem that needs solving the vision gives an answer.

10-Step model overview

4. Roles: roles narrow of the stakeholder base to consider those who will actually interact with the system as envisaged by the vision.
5. Personas: personas expand and elaborate certain roles, adding texture so requirements analysts, user design specialists and software developers can better understand and empathise towards those who will use the system.

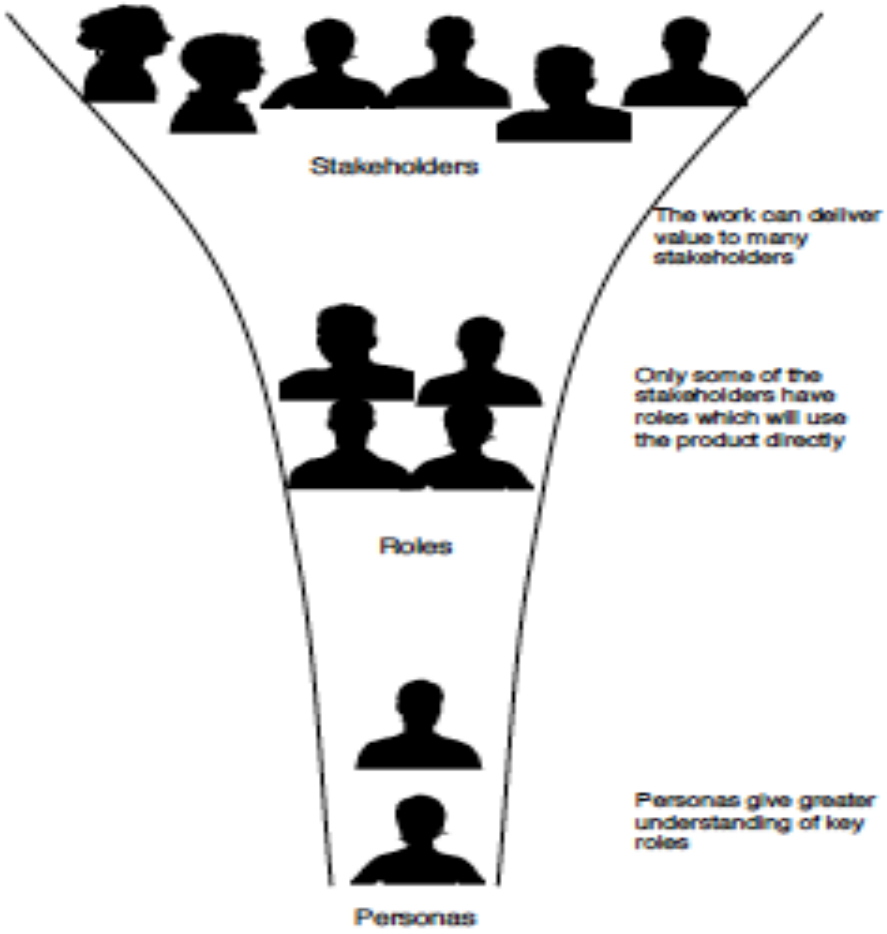


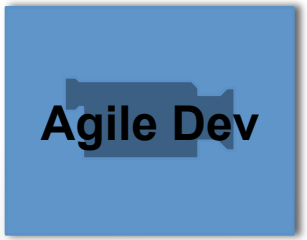
Figure 3 – From stakeholders to personas

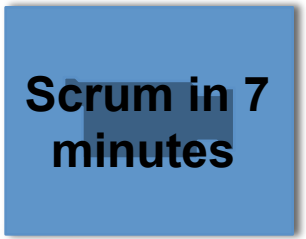
10-Step model overview

6. Create and manage stories: when objectives and users are well understood it is time to start specifying what they system will do. **This is the step into which much of the existing Agile literature fits: writing User Stories, Managing the Product Backlog and so on.**
7. Acceptance tests: once the essence of a story is captured some description of what constitutes done for the story needs be given.
8. Development: once a need is identified, understood and acceptance criteria specified it is time to actually do the work, develop the software.

10-Step model overview

9. Delivery: once a need is met the product needs to be delivered to the customer.
10. Value Management: last but by no means least is the need to close the loop and check that value is actually delivered.





User Stories, Product and Sprint Backlog

creates a product backlog, which usually contains user stories, to represent what its customers need and value.

The product backlog and the sprint backlog serve similar but distinct purposes. The product backlog is primarily managed by your product owner and contains a high-level view of all the work that your team must complete to create the product.

Team creates the sprint backlog, which contains a detailed list of all the tasks that your team must complete to finish the user stories for the sprint

Comparison Product and Sprint backlog

Item	Product Backlog	Sprint Backlog
Level of detail	Less Detail	Very Detailed
Estimation Units	Story Points	Hours
Document Ownership	Product Owner	Team
Revised	Weekly	Daily
Duration	Project	Sprint
Workbook	Product Backlog Workbook	Iteration Backlog Workbook

User Stories

User Stories are best written in following format.

As a <User>, I want to <Have> so that <Benefit>

- As a new user, I want to register by creating a username and password so that the system can remember my personal information.
- As a registered user, I can log in with my username and password so I can trust the system.
- As a registered user, I can change my password so that I can keep it secure or make it easier to remember.
- As a registered user, I want the system to warn me if my password is easy to guess so that my account is harder to break into.
- As a forgetful user, I want to be able to request a new password so that I am not permanently locked out if I forget it.

A Sample Format for a Spreadsheet-Based Product Backlog

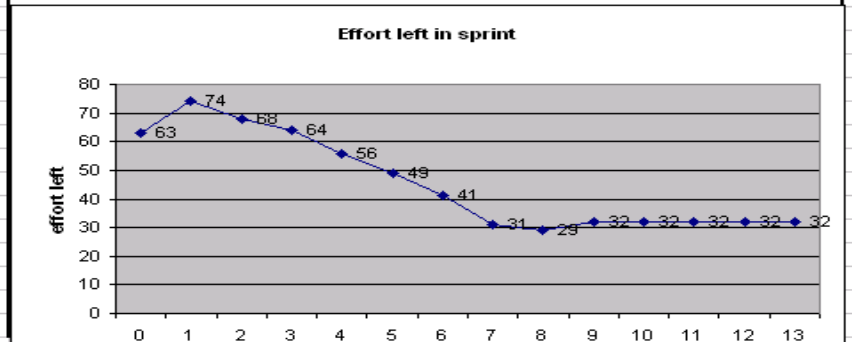
ID	Theme	As a/an	I want to...	so that...	Notes	Priority	Status
2	Game	moderator	create a new game by entering a name and an optional description	I can start inviting estimators	If games cannot be saved and returned to, the description is unnecessary	Required	done
2	Game	moderator	invite estimators by giving them a url where they can access the game	we can start the game	The url should be formatted so that it's easy to give it by phone.		done
5	Game	estimator	join a game by entering my name on the page I received the url for	I can participate			done
6	Game	moderator	start a round by entering an item in a single multi-line text field	we can estimate it			done
8	Game	estimator	see the item we're estimating	I know what I'm giving an estimate for			done
~~40~~	~~Game~~	~~participant~~	~~always have the cards in the same order across multiple draws~~	~~it's easy to compare estimates~~	-	Replaced with A08 because I didn't want the story to talk about "the same order" as that might be a UI implementation detail	~~todo~~
35	Non-functional	user	have the application respond quickly to my actions	I don't get bored			done
36	Non-functional	user	have nice error pages when something goes wrong	I can trust the system and it's developers			done
A11	Non-functional	Researcher	results to be stored in a non-identifiable way	I can study the data to see things like whether estimates converged around the first opinion given by "estimator A" for example	No names or story text should be stored but we should store each card of each hand, know who played it, and know the final accepted estimate		
A05	Game	moderator	edit an item in the list of items to be estimated	so that I can make it better reflect the team's understanding of the item			
22	Archive	moderator	export a transcript of a game as a CSV file	I can further process the stories and estimates	Exported file should be directly importable back into the system.		done

Sprint Backlog

Sprint 3. Plug in the Real Weather

Story ID	Story/task	days in sprint / effort left													
		0	1	2	3	4	5	6	7	8	9	10	11	12	13
		63	74	68	64	56	49	41	31	29	32	32	32	32	32
10	**Fetch one day temperature data from the weather provider system**														
	Make our server connect and authenticate to the provider system	4	4	1								3	3	3	3
	Read provider's data directory	8	7									0	0	0	0
	Parse the current temperature out of the data	6	6									1	1	1	1
	Push the temperature data to the client	16	16	1								0	0	0	0
11	**Fetch rain, snow, etc details from the provider**														
	Parse snow/rain data from the provider's data	4	4									0	0	0	0
	Push the snow/rain data to the client	4	4									0	0	0	0
	Redesign client screen a bit											3	3	3	3
	Refactor the server code										4	4	4	4	4
12	**Fetch several days data from the provider**														
	Parse the weather data in day packs	10	10	10	10	10	10	10	10	10	10	10	10	10	10
	Push several days data to the client	3	3	3	3	3	3	3	3	3	3	3	3	3	3
13	**Auto-refresh feature**														
	Make the client ping server once per 4 hours	6	6	6	6	6	6	6	6	6	6	6	6	6	6
	Make the server update the client	2	2	2	2	2	2	2	2	2	2	2	2	2	2

The team underestimated the difficulties of connecting to the provider's CORBA server. Even though some work has been already done, the remaining amount is much bigger, than the original estimation for the whole task



COLORING

Ujian (Kelompok)

UJIAN 1

I. Berdasarkan Project yang akan didevelop atau yang telah didevelop atau yang sedang didevelop
II. Buat User Stories, Product Backlog dan Sprint Backlog
III. Pembuatan dapat menggunakan manual dokumen atau tools atau template yang bisa didapatkan secara free

UJIAN 2

I. Ambil contoh Project → browsing dll
II. Ambil contoh User stories, product Backlog dan Sprint Backlog → berkelanjutan

UJIAN

Format :

1. Nama Sistem, nama Kelompok
2. Gambaran Umum Perangkat Lunak
3. Metode Pengembangan Perangkat Lunak
4. User Stories
5. Backlog Product
6. Sprint Product
7. Penutup