INF5181: Process Improvement and Agile Methods in Systems Development

Lecture 03:

Processes and Process Modeling (Section B)

UNIVERSITETET I OSLO

Dr. Dietmar Pfahl

email: dietmarp@ifi.uio.no

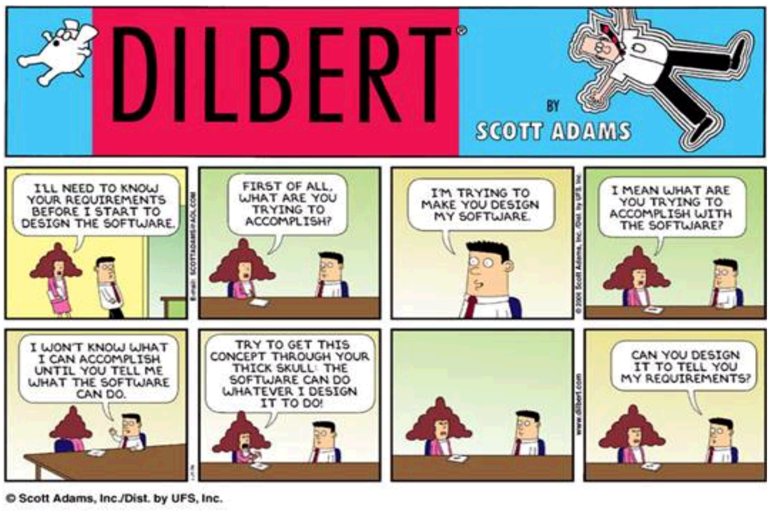
Fall 2011

Structure of Lecture 03

- Hour 1:
  - Light-weight (agile) processes / Evolutionary development ←
  - XP, Crystal and Scrum
- Hour 2:
  - Scrum (cont'd)
  - Choosing the right process (model)
- Hour 3:
  - Homework exercise
  - Question/answer session about project

UNIVERSITETET I OSLO

Requirements and Customers


DILBERT
BY SCOTT ADAMS
I'LL NEED TO KNOW YOUR REQUIREMENTS BEFORE I START TO DESIGN THE SOFTWARE.
FIRST OF ALL, WHAT ARE YOU TRYING TO ACCOMPLISH?
I'M TRYING TO MAKE YOU DESIGN MY SOFTWARE.
I MEAN WHAT ARE YOU TRYING TO ACCOMPLISH WITH THE SOFTWARE?
I WON'T KNOW WHAT I CAN ACCOMPLISH UNTIL YOU TELL ME WHAT THE SOFTWARE CAN DO.
TRY TO GET THIS CONCEPT THROUGH YOUR THICK SKULL: THE SOFTWARE CAN DO WHATEVER I DESIGN IT TO DO!
CAN YOU DESIGN IT TO TELL YOU MY REQUIREMENTS?
© Scott Adams, Inc./Dist. by UFS, Inc.

The Agile Manifesto

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over **processes and tools**
Working software over **comprehensive documentation**
Customer collaboration over **contract negotiation**
Responding to change over **following a plan**

That is, while there is value in the items on the **right**, we value the items on the **left** more.

http://www.agilemanifesto.org/

Structure of Lecture 03

- Hour 1:
  - Light-weight (agile) processes / Evolutionary development
  - XP, Crystal and Scrum
- Hour 2:
  - Scrum (cont'd)
  - Choosing the right process (model)
- Hour 3:
  - Homework exercise
  - Question/answer session about project

UNIVERSITETET I OSLO

Extreme Programming

- Origin: Kent Beck, Ward Cunningham, Ron Jeffries (end of 1990s)
- Idea: "light weight" process model, agile process
- Characteristic:
  - "Minimum" of accompanying measures (documentation, modeling , …)
  - Team orientation (e.g., common responsibility for all development artifacts)
  - Small teams (12-14 persons)
  - Involvement of user/client at an early stage
  - Social orientation
- Scope: Prototype projects, small projects, low criticality of the results

UNIVERSITETET I OSLO

XP – Rules and Practices

http://www.extremeprogramming.org/rules.html

Planning

User stories are written (by the customer!).
Release planning creates the schedule.
Make frequent small releases.
The Project Velocity is measured.
The project is divided into iterations.
Iteration planning starts each iteration.
Move people around.
A stand-up meeting starts each day.
Fix XP when it breaks.

Designing

Simplicity.
Choose a system metaphor.
Use CRC* cards for design sessions.
Create spike solutions to reduce risk.
No functionality is added early.
Refactor whenever and wherever possible.

* CRC = Class Responsibility Collaborator

Coding

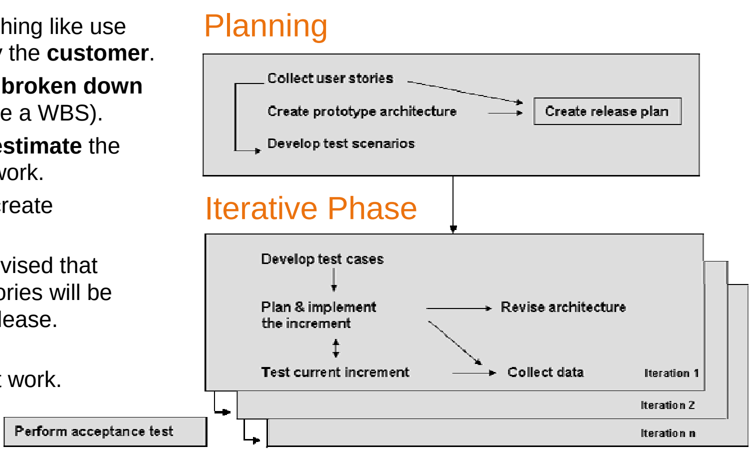
The customer is always available.
Code must be written to agreed standards.
Code the unit test first.
All production code is pair programmed.
Only one pair integrates code at a time.
Integrate often.
Use collective code ownership.
Leave optimization till last.
No overtime.

Testing

All code must have unit tests.
All code must pass all unit tests before it can be released.
When a bug is found (acceptance) tests are created.
Acceptance tests are run often and the score is published.

Extreme Programming – Overview

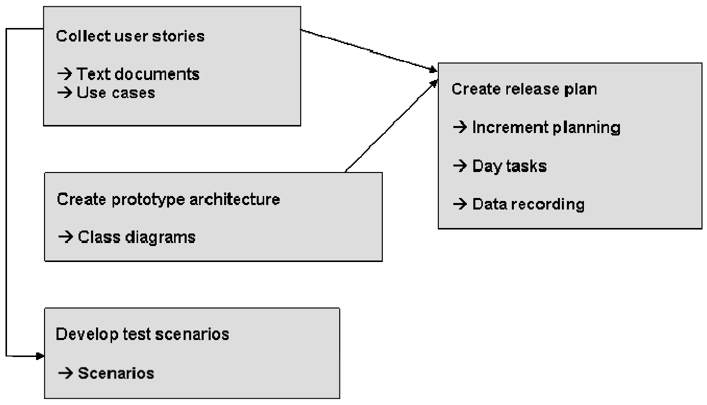
- **User stories** (something like use cases) are written by the **customer**.
- Complex stories are **broken down** into simpler ones (like a WBS).
- Stories are used to **estimate** the required amount of work.
- Stories are used to create **acceptance tests**.
- A **release plan** is devised that determines which stories will be available in which release.
- Don't hesitate to change what doesn't work.



http://www.extremeprogramming.org/rules.html

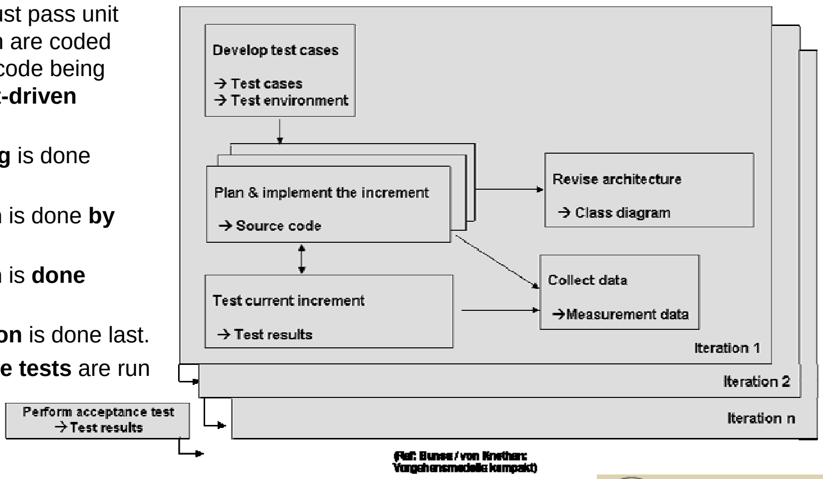
Extreme Programming – Planning

- Each release is preceded by a **release planning meeting**.
- Each day begins with a **stand-up meeting** to share problems and concerns.
- **CRC cards** are used for design. [XP and CRC were created by the same person, Kent Beck.]
- **Spike solutions** are done to assess risks.
- The **customer** is always available.



Extreme Programming – Iterative Phase

- All code must pass unit tests, which are coded before the code being tested (**test-driven design**).
- **Refactoring** is done constantly.
- **Integration** is done **by one pair.**
- **Integration** is **done frequently.**
- **Optimization** is done last.
- **Acceptance tests** are run often.



(Ref: Bunse / von Knethen: Vorgehensmodelle kompakt)

Requirements vs. User Stories

Traditional requirement – "shall" statements:

- "The system shall provide a user configurable interface for all user and system manager functions"
- "The user interface shall be configurable in the areas of:
  - Screen layout
  - Font
  - Background and text color

Corresponding "User Story":

- "As a system user or system manager, … **1**
- … I want be able to configure the user interface for screen layout, font, background color, and text color, … **2**
- … So that I can use the system in the most efficient manner" **3**

who - what - why

From Requirement to User Story – Functional Requirements

Requirement:

- The system shall provide the capability for making hotel reservations.

User Story 1:

- As a <u>premiere member</u>, I want to <u>search for available discounted rooms</u>.

User Story 2:

- As a <u>vacationer</u>, I want to <u>search for available rooms</u>.

User Story 3:

- As a <u>vacationer</u>, I want to <u>save my selections</u>.

From Requirement to User Story – Non-Functional Requirements

Requirement:

- The system shall …

User Story 4:

- As a vacationer and user of the hotel website, I want the system to be available 99.99% of the time.

User Story 5:

- As a vacationer, I want web-pages to download in <4 seconds.

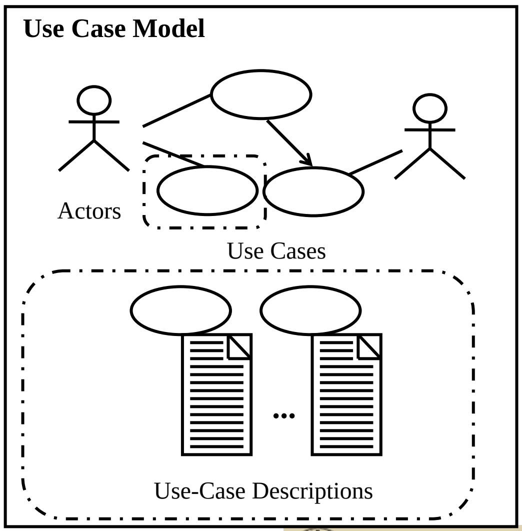
User Story 6:

- As the hotel website owner, I want 10,000 concurrent users to be able to access the site at the same time with no impact to performance.

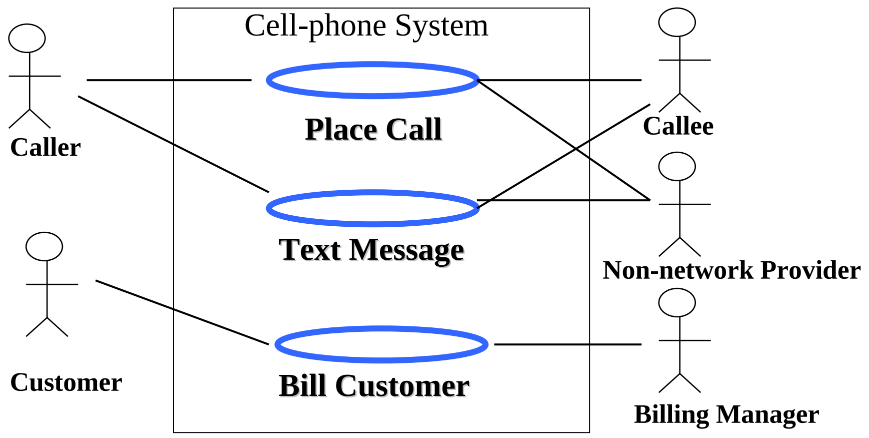
Use Case Model: Use Case Diagrams and Descriptions

Use Case Description:

- Name of Use Case
- Actors associated with Use Case
- Pre-conditions
- Post-conditions
- Normal Flow of Events (Basic Scenario)
- Alternative Flow of Events (Alternative Scenarios)
- …



Use Case Model – Example



A model of what the system is supposed to do (use case), the system's surroundings (actors), and their associations.

Use Case: Place Call

Actors: Caller, Callee, Network Provider

Pre-conditions: A caller wants to make a call to a callee. The cell phone is switched on and connected to a cell phone network. The phone is idle.

Post-conditions: On successful completion, the phone is idle. The caller has been connected to the callee for voice communication.

Basic Scenario:

- The caller activates the "call" option. (this may be by opening the phone or selecting some UI element.)
- The system displays a blank list of digits and indicates it is in "call" mode.
- The user enters digits (ALT 1).
- The system displays the entered digits.
- The user selects the "dial" option (ALT 2).
- The system sends the sequence of digits to the network provider.
- The network provider accesses the network and makes a connection (ALT 3, ALT 4).
- The callee answers (ALT 5).
- The network provider completes the voice connection.
- The caller and callee engage in voice communications.
- The caller hangs up (ALT 6).
- The system returns to idle mode.
- End of Use Case.

Use Case: Place Call

Actors: Caller, Callee, Network Provider

Pre-conditions: A caller wants to make a call to a callee. The cell phone is on and connected to a cell phone network. The phone is idle.

Post-conditions: On successful completion, the phone is idle. The caller has been connected to the callee for voice communication.

Alternative Scenarios:

ALT 1: The user uses speed dial.
- A1-1: The user enters a single digit and selects “dial”.
- A1-2: The system accesses the phone number associated with the digit (ALT 1.1).
- A1-3: Use case continues at step 6.

ALT 1.1: No speed dial number is associated with the entered digit.
- A1.1-1: The system ignores the “dial” command and displays the digit.
- A1.1-2: Use case continues at step 4.

ALT 2: The user cancels the operation.
- A2-1: Use case continues at step 12.

UNIVERSITETET I OSLO

CRC Card

- CRC = Class-Responsibility-Collaboration
- Used in OOA to identify classes, their responsibilities, and their collaborations

- Format:

Class Name	
Superclasses	
Subclasses	
Responsibilities	Collaborators

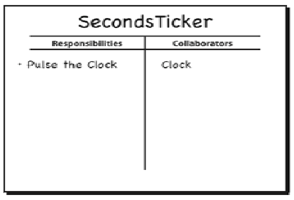
UNIVERSITETET I OSLO

CRC Cards for a ‘Clock’

- We want to design a clock.
- The clock should:
  - Have a way to set the current time
  - Display the time in hours, minutes, and seconds in different formats
  - Update the time to keep it current

Scenario 1: The Ticker Ticks

- The SecondsTicker pulses the Clock

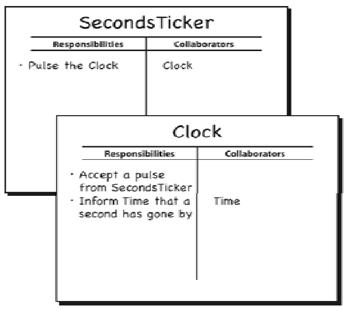


SecondsTicker

Responsibilities	Collaborators
- Pulse the Clock	Clock

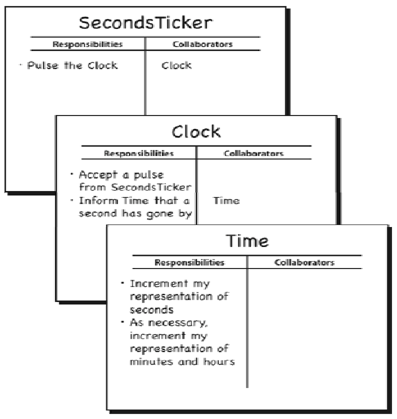
Scenario 1: The Ticker Ticks

- The SecondsTicker pulses the Clock
- The Clock updates Time



Scenario 1: The Ticker Ticks

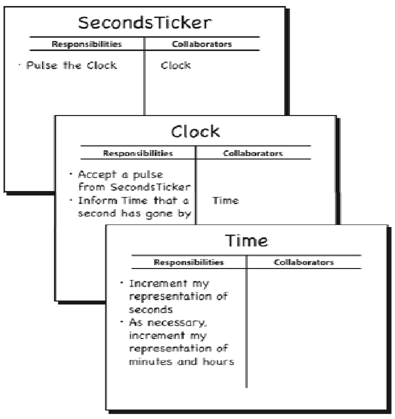
- The SecondsTicker pulses the Clock
- The Clock updates Time
- Time updates itself


SecondsTicker
Responsibilities
Collaborators
· Pulse the Clock
Clock
Clock
Responsibilities
Collaborators
· Accept a pulse from SecondsTicker
· Inform Time that a second has gone by
Time
Time
Responsibilities
Collaborators
· Increment my representation of seconds
· As necessary, increment my representation of minutes and hours

Scenario 1: The Ticker Ticks

- The SecondsTicker pulses the Clock
- The Clock updates Time
- Time updates itself

End of Scenario



SecondsTicker

Responsibilities	Collaborators
· Pulse the Clock	Clock

Clock

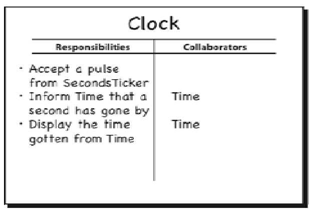
Responsibilities	Collaborators
· Accept a pulse from SecondsTicker	
· Inform Time that a second has gone by	Time

Time

Responsibilities	Collaborators
· Increment my representation of seconds	
· As necessary, increment my representation of minutes and hours	

Scenario 2: Clock Responds with the Time

- Display the time

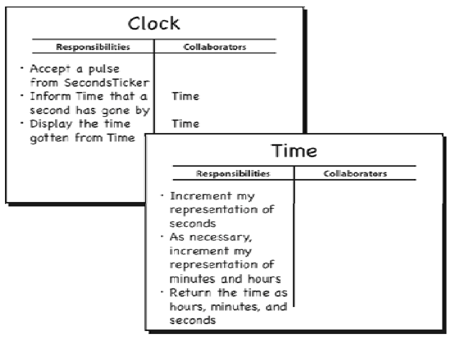


Clock

Responsibilities	Collaborators
· Accept a pulse from SecondsTicker	
· Inform Time that a second has gone by	Time
· Display the time gotten from Time	Time

Scenario 2: Clock Responds with the Time

- Display the time
- Collaborator: Return hours, minutes, & seconds



Clock

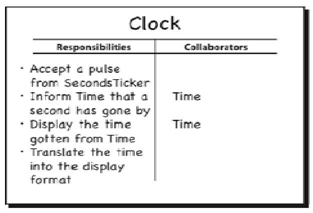
Responsibilities	Collaborators
· Accept a pulse from SecondsTicker	
· Inform Time that a second has gone by	Time
· Display the time gotten from Time	Time

Time

Responsibilities	Collaborators
· Increment my representation of seconds	
· As necessary, increment my representation of minutes and hours	
· Return the time as hours, minutes, and seconds	

Scenario 2: Clock Responds with the Time

- Display the time
- Collaborator: Return hours, minutes, & seconds
- Translate the time into the display format



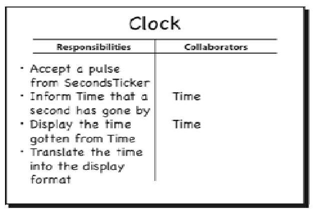
Clock

Responsibilities	Collaborators
· Accept a pulse from SecondsTicker	
· Inform Time that a second has gone by	Time
· Display the time gotten from Time	Time
· Translate the time into the display format	

Scenario 2: Clock Responds with the Time

- Display the time
- Collaborator: Return hours, minutes, & seconds
- Translate the time into the display format

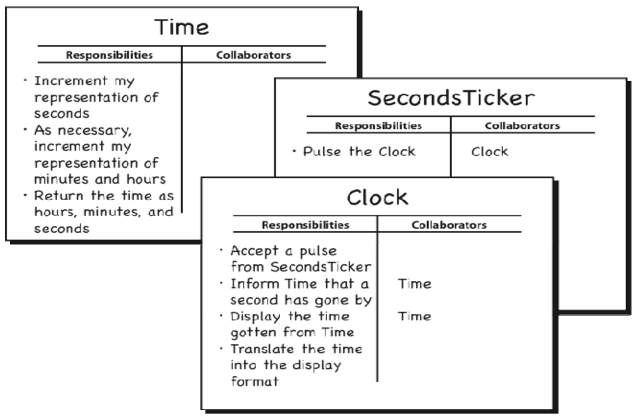
End of Scenario



Clock

Responsibilities	Collaborators
· Accept a pulse from SecondsTicker	
· Inform Time that a second has gone by	Time
· Display the time gotten from Time	Time
· Translate the time into the display format	

OOA for a Clock



Time

Responsibilities	Collaborators
· Increment my representation of seconds	
· As necessary, increment my representation of minutes and hours	
· Return the time as hours, minutes, and seconds	

SecondsTicker

Responsibilities	Collaborators
· Pulse the Clock	Clock

Clock

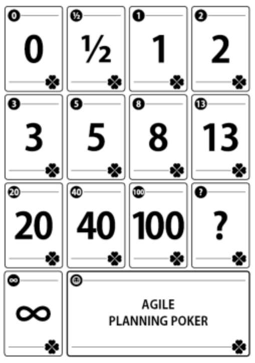
Responsibilities	Collaborators
· Accept a pulse from SecondsTicker	
· Inform Time that a second has gone by	Time
· Display the time gotten from Time	Time
· Translate the time into the display format	

Why CRC Cards?

- Forces you to think in “objects”
- Help you identify objects and their responsibilities
- Help you understand how the objects interact
- Cards form a useful record of design activity
- Cards work well in group situations and are understandable by non-technical stakeholders.

Planning Poker /1

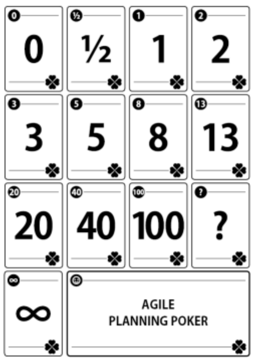
- Participants in planning poker include all of the developers on the team
- Step 1: Give each estimator a deck of cards
- Step 2: Moderator reads description of User Story to be estimated.
- Step 3: Product owner answers any question the estimators may have about the User Story.
- Step 4: Each estimator privately selects a card representing his or her estimate. Cards are not shown until each estimator has made a selection.
- …



Planning Poker /2

- Step 5: When everyone has made an estimate, the cards are simultaneously turned over.
- Step 6: If estimates differ, the highest and lowest estimates are explained by the estimators - otherwise the estimation is completed for this User Story.
- Step 7: The group can discuss the story and their estimates for a few more minutes. The moderator can take any notes he/she thinks will be helpful when this story is being programmed and tested. After the discussion, each estimator re-estimates by selecting a card.

  → Go to Step 5.



Note: In many cases, the estimates will already converge by the second round. But if they have not, continue to repeat the process. The goal is for the estimators to converge on a single estimate that can be used for the story. It rarely takes more than three rounds, but continue the process as long as estimates are moving closer together.

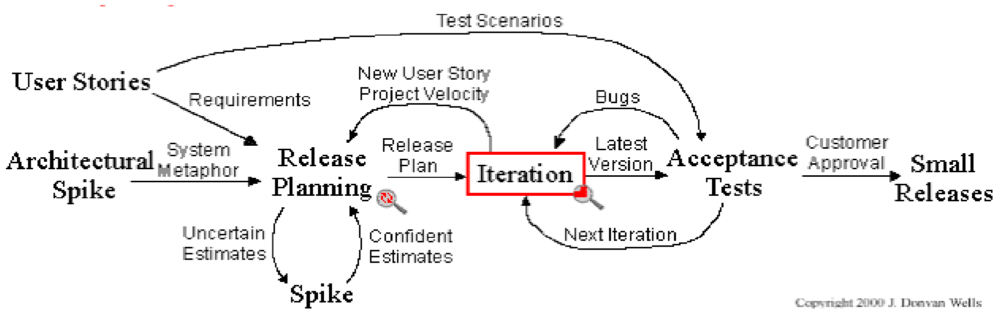
Refactoring

- Refactoring is a disciplined technique for restructuring an existing body of code, altering its internal structure without changing its external behavior. (Invented by Martin Fowler)
- Many refactorings can be automated
- Catalogue of refactorings:

  http://www.refactoring.com/catalog/index.html

- Note: It is not always clear (a) how to detect refactoring opportunities and (b) what refactoring(s) are most appropriate (→ 'code smells': http://en.wikipedia.org/wiki/Code_smell)

Extreme Programming – Evolutionary Process Model



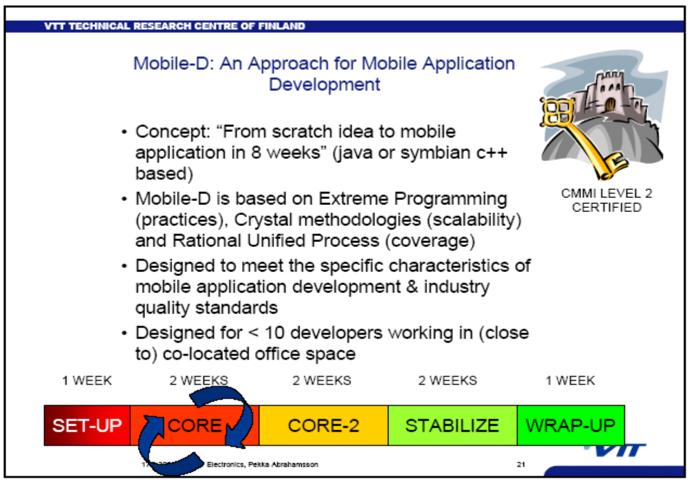
UNIVERSITETET I OSLO

Mobile-D

Defined by VTT for the mobile phone industry in Finland

More on Crystal methodologies can be found at:

http://alistair.cockburn.us/index.php/Crystal_methodologies



VTT TECHNICAL RESEARCH CENTRE OF FINLAND

Mobile-D: An Approach for Mobile Application Development

- Concept: "From scratch idea to mobile application in 8 weeks" (java or symbian c++ based)
- Mobile-D is based on Extreme Programming (practices), Crystal methodologies (scalability) and Rational Unified Process (coverage)
- Designed to meet the specific characteristics of mobile application development & industry quality standards
- Designed for < 10 developers working in (close to) co-located office space

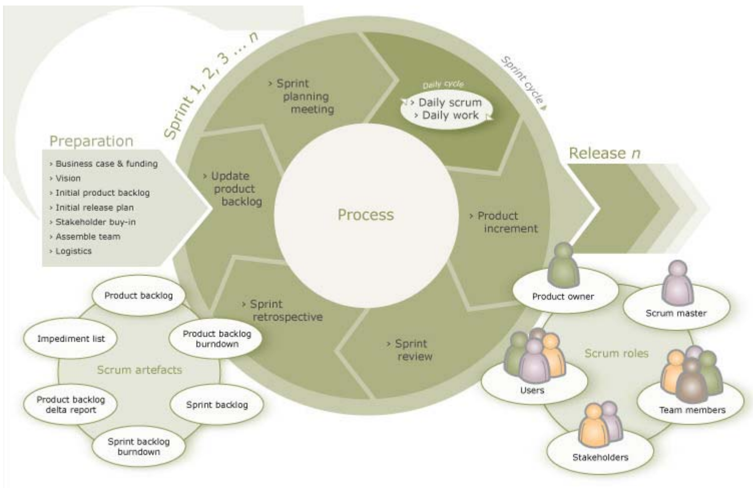
CMMI LEVEL 2 CERTIFIED

1 WEEK	2 WEEKS	2 WEEKS	2 WEEKS	1 WEEK
SET-UP	CORE	CORE-2	STABILIZE	WRAP-UP

Electronics, Pekka Abrahamsson 21

UNIVERSITETET I OSLO

Scrum

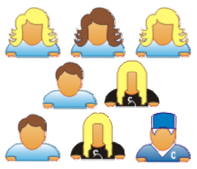


http://www.scrumforteamsystem.com/processguidance/v1/Scrum/Scrum.html

Scrum – Roles: ”Pigs” and ”Chicken”

"Pig" roles

- Pigs are the ones committed to the project in the Scrum process; they are the ones with "their bacon on the line".
  - Product Owner
  - Scrum Master (or Facilitator)
  - Team



"Chicken" roles

- Chicken roles are not part of the actual Scrum process, but must be taken into account.
  - Users
  - Stakeholders (customers, vendors)
  - Managers
- Note: An important aspect of an Agile approach is the practice of involving users, business and stakeholders into part of the process. It is important for these people to be engaged and provide feedback into the outputs for review and planning of each sprint.

Scrum – Roles

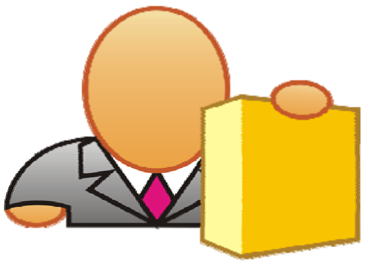
"Pig" roles:

- Product Owner
  - The Product Owner represents the voice of the customer ensuring that the Team works on the right things from a business perspective.
  - The Product Owner writes user stories, prioritizes them, then places them in the product backlog.
- Scrum Master (or Facilitator)
  - Scrum is facilitated by a ScrumMaster, whose primary job is to remove impediments to the ability of the team to deliver the sprint goal.
  - The ScrumMaster is not the leader of the team (as they are self-organizing) but acts as a buffer between the team and any distracting influences.
  - The ScrumMaster ensures that the Scrum process is used as intended. The ScrumMaster is the enforcer of rules.
- Team
  - The team has the responsibility to deliver the product.
  - A team is typically made up of 5–9 people with cross-functional skills to do the actual work (designer, developer, tester, etc.).

"Chicken" roles:

- Users
  - The software is being built for someone.
- Stakeholders (customers, vendors)
  - The people that will enable the project, and for whom the project will produce the agreed-upon benefit(s) which justify it. They are only directly involved in the process at sprint reviews.
- Managers
  - People that will set up the environment for the product development organizations.

Product Owner



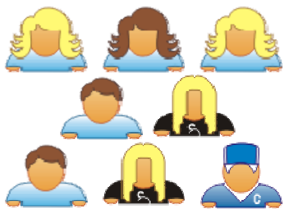
- Defines the features of the product
- Decides on release date and content
- Is responsible for the profitability of the product (ROI)
- Prioritizes features according to market value
- Adjusts features and priority every iteration, as needed
- Accepts or rejects work results

Scrum Master

- Represents management to the project
- Responsible for enacting Scrum values and practices
- Removes impediments
- Ensures that the team is fully functional and productive
- Enables close cooperation across all roles and functions
- Shields the team from external interferences

UNIVERSITETET I OSLO

Team



- Typically 5-9 people
- Cross-functional:
  - Programmers, testers, user experience designers, etc.
- Members should be full-time
  - May be exceptions (e.g., database administrator)
- Teams are self-organizing
  - Ideally, no titles but rarely a possibility
- Membership should change only between sprints

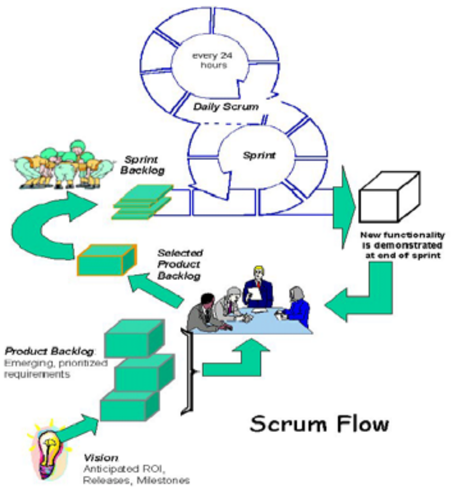
UNIVERSITETET I OSLO

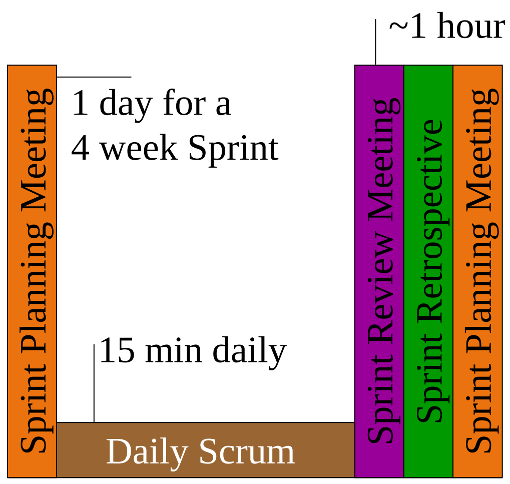
Structure of Lecture 03

- Hour 1:
  - Light-weight (agile) processes / Evolutionary development
  - XP, Crystal and Scrum
- Hour 2:
  - Scrum (cont'd)
  - Choosing the right process (model)
- Hour 3:
  - Homework exercise
  - Question/answer session about project

UNIVERSITETET I OSLO

Scrum Iterations and Sprints


every 24 hours
Daily Scrum
Sprint
Sprint Backlog
New functionality is demonstrated at end of sprint
Selected Product Backlog
Product Backlog
Emerging, prioritized requirements
Scrum Flow
Vision
Anticipated ROI, Releases, Milestones

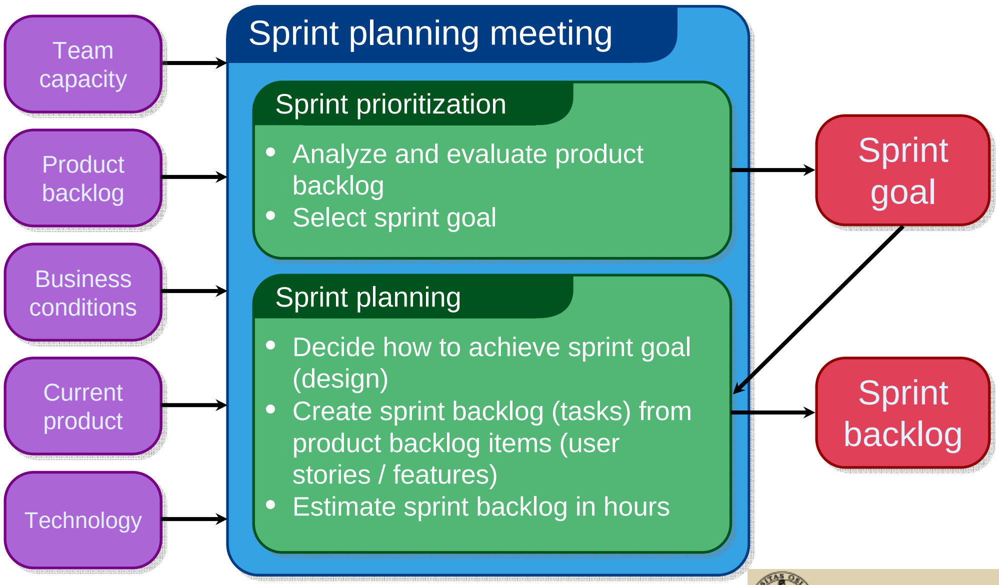


UNIVERSITETET I OSLO

Scrum – Meetings

- Daily Scrum
  - Each day during the sprint, a project status meeting occurs. This is called a "scrum", or "the daily standup". Daily scrum guidelines:
    - The meeting starts precisely on time. Often there are team-decided punishments for tardiness (e.g. money, push-ups, hanging a rubber chicken around your neck)
    - All are welcome, but only "pigs" may speak
    - The meeting is time-boxed (15 minutes) regardless of the team's size
    - All attendees should stand (it helps to keep meeting short)
    - The meeting should happen at the same location and same time every day
  - During the meeting, each team member answers three questions:
    - What have you done since yesterday?
    - What are you planning to do by today?
    - Do you have any problems preventing you from accomplishing your goal?
    - It is the task of the ScrumMaster to remind the team of these questions.
- Sprint Planning Meeting
  - Select what work is to be done
  - Prepare the Sprint Backlog that details the time it will take to do that work
  - 8 hour limit
- Sprint Review Meeting
  - Review the work that was completed and not completed
  - Present the completed work to the stakeholders (a.k.a. "the demo")
  - Incomplete work cannot be demonstrated
  - 4 hour time limit
- Sprint Retrospective
  - All team members reflect on the past sprint.
  - Make continuous process improvement.
  - Two main questions are asked in the sprint retrospective: What went well during the sprint? What could be improved in the next sprint?
  - 3 hour time limit

UNIVERSITETET I OSLO


Team capacity
Product backlog
Business conditions
Current product
Technology
Sprint planning meeting
Sprint prioritization
Analyze and evaluate product backlog
Select sprint goal
Sprint planning
Decide how to achieve sprint goal (design)
Create sprint backlog (tasks) from product backlog items (user stories / features)
Estimate sprint backlog in hours
Sprint goal
Sprint backlog

UNIVERSITETET I OSLO

Sprint Goal – Examples

- A short statement of what the work will be focused on during the sprint

Database Application

Make the application run on SQL Server in addition to Oracle.

Life Sciences

Support features necessary for population genetics studies.

Financial services

Support more technical indicators than company ABC with real-time, streaming data.

UNIVERSITETET I OSLO

Sprint Planning Meeting

- Team selects items from the product backlog they can commit to completing
- Sprint backlog is created
  - Tasks are identified and each is estimated (1-16 hours)
  - Collaboratively, not done alone by the Scrum Master
- High-level design is considered

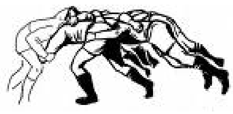
As a vacation planner, I want to see photos of the hotels.

$\rightarrow$

Code the middle tier (8 hours)
Code the user interface (4)
Write test fixtures (4)
Code the foo class (6)
Update performance tests (4)

UNIVERSITETET I OSLO

Daily Scrum



- Parameters
  - Daily
  - 15-minutes
  - Stand-up
- Not for problem solving
  - Whole world is invited
  - Only team members, Scrum Master, product owner, can talk
- Helps avoid other unnecessary meetings

Daily Scrum – 3 Questions

NB:

- These questions are not status reports for the Scrum Master
- They are commitments in front of peers

1. What did you do yesterday?
2. What will you do today?
3. Is anything in your way?

Sprint Review Meeting

- Team presents what it accomplished during the sprint
- Typically takes the form of a demo of new features or underlying architecture
- Informal
  - 2-hour prep time rule
  - No slides
- Whole team participates
- Invite the world

Sprint Retrospective

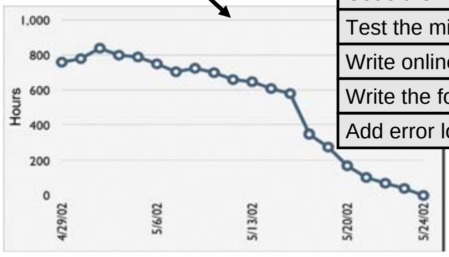
- Periodically take a look at what is and is not working
- Typically 15–30 minutes
- Done after every sprint
- Whole team participates
  - Scrum Master
  - Product Owner
  - Team
  - Possibly customers and others

Scrum – Main Artifacts

- Product backlog
- Sprint backlog
- Burn down chart

Backlog item	Estimate
Allow a guest to make a reservation	3
As a guest, I want to cancel a reservation.	5
As a guest, I want to change the dates of a reservation.	3
As a hotel employee, I can run RevPAR reports (revenue-per-available-room)	8
Improve exception handling	8
	30

Tasks	Mon	Tues	Wed	Thur	Fri
Code the user interface	8	4	8		
Code the middle tier	16	12	10	4	
Test the middle tier	8	16	16	11	8
Write online help	12				
Write the foo class	8	8	8	8	8
Add error logging			8	4	



Scrum – Artifacts

Product backlog

- The product backlog is a high-level document for the entire project. It contains backlog items: broad descriptions of all required features, wish-list items, etc. It is the "What" that will be built. It is open and editable by anyone and contains rough estimates of both business value and development effort. Those estimates help the Product Owner to gauge the timeline and, to a limited extent, priority.
  - For example, if the "add spellcheck" and "add table support" features have the same business value, the one with the smallest development effort will probably have higher priority, because the return-on-investment is higher.
- The product backlog is property of the Product Owner. Business value is set by the Product Owner. Development effort is set by the Team.

Sprint backlog

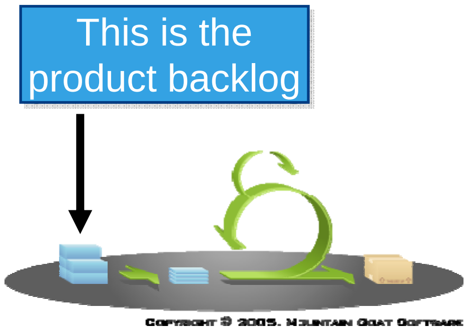
- The sprint backlog is a greatly detailed document containing information about how the team is going to implement the requirements for the upcoming sprint. Tasks are broken down into hours, with no task being more than 16 hours. If a task is greater than 16 hours, it should be broken down further. Tasks on the sprint backlog are never assigned; rather, tasks are signed up for by the team members as they like.
- The sprint backlog is property of the Team. Estimations are set by the Team.

Burn down chart

- The burn down chart is a publicly displayed chart showing remaining work in the sprint backlog. Updated every day, it gives a simple view of the sprint progress.

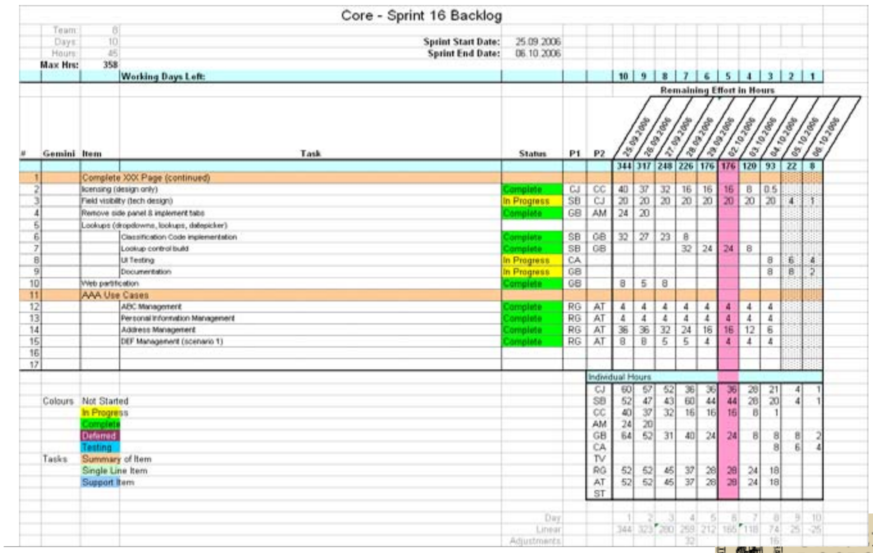
Product Backlog

- The requirements
- A list of all desired work on the project
- Ideally expressed such that each item has value to the users or customers of the product
- Prioritized by the product owner
- Reprioritized at the start of each sprint


This is the product backlog
Copyright © 2005, Mountain Goat Software

UNIVERSITETET I OSLO

Sprint Backlog – Example



Core - Sprint 16 Backlog

Team	8
Days	10
Hours	45
Max Hrs:	358
Sprint Start Date:	25.09.2006
Sprint End Date:	06.10.2006

#	Gemini	Item	Task	Status	P1	P2	25.09.2006	26.09.2006	27.09.2006	28.09.2006	29.09.2006	02.10.2006	03.10.2006	04.10.2006	05.10.2006	06.10.2006
			Working Days Left:				10	9	8	7	6	5	4	3	2	1
							Remaining Effort in Hours									
							344	317	248	226	176	176	120	93	22	8
1		Complete XXX Page (continued)														
2		licensing (design only)		Complete	CJ	CC	40	37	32	16	16	16	8	0.5		
3		Field visibility (tech design)		In Progress	SB	CJ	20	20	20	20	20	20	20	20	4	1
4		Remove side panel & implement tabs		Complete	GB	AM	24	20								
5		Lookups (dropdowns, lookups, datepicker)														
6			Classification Code implementation	Complete	SB	GB	32	27	23	8						
7			Lookup control build	Complete	SB	GB				32	24	24	8			
8			UI Testing	In Progress	CA									8	6	4
9			Documentation	In Progress	GB									8	8	2
10		Web partification		Complete	GB		8	5	8							
11		AAA Use Cases														
12			ABC Management	Complete	RG	AT	4	4	4	4	4	4	4	4		
13			Personal Information Management	Complete	RG	AT	4	4	4	4	4	4	4	4		
14			Address Management	Complete	RG	AT	36	36	32	24	16	16	12	6		
15			DEF Management (scenario 1)	Complete	RG	AT	8	8	5	5	4	4	4	4		
16																
17																

Individual Hours										
CJ	60	57	52	36	36	36	28	21	4	1
SB	52	47	43	60	44	44	28	20	4	1
CC	40	37	32	16	16	16	8	1		
AM	24	20								
GB	64	52	31	40	24	24	8	8	8	2
CA								8	6	4
TV										
RG	52	52	45	37	28	28	24	18		
AT	52	52	45	37	28	28	24	18		
ST										

Colours	Not Started	
	🟨	In Progress
	🟩	Complete
	🟪	Deferred
	🟦	Testing
Tasks	🟧	Summary of Item
	🟩	Single Line Item
	🟦	Support Item

Day	1	2	3	4	5	6	7	8	9	10
Linear	344	323	280	259	212	165	118	74	25	-25
Adjustments				32				16		

UNIVERSITETET I OSLO

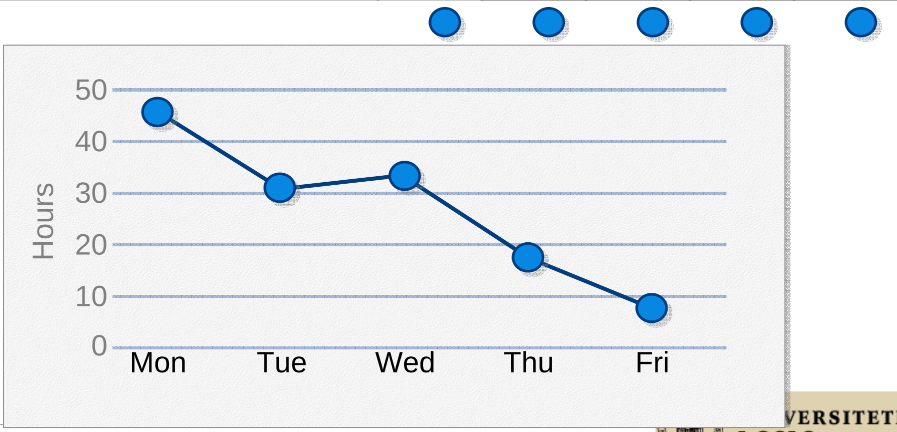
Managing the Sprint Backlog

- Individuals sign up for work of their own choosing
  - Work is never assigned!
- Estimated work remaining is updated daily
- Any team member can add, delete or change the sprint backlog
- Work for the sprint emerges
- If work is unclear, define a sprint backlog item with a larger amount of time and break it down later
- Update work remaining as more becomes known

- Visualisation → Burndown chart (see next slide)

UNIVERSITETET I OSLO

Tasks	Mon	Tues	Wed	Thur	Fri
Code the user interface	8	4	8		
Code the middle tier	16	12	10	7	
Test the middle tier	8	16	16	11	8
Write online help	12				



Scalability of Scrum

Scrum of Scrums of …

- Typical individual team is 7 ± 2 people
  - Scalability comes from teams of teams
- Factors in scaling
  - Type of application
  - Team size
  - Team dispersion
  - Project duration
- Scrum has been used on multiple 500+ person projects (e.g., SAP)

Structure of Lecture 03

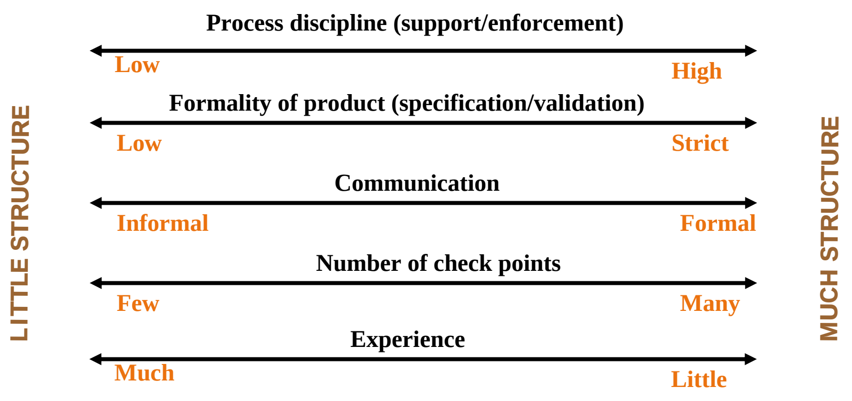
- Hour 1:
  - Light-weight (agile) processes / Evolutionary development
  - XP, Crystal and Scrum
- Hour 2:
  - Scrum (cont'd)
  - Choosing the right process (model) ←
- Hour 3:
  - Homework exercise
  - Question/answer session about project

Choosing a Process Model is Difficult !

- What you should first decide is whether you actually need a prescriptive process model.
- To make the choice it is important to know your organization/project.
  - What characteristics does the project have?
  - What characteristics affect the choice of the process model?
  - Can we use the same model everywhere, or do we need variants (a repertoire of different models)?

UNIVERSITETET I OSLO

How Much Structure is needed/wanted?



UNIVERSITETET I OSLO

How Much Structure is needed/wanted?

LITTLE STRUCTURE

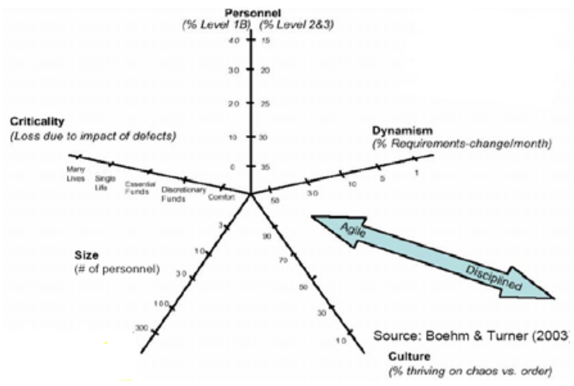
MUCH STRUCTURE

Small	Size (organization)	Big
Small	Size (product)	Big
High	Age (team)	Low
Little	Problem complexity	Big
Little	Demand for precision	Big
Short	Project length	Long

UNIVERSITETET I OSLO

How much Agility is Recommended?

- Source: Boehm, B.; Turner, R.; Observations on balancing discipline and agility, Proceedings of the Agile Development Conference, 2003. ADC 2003. Page(s):32-39


Personnel
(% Level 1B) (% Level 2&3)
Criticality
(Loss due to impact of defects)
Many Lives
Single Life
Essential Funds
Discretionary Funds
Comfort
Dynamism
(% Requirements-change/month)
Size
(# of personnel)
Culture
(% thriving on chaos vs. order)
Agile
Disciplined
Source: Boehm & Turner (2003

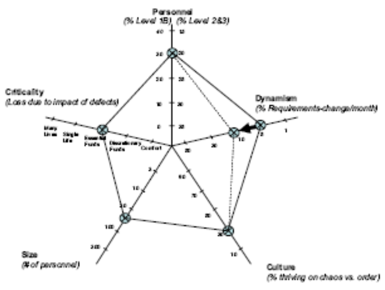

Personnel
(% Level 1B) (% Level 2&3)
Criticality
(Loss due to impact of defects)
Dynamism
(% Requirements-change/month)
Size
(# of personnel)
Culture
(% thriving on chaos vs. order)

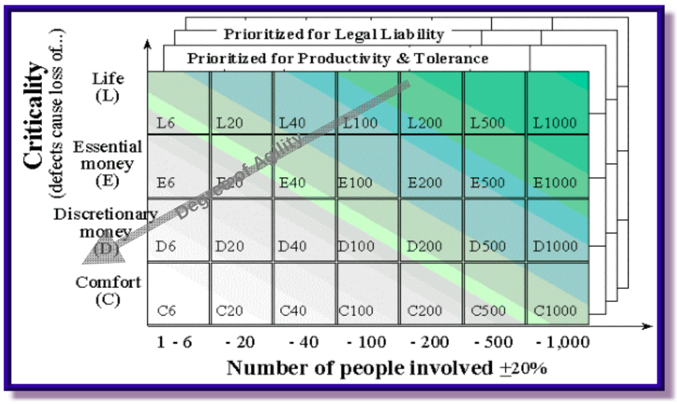
Figure 2. Sample highly-mixed profile

UNIVERSITETET I OSLO

Alistair Cockburn – Project Categorizing

"Any one methodology is likely to be appropriate for only one of the boxes on one of the planes. Thus, at least 150 or so methodologies are needed!"

[Alistair Cockburn: Selecting a Project 's Methodology. IEEE Software 17(4): (2000)]



UNIVERSITETET I OSLO

'Rules of thumb' for selecting process

- IF uncertainty is high
  - THEN use *agile* process
- IF complexity is high but uncertainty is not
  - THEN use *incremental* process
- IF uncertainty and complexity both low
  - THEN use *waterfall* process
- IF schedule is tight/fixed
  - THEN use *agile* or *incremental* approach
- …

UNIVERSITETET I OSLO

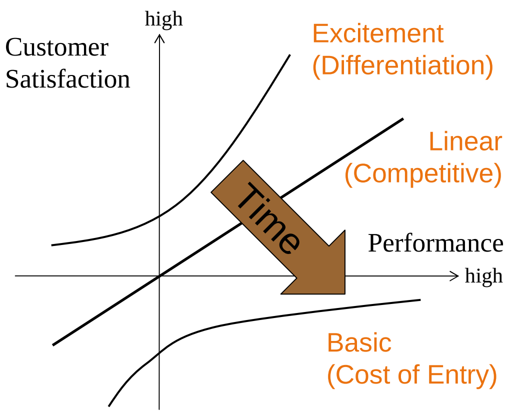
Structure of Lecture 03

- Hour 1:
  - Light-weight (agile) processes / Evolutionary development
  - XP, Crystal and Scrum
- Hour 2:
  - Scrum (cont'd)
  - Choosing the right process (model)
- Hour 3:
  - Homework exercise
  - Question/answer session about project

Homework

- Task:
  - Model the process of surveying "Customer Satisfaction" using the Kano-Model
  - Specify activities, artifacts, roles, tools/techniques/methods
  - Use either the graphical or the table notation

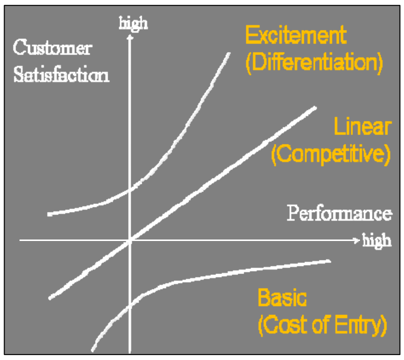
The Kano-Model



Five dimensions of quality:

- ”Basic quality” – satisfies basic “must-have” needs which probably do not even need to be specified.
- ”Competitive quality” - satisfies expressed needs (usually in requirement specification).
- ”Excitement quality” - satisfies latent needs, needs which are there but which the user hasn’t expressed and/or is himself/herself aware of
- ”Indifference quality” - needs which are covered but which user is indifferent to
- ”Reverse quality” - qualities which the customer do not want

The Kano-Model – Surveying Users



- To assess whether a feature is basic, linear, or exciting we can:
  - Sometimes guess
  - Survey a small set of users (20-30)
- We ask two questions:
  - A functional question:
    How do you feel if a feature is present?
  - A dysfunctional question:
    How do you feel if that feature is absent?

Functional and Dysfunctional Forms

Functional form of question → If your editor includes a voice recognition function, how do you feel?

I like it that way	
I expect it to be that way	
I am neutral	
I can live with it that way	
I dislike it that way	

Dysfunctional form of question → If your editor does not include a voice recognition function, how do you feel?

I like it that way	
I expect it to be that way	
I am neutral	
I can live with it that way	
I dislike it that way	

UNIVERSITETET I OSLO

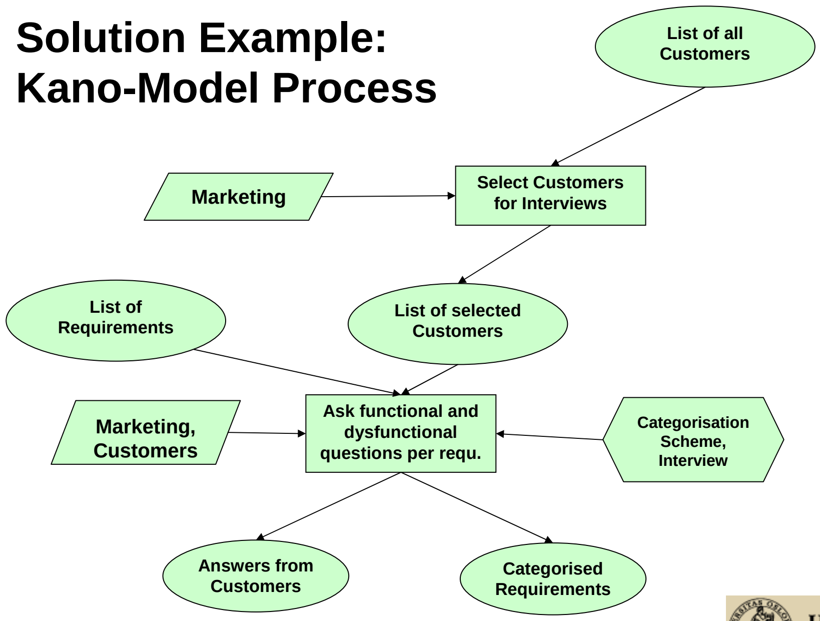
Categorizing an Answer Pair

		Dysfunctional Question				
		Like	Expect	Neutral	Live with	Dislike
Functional Question	Like	Q	E	E	E	L
	Expect	R	I	I	I	B
	Neutral	R	I	I	I	B
	Live with	R	I	I	I	B
	Dislike	R	R	R	R	Q

B: Basic (Mandatory)
L: Linear
E: Excitement
R: Reverse
I: Indifferent
Q: Questionable

UNIVERSITETET I OSLO

Solution Example: Kano-Model Process


List of all Customers
Marketing
Select Customers for Interviews
List of Requirements
List of selected Customers
Marketing, Customers
Ask functional and dysfunctional questions per requ.
Categorisation Scheme, Interview
Answers from Customers
Categorised Requirements

UNIVERSITETET I OSLO

Structure of Lecture 03

- Hour 1:
  - Light-weight (agile) processes / Evolutionary development
  - XP, Crystal and Scrum
- Hour 2:
  - Scrum (cont'd)
  - Choosing the right process (model)
- Hour 3:
  - Homework exercise
  - Question/answer session about project

UNIVERSITETET I OSLO

Project Assignment – Task

Task:

- Prepare a (realistic) software process improvement plan for a software/systems development organization
- A project template with detailed guidelines is available
- The scope of the SPI plan could be (examples):
  - complete process
  - a sub-process of the complete process
  - an activity of a sub-process
  - a method/technique used in an activity
  - …

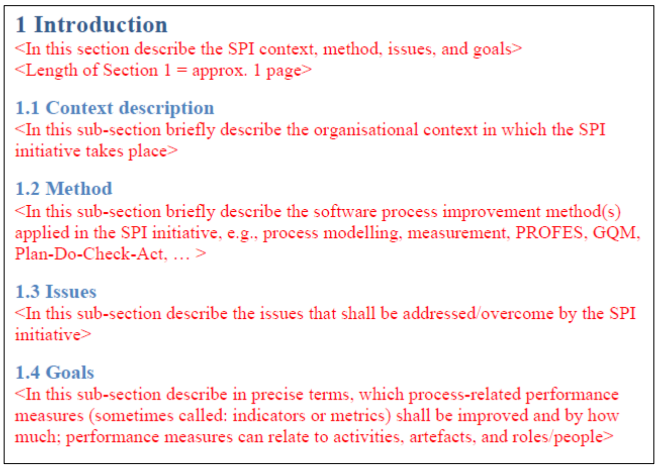
Project Schedule

2 marks penalty If presentation is NOT given

- 06-Oct-2011: **Student Presentation** (5 min, mandatory)
  - Should cover Section 1 of Report Template
- 20-Oct-2011: **Draft Report** (mandatory)
  - Should cover Sections 1 to 3 of Report Template
  - Deliver by email to dietmarp@ifi.uio.no no later than 13:30
  - You will receive feedback (by email) within 2 weeks
- 06-Dec-2011: **Final Report** (mandatory)
  - Should cover all Sections of Report Template
  - Deliver by email to dietmarp@ifi.uio.no no later than 19:59

Project Assignment – Report Template

Introduction


1 Introduction
<In this section describe the SPI context, method, issues, and goals>
<Length of Section 1 = approx. 1 page>

1.1 Context description
<In this sub-section briefly describe the organisational context in which the SPI initiative takes place>

1.2 Method
<In this sub-section briefly describe the software process improvement method(s) applied in the SPI initiative, e.g., process modelling, measurement, PROFES, GQM, Plan-Do-Check-Act, … >

1.3 Issues
<In this sub-section describe the issues that shall be addressed/overcome by the SPI initiative>

1.4 Goals
<In this sub-section describe in precise terms, which process-related performance measures (sometimes called: indicators or metrics) shall be improved and by how much; performance measures can relate to activities, artefacts, and roles/people>

Project Assignment – Topic Ideas

Examples of problems and related improvement goals:

- *Customers find too many defects* – Improve software quality
- *Inaccurate planning / estimates* – Improve planning methods/models
- *New technologies or standards make their way into the market (e.g., model-driven development/testing)* – Adapt existing processes to accommodate the new technology/standard
- *Software is hard to maintain / difficult to evolve* – Improve software architecture
- *Increasing competition* – Speed-up development, issue releases more frequently
- *Customer are dissatisfied with deliveries* – More customer participation and more flexible process
- *"Old-fashioned", heavy development process* – Modernize dev. processes, methods, and tools
- *Little diffusion of competence, low motivation* – Improve training & enhance involvement of people

FIND A REALISTIC APPROACH TO SOLVING A REALISTIC PROBLEM.
MAKE USE Of YOUR IMAGINATION (but choose "probable" problems/goals/solutions).

Next Lecture

- Topic: Flow-based Agile Development (KANBAN)
- **Date: 29 Sep 2011**

- Instructor:
  – Dag Sjøberg

UNIVERSITETET I OSLO