
LUT
University
LAND
OF THE
CURIOUS

LUT University

LECTURE 11

SOFTWARE PROJECT PLANNING

COURSE LECTURE SCHEDULE

Teachers:
Maria
Susan
Sami
Hyrynsalmi

Date	Topic	Book Chapter(s)
Wed 8.9.	Course introduction	
Tue 14.9.	Introduction to Software Engineering	Chapter 1
Tue 21.9.	Software Processes	Chapter 2
Mon 27.9	Agile Software Engineering	Chapter 3
Tue 5.10.	Requirements Engineering	Chapter 4
Mon 11.10.	Architectural Design	Chapter 6
Wed 20.10.	Modeling and implementation	Chapters 5 & 7
Mon 1.11.	Testing & Quality	Chapters 8 & 24
Mon 8.11.	Software Evolution & Configuration Management	Chapters 9 & 25
Mon 15.11.	Software Project Management	Chapter 22
Mon 22.11.	**Software Project Planning**	**Chapter 23**
Mon 29.11.	Global Software Engineering	
Wed 8.12.	Software Business	
Mon 13.12.	Project Presentations	

GOALS FOR THIS LECTURE:

After this lecture, you know

1. Basics of plan-driven & agile planning
2. Effort estimation techniques

SOFTWARE PROJECT PLANNING

PROJECT PLANNING AT THREE STAGES

- Project proposal stage:
  - Bidding -> price
  - Effort cost, hardware & software costs, travel & training
- Start-up phase: plan for the project
  - Resources, who
  - Stages / releases / iterations
- Updating the plan
  - Throughout the project
  - Requirements change, you learn more about the system etc.

PLAN-DRIVEN DEVELOPMENT

- Plan-driven or plan-based development is an approach to software engineering where the development process is planned in detail
  - Plan-driven development is based on engineering project management techniques and is the ‘traditional’ way of managing large software development projects.
- A project plan is created that records the work to be done, who will do it, the development schedule and the work products.
- Managers use the plan to support project decision making and as a way of measuring progress.

PLAN-DRIVEN DEVELOPMENT

- Plan-driven or plan-based development is an approach to software engineering where the development process is planned in detail
  - Plan-driven development is based on engineering project management techniques and is the ‘traditional’ way of managing large software development projects.
- A project plan is created that records the work to be done, who will do it, the development schedule and the work products.
- Managers use the plan to support project decision making and as a way of measuring progress.

What do you see as problems in this approach?

PLAN-DRIVEN DEVELOPMENT – PROS AND CONS

- The arguments in favor of a plan-driven approach are that early planning allows organizational issues (availability of staff, other projects, etc.) to be closely taken into account, and that potential problems and dependencies are discovered before the project starts, rather than once the project is underway.
- The principal argument against plan-driven development is that many early decisions have to be revised because of changes to the environment in which the software is to be developed and used.

PROJECT PLAN

In a plan-driven development project, a project plan sets out the resources available to the project, the work breakdown and a schedule for carrying out the work.

Plan Sections:

1. Introduction: objectives, constraints (e.g. budget, time)
2. Project organization: people, organization, roles
3. Risk analysis: risks, likelihood, reduction strategies
4. Hardware and software resource requirements: what to buy
5. Work breakdown
6. Schedule
7. Monitoring and reporting

THE PLANNING PROCESS

- Project planning is an iterative process that starts when you create an initial project plan during the project startup phase.
- Plan changes are inevitable.
  - As more information about the system and the project team becomes available during the project, you should regularly revise the plan to reflect requirements, schedule and risk changes.
  - Changing business goals also leads to changes in project plans. As business goals change, this could affect all projects, which may then have to be re-planned.

AGILE PLANNING

- Agile methods of software development are iterative approaches where the software is developed and delivered to customers in increments.
- Unlike plan-driven approaches, the functionality of these increments is not planned in advance but is decided during the development.
  - The decision on what to include in an increment depends on progress and on the customer's priorities.
- The customer's priorities and requirements change so it makes sense to have a flexible plan that can accommodate these changes.

AGILE PLANNING

- **Product vision** – overall vision for the product
- **Product roadmap** – displays the goals, direction, and priorities of a product in a visual way (Product Manager)
- **Release plan** - breaks down how and when the organization releases product features or functionalities (Product Manager, Product Owner, Developers)
  - Release planning looks ahead for several months and decides on the features that should be included in a release of a system.
- **Iteration (sprint) planning -** has a shorter term outlook, and focuses on planning the next increment of a system. This is typically 2-4 weeks of work for the team. (Team including PO)
- **Backlog refinement** – reviewing and refining the backlog

MVP – MINIMUM VIABLE PRODUCT

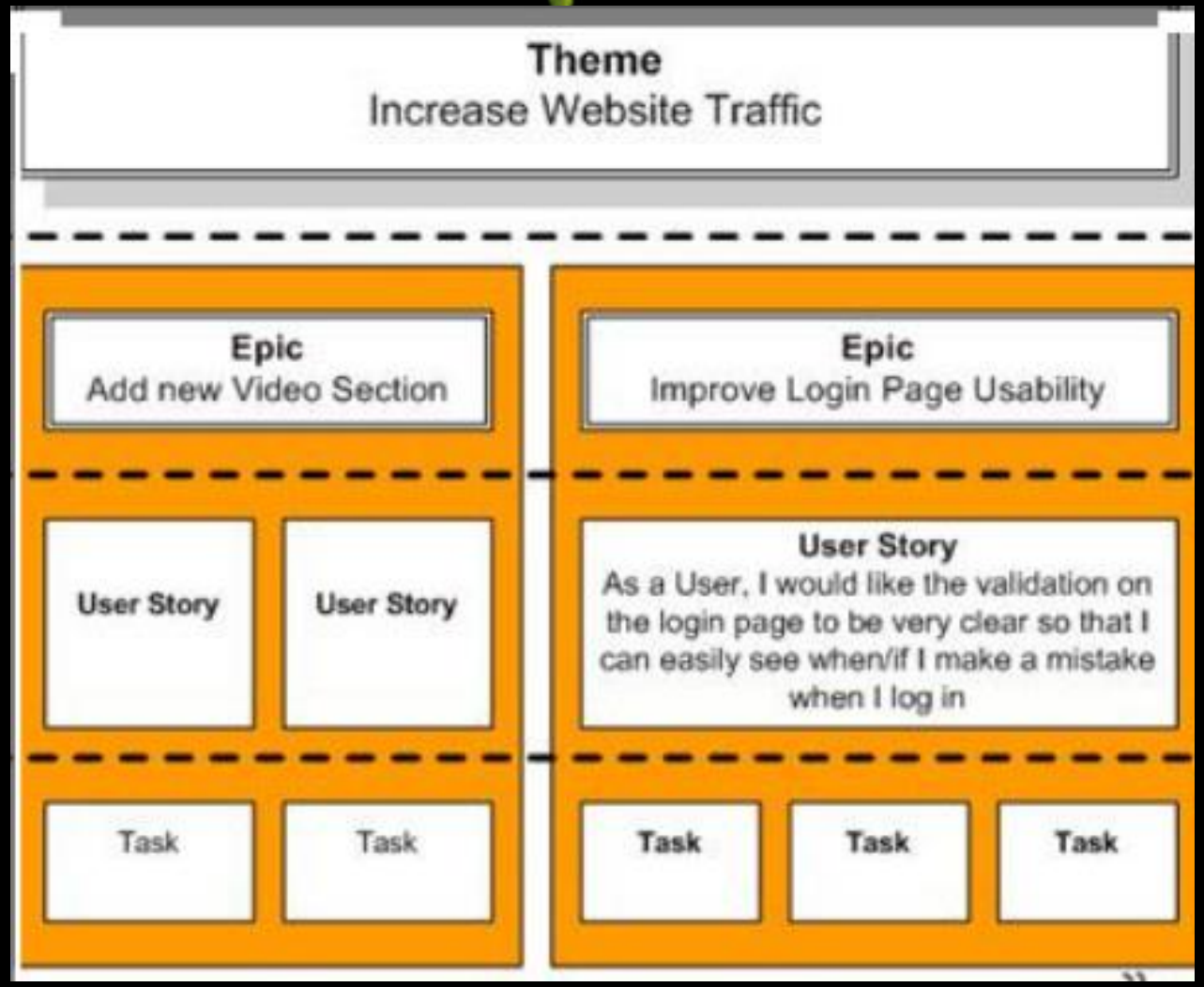
- A product that has just the key functionalities to fix customers' main problems.
- This solution has the potential to assist in validating an innovative suggestion for any product.

AGILE PLANNING: USER STORY

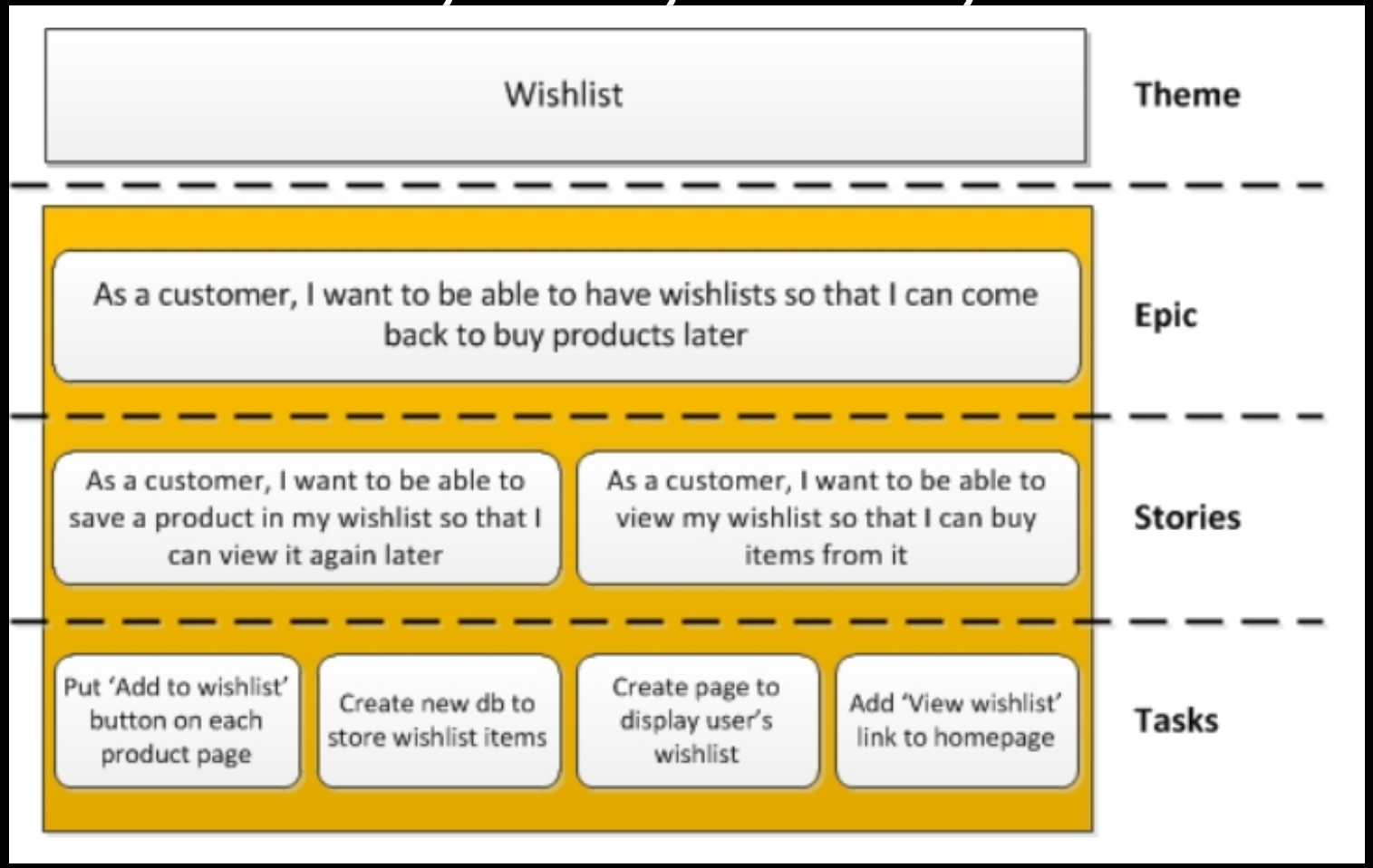
- **Stories are often written in a specific format:**
- **As a <who>, I want <what> so that <why>**
  - **As a** [end user of the required feature]
  - **I want** [actual thing the user wants to be able to do once the feature is live – so often contains a verb]
  - **So that** [why they want this feature / the benefit this feature brings]
- An example story: *"As a customer, I want to be able to save a product in my wishlist so that I can view it again later."*

AGILE PLANNING ITEMS

- **Theme**: a collection of stories
- **Epic**: a requirement that is too big to deliver in a single sprint: “big story”
- User stories are split into **tasks**



EXAMPLE: THEME, EPIC, STORY, TASK



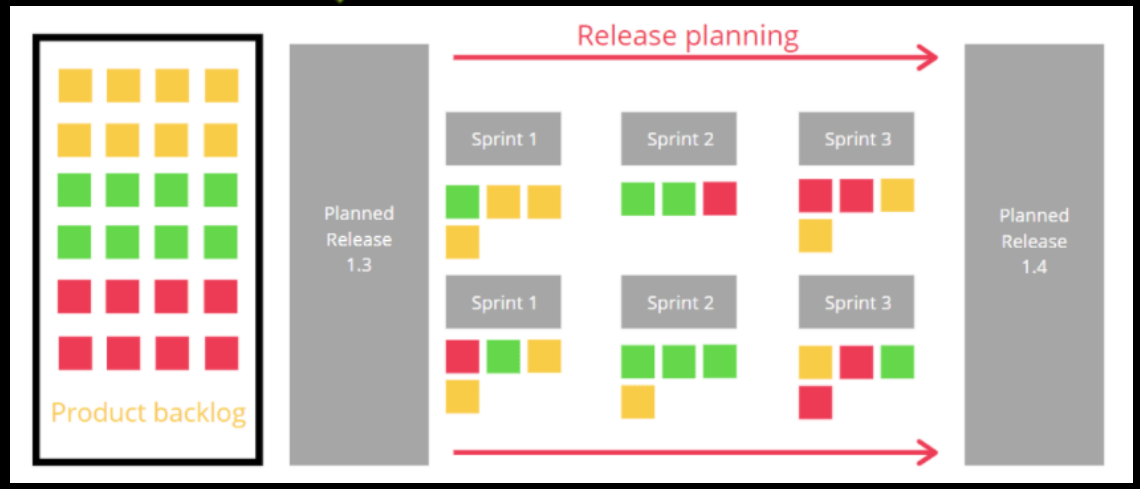
Example by David Lowe

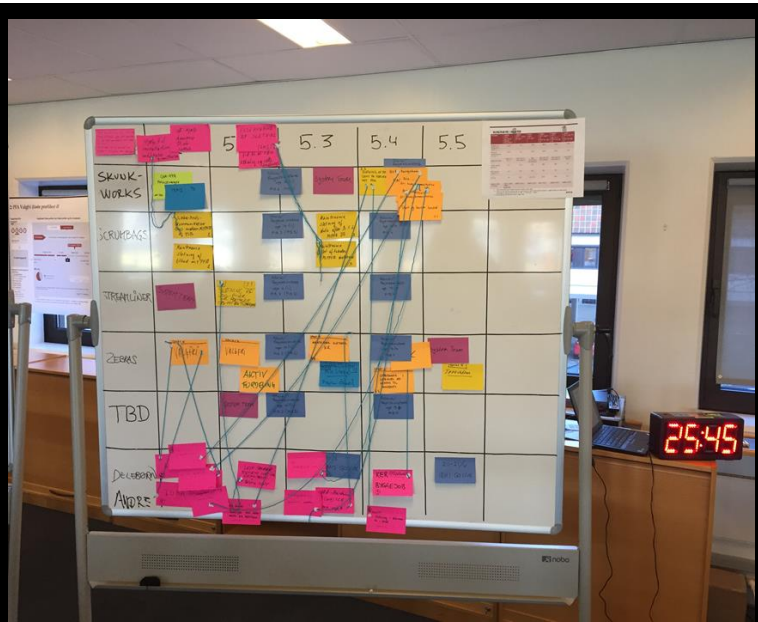
“INVEST” PRINCIPLE FOR CREATING USER STORIES

- **I – Independent**: The story should be independent from other stories so that, should something block this story, the other items in the iteration are not affected (domino effect).
- **N – Negotiable:** The iron triangle (quality, scope, time and resources) sometimes needs to be broken and the business needs to be prepared to negotiate.
- **V – Valuable:** Should provide value to the business
- **E – Estimate-able:** There must be sufficient information available to allow the team to estimate the story.
- **S – Size / Small:** The story should be a good size (i.e. not an epic) to allow the team to be able to complete it within an iteration.
- **T – Testable:** If you’ve got acceptance criteria / Conditions of Satisfaction, then you should be able to test it (note: this is not how to test the story, but what means the story passes).

AGILE RELEASE PLANNING

- Agile release planning is a product management method where you plan incremental releases of a product
- You prepare for staged releases and then break those down into several different sprints or iterations
- Who? Product Managers, Product Owners, teams
- When? E.g. quarterly
- How to do release planning? E.g. 1-2 day session with all stakeholders
- https://www.youtube.com/watch?v=oZ6yA1BS0ns





SPRINT PLANNING

- Attended by the Product Owner, Scrum Master, Developers
- Input: product backlog
  - PO describes the highest priority items, team asks questions
  - Splitting bigger items
- Output:
  - A sprint goal
  - A sprint backlog
- A sprint backlog is a list of the product backlog items the team commits to delivering plus the list of tasks necessary to delivering those product backlog items
  - Each task on the sprint backlog is usually estimated
- https://www.mountaingoatsoftware.com/agile/scrum/meetings/sprint-planning-meeting (3:18 min)

BACKLOG REFINEMENT

- when the Product Owner and some, or all, of the rest of the team review items on the backlog to ensure the backlog contains the appropriate items, that they are prioritized,
- occurs on a regular basis and may be an officially scheduled meeting or an ongoing activity
- Some of the activities during the backlog refinement:
  - removing user stories that no longer appear relevant
  - creating new user stories in response to newly discovered needs
  - re-assessing the relative priority of stories
  - assigning estimates to stories which have yet to receive one
  - correcting estimates in light of newly discovered information
  - splitting user stories which are high priority but too coarse grained to fit in an upcoming iteration

EFFORT ESTIMATION

EFFORT ESTIMATION IN GENERAL

- Important for feasibility, prioritization and scheduling
- Law of diminishing returns holds
- ”Garbage in – garbage out” regardless of the method
- The best effort estimation method depends on context
- Software projects do not scale linearly
- Combining effort estimation methods increases precision more than additional effort with a single method
- pX estimate describes the effort estimate that has X probability to not exceed the (true) effort, e.g. p50 means that there is 50% probability that the (true) effort does not exceed the estimate

EXPERT JUDGEMENT

- They who have the expertice should create the estimates
  - Developer, manager, tester, technical writer, etc.
- Easy to implicitly include detrimental factors (cost-to-win, wishfull thinking, client/customer expectations)
  - Keep planning and bidding separate
- Estimate the accuracy of the estimate in addition to the estimate (pX estimate)
- Provide and record justification for your estimates (for later review and for learning)
- Training may help more than (just) experience

ESTIMATION BY ANALOGY

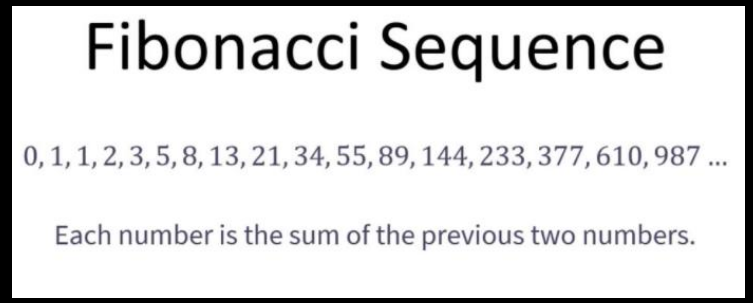
- Systematic form of expert judgement
- Easy to understand intuitively
- Requires that analogous information from previous requirements or projects is available
- Identifying significant characteistics
  - Application domain, number of inputs, number of screens, etc.
  - Indentify the characteristic that most effect the size
- Software tools can help
  - (Automatic) recording of characteristics
  - (Automated) identification of similar requirements/projects

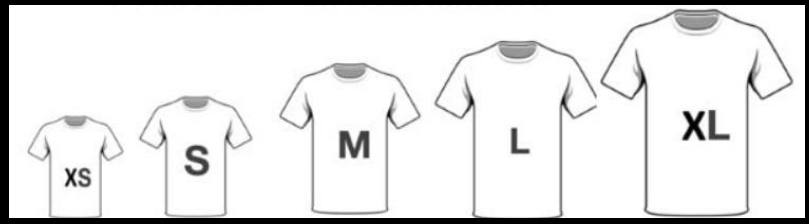
ALGORITHMIC METHODS

- Function point methods quantify the software (requirements) based on a measure specific to the estimation method
- Horde of different kinds of methods exists
  - If you are going to use an algorithmic model, try to find one that fits best in your context and has some empirical validation behind it
  - Consider the architecture (MVC, client-server, layered, etc.)
  - Consider the programming language paradigm (OO, functional, imperative)
- Added complexity may not increase precision
- May be unstable and create false sense of precision

ESTIMATION IN AGILE & PLANNING POKER

- Agile context affects the method selection
  - Requirements definition is lightweight
  - Emphasis on reacting to change instead of precise planning
  - Estimation is a collaborative team-centric endeavour
  - User stories and tasks estimated on different scales
- Planning poker
  - Collaborative way of estimating that combines several approaches
  - Typically produces good-enough (in agile context) estimates
  - Approximate Fibonacci numbers (1, 2, 3, 5, 8, 13) or the exponential scale (1, 2, 4, 8, 12) are the most common scales

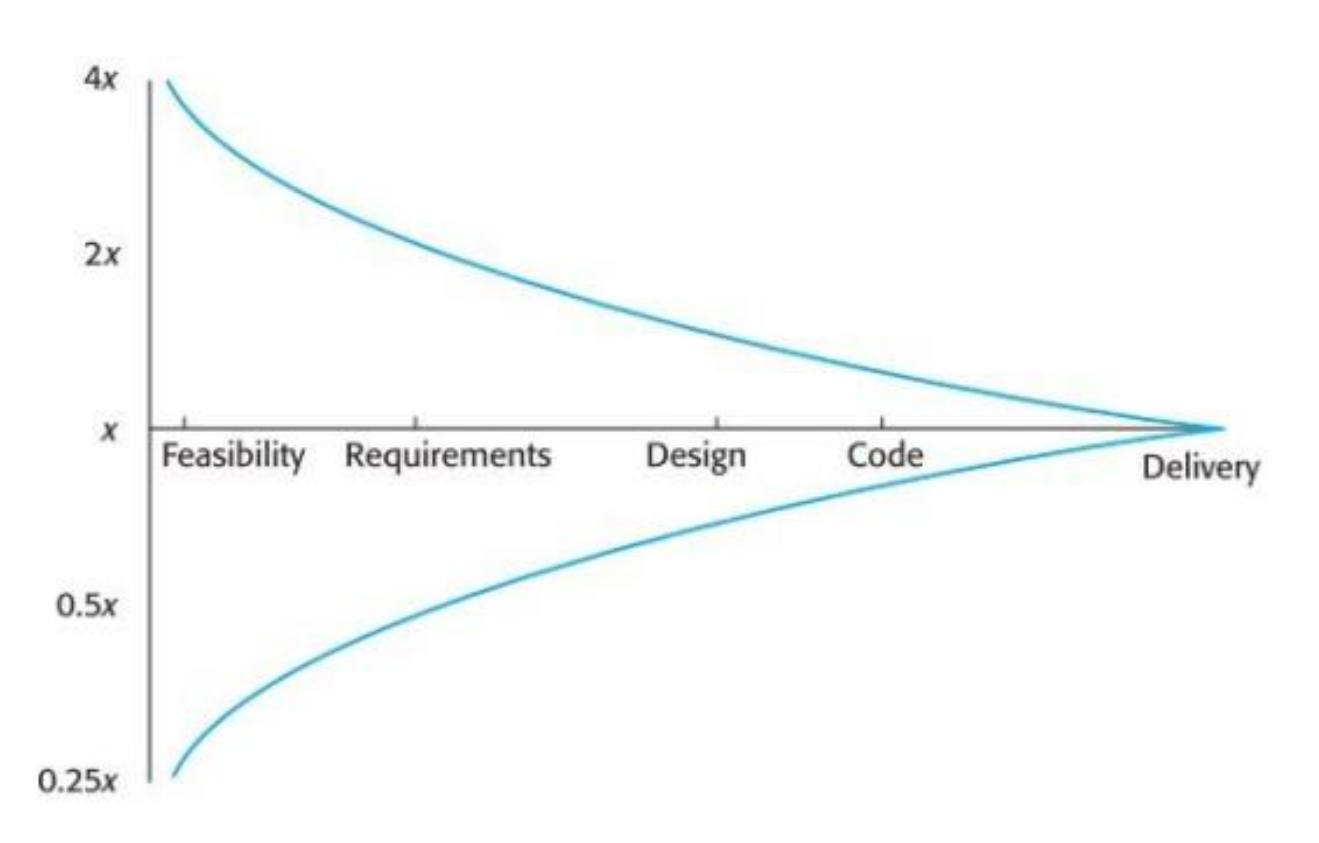




PLANNING POKER VIDEO

- By Mike Cohn, Mountain Goat Software
  - Video (5:31 min) https://www.youtube.com/watch?v=gE7srp2BzoM
  - Short text: https://store.mountaingoatsoftware.com/pages/planning-poker-in-detail

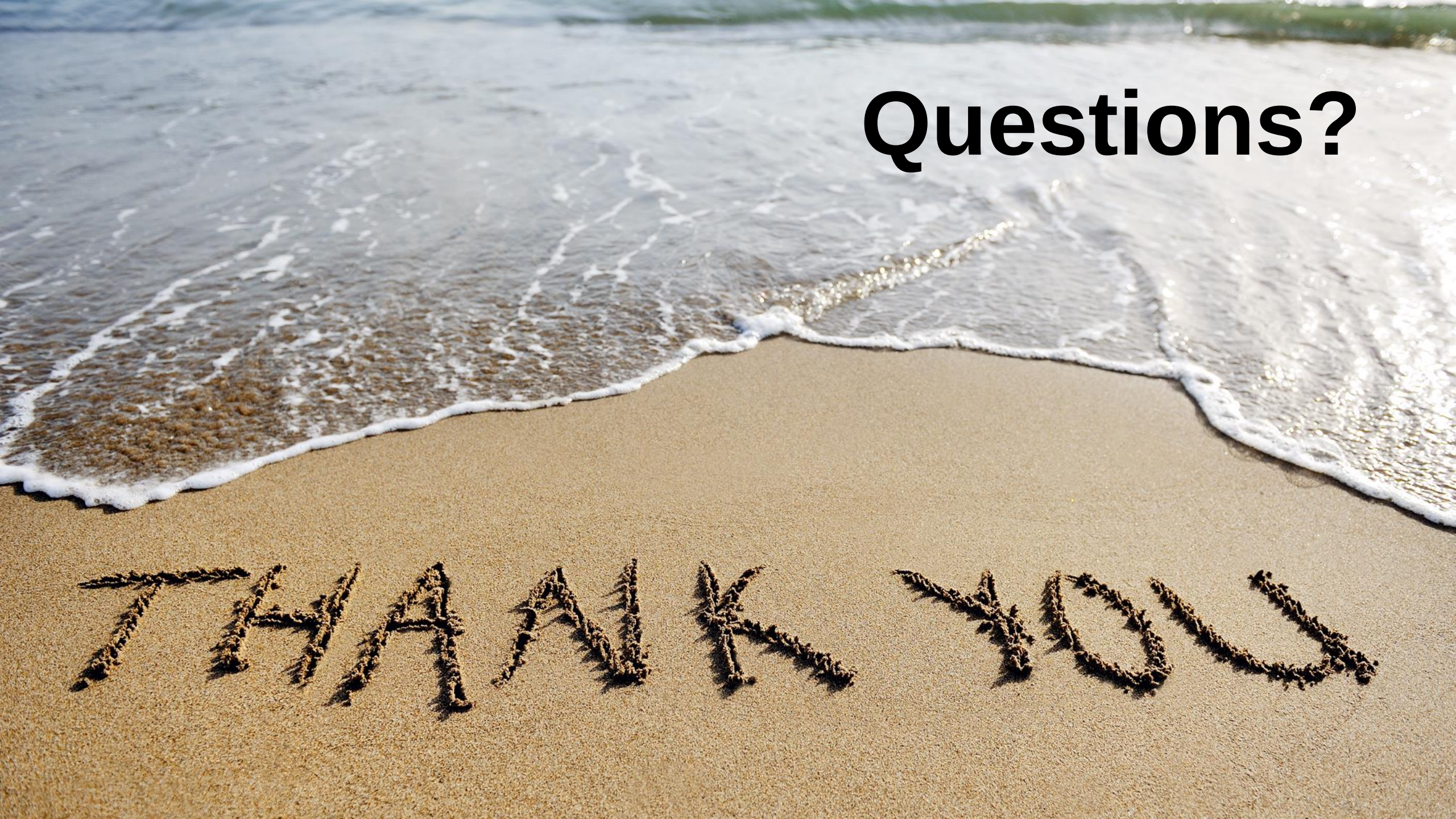
ESTIMATE UNCERTAINTY



Estimate Uncertainty (Sommerville, p.683)

ESSAY – SOFTWARE DEVELOPMENT EFFORT ESTIMATION

- **Discuss software development effort estimation:** Why is effort estimation needed? Why is effort estimation difficult? Compare briefly different software development effort estimation methods (how is estimation done is each, what are the pros and cons). What would you recommend to a new software project manager as a best way to create an effort estimate for a new medium size project? Motivate your recommendation.
- As material use the attached articles, the course book chapter and the planning poker video (and whatever other material you find useful):
  - Chapter 23, Sommerville (course book)
  - Jørgensen, M., 2014. What we do and don't know about software development effort estimation. IEEE software, 31(2), pp.37-40.
  - Mike Cohn, Mountain Goat Software: Planning Poker Estimating in Detail: https://store.mountaingoatsoftware.com/pages/planning-poker-in-detail
  - Mike Cohn, Mountain Goat Software: https://www.youtube.com/watch?v=gE7srp2BzoM
- Length: max 700 words
- Use references


Questions?
THANK YOU


LUT
University