AGILE SOFTWARE ENGINEERING

JOHN STOUBY PERSSON

AALBORG UNIVERSITY
DENMARK

Lecture objectives

Overview of the content and activities in the agile software engineering (ASE) course.

Understand the software engineering problem and that it has fundamentally different answers.

Knowledge of different paradigms for organizing software engineering (process models).

Something about John Stouby Persson

2015 Associate professor, Department of Computer Science

2010 PhD in Computer Science and Engineering (Information Systems)

2006 MSc in Informatics

John's research of information and software technology centers on the problems people experience in practice. He focuses on the problems that are important for professionals such as software engineers, interaction designers, computer scientists, project managers, data scientists, and chief information officers. His problem-based research also aims at improving his problem-based teaching to support reflection in and on professional practice.

He mostly does collaborative practice research of problems in managing the development of information systems with public or private organizations. More specifically, he has collaborated on improving the management of globalized work processes, valuation of information systems, and pragmatic inquiry.

Course objectives

- Provide insight into leading paradigms of agile software engineering, including but not limited to agile processes, risk management, quality management, testing, and DevOps
- Understand the primary software engineering processes, in particular plan-driven, agile, and re-use processes
- Understand the primary tools and techniques and how they support the primary processes
- Develop the capacity to use this understanding to reflect on and improve software development practices
- Establish precise use of terms and concepts
- 20 min oral exam in January

(AGILE)
SOFTWARE ENGINEERING
INTRODUCTION

What's the problem for software engineering

- Delays
- Budget overruns
- Less quality than expected or agreed

- Planning is difficult – and not always possible
- Software quality is not easily assessed

Danish examples of the problem

Skat indrømmer: Videreudvikling af Netcompanys EFI-afløser kan ikke løse centrale problemer

Offentlig it | 18. august kl. 04:11 | 15

Illustration: Nanna Skytte.

Det er ikke forsvarligt at løse store udfordringer i Skat ved at videreudvikle EFI-afløseren PSRM, udtaler Skatteministeriet til Rigsrevisionen. Ministeriets arbejde med at færdigudvikle systemet er utilfredsstillende, fastslår Rigsrevisionen i nyt notat.

Redegørelse om Proask: 283 mio. blev spildt på en forkert vision om automatisering

(Illustration: REDPIXEL.PL/Bigstock)

Proask var det forkerte it-projekt fra begyndelsen, men dårlig projektstyring betød, at fejltagelserne ikke blev opdaget i tide, konkluderer Beskæftigelsesministeriet.

Jesper Stein Sandal @jespersandal Onsdag, 13. maj 2015 - 8:54 10

What do you see as the biggest challenge to successful software engineering?

Types of solutions

- Methods
  - Structured
  - Data-flow
  - Object-oriented
  - Prototyping
  - Business process modelling
  - …
- Processes
  - Waterfall
  - Agile, XP, Scrum, Crystal
  - Mixed, RUP, UP
- Project management
  - Project planning
  - Risk management
  - Quality management
  - Configuration management
  - Test planning
  - Change management
  - Subcontractor management
  - …
- Tools
  - Integrated development environments (language dependent)
  - Testing tools,
  - Versioning and configuration tools, SVN, Git,
  - …
- Training, education, competence, people
  - …

Sommerville's view on software engineering (Chapter 1.1)

Question	Answer
What is software?	Computer programs and associated documentation. Software products may be developed for a particular customer or may be developed for a general market.
What are the attributes of good software?	Good software should deliver the required functionality and performance to the user and should be maintainable, dependable and usable.
What is software engineering?	Software engineering is an engineering discipline that is concerned with all aspects of software production from initial conception to operation and maintenance.
What are the fundamental software engineering activities?	Software specification, software development, software validation and software evolution.
What is the difference between software engineering and computer science?	Computer science focuses on theory and fundamentals; software engineering is concerned with the practicalities of developing and delivering useful software.
What is the difference between software engineering and system engineering?	System engineering is concerned with all aspects of computer-based systems development including hardware, software and process engineering. Software engineering is part of this more general process.
What are the key challenges facing software engineering?	Coping with increasing diversity, demands for reduced delivery times and developing trustworthy software.
What are the costs of software engineering?	Roughly 60% of software costs are development costs, 40% are testing costs. For custom software, evolution costs often exceed development costs.
What are the best software engineering techniques and methods?	While all software projects have to be professionally managed and developed, different techniques are appropriate for different types of system. For example, games should always be developed using a series of prototypes whereas safety critical control systems require a complete and analyzable specification to be developed. There are no methods and techniques that are good for everything.
What differences has the Internet made to software engineering?	Not only has the Internet led to the development of massive, highly distributed, service-based systems, it has also supported the creation of an "app" industry for mobile devices which has changed the economics of software.

Software Engineering Ethics (Chapter 1.2)

You must accept that your job involves wider responsibilities than simply the application of technical skills!

You are lucky! You transform the society through good well paid jobs!

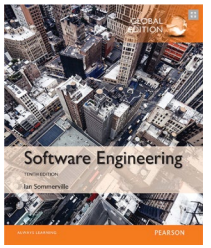
Pay attention to:

- Confidentiality of employers and clients
- Competence – do not knowingly accept work outside your competencies
- Intellectual Property Rights – be aware of copyright and patents
- Computer misuse – do not misuse other people's computer
- Dark patterns – do not accept work, that asks you to trick users to do something they don't want (see video with Martin Fowler)

Sommerville, chapter 2

SOFTWARE PROCESSES (PART 2)

What is a software process



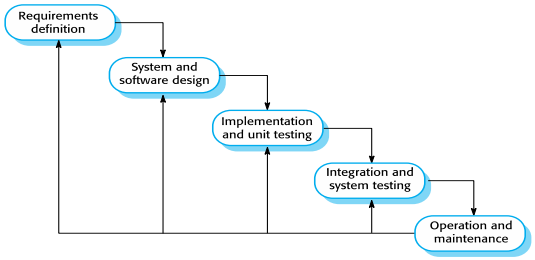
- Software process: a set of related activities required to develop a software system.
- Many different software processes but all involve activities to:
  - Specification – defining what the system should do;
  - Design and implementation – defining the organization of the system and implementing the system;
  - Validation – checking that it does what the customer wants;
  - Evolution – changing the system in response to changing customer needs.
- A software process model is an abstract representation of a process. It presents a description of a process from a particular perspective.

Sommerville's Theory of Activities

1. Specification, where customers and developers defines the properties and requirements of the software product or service
2. Design and implementation, where the developers design and program the software
3. Validation & verification, where it is checked to ensure the product/service conforms to customers' expectations and experience (validation) and specification (verification) through integration and system testing
4. Evolution, where the product/service is modified to reflect changing customer and market requirements

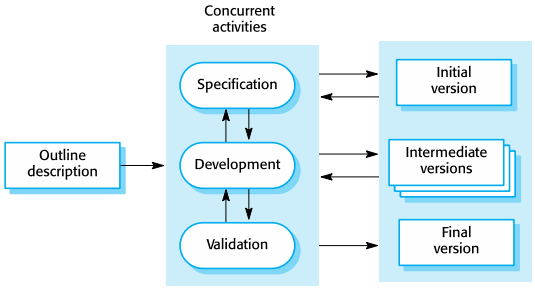
- Customer, i.e., buyers, clients, users, employees, citizens, …

Three main Software Process Models (Chapter 2.1)



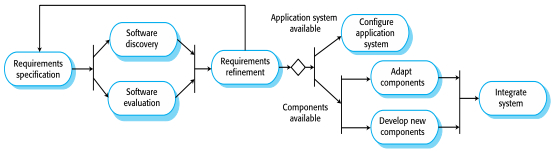
A. The waterfall model

- Plan-driven model. Separate and distinct phases of specification, design, implementation, test and operations.



B. Incremental development

- Specification, development and validation are interleaved. May be plan-driven or agile.

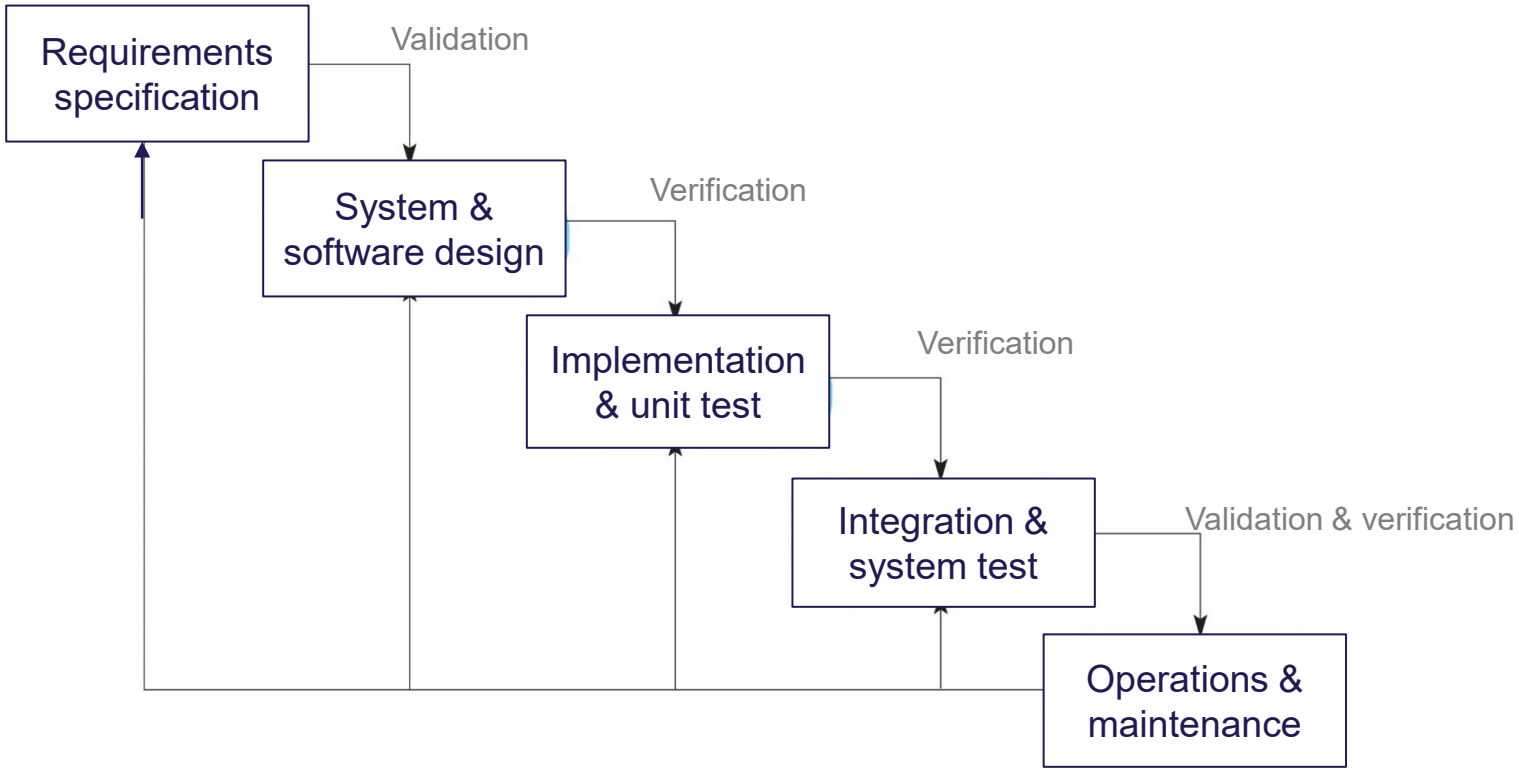


C. Integration and configuration

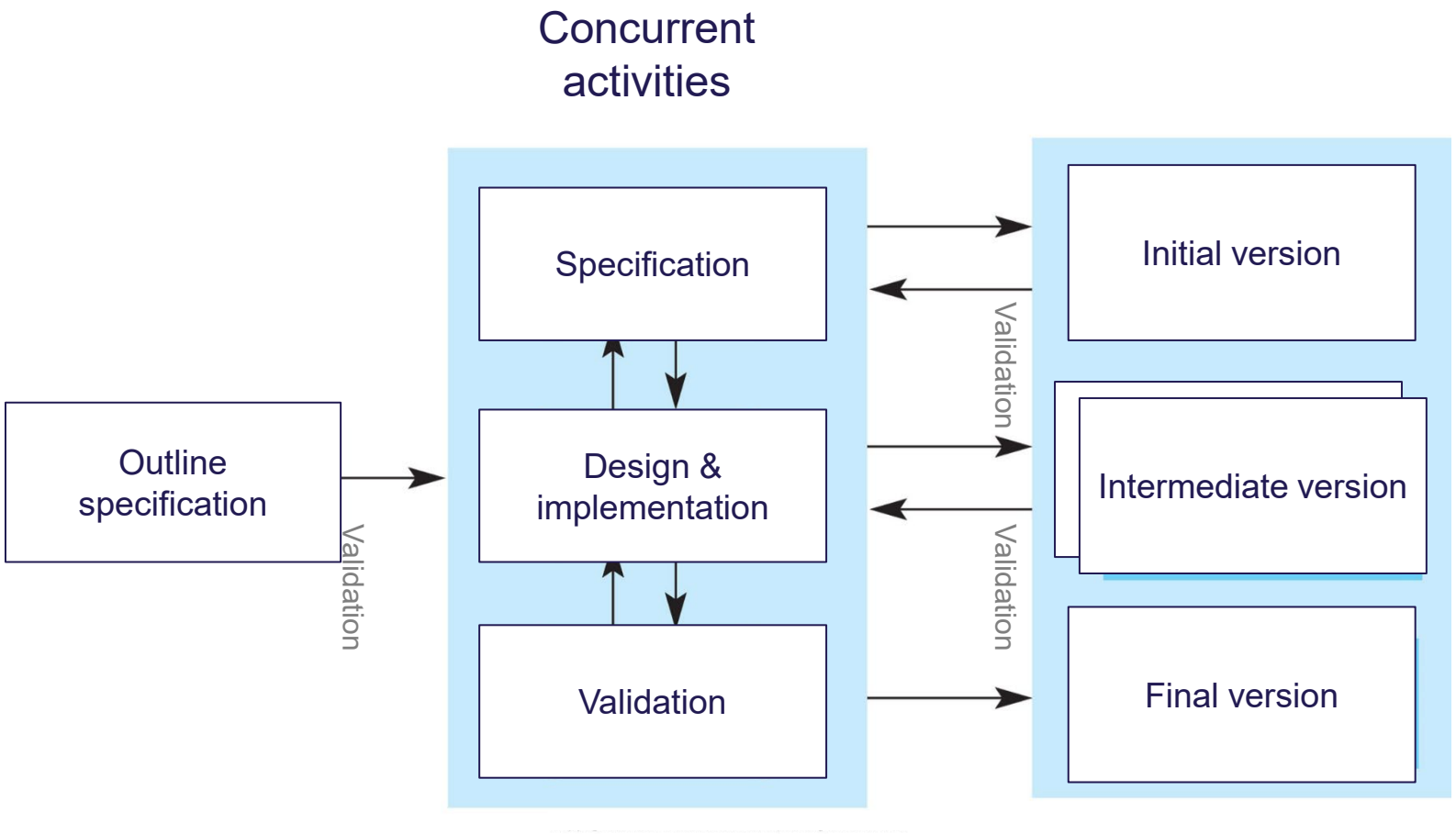
- The system is assembled from existing configurable components. May be plan-driven or agile.

In practice, most large systems are developed using a process that incorporates elements from all of these models.

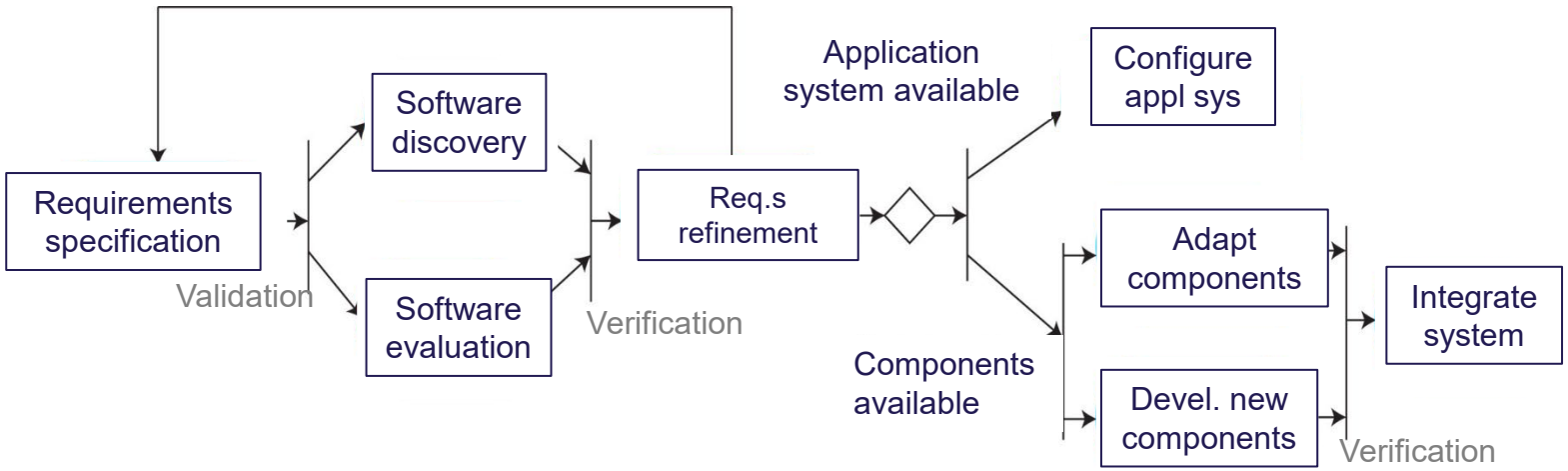
A. Waterfall Model



B. Incremental Model

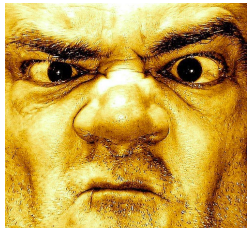


C. Integration & configuration

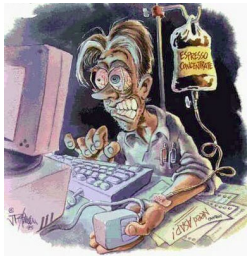


Coping with change and process improvement (Chapter 2.3)

- Change is inevitable in all large software projects.
  - Business changes lead to new and changed system requirements
  - New technologies open up new possibilities for improving implementations
  - Changing platforms require application changes
- Change leads to rework so the costs of change include both rework (e.g. re-analyzing requirements) as well as the costs of implementing new functionality

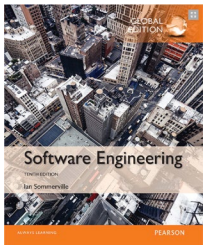


- **3 things we wish were true**
  - **The customer knows what he wants**
  - **The developers know how to build it**
  - **Nothing will change along the way**



- **3 things we have to live with**
  - **The customer discovers what he wants**
  - **The developers discover how to build it**
  - **Many things change along the way**

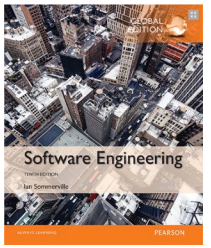
Coping with changing requirements (Chapter 2.3)

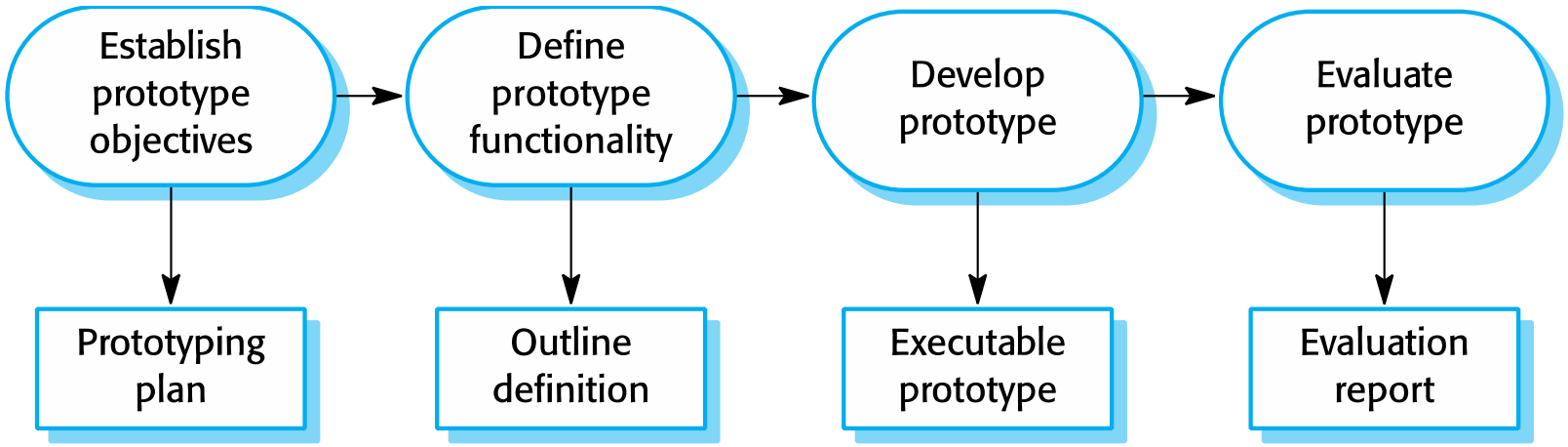


I. System prototyping, where a version of the system or part of the system is developed quickly to check the customer's requirements and the feasibility of design decisions. This approach supports change anticipation.

II. Incremental delivery, where system increments are delivered to the customer for comment and experimentation. This supports both change avoidance and change tolerance.

I. The process of prototype development (Chapter 2.3)



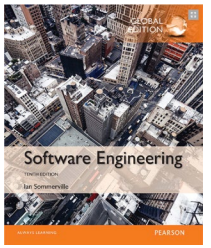


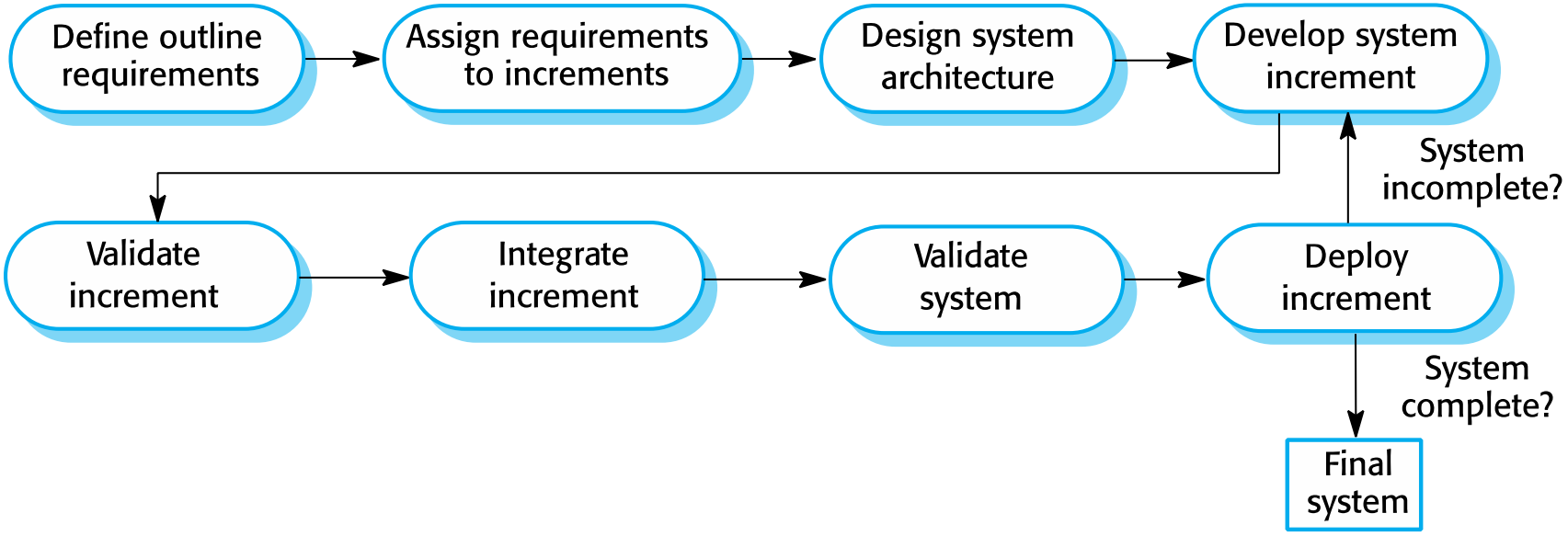
Types of prototypes can be:

- Horizontal (e.g. UI)
- Vertical (e.g. slice of functionality)

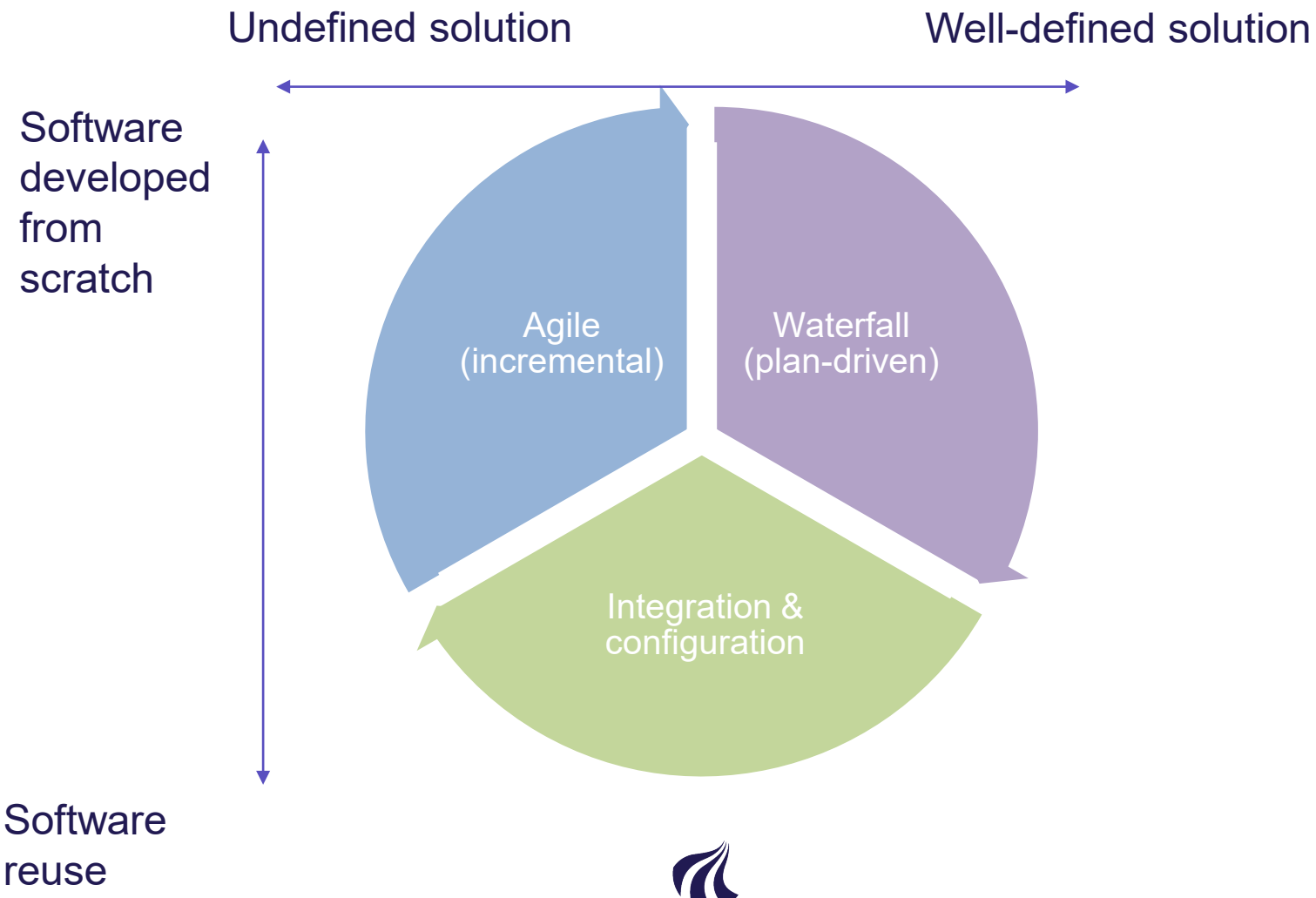
AALBORG UNIVERSITY
DENMARK

II. Incremental delivery (Chapter 2.3)



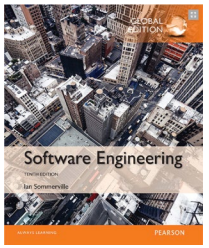


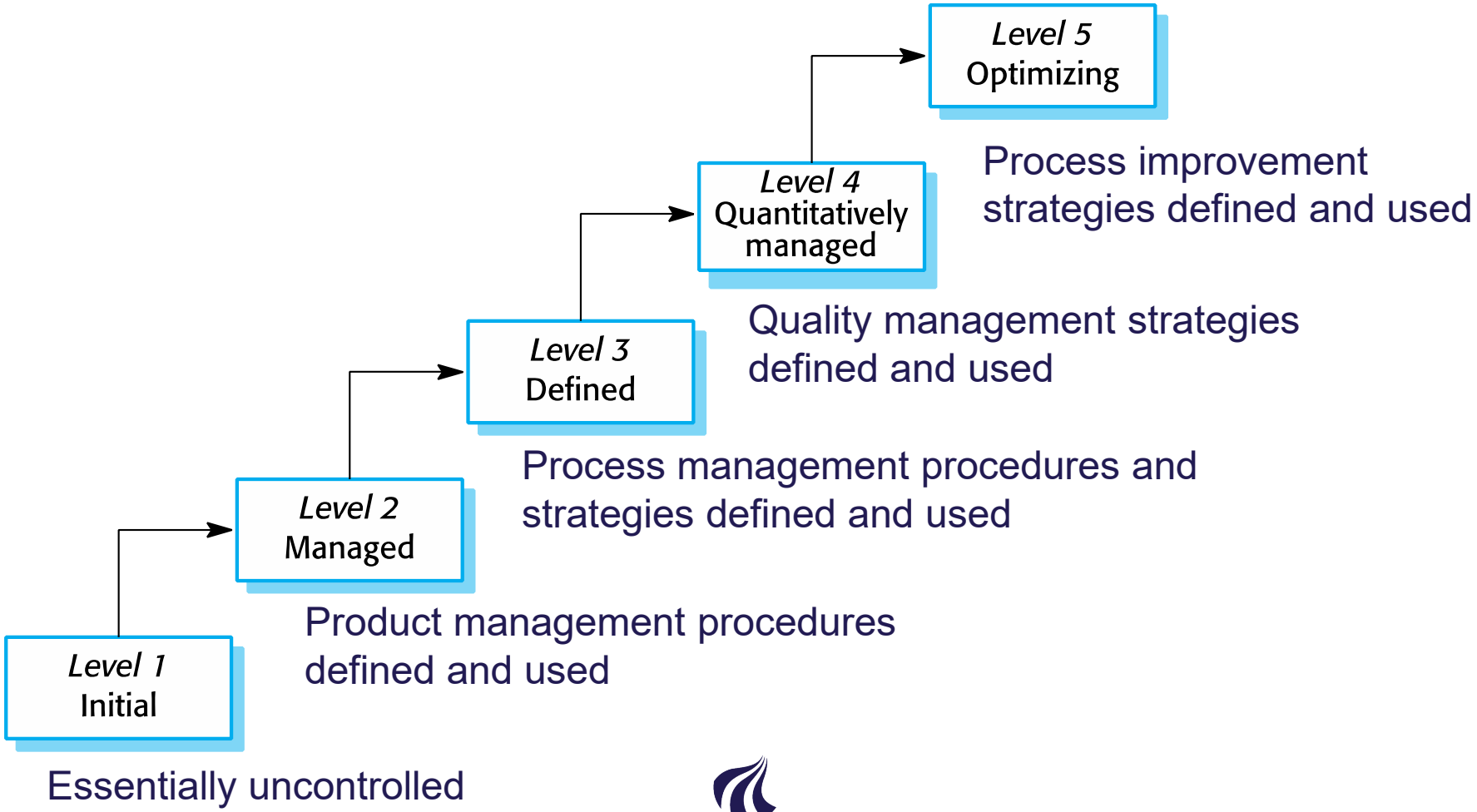
Which model in which situation?



Process improvement: Capability maturity levels

(Chapter 2.4)





Group exercises (10:15-12:00)

Use the document on Moodle (Lecture 01 Group Exercises).

Assignment 1: Exploration of problems (20%)

- Discuss 2-3 cases of IT projects that you know either directly or from the news.

Assignment 2: Differences between main processes (40%)

- Benefits and drawbacks of the three main processes described in Sommerville ch. 2.
- Their implementation of the four generic activities: Specification, Design and Implementation, Validation, and Operation

Assignment 3: In your project (40%)

- What will our main process be (waterfall, incremental, integration & configuration, or a mix)?
- Why?

Help on-demand: send an email with {ASE and room number} in the subject line to the lecturer John S. Persson john@cs.aau.dk
or teaching assistant Jonas C. Lindberg jlindb18@student.aau.dk

Lecture objectives

Overview of the content and activities in the agile software engineering (ASE) course.

Understand the software engineering problem and that it has fundamentally different answers.

Knowledge of different paradigms for organizing software engineering (process models).