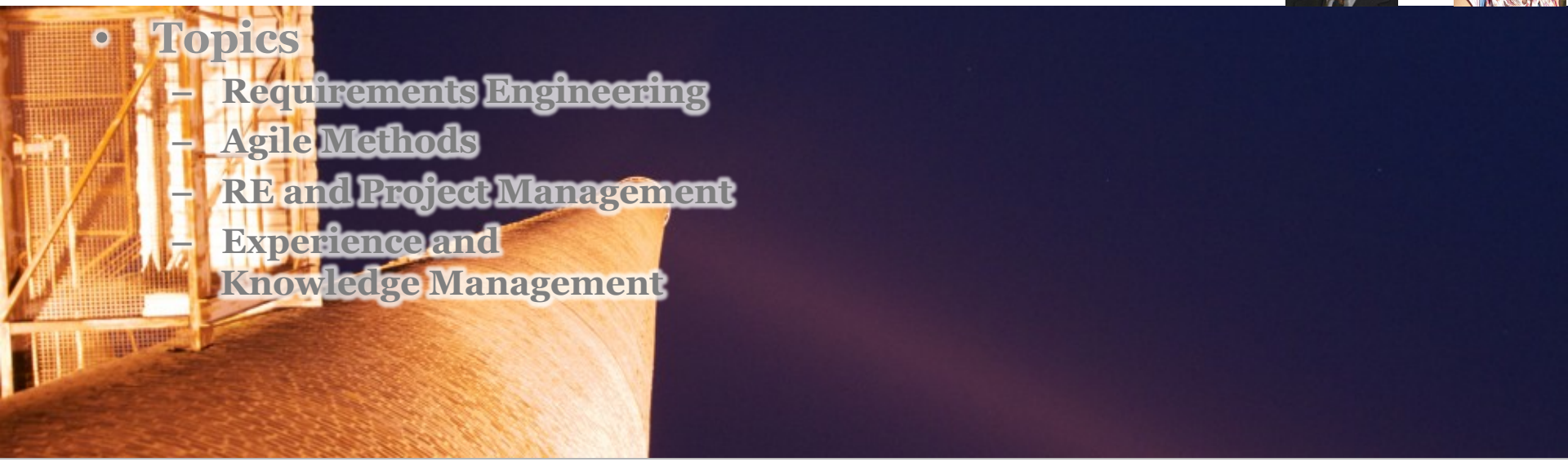
Agile Software Development (DIT191 / EDA397)

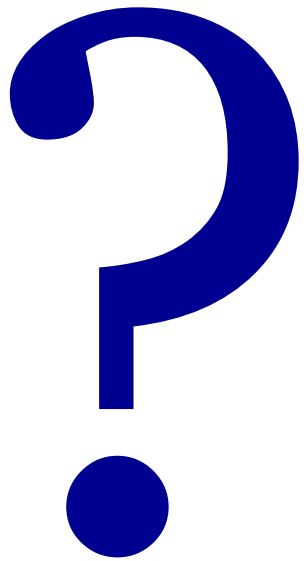
Eric Knauss
<eric.knauss@cse.gu.se>

Hi, nice meeting you! (Short CV)

- Born in Hattingen, Germany
- University in Dortmund and Hanover
- PhD in Hanover: **Improving Requirements Documentation based on Heuristics and Experience**
- Postdoc in Hanover: **Global Software Development**
- Postdoc in Victoria BC: **RE in Distributed Large-Scale Software Projects**



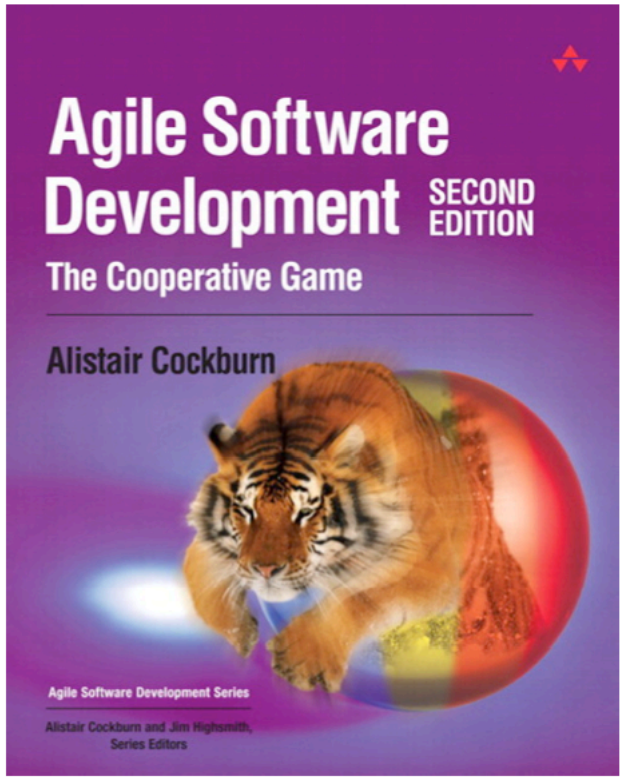
- **Topics**
  - **Requirements Engineering**
  - **Agile Methods**
  - **RE and Project Management**
  - **Experience and Knowledge Management**



- What about you?
  - Who are you (name, university, programme)?
  - What would make this course great for you?

Course setup

- Teaching Assistant
  - Emil Alégroth (emil.alegroth@chalmers.se)
- Course representants
- In parallel:
  - PhD course
- Course Material
  - http://oerich.github.io/EDA397/
  - Cockburn,A., (2009) Agile Software Development, 2ed
  - Papers



Course setup (Practical details)

- Schedule
  - 0-3 lectures per week
  - 0-2 workshops per week
  - =3 scheduled activities per week
    - Even if there is no lecture, we will be available and you can (should!) use the rooms/time to work!

- Examination
  - Project (teams)
    - Final product
    - Artifacts
  - Report (Individual)
    - Experience / Post-Mortem report
  - Written exam

Project

- Develop an Android app for a customer
- Work in predetermined teams
  - You will be assigned teams
  - You decide about your development methodology
  - You meet with the customer during course week 2
    - and get all the details!

How to assign teams? V

- We strive to create a realistic scenario/ environment
  - We rely on a number of real-world services and tools, e.g.
  - Android (SDK) GitHub / BitBucket ..

Course setup

- Three sprints
- Sprint 1: Getting started
- Sprint 2: Getting work done
- Sprint 3: Theory and advanced concepts

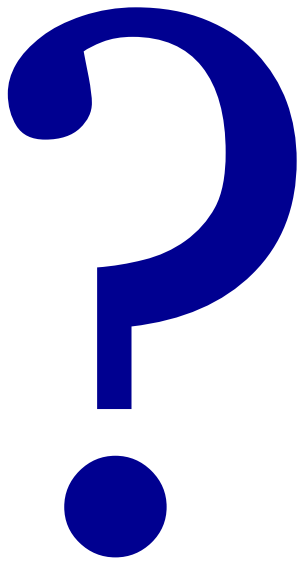
Sprint 1: Getting started

What is agility in Software Development?

Motivation

- What is the “Software crisis”?
  - Software development inefficient
  - Software does not meet requirements
  - Projects over time/budget
  - Projects were unmanageable and software unmaintainable
- What can be done about it?

→ Software Engineering
  - Application of engineering to Software
  - Systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software
  - Assure quality of process and product

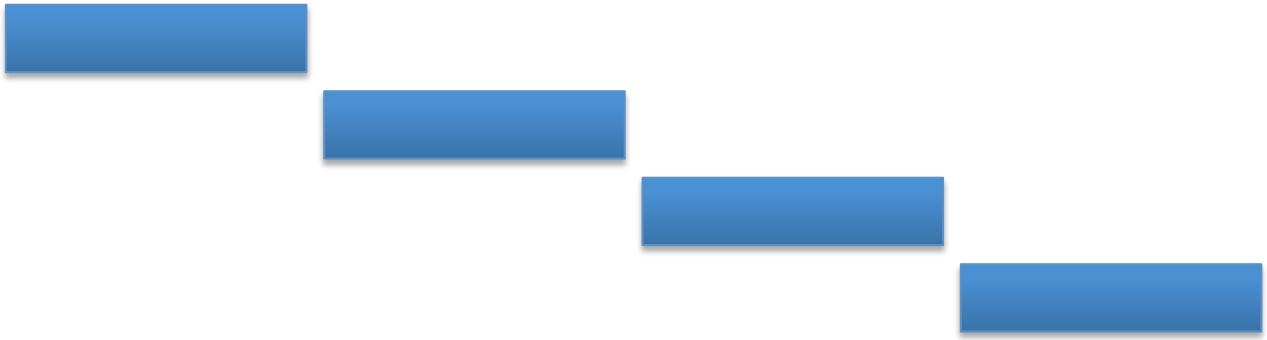


- Do you know examples of
  - Systematic, disciplined, quantifiable approaches to the development, operation, and maintenance of software?
  - Applying engineering to software?

Where are we coming from?

Systematic sequential development

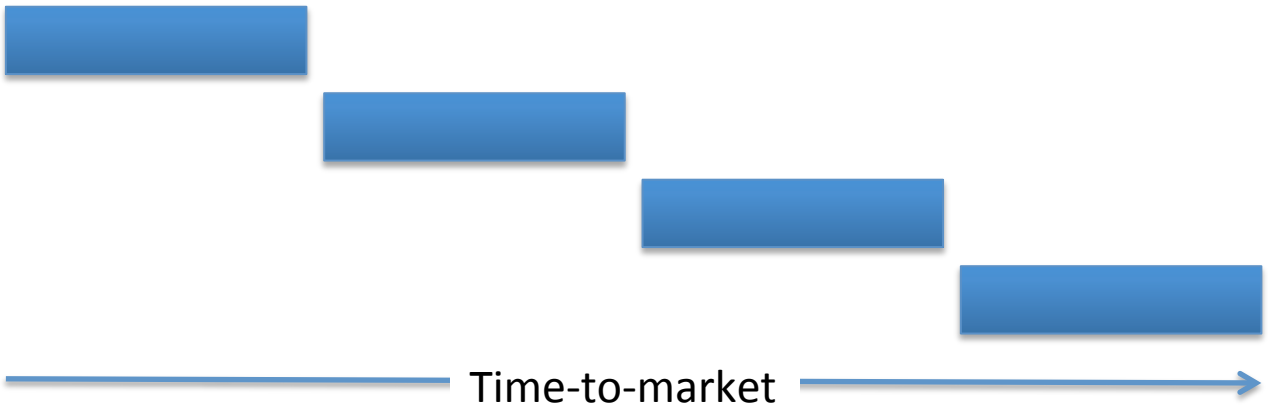
- Requirements
- Design
- Programming
- Test



- Advantages
  - Simple
  - Controllable
  - Cost efficient
- Problems
  - Time-to-market
  - What about change?

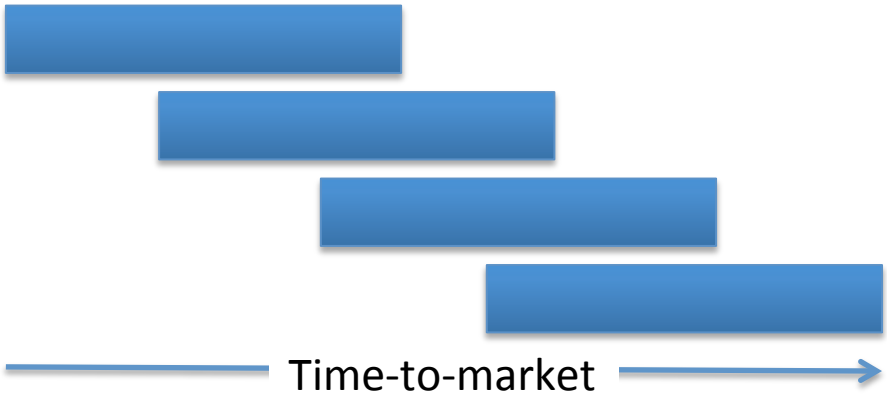
Towards concurrent development

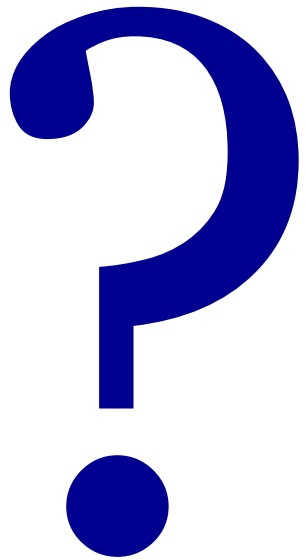
- Requirements
- Design
- Programming
- Test



What can we do if time to market and robustness against late changes are more important than cost-efficiency?

- Requirements
- Design
- Programming
- Test



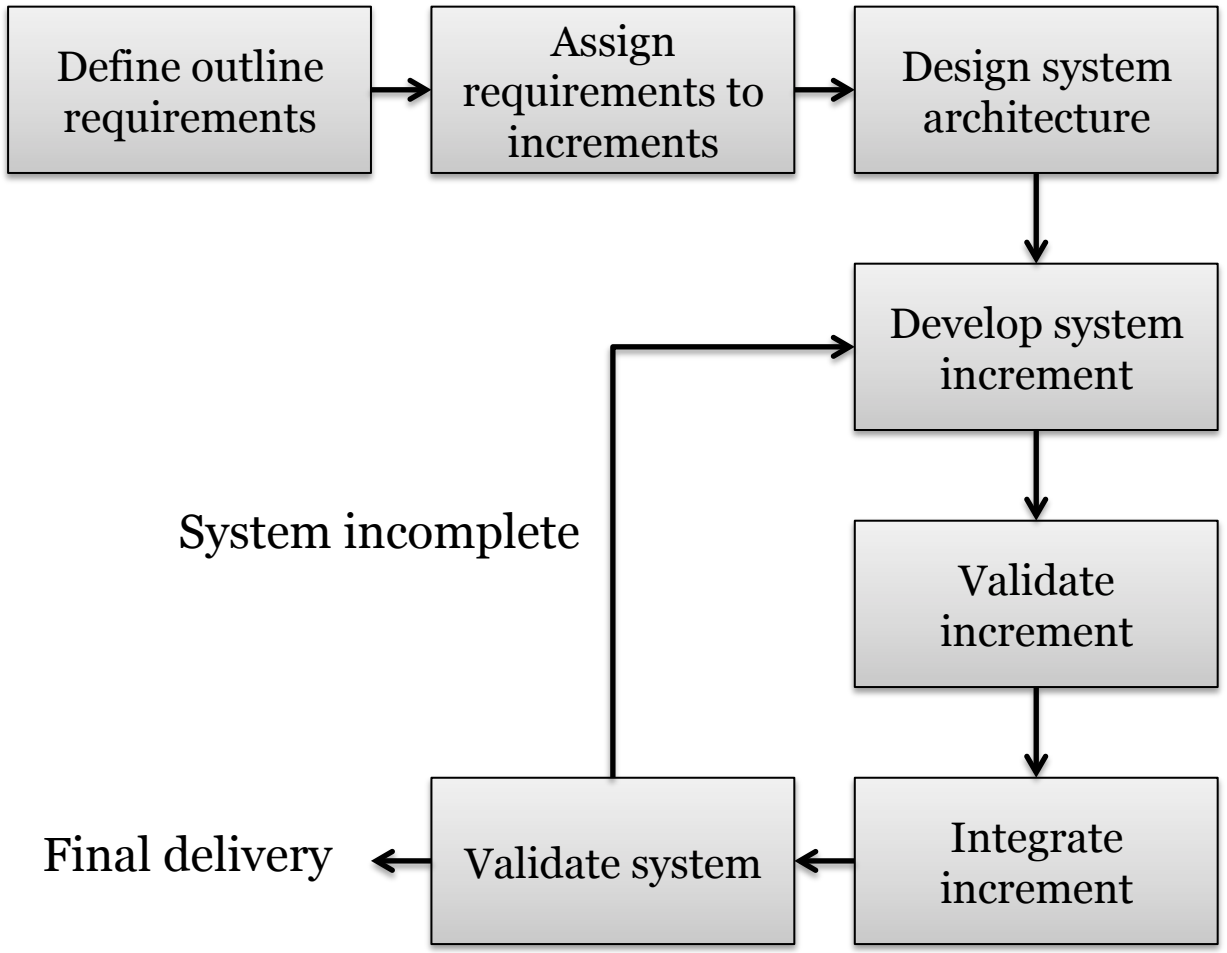


- What is the consequence of concurrent development?
  - (hint: why are concurrent tasks depicted longer than sequential tasks?)
- What has this to do with agile?
- What do you know about agile?

Iteration Models

- System requirements *always* evolve during a project
- Iterations are part of larger development projects
- Iterations can be applied to any generic development process model

Incremental delivery



Incremental delivery

- Customer value can be delivered with each increment so system functionality is early available for customer's feedback
- Early increments act as prototype to help elicit requirements for later increments
- Reduced risk of project failure
- The highest priority system services tend to receive the most testing