Softwareudvikling 2017
software development

Introduction

Boris Düdder, Datalogisk Institut
6.2.2017

UNIVERSITY OF COPENHAGEN

Agenda

- **Organization**
- **Competencies**
- **Lecture**
- **Tutorials**
- **Exercises**
- **Exam**

- Subject area
- Lecture contents

Team

- Prof. Fritz Henglein, Kursusansvarlig
- Prof. Erik Jul, Underviser
- Boris Düdder, Underviser
- Oleks Shturmov, Koordinerende instruktor
- Kristian Fogh Nissen, Instruktor
- Mads Ulrik Svendsen, Instruktor
- Alexander Christensen, Instruktor
- Line Hagenow, Instruktor
- Benjamin Rotendal, Instruktor
- Sven Frenzel, Instruktor

Learning Goals and Competencies

- Students can design and develop non-trivial programs using an engineering approach.
- Students can write a report on the project development.

Learning Contract: We offer …

- Technical introduction to software development as part of software engineering.
  - Focus: object-oriented software systems
- Committed support:
  - Interesting lecture.
  - Regular consultation hours.
  - Supervised exercises.
  - Fast feedback.
  - Transparent requirements.
  - Opportunities for direct feedback.

Learning Contract: We expect …

Active looking into lecture contents:

- Active participation in the lecture.
- Preparation and follow-up of the lecture.
- Active participation in the exercises.
  - Programming as a craft must be practiced.
- Coping with exercises and additional materials.

Course Schedule (Block 3+4)

- Monday
  - 8-10 lectures and tutorials on demand
  - 10-12 group meetings
- Tuesday
  - 13-16 regular lectures
- Friday
  - 9-12 exercises with instructors

Tutorials

- Language in tutorials: English / Danish
- Mondays, 8:00-10:00
- Lectures:
  - Erik Jul
  - Boris Düdder
- Room
  - aud - Store UP1 - 5-1-02, Universitetsparken 1-3, DIKU

Lectures

- Language in lecture: English / Danish
- Tuesdays, 13:00-16:00
- Lectures:
  - Erik Jul
  - Boris Düdder
- Room
  - aud - Store UP1 - 5-1-02, Universitetsparken 1-3, DIKU

Meetings

- Language in meetings: English / Danish …
- Mondays, 10:00-12:00
- With instructors and lecturers
- Rooms
  - øv - 1-0-26, Universitetsparken 1-3, DIKU
  - øv - 1-0-30, Universitetsparken 1-3, DIKU
  - øv - 1-0-18, Universitetsparken 1-3, DIKU
  - øv - 1-0-22, Universitetsparken 1-3, DIKU
  - øv - 1-0-10, Universitetsparken 1-3, DIKU
  - øv - 1-0-34, Universitetsparken 1-3, DIKU

Meetings / Exercises

- Language in meetings: English / Danish …
- Fridays, 9:00-12:00
- With instructors
- Rooms
  - øv - 1-0-26, Universitetsparken 1-3, DIKU
  - øv - 1-0-30, Universitetsparken 1-3, DIKU
  - øv - 1-0-18, Universitetsparken 1-3, DIKU
  - øv - 1-0-22, Universitetsparken 1-3, DIKU
  - øv - 1-0-10, Universitetsparken 1-3, DIKU
  - øv - 1-0-04, Universitetsparken 1-3, DIKU

Room change 04 instead of 34!

Meetings (One Group)

- Language in meetings: English / Danish …
- Tuesday, 10:00-12:00
- Thursday, 9:00-12:00
- With instructor
- Rooms
  - øv - 1-0-14, Universitetsparken 1-3, DIKU

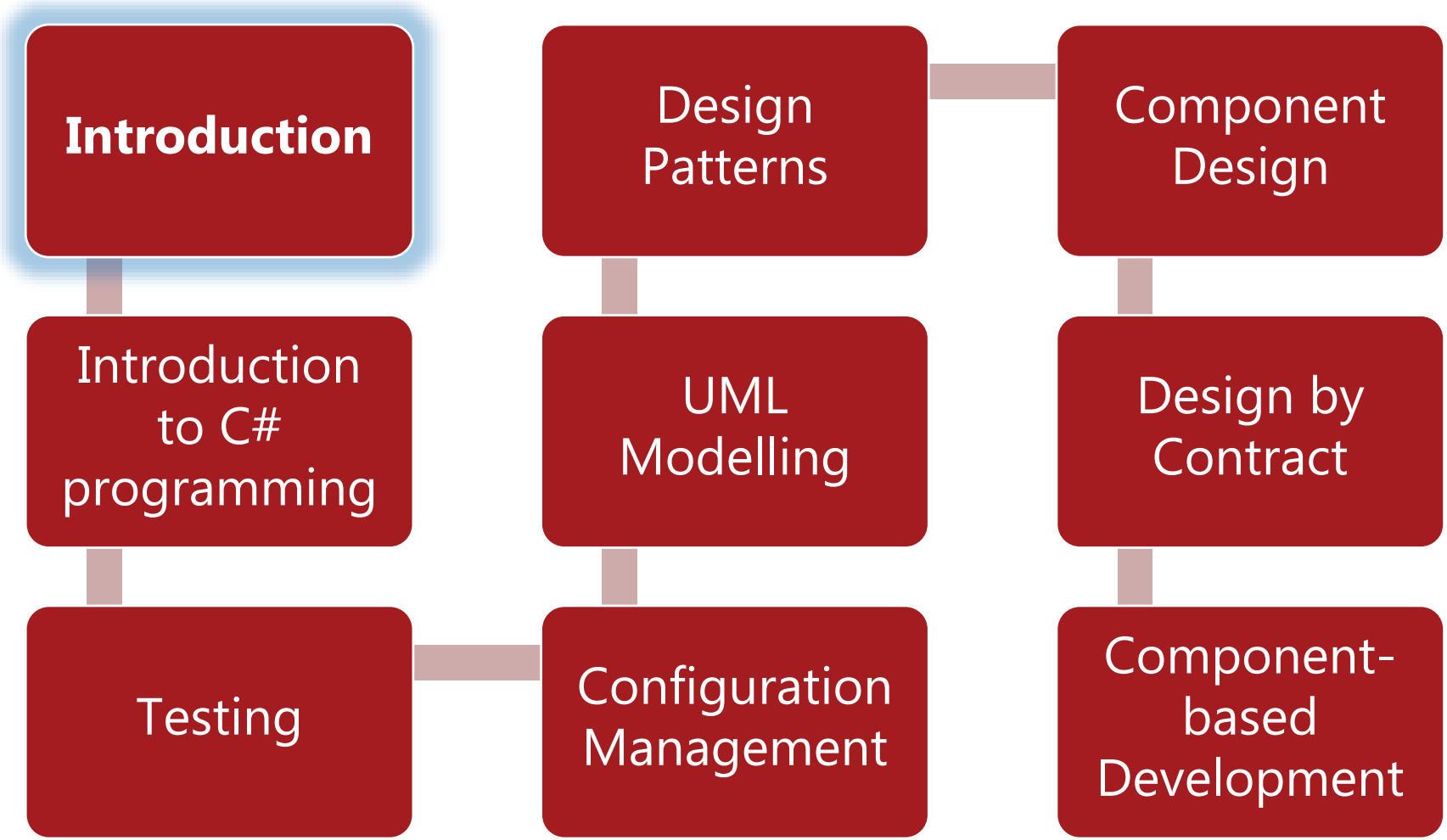
Exam

- Submitted final report on group project.
- Oral exam of 20 minutes after block 4.

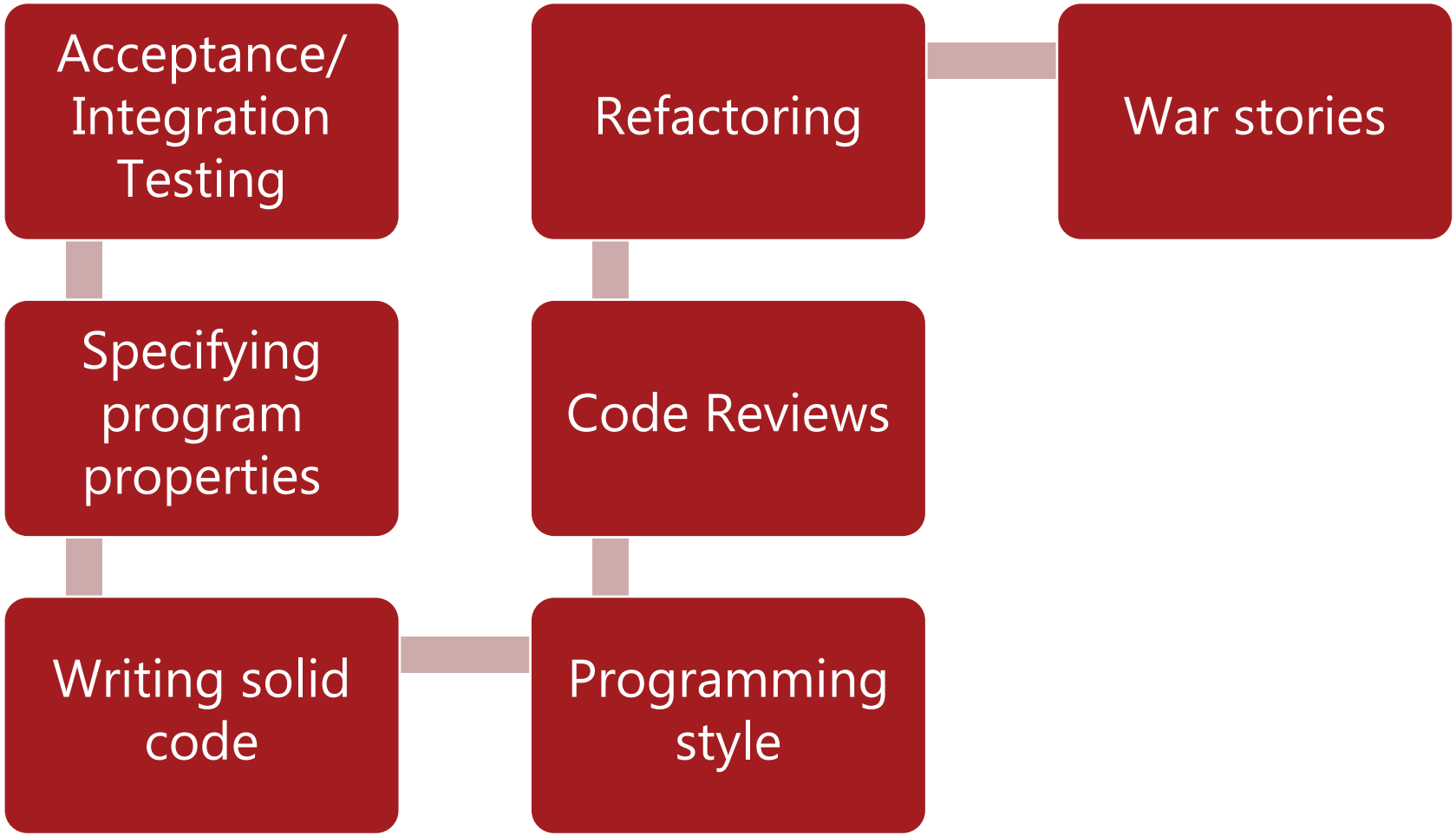
Proposal for a General Student Timeplan SU17

Day and Time	Activity
Monday, 8-10	Attend lecture and tutorial
Monday, 10-12	Attend group meeting
Tuesday, 13-16	Attend lecture
Tuesday	Work on the proposed literature
Thursday	Make homework and prepare exercises
Friday, 9-12	Participate in exercises
Friday, 15:00	Submission of assignments

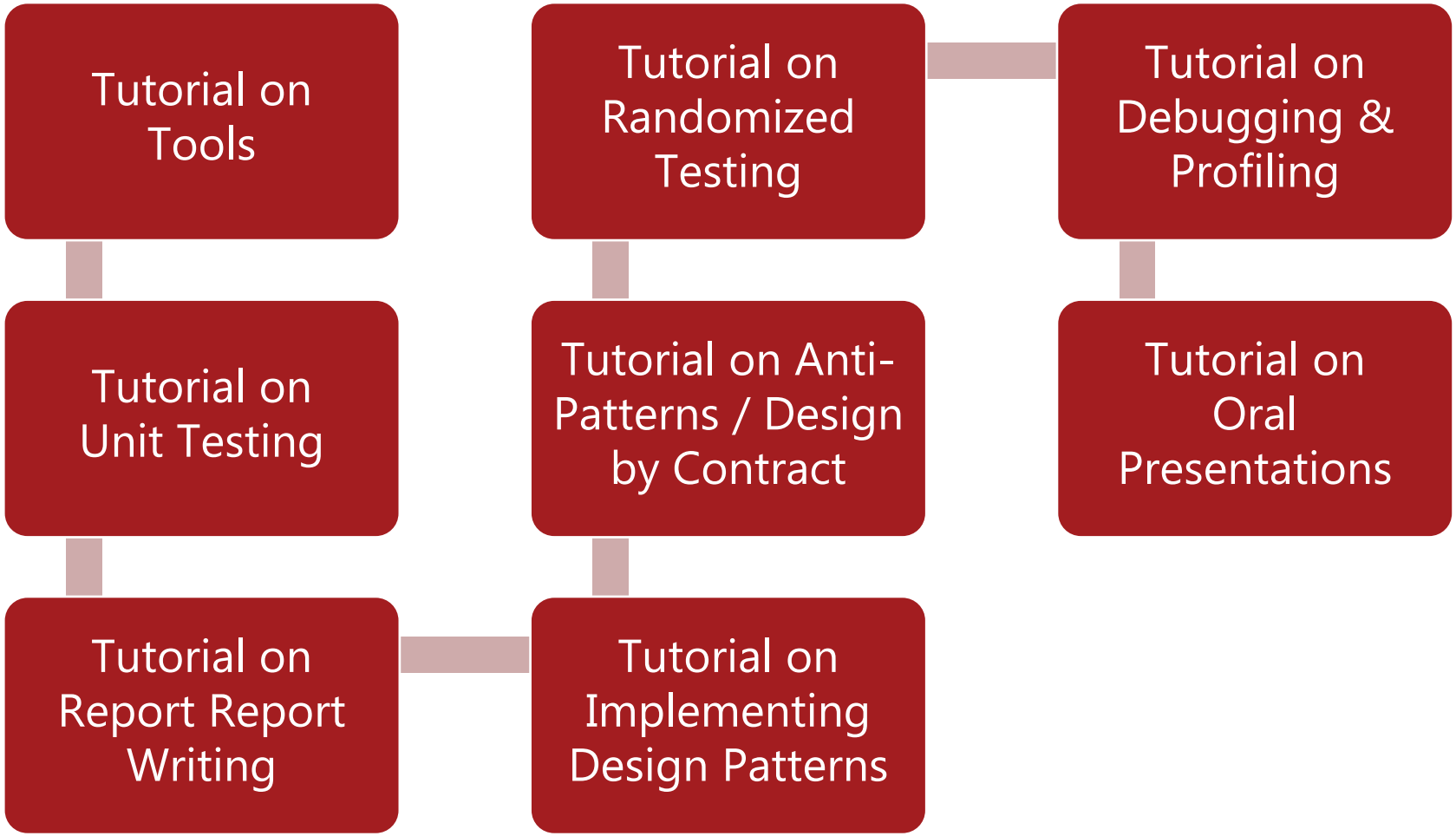
Course outline block 3: Lectures



Course outline block 4: Lectures



Course outline block 3+4: Tutorials



Common Pitfalls for New Programmers

Here are a few common pitfalls [illegible] programmers face:

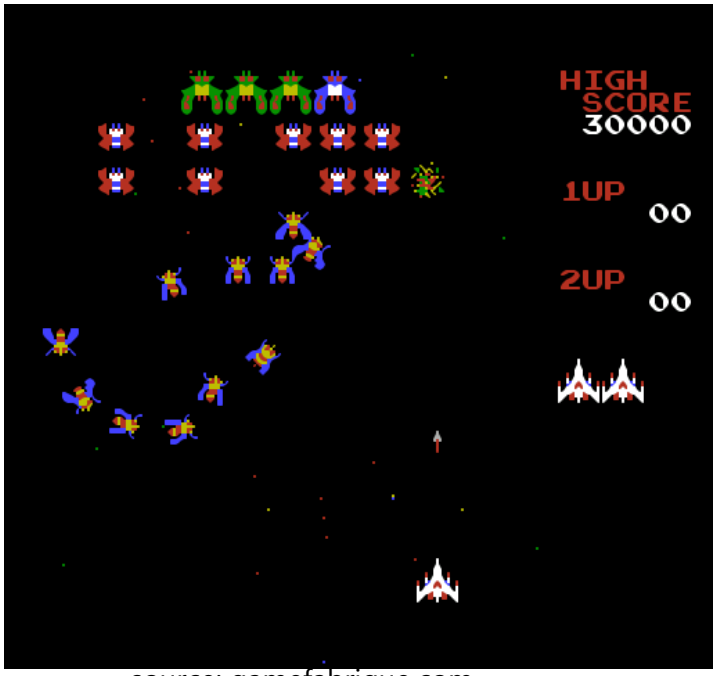
1. A lack of con[illegible]
   Programmi[illegible]tence.
2. Hesitanc[illegible] about and utilize a good e[illegible]
   It will m[illegible]t of time.
3. Viewing [illegible]mmonplace events:
   Realizing [illegible] out of the box.
4. Seeking out t[illegible]elp with questions:
   Create a supportive tea[illegible]k together in a common location, help each other remember to walk away every now and then.

We try to help you over these pitfalls in our exercises!

Source: https://www.quora.com/What-are-the-most-common-pitfalls-that-new-programmers-face

Exercises

- Language: Danish / English
- Two projects: C# games with self-contained framework
- Independent coding and testing exercises
- Platform Mono
- Groups of **3** students
- Both mandatory exercises (80%) and challenges (20%)
- Weekly submissions in block 3 bi-weekly in block 4



source: gamefabrique.com

Assignments = Exercises

- Deadline for submissions / hand-in
  - Fridays 15:00
  - GitLab submission is used for check deadline
- Reports must be written in Latex use SU17.sty
- Resubmission every week = two assignments in one week
- Points 0, 0.5 and 1
  - Resubmission only on 0.5

Recommended literature

- Robert C. Martin and Micah Martin. Agile Principles, Patterns, and Practices in C#, Pearson Education, Inc., 2007
- Peter Sestoft, Henrik Hansen, C# Precisely, 2nd ed., MIT Press, 2011
- Steve McConnell. Code Complete: A Practical Handbook of Software Construction, 2nd Edition, Microsoft Press, 2004.
- Peter Bell and Brent Beer. Introducing GitHub—A Non-Technical Guide, O'Reilly Media, Inc., 2015.

Recommended literature

- Martin Fowler, UML Distilled: A Brief Guide to the Standard Object Modeling Language, 3rd Edition, Addison-Wesley Professional, 2003
- Martin Fowler, Uml Distilled: A Brief Guide To The Standard Object Modeling Language, 3/E Paperback, 2015
- David West and Brett McLaughlin, Head First Object-Oriented Analysis and Design, O'Reilly Media, 2006

Additional literature

- Alistair Cockburn. Structuring use cases with goals, 1997.
- Robert C. Martin. Clean Code: A Handbook of Agile Software Craftsmanship, 1st Edition, Prentice Hall, 2008.
- Erich Gamma, et al. Design Patterns: Elements of Reusable Object-Oriented Software, 1st Edition, Addison-Wesley Professional, 1994
- Andrew Troelsen and Philip Japikse. C# 6.0 and the .NET 4.6 Framework. 7th ed., Apress, 2016

Agenda

- Organization
- Competencies
- Lecture
- Tutorials
- Exercises
- Exam

- **Subject area**
- Lecture contents

Software Development (SD)

Reasons why this might be your most important lecture in your study:

- SD skills are expected in subsequent courses
- Programming and software development is part of most other disciplines in computer science (data science, machine learning, …)
- Enormous industrial demand
- Various career paths in industry

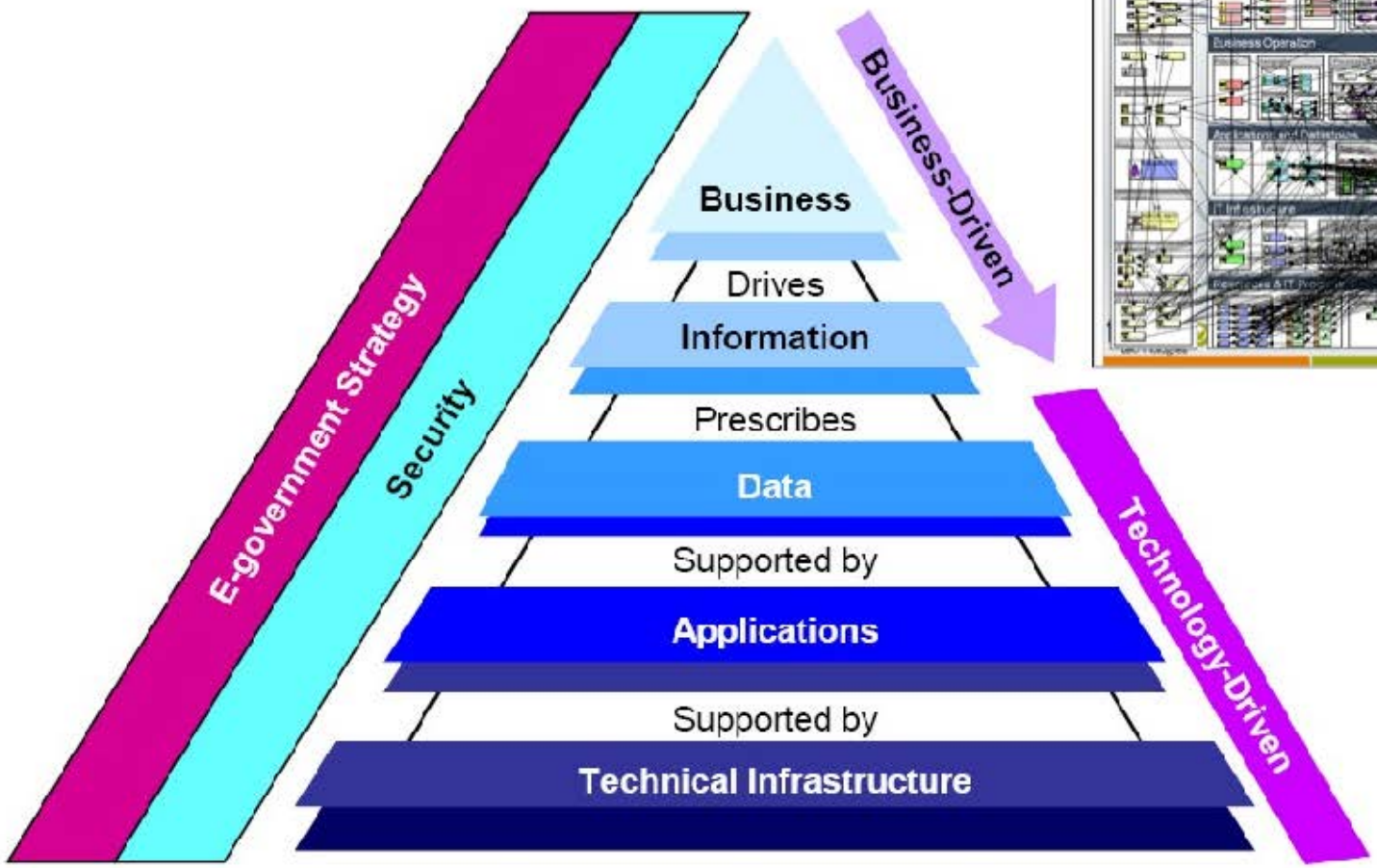
More than Programming

- Software development is more than programming
- Part of an engineering approach to development of software systems

But why?

- Complexity in multiple dimensions
  - Lines of code / number of modules
  - Heterogeneous systems (programming languages, platforms, frameworks, hardware, ...)
  - Complex control flows in distributed systems
  - Complex interactions of modules

Complex Interactions



Source: wikimedia.org

Example Projects in Mio. Lines of Code

iOS app - simple game	0,01
Unix v 1.0 (1971)	0,01
Win32/Simile virus	0,01
iOS app - photo editing	0,04
Pacemaker	0,08
Photoshop v1 (1990)	0,12
Camino	0,2
Quake 3 engine	0,31
Space Shuttle	0,4
18000 pages of printed text	1
Crysis	1
War And Peace x 14, Ulysses x 25, The Catcher in The Rye x 63	1
Syphilis	1,14
Age of Empires Online	1,2
CESM Community Earth System Model	1,2
F-22 Raptor	1,7
Linux Kernel 2.2.0 (1999)	1,8
Hubble Space Telescope	2
Unreal Engine 3	2
Lines of code de-bugged in the Jurassic Park network by Dennis Nedry	2
Windows 3.1 (1992)	2,5
Drones (control software)	3,5
ROOT software (at the LHC)	3,5
Photoshop CS 6 (2012)	4,5
Windows NT 3.1 (1993)	4,5
HD DVD Players on XBox	4,7
HealthCare.gov - needed to repair	5
Mars Curiosity Rover	5
Linux kernel 2.6.0 (2003)	5,2
Google Chrome (estimate 2) (2011)	5,4
World of WarCraft (Server)	5,5
Windows XP Service Pack 1	6,1
Boeing 787	6,5

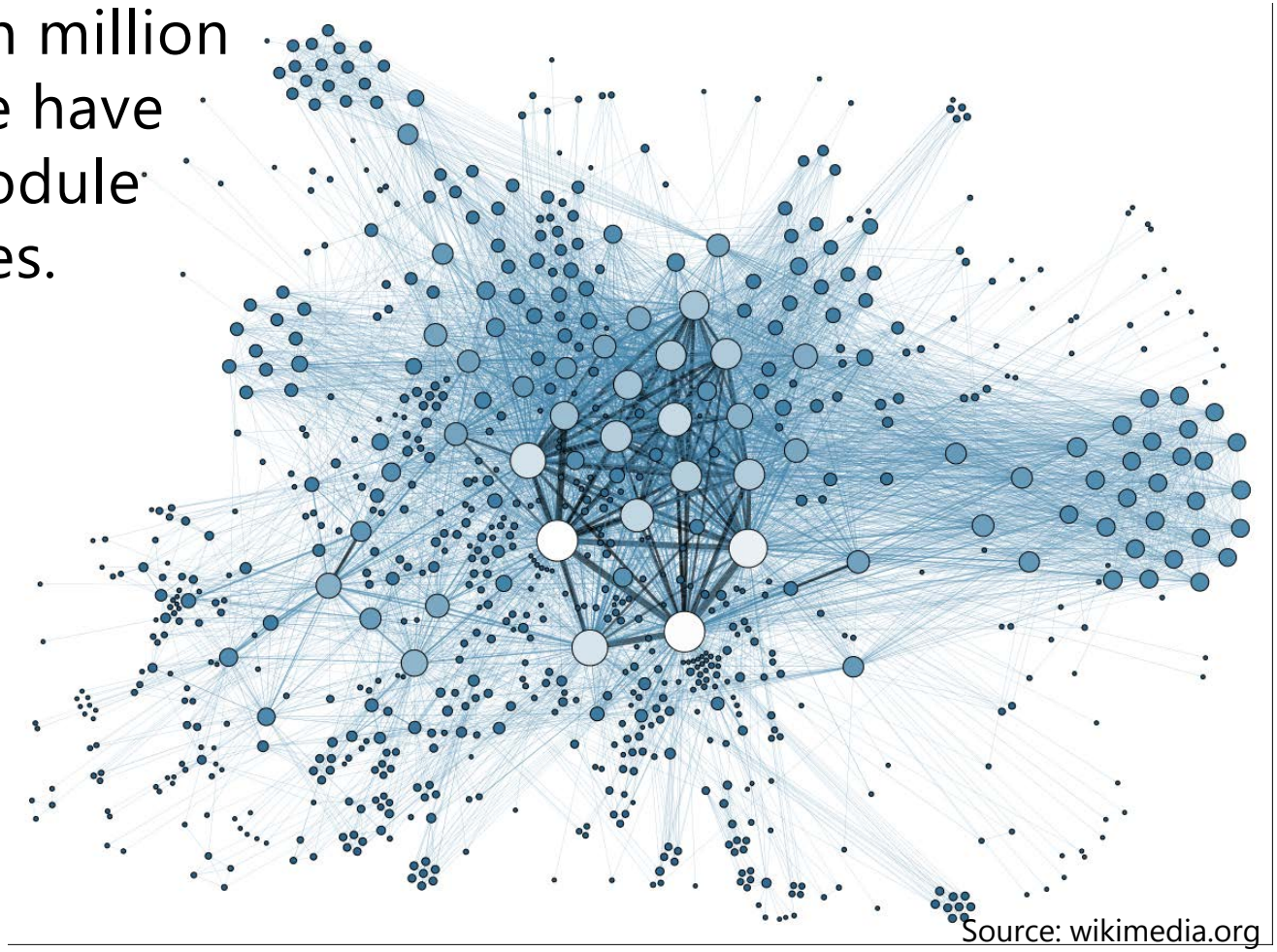
Example Projects in Mio. Lines of Code

Google Chrome	6,7
Windows NT 3.5 (1994)	7,5
Windows NT 3.51 (1995)	9,5
Firefox	9,7
Chevy Volt (electric car)	10
Intuit Quickbooks	10
Windows NT 4.0 (1996)	11,5
Android	12
Mozilla Core	12,5
MySQL	12,5
Boeing 787, total flight software	14
Android (upper estimate)	15
Linux 3.1 (recent version, 2013)	15
Apache Open Office	23
F-35 Fighter	24
Microsoft Office (2001)	25
Windows 2000 (2000)	29
Microsoft Office for Mac (2006)	30
Symbian	37,6
Windows 7	40
Windows XP (2001)	40
Microsoft Office (2013)	45
Large Hadron Collider	50
Microsoft Visual Studio 2012	50
Windows Vista (2007)	50
Facebook (without backend code)	62
US Army's Future Combat System	63,8
Debian 5.0 codebase	68
Mac OS X 10.4	86
Software in typical new car, 2013	100
Debian 5.0 (all software in package)	324
Healthcare.gov	500
Google	2.000

Source: http://www.informationisbeautiful.net/visualizations/million-lines-of-code/

Software Systems as Networks

Systems with million lines of code have complex module dependencies.



Source: wikimedia.org

Countermeasures

- Projects stretch the limits of state-of-the-art capabilities → Engineering approach necessary

- Programming languages and tools
  - Object-oriented languages (Java, C++, Python, C#)
- Organization
  - Object-oriented programming, libraries, frameworks, …
- Process models
  - Waterfall model, SCRUM, RUP, …
- (Project-)Management
  - Prince2, PMBOK, …

Software Engineering

Software engineering is the application of a systematic, disciplined, quantifiable approach to the design, development, operation, and maintenance of software, and the study of these approaches; that is, the application of engineering to software.

[IEEE Standard Glossary of Software Engineering Terminology, 1990]

Software Engineering

Research, design, develop, and test operating systems-level software, compilers, and network distribution software for medical, industrial, military, communications, aerospace, business, scientific, and general computing applications.

[ACM (2007). "Computing Degrees & Careers“]

Sub disciplines of Software Engineering

- Requirements engineering
- Software design
- Software construction
- Software testing
- Software maintenance
- Software configuration management
- Software engineering management
- Software development process
- Software engineering models and methods

The process of defining the architecture, components, interfaces, and other characteristics of a system or component. It is also defined as the result of that process.

The detailed creation of working, meaningful software through a combination of coding, verification, unit testing, integration testing, and debugging.

An empirical, technical investigation conducted to provide stakeholders with information about the quality of the product or service under test.

Sub disciplines of Software Engineering

- Software quality
- Software engineering professional practice
- Software engineering economics
- Computing foundations
- Mathematical foundations
- Engineering foundations

[Software Engineering Body of Knowledge (SWEBOK Version 3), 2014]

Agenda

- Organization
- Competencies
- Lecture
- Tutorials
- Exercises
- Exam

- Subject area
- **Lecture contents**

C# Programming

- Modern object-oriented programming language
- .NET-framework
- IDEs Visual Studio on Windows OS (Linux/Mac)

Testing

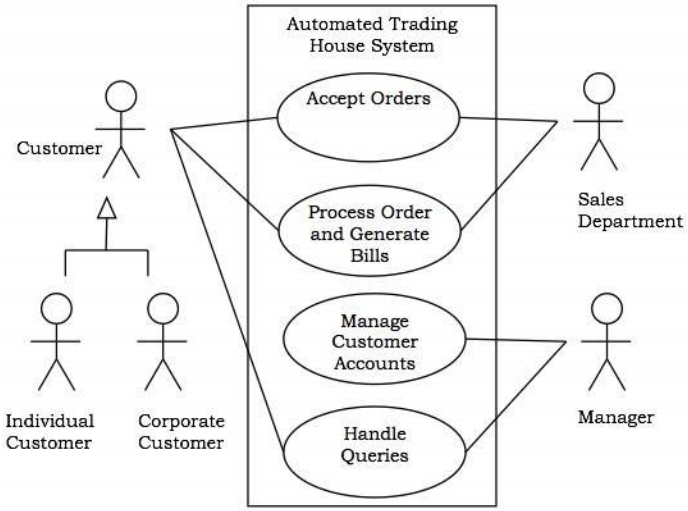
- White-box testing
- Acceptance testing
- Integration testing

Design Patterns

- Best-practice object-oriented solutions to problems
  - Describe how we should solve a particular problem
- Anti-patterns
  - Describe how we should <u>not</u> solve problems

Modelling

- UML Modelling
  - Various diagram types (use-case, class, object, sequence, activity, component diagrams)
- Component Design
- Design by Contract
- Component-based Development
- Specifying program properties



Style

- Writing solid code
- Programming style
- Code Reviews
- Refactoring

Group forming phase today 10:00-12:00 (after the lecture!)

- Rooms for groups already formed
  - øv - 1-0-26, Universitetsparken 1-3, DIKU (Mac) (Benjamin)
  - øv - 1-0-30, Universitetsparken 1-3, DIKU (Mac) (Kristian)
  - øv - 1-0-18, Universitetsparken 1-3, DIKU (Windows/Linux) (Alexander)
  - øv - 1-0-22, Universitetsparken 1-3, DIKU (Windows/Linux) (Mads)
  - øv - 1-0-10, Universitetsparken 1-3, DIKU (other OS) (Line)
- Room for students without a group
  - øv - 1-0-34, Universitetsparken 1-3, DIKU (Line & Kristian)

- Exercises on C# and MonoDevelop/Xamarin Studio

After group forming

- Walk through tutorial on installing the development environment
- Fill out questionnaire on absalon
- C# exercise

Next lecture

C# Programming