Hugbúnaðarverkefni 1 / Software Project 1

1. Introduction

HBV501G – Fall 2018

Matthias Book

HÁSKÓLI ÍSLANDS
VERKFRÆÐI- OG NÁTTÚRUVÍSINDASVIÐ
IÐNAÐARVERKFRÆÐI-, VÉLAVERKFRÆÐI- OG TÖLVUNARFRÆÐIDEILD

Doctor Who?

- **Dr. Matthias Book, Professor of Software Engineering**

- **Contact information**
  - Office: Tæknigarður 208
  - E-mail: book@hi.is
  - No fixed office hours, make appointments by e-mail anytime

- **Background**
  - Studied Computer Science for Engineers at Universities of Dortmund and Montana
  - Doctoral degree from University of Leipzig
  - Researcher and lecturer at Universities of Duisburg-Essen and Chemnitz
  - Research manager at software development company adesso in Dortmund
  - Teaching at University of Iceland since 2014

Research Interests

- **Interaction among software project stakeholders**
  - Facilitating effective communication between team members from heterogeneous backgrounds (business, technology, management…)
  - Identifying risks, uncertainties and value drivers early in a project, and focusing collaboration on these aspects rather than on business/technology trivia
- **Multi-modal and sketch-based user interfaces**
  - Specifying and controlling user interactions with software systems through 2D and 3D gestures, voice commands etc.
  - Sketch-based software engineering: Understanding and manipulating software artifacts through sketches on code, models, user interfaces
- **Software engineering for high-performance computing**
  - Using software engineering methods and tools to efficiently develop scientific software

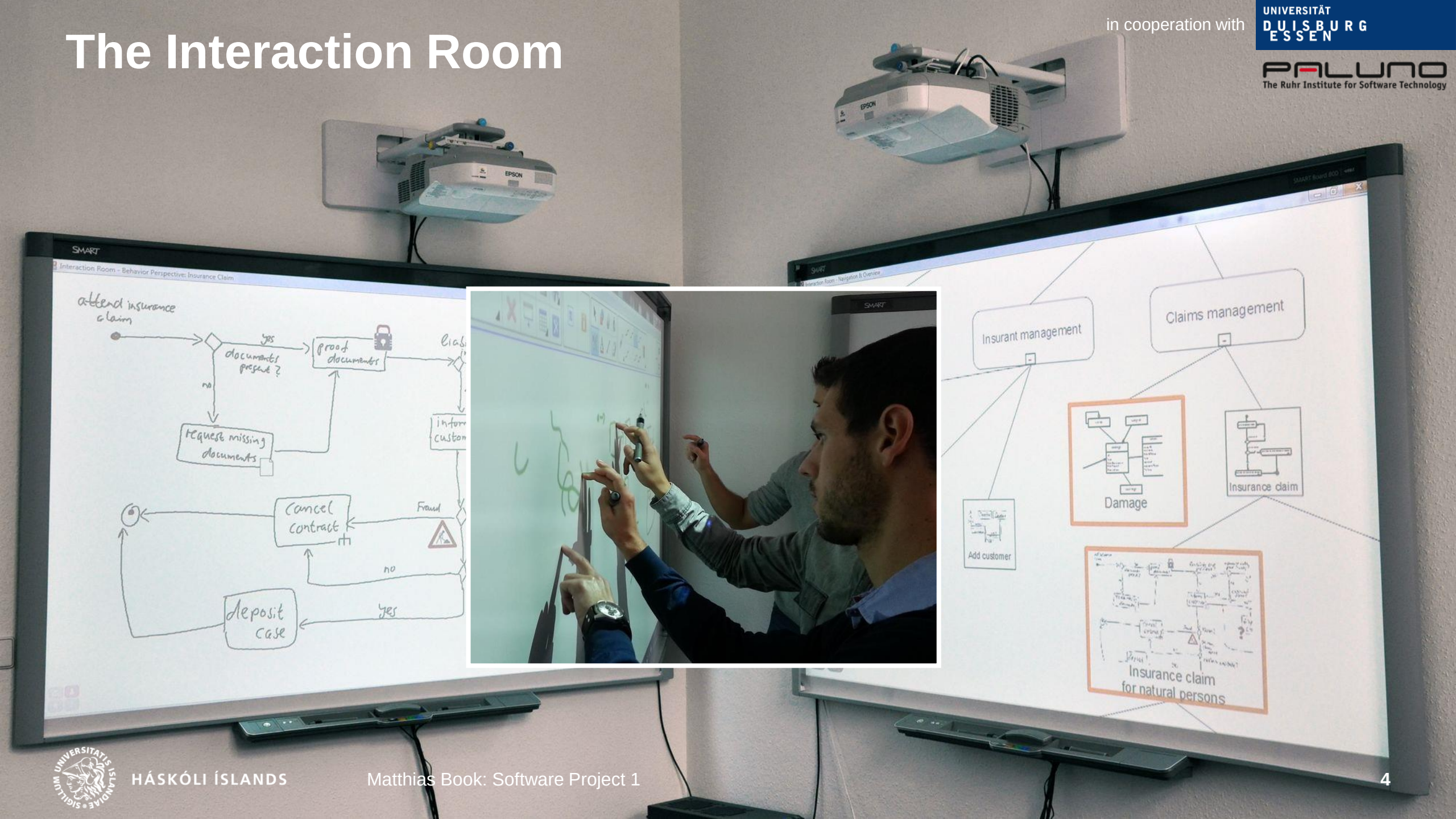
- B.Sc., M.Sc., Ph.D. projects on these topics available – or suggest your own!

The Interaction Room

in cooperation with

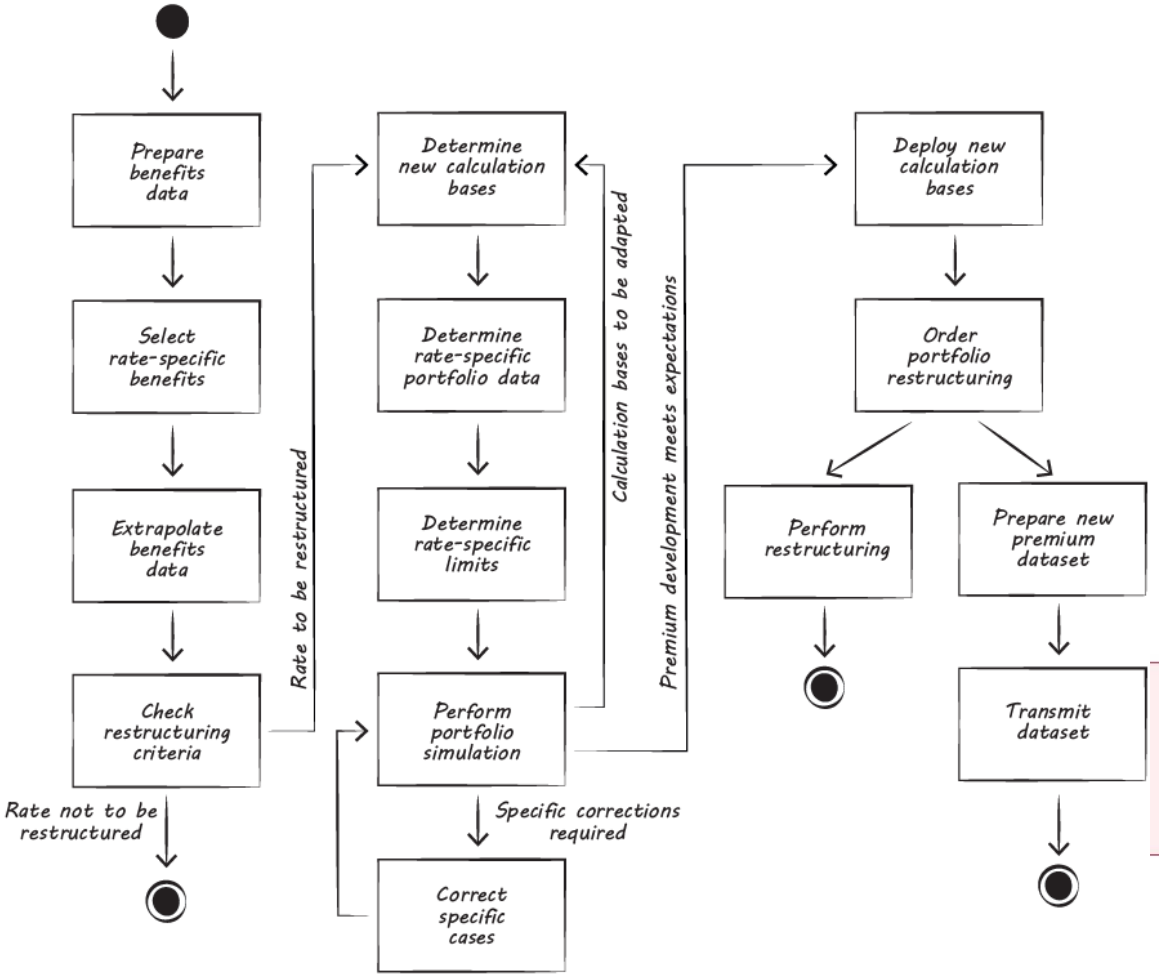
UNIVERSITÄT DUISBURG ESSEN

PALUNO
The Ruhr Institute for Software Technology



Sketch-based Software Engineering

in cooperation with UCI University of California, Irvine



Sketch-based Test Case Specification

Goal: Enable developers to use sketching

- **not just to help them think**
  about software engineering activities
- **but as an interaction modality**
  to perform and complete activities directly

Sketch-based Code Merging

```
        <constraint
            firstAttribute="trailing"
            secondItem="LFs-rG-Cv6"
            secondAttribute="trailing"
            id="w6V-ZV-c3M"/>
    </constraints>
</view>
<connections>
    <outlet property="displaySubtitle"
        destination="PPq-nc-dK0" id="p3z-
        Mf-Llw"/>
    <outlet property="bottomView"
        destination="7Wa-hI-bwz" id="wcM-
        eJ-2e5"/>
    <outlet property="finishSuiDuButton"
        destination="dWz-Xu-8U4" id="TtA-
        CN-vFO"/>
    <outlet property="mainView"
        destination="LFs-rG-Cv6" id="OZC-
        zg-yYF"/>
    <outlet property="numberLabel"
        destination="zog-c3-fbX" id="wJq-
        Xf-MiY"/>
    <outlet property="weiTiaoView"
        destination="58Q-XU-7cJ" id="fre-
        Wb-wsR"/>
</connections>
</viewController>
<placeholder
```

```
188
189
190
191
192
193
194
195
196
197
198
199
```

```
        <constraint firstAttribute=
            "trailing" secondItem="LFs-rG-
            Cv6"
            secondAttribute="trailing"
            id="w6V-ZV-c3M"/>
    </constraints>
</view>
<connections>
    <outlet property="bottomView"
        destination="7Wa-hI-bwz" id="wcM-
        eJ-2e5"/>
    <outlet property="finishSuiDuButton"
        destination="dWz-Xu-8U4" id="TtA-
        CN-vFO"/>
    <outlet property="mainView"
        destination="LFs-rG-Cv6" id="OZC-
        zg-yYF"/>
    <outlet property="numberLabel"
        destination="zog-c3-fbX" id="wJq-
        Xf-MiY"/>
    <outlet property="weiTiaoView"
        destination="58Q-XU-7cJ" id="fre-
        Wb-wsR"/>
</connections>
</viewController>
<placeholder placeholderIdentifier=
    "IBFirstResponder" id="bFU-wk-q91"
    userLabel="First Responder"
    sceneMemberID="firstResponder"/>
```

Course Scope

- After a first taste of programming-in-the-large…
  - One simple approach to agile software development (HBV401G)
- …now a more in-depth look at different software engineering paradigms:
  - **Plan-driven development and web engineering (HBV501G)**
  - Agile development and mobile software engineering (HBV601G)

- **Course aims:**
  - Learn about the different software engineering methods at your disposal for
    - requirements, estimation, architecture, design, integration, testing, management
  - Gather more practical experience with these approaches
    - in the context of two different technologies (**web** and mobile)
    - in the context of two different system landscapes (**green-field** and brown-field)
  - ➢ Make well-informed method decisions and avoid pitfalls in your future industry projects

Are You in the Right Course?

- **Proper sequence:**
  - HBV401G Software Development (prerequisite: HBV201G, TÖL101G, TÖL203G)
  - ➢ **HBV501G Software Project 1 (prerequisite: HBV401G)**
  - HBV601G Software Project 2 (prerequisite: HBV501G)

- **Not recommended to skip HBV401G before taking HBV501G**
  - Knowledge and project experience from 2nd year courses is expected
  - HBV402G (Software Development A) is not equivalent preparation
  - HBV501G and HBV601G are designed to be taken in sequence (continuous project)
  - You may not have enough time in the spring semester to accomplish required project work for HBV401G and HBV601G in parallel

Course Schedule (tentative)

Week	Thu: Team Consultations	Fri: Lectures	Sun: Assignments due
1		Introduction	
2	Phase 1: Inception	Rational Unified Process	Team Formation (9 Sep)
3	Phase 1: Inception	Requirements Engineering	
4	Phase 1: Inception	Spring Web Applications	#1: Vision & Use Cases (23 Sep)
5	Vision & UC Present. (27 Sep)	Spring Web Persistence	
6	Phase 2: Elaboration	Behavior Models	
7	Phase 2: Elaboration		#2: Behavior Model (14 Oct)
8	Behavior Model Present. (18 Oct)	Domain Models	
9	Phase 2: Elaboration	Design Models	
10	Phase 3: Construction	Design Patterns	#3: Design Model (4 Nov)
11	Design Model Present. (8 Nov)	Design Patterns	Exchange code w other team (11 Nov)
12	Phase 3: Construction	Web Presentation Layer	#4: Code Review Report (18 Nov)
13	Code Review Present. (22 Nov)	Interaction Room	
14	Project Presentation (29 Nov)	Final Exam Prep	#5: Final Project (2 Dec)

Course Structure

- **Lectures**
  - Friday 13:20-14:50, HT-104
- **Project team consultations**
  - Weekly meetings with tutors
    - to discuss requirements, design, questions
    - to present assignment deliverables
  - Thursday 08:30-11:30, V-152
    - Teams 1-10: 08:30-09:30
    - Teams 11-20: 09:30-10:30
    - Teams 21-30: 10:30-11:30
  - Please bring your laptop!
- **Project team meetings**
  - Scheduled freely among team members
- **Project teams**
  - 3-4 students per team
- **Project topics**
  - Your own project idea
    - e.g. a service, community, game, etc.
- **Basic architectural requirements**
  - Server-side back-end logic
    - implemented using Java Spring framework
  - Client-side front-end
    - HBV501G: HTML/JS in web browser
    - HBV601G: native Android app

Some Project Ideas

- News aggregator
- Educational game
- Entertainment game
- Consumer survey
- Dictionary
- Geo information
- Recipe database
- Image sharing
- Playlist management
- Scheduling
- Community
- Job market
- Sports statistics
- Workout tracker
- Language learning
- Tournament management
- Rehabilitation support
- etc.

Team Formation

- **How:** Sign up on Doodle (https://doodle.com/poll/v56843zigu2vkbub)
  - Enter your name and HÍ e-mail account – example: “Jón Jónsson (jj1)”
  - Choose your desired team (max. 4 members per team; 3 recommended)
    - Note: Your team number determines your consultation time slot (see slide 9)
    - Ideally, form teams offline and let one person enter the names of all team members at once
    - If you can’t find team members offline:
      - If you are early: Sign up for an empty group and wait for people to join
      - If you are late: Join a group that has only 1-2 members
      - If you can find only groups with 3 members:
        Introduce yourself, ask what they are planning to build, and if it’s ok to join them
    - If you are looking for teams or team members: Communicate through Piazza
  - To change your group affiliation: Delete your previous Doodle entry and create a new one
    - Do not edit other people’s Doodle entries! Resolve assignment conflicts by e-mail
- **When:** by next Wednesday (5 Sep)
  - So you can start working on your project idea together in the first consultation on Thursday
  - Team formation needs to be finalized by Sun 9 Sep at the latest

Assignments

- 5 team assignments:
  1. Project vision & requirements: due Sun 23 Sep
  2. Behavior model & software skeleton: due Sun 14 Oct
  3. Design model: due Sun 4 Nov
  4. Code review: due Sun 11 Nov (exchange) / 18 Nov (report)
  5. Final product: due Sun 2 Dec

- **Expected quality:** Assignment deliverables should be
  - of **realistic scope** for your project
    - i.e. not specification for its own sake
    - but also don't tell us "No need to write this down, we've got it all in our head"
  - of **professional quality**, i.e.
    - clean and syntactically correct
    - suitable to show your boss

- **Expected level of detail:** Should reflect team's level of (un)certainty
  - If you are sure about something:
    - include relevant details in the document
  - If you are unsure about something:
    - If now would be a good time to figure it out: Solve it in team and include in doc.
    - If you prefer to leave it open at this time: Mention why & what would be options, and be prepared to discuss with tutor

Preview: Assignment 1: Vision Document & Use Cases

- Two of the most important artefacts created in the Inception phase are:

1. The **Vision and Scope document**
   - Describing what you want to build, what its key features/capabilities will be, who will use it…
   - Imagine having to convince your boss or an investor to provide funding for the project
     - What would they want to know to be convinced?
2. The initial **Use Case document**
   - Describing the most important use cases of your product
   - i.e. the primary things that your system is supposed to be able to do

- Producing these documents and explaining the considerations that went into them will be your job in Team Assignment 1.

Preview: Assignment 2: Behavior Model & Skeleton

- After defining the project vision and familiarizing yourself with implementation technologies, the next step is designing the technical solutions that shall be implemented to satisfy the application requirements.
- The **behavior model** includes UML sequence, activity and/or state diagrams
  - explaining how classes work together, how objects or components change state, etc.
- A **software skeleton** serves as a backbone for your future system, showing that you are familiar with the technology and can add features incrementally..
- Producing the behavioral model and software skeleton, and explaining the considerations that went into them, will be your job in Team Assignment 2.

Preview: Assignment 3: Design Model

- After defining the project vision and familiarizing yourself with implementation technologies, the next step is designing the technical solutions that shall be implemented to satisfy the application requirements.

The **design model** includes UML class diagrams

- defining the technical classes your software system will be comprised of, and their relationships

- Producing the design model and explaining the considerations that went into it will be your job in Team Assignment 3.

Preview: Assignment 4: Code Review

- To get experience in working with unfamiliar code, you'll review another team's code toward the end of the semester as Team Assignment 4.

- Make the **project documents and code** you created available to another team.
- Take **one week** to **review** the other team's code and briefly **document** your findings:
  - Examine how they structured their system. Do vision document and design model help to understand it?
  - Make suggestions for improvement in component design, technology use and coding style.
  - What did you like? Which questions did you run into? What would you recommend to change?
- In the consultations, **discuss** your findings with the other team and your tutor.

Preview: Assignment 5: Final Project

- In the last week, demonstrate and explain your product in class:
  1. **Product:** What does your system do? Demonstrate the key use cases of your product.
  2. **Architecture:** How does your product work? Explain architecture & key design decisions.
  3. **Process:** How did you build the system? Relate and interpret challenges you faced.

- Afterwards, submit in Ugla:
  - The **source code** of your final product, including everything required to build and run it
  - The **slides** of the presentation you gave in class

General Assignment Format

- Required **deliverables** (documents / models / code) must
  - be produced by all team members together
  - be submitted in one PDF document by specified deadline in Ugla
  - contain your team number, the names and kennitölur of all team members
  - indicate who will present the assignment
- **Submissions** are due at 23:59 on Sundays
  - **No individual extensions** – missing submissions receive a grade of 0!
  - But undefined grace period until whenever tutors download submissions from Ugla
  - Only the team member who will present must submit a document for the whole team
- The **presentation** must
  - be given by one representative of the team (a different one for each assignment)
  - be based on the submitted document (don’t prepare extra slides)
  - take around 5-10 minutes (plus some questions asked by the tutor)

Project Grading

- The project grade depends on the **deliverables** submitted and the **presentation** given for each assignment.
  - Grading criteria will be published together with assignment.
- All team members receive same grade for **deliverables** submitted for an assignment
  - Each assignment weighs **17%** of project grade
- Over the course of the semester, each team member must lead the **presentation** of at least one assignment to tutor
  - Focus: Don't just tell us what you did, but *why* you decided to do it this way.
  - The presenting team member receives an individual grade for their presentation ("6th assignment", weighs **15%** of project grade)
    - If someone shares or gives several presentations, weights are adapted accordingly
- The resulting project grade weighs 30-70% of the final course grade.

Grade Weights

- Assessment of team contribution
  - At the end of the semester, all team members assess how much each of their team mates contributed to the project.
  - Contribution votes are normalized to obtain each team member's contribution factor.

- Depending on team members' individual contributions, the weight of their project and final exam grades will be individually adjusted between 30% and 70%
  - Below-average contribution → lower weight of project grade, higher weight of exam grade
  - Average contribution → equal weight of project and exam grade (50:50)
  - Above-average contribution → higher weight of project grade, lower weight of exam grade

- More details toward end of course

Final Exam

- **Date & Time:** TBA
- **Focus:** Understanding of software engineering concepts and methods
- **Scope:** Lecture slides (i.e. contents of Námsefni folder)
  - Note: The spoken part is relevant too!
- **Style:** Written exam
  - Write into given spaces on exam sheets
  - Mark exam sheets only with your exam number, *not* your name
- **Weight:** 30-70% of final course grade

- **Tools:**
  - One sheet of handwritten material allowed
    - i.e. blank A4 sheets with your own ink
    - no photocopied notes
    - no margin notes in printed lecture slides
  - Dictionary allowed (in book form)
  - No electronic devices allowed
- **Questions:**
  - Explain / argue / discuss / calculate...
  - No optional questions
  - But answers that exceed expectations can make up for deficiencies elsewhere
- **Answers:**
  - in English
  - short paragraphs of whole sentences
  - small code fragments / models

Optional In-Class Quizzes

- To encourage class attendance: Small quizzes in most lectures
  - Solved and handed in during class
  - Graded on usual scale (0…11), with 0 for any quizzes that are not handed in
    - 2 quizzes may be skipped without impacting the quiz grade

- Grades of all in-class quizzes will be averaged into one final **quiz grade**
- Quiz grade can improve your final exam grade:
  - All the normal final exam questions add up to 100% (as usual)
  - Quiz grade will be counted as an optional additional final exam question worth 7.5%

➢ If you don't participate in any quizzes, you can still get top marks on the exam

➢ If you do participate in quizzes, the quiz grade can improve your exam grade by up to 11 on a 7.5% question, and thereby make up for deficiencies elsewhere

Grading Scheme

- All contributing factors are graded on a scale of 0…11
  - Grading criteria for assignment deliverables
  - Individual presentation quality
  - Questions on in-class quizzes
  - Questions on final exam
- All factors are averaged using the published weights, without rounding
  - Exceptional performance (11) on some factors can outweigh lower performance elsewhere
- Resulting final grade is rounded to nearest half point
  - In case of a passed exam, final grade is capped at 10.0
  - In case of a failed exam, final grade is capped at 4.5

- Need a passing project grade to be allowed to final exam
  - If project grade is not sufficient, need to do a project again next year
- Need a passing project and exam grade to pass the course
  - Failed exam can be re-taken during the resit period
  - No need (and not possible) to redo a passed project
    - Project grade remains valid for only one year though

Previous Course Feedback / Expectation Management

- **“Too little emphasis on programming, too much SE / OO / UML”**
  - HBV501G is a software engineering, not a web programming class
  - OO is most common conceptualization & implementation of basic modularization principles
    - Good practice to apply these principles even if the programming language doesn’t enforce them
  - UML is communication standard of international software engineering community
    - Lots of valid criticism of UML – that’s why I’m teaching only what you really need to know
  - Companies expect you to be not just a programmer, but a **software engineer** – i.e. someone who can
    - talk to users, managers and other engineers, and understand what they need
    - come up with efficient solutions, and make them happen on schedule/budget in a team that can still work with the code after you are gone
  - This is fundamental methodical knowledge for working in commercial projects (most of which are legacy, not startup projects), and for pursuing a Master’s degree in Iceland and abroad
- **“HBV assignments are repetitive”**
  - Key software engineering skills (identifying requirements, designing software components, applying design patterns, managing teamwork) grow only with practical experience

Previous Course Feedback / Expectation Management

- **“Why produce all these complex documents for a simple class project?”**
  - Plan-driven processes by nature are more document-heavy than agile processes
  - I agree it’s a bit artificial in a course, but you need to practice this approach as well
  - Practicing a new method is easier on a simple example project
  - There is value in the documents – see them as a tool, not a chore
    - *“The vision document was the one thing we kept coming back to” (a team’s project retrospective)*
- **“Assignments are unclear”**
  - Read tasks carefully, ask questions and get feedback on drafts early
  - **Adopt professional perspective:** “What would make your boss and team mates happy?”
    - precise writing, clean modeling, clean coding, honoring deadlines and commitments, thinking ahead, finding solutions, explaining your reasoning, helping team mates
- **“Lectures are too text-heavy”**
  - No single good textbook, so slides intended for review, exam prep, reference on the job
    - *“The HBV classes are saving my life here” (former student now working in a bank)*

Course Support

- **Questions and feedback** welcome anytime! Preferably:
  - in class
  - in team consultations
  - in Piazza (enroll at https://piazza.com/hi.is/fall2018/hbv501g)

- **See Ugla for**
  - Slides (for download after each lecture)
  - Important course announcements

- **Teaching staff**
  - Matthias Book (book@hi.is)
  - Daníel Páll Jóhannsson (dpj@hi.is)
  - Andri Valur Guðjohnsen (avg4@hi.is)

Suggested Literature

(for more in-depth reading – not mandatory to buy for course)

- **Requirements Engineering**
  - Karl Wiegers, Joy Beatty: Software Requirements. Microsoft Press, 2013
- **Java Web Application Development**
  - Spring Framework Documentation. http://spring.io/docs
  - Tejaswini Mandar Jog: Learning Spring 5.0. Packt, 2017
  - Java Enterprise Edition Documentation. https://docs.oracle.com/javaee/7/index.html
  - Nicholas S. Williams: Professional Java for Web Applications. Wrox, 2014
- **Object-oriented Analysis and Design**
  - Craig Larman: Applying UML and Patterns. Prentice Hall, 2004
  - Eric Freeman, Bert Bates: Head First Design Patterns. O'Reilly, 2004
- **Unified Process**
  - The Rational Unified Process. http://sce.uhcl.edu/helm/rationalunifiedprocess/
  - Per Kroll, Philippe Kruchten: The Rational Unified Process Made Easy – A Practitioner's Guide to the RUP. Addison-Wesley, 2003

Course Advertisement

Security Requirements Engineering for Cyber-Physical Systems (TÖL505M)

- 2-credit course for Bachelor and Master students
- 4 weeks (10 Sep – 5 Oct), 4 lectures, 2 assignments
  - Tentative schedule (likely subject to change):
    Wednesdays 10:00-11:30 & Fridays 13:20-14:50 @ Interaction Room (Tæknigarður 227)
- Lecturer: Shafiq ur Rehman, MSc (University of Duisburg-Essen, Germany)
  - Contact: shafiq@hi.is, book@hi.is

- **Contents:**
  - Basic definitions of security threats, security goals and risk assessment
  - Overview of concepts associated with industrial cybersecurity
  - Main security threats to software development lifecycle
  - Types of attacks on critical infrastructure
  - Main standards and methodologies of security requirements engineering

Let’s get started!

book@hi.is

HÁSKÓLI ÍSLANDS
VERKFRÆÐI- OG NÁTTÚRUVÍSINDASVIÐ
IÐNAÐARVERKFRÆÐI-, VÉLAVERKFRÆÐI-
OG TÖLVUNARFRÆÐIDEILD