Hugbúnaðarverkefni 1 / Software Project 1

14. Final Exam Preparation

HBV501G – Fall 2018

Matthias Book

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

Quiz #11 Solution: Design Patterns

- **What is the purpose (1-5) of the following design patterns (a-e)?**

a) Command **(5)**
b) Decorator **(3)**
c) Observer **(4)**
d) Singleton **(1)**
e) Strategy **(2)**

1. Ensure a class only has one instance.
2. Define a family of algorithms, encapsulate each one, and make them interchangeable in objects that use them.
3. Attach additional responsibilities to an object dynamically.
4. Notify all dependents of an object when its state changes.
5. Decouple an object making requests from objects that know how to perform those requests.

Definitions: Freeman/Robson, Head First Design Patterns

Recap: Presentation Lessons

- Engage your audience
  - Speak freely, don’t read off slides/cards
  - Interact with the audience and the slides
  - Excite yourself about the topic
- Tailor your talk to your audience
  - What do they know already?
  - What don’t they know yet?
  - What do they need and want to know?
- Be clear
  - High-contrast colors, readable font size
  - Keep your slides simple
    - just a few text bullets
    - even better: clear illustrations
- Use a style you are comfortable with
  - Visualize what you want to explain in a way that’s natural for you to talk about
    - Slides, (clean) hand-drawn sketches…
- Be prepared
  - Practice to get a feeling for your talk
    - Pacing, time limits vs. amount of material
  - Rehearse any sticky parts
    - Intro, wrap-up, topic transitions, sketches
  - Don’t try to memorize it all – get fluent in the topic, and you’ll be fluent in your talk
  - Try technical setup ahead if possible

General observation from yesterday:
This looked good already – but keep using any chance you get to practice!

Recap: Assignment 5: Grading Criteria

1. **Product Demonstration (80%)**
   - ✓ Software runs smoothly
   - ✓ Key use cases are working
2. **Architecture & Design (10%)**
   - ✓ Architecture described, illustrated clearly
   - ✓ Key design decisions described and illustrated clearly
   - ✓ Room for improvement discussed critically
3. **Software Process (10%)**
   - ✓ Design & development process over the course of the semester described clearly
   - ✓ Handling of technical / methodical / collaboration challenges discussed critically

- **Submitted code (due Sun 2 Dec)**
  - Not graded explicitly, but checked for conformity with presented prototype and architectural requirements
    - Inconsistency of submitted code with presented product may lead to reduction of Product Demonstration grade
    - Completely messy code or code not using server-side Spring MVC components may lead to reduction of Architecture & Design grade
- **Presentation Grade**
  - ✓ Presenter shows initiative, explains things clearly, shows understanding of the project, can answer questions competently

Lessons Learned

Software Engineering Lessons Learned

Last semester's closing questions:

- What lessons have you learned from the project experience?
- What might prevent you from following through on your good intentions in future projects?
- And how would you avoid those obstacles?

Retrospective on this semester:

- Which things worked better than last semester?
- Did you manage to follow through on your good intentions?
- What prevented you from doing so?
- How would you avoid *that* in the future?

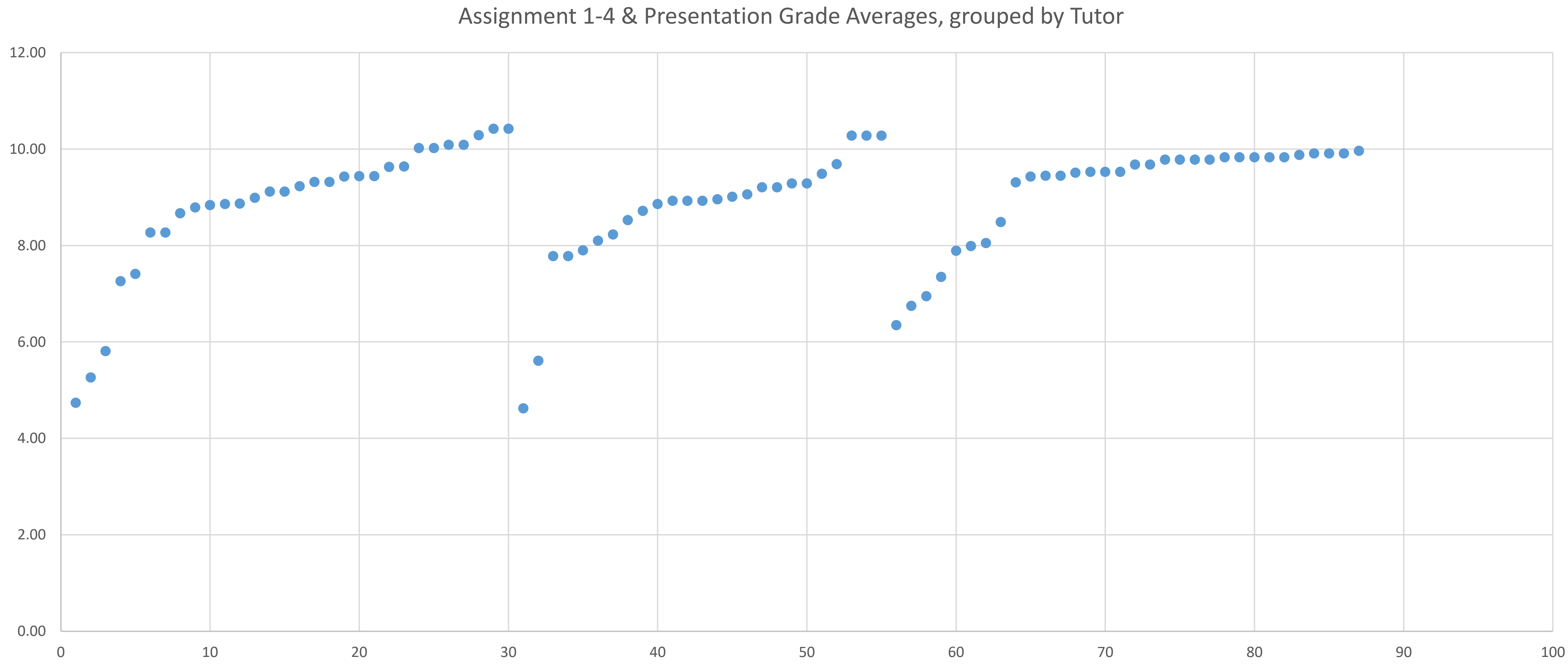
Notice a Pattern?

- The advice that sounds most boring and trivial when I talk about it in class…
  - Communicate and be proactive
  - Write down precise requirements
  - Think well about your domain and design models
  - Make a project plan and strive to follow it as far as possible
- …is also what *always* shows up on teams' "What we'll do better next time" slides

- Pessimistic view
  - Ignoring this advice in future projects will always be tempting
  - There'll always be scheduling issues that appear to justify cutting corners
- Optimistic view
  - Following this advice is not rocket science!
  - With a bit more discipline, you can get better results with less stress.

Example lesson:
"Never code alone"
– Team 19

Preliminary Project Grades (still missing Assignment 5)



Did you Learn Anything in this Course?

- Much of the material we covered is designed to suit large, complex projects
  - RUP – a heavyweight software process model
    - Lots of artefacts (when you could also just put some cards on a Kanban board)
  - UML – a heavyweight modeling language
    - Lots of diagram types (when you could also just sketch some box & line diagrams)
  - Java Spring – a heavyweight server-side web framework
    - Complex design despite lots of abstraction (when client-side frameworks yield neat results faster)

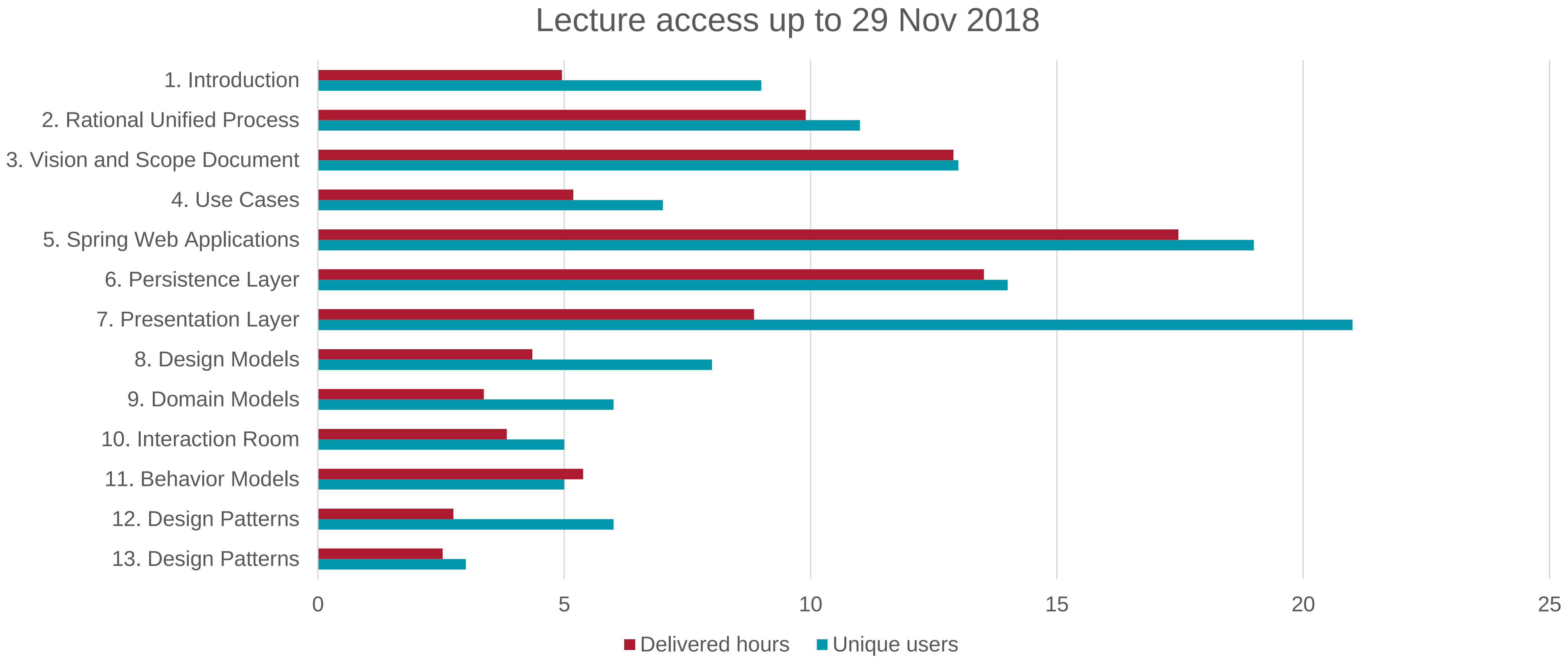
➢ May not have appeared ideally or immediately applicable to in-class projects

- Tailoring any of the above to your needs is perfectly fine and recommended
  - But one must experience and understand a technique in order to tailor in properly
- Ultimately, you *will* likely be working on large, complex industry projects
  - And then be expected to know how to deal with them without improvising and scrambling

Teaching Retrospective

- **Things that seem to have worked well**
  - In-class quizzes to encourage class attendance and “activation” of concepts
  - Class recordings to enable students to participate remotely or recap lectures
  - Communication of grading standards in class and coordination among tutors
  - Earlier introduction to technology to enable teams to start prototyping
- **Things I’m still working on optimizing (feedback welcome)**
  - Degree of freedom in technology choices (Spring shouldn’t be “necessary evil”)
  - Using consultation times more productively
    - Mandatory discussion of drafts with teams?
    - In-class discussion of selected assignment submissions?
  - Using lecture times more productively
    - Using currently unused 1-hour lecture slot for voluntary student presentations on tech topics?
    - Suitability of “flipped classroom” approach?

Class Recordings



Kennslukönnun

Evaluate this course on Ugla!

(until 1 December)

Recap: The Nature of Software Development

“Because software is embodied knowledge,
and that knowledge is initially dispersed, tacit, latent, and incomplete,
software development is a social learning process.”

Howard Baetjer, Jr.: Software as Capital. IEEE Computer Society Press, 1998

Project Grading and Final Exam

Recap: 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.

Recap: Grade Weights

- Peer assessment of team mates' contributions to project
  - 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.

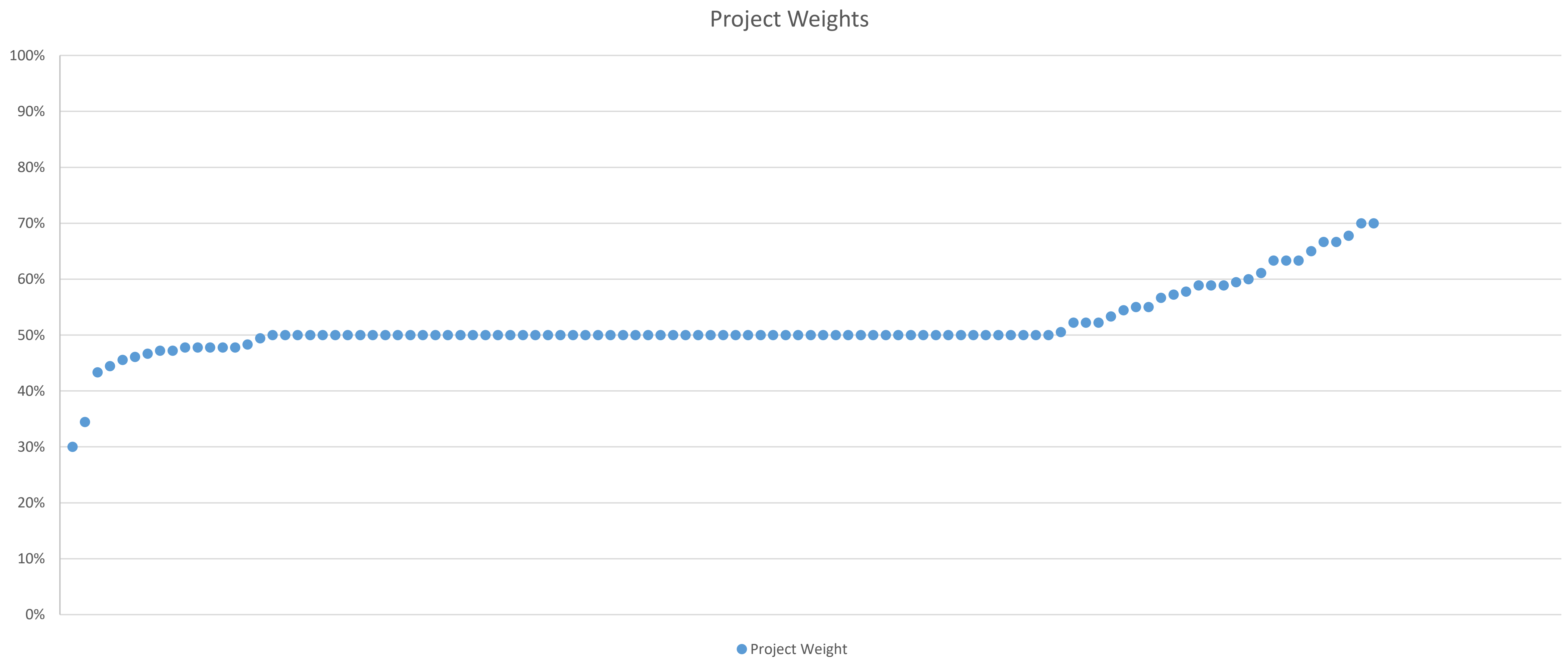
- Depending on team members' project contributions, the weight of their project and final exam grades will be individually adjusted between 30% and 70%:
  - Below-average contribution → lower weight of project grade, higher weight of exam grade
  - Average contribution → equal weight of project and exam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

- In rare cases where the above rules would punish someone who contributed above average, or reward someone who contributed below average, the traditional 50:50 weight distribution is used instead.

Peer Assessment

- Each team member can assess the contribution of each of their team mates:
  - +++ : contributed **much more** than others
  - ++ : contributed **more** than others
  - + : contributed **slightly more** than others
  - o : contributed **same** as others
  - - : contributed **slightly less** than others
  - -- : contributed **less** than others
  - --- : contributed **much less** than others
- Assessments for each team member will be averaged,
  and the assessments will be normalized across all members of a team
  - ➢ if you and your team mates all rate each other as “+++”, your normalized contribution will be “o”
- Ratings should reflect a fair assessment of your own and your team mates’ contribution.
  - Ultimate decision of how assessments and assignments are counted remains with tutors.
- Submit your ratings under the “Peer Assessment” assignment
  in Ugla by **Wed 5 Dec 23:59** (plain text, no need to upload a document).

Example: Last Year’s Project Weights based on Peer Assessment



Recap: Optional In-Class Quizzes

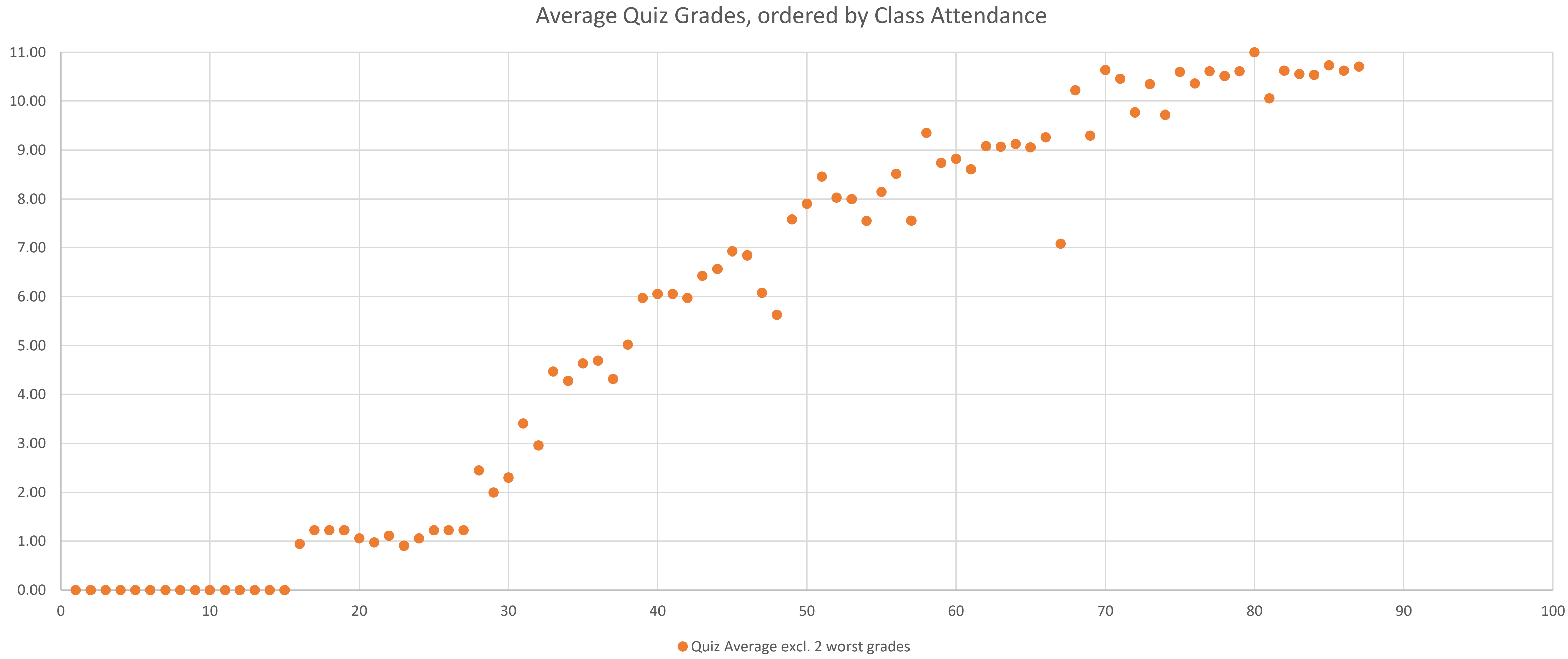
- To encourage class attendance: Small quizzes in most lectures
  - Solved and handed in during class
  - Graded on usual scale, with 0 for any unanswered questions, 5…11 for answered questions
    - 2 worst quiz grades (e.g. from skipped classes) will be ignored

- 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

This Year’s Average Quiz Grades



Recap: Final Exam

- **Date & Time: 13 Dec 2018, 13:30-16:30**
  - Makeup and Resit exams in January
- **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 sheet with only your own ink (double-sided use ok)
    - no photocopied notes or printed material
  - Dictionary allowed (in book form)
  - No electronic devices allowed
- **Questions:**
  - Explain / argue / discuss / calculate...
  - No optional questions (except quiz result)
  - But answers that exceed expectations can make up for deficiencies elsewhere
- **Answers:**
  - in English, in your own words
  - short paragraphs of whole sentences
  - possibly small models

Types of Questions

- **“Explain…”**
  - Give an explanation of what something is or how something works in general.
  - Note: A concrete example is not a substitute for a general explanation.
- **“Argue…”**
  - Make up your mind about a certain issue.
  - Present arguments for your opinion on the issue.
- **“Discuss…”**
  - Consider both sides of an issue.
  - Present arguments for both sides.
  - You can state your own opinion, but should present counter-arguments as well.
- **“Suggest…”**
  - Come up with a solution for an issue.
  - Briefly explain what should be done.
- **“Give an example…”**
  - Give a brief example that illustrates the considered concept.
  - When giving examples for several distinct concepts, make sure the examples highlight the distinguishing characteristics.
- **“Point out issues…”**
  - Examine a given artifact, and explain what is wrong with it.
  - Don’t just say what is wrong, but why it is.
- **“Draw…”**
  - Draw a UML diagram reflecting the relevant aspects of the given scenario.
  - Pay attention to proper syntax!
- **“Implement…”**
  - Write/modify Java code. For modifications, place question sheet into answer book.
  - Pay attention to proper syntax!

Sample Questions

- The following review questions are representative of the types of questions that could be on an exam.
- The answers to some of the following review questions can be found immediately on the lecture slides.
  - Caution: An open-book exam will obviously contain very few of these!
- Most questions require more individual reasoning and/or application of learned knowledge.
  - Be precise, justify your answers.

- **Read exam questions carefully**
  - Are you asked to explain or give an example or both?
  - Are you asked to argue for or discuss an issue?
  - Are you asked to explain or discuss an issue, or both?
  - How many concepts are you asked to consider?

- **Answer precisely**
  - i.e. brief, focused, comprehensive

Sample Questions: Software Process Models

- Explain what is defined by a software process model.

- Argue if strictly sequential process models are suitable for large projects.

- Argue for which kinds of projects a plan-driven process model should be followed.

- Discuss how Eisenhower’s statement “I have always found that plans are useless, but planning is indispensable” could be interpreted in the context of software engineering.

Sample Questions: The Unified Process

- Explain how the phases of the Unified Process differ from the phases of the waterfall model.
- Argue whether the Unified Process is a sequential process model.
- Explain how the Inception and Elaboration phases support risk management in large projects.

Sample Questions: Project Vision and Use Cases

- Give an example of a Vision Statement that could appear in a RUP Vision & Scope Document.

- Point out any issues with the following use cases: […]

- Explain how the size test can be used to check the validity of a use case, and argue how strictly it should be applied.

- Discuss if user interface descriptions should be part of a use case.

Sample Questions: Domain Models

- Draw a UML class diagram (as part of a domain model) modeling the structural aspects of the following scenario: […]

- Point out any modeling errors in the following UML activity diagram: […]

- Explain the difference between what is expressed by UML activity diagrams and what is expressed by UML state machine diagrams, and give one example for what you would model with each of them.

- Argue whether it is possible to use composite states without fork pseudo states.

Sample Questions: Design Models and Patterns

- Explain the different meanings of arrows in UML sequence diagrams in a domain model and in a design model.
- Explain the issue addressed by the Protected Variations Principle, and the solution strategy recommended by it.
- Explain which design pattern has been used in the following UML class diagram, and argue whether you would have chosen a different pattern for the same purpose: […]
- Explain the differences and commonalities of the Strategy and State patterns.

Sample Questions: Spring Web Applications

- Draw a UML sequence diagram showing how the Spring Web MVC framework implements the Model-View-Controller pattern.

- Explain the purpose of an Object-Relational Mapper.

- Explain the difference between annotating a JavaBean with **`@Entity`** vs. annotating it with **`@Embeddable`** when using the Java Persistence API.

- Argue whether it is reasonable that the JSP Expression Language provides convenient mechanisms for reading, but not for modifying object properties.

Recap: Overall Grading Policy

- All contributing factors are graded on a scale of 0…11
  - Grading criteria for assignment deliverables
  - Individual presentation quality
  - In-class quiz questions
  - 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 in January
    - If the resit exam is failed as well, next chance is next December (but you can take HBV601G already)
  - No need (and not possible) to redo a passed project

Wrap-up

Special Thanks

Andri Valur Guðjohnsen

Daníel Páll Jóhannsson

Teaching Assistants Wanted

- I am looking for TAs for next semester’s course:
  - **HBV401G Software Development**
- If you are interested or know someone who is, please contact me at **book@hi.is**

- **Tasks**
  - Advising student teams
  - Scoring assignments
- **Opportunities**
  - Help to shape the course
  - Brush up your CV
- **Requirements**
  - Successful completion of respective course
  - Ability to advise and evaluate student teams

Recap: HBV501G 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

Preview: HBV601G Scope

- **Agile development in industrial context**
  - Requirements engineering
  - Software estimation
  - Software architecture
  - Testing
  - Contracting

- HBV401G gave a first taste of working as a **team** in a simple agile approach
- HBV501G focused on a document-heavy **plan-driven** process
- HBV601G presents a spectrum of **agile** methods for all project phases

- **Mobile software development**
  - Designing for mobile platforms
  - Developing Android apps

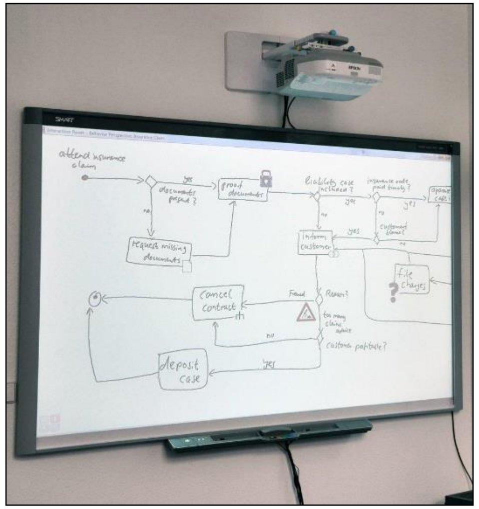
- HBV401G focused on building a simple **desktop** application
- HBV501G focused on framework-heavy **web** development
- HBV601G focuses on the peculiarities of **mobile** platforms

- **Brown-field software development**
  - Adapting and extending existing code

- HBV401G focused on **integrating** components of several teams
- HBV501G focused on **designing** a complex system from scratch
- HBV601G introduces the challenges of **adapting** existing code

Seminar / Project / Thesis Topics Available

- **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 interaction with large interactive displays**
  - Specifying and controlling user interactions with software systems through 2D and 3D gestures, voice commands etc.



- **Software engineering for mobile applications**
  - User experience, implementation challenges, ubiquitous computing
- **Intl. collaboration** with Univ. of Duisburg-Essen, Germany & University of California, Irvine
- If you are interested or know someone who is, please contact me at **book@hi.is**

Keep in Touch!

- For questions, projects, events, guest talks, student jobs, collaboration…

✉ book@hi.is

http://is.linkedin.com/in/matthiasbook

http://www.twitter.com/matthiasbook

Takk fyrir!

book@hi.is

HÁSKÓLI ÍSLANDS

VERKFRÆÐI- OG NÁTTÚRUVÍSINDASVIÐ

IÐNAÐARVERKFRÆÐI-, VÉLAVERKFRÆÐI-
OG TÖLVUNARFRÆÐIDEILD