Hugbúnaðarverkefni 2 / Software Project 2

1. Introduction

HBV601G – Spring 2019

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

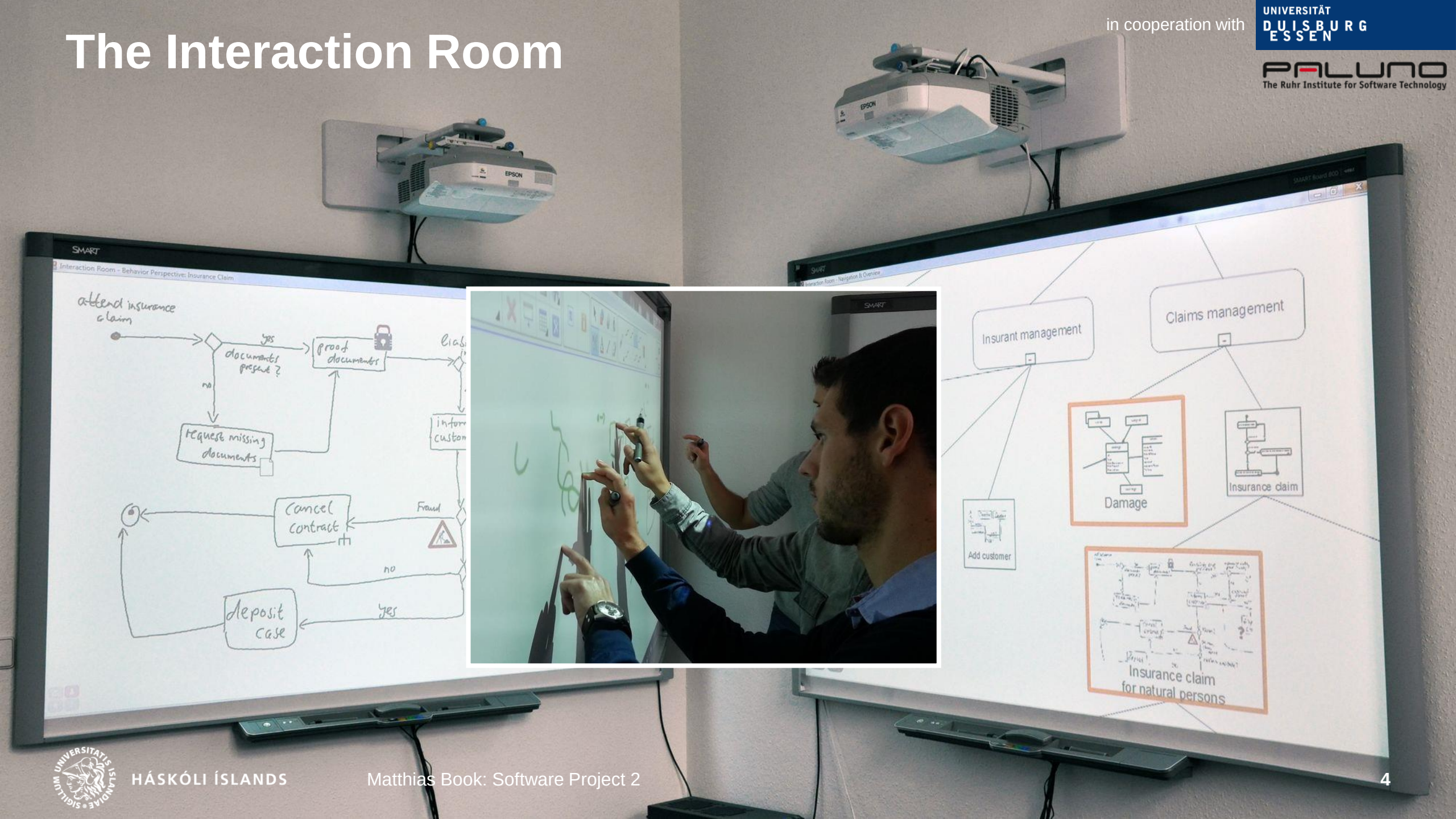
- **Interaction among software project stakeholders**
  - Facilitating effective communication between team members from heterogeneous backgrounds (business, technology, management…)
  - Identifying risks, uncertainties and value drivers early in a project, and focusing collaboration on these aspects 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

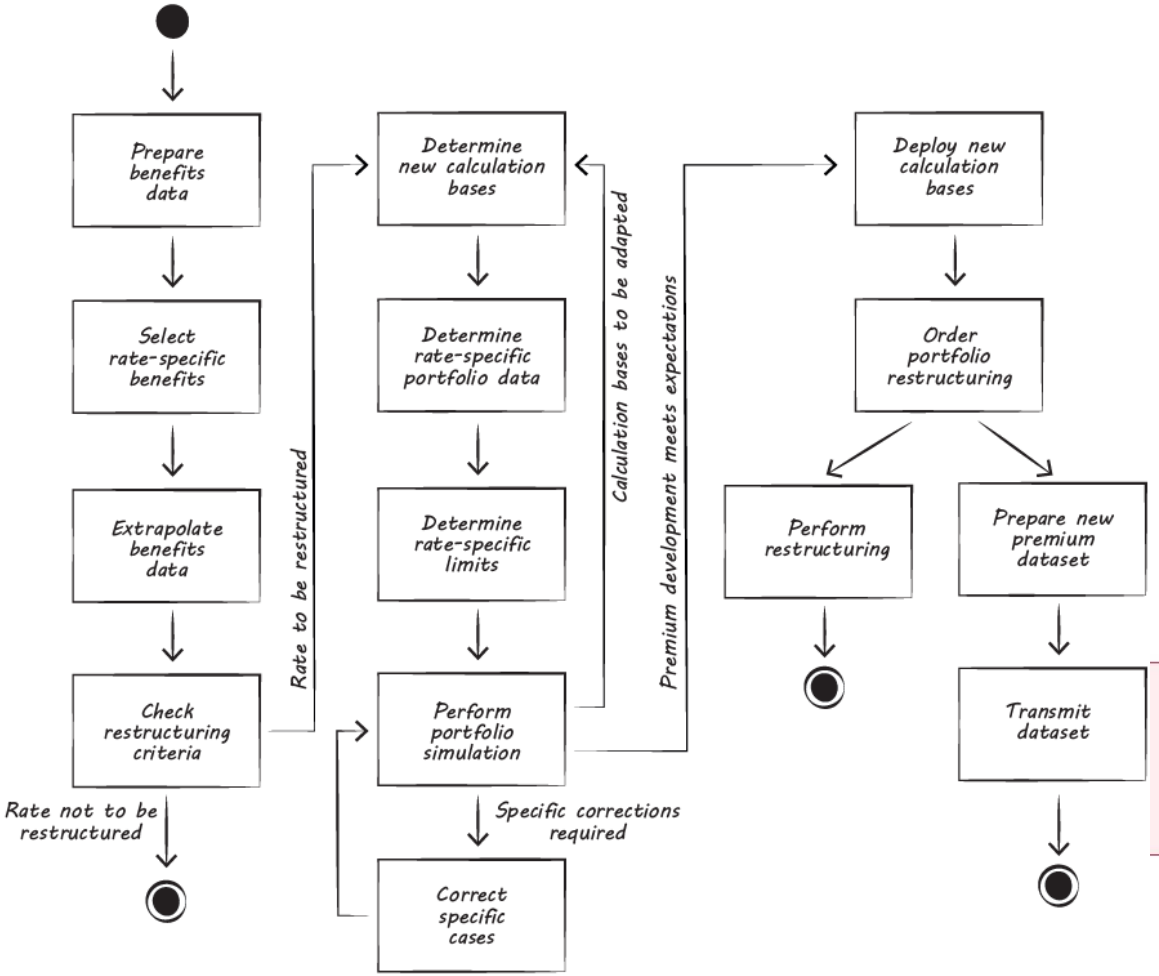
in cooperation with UNIVERSITÄT DUISBURG ESSEN

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

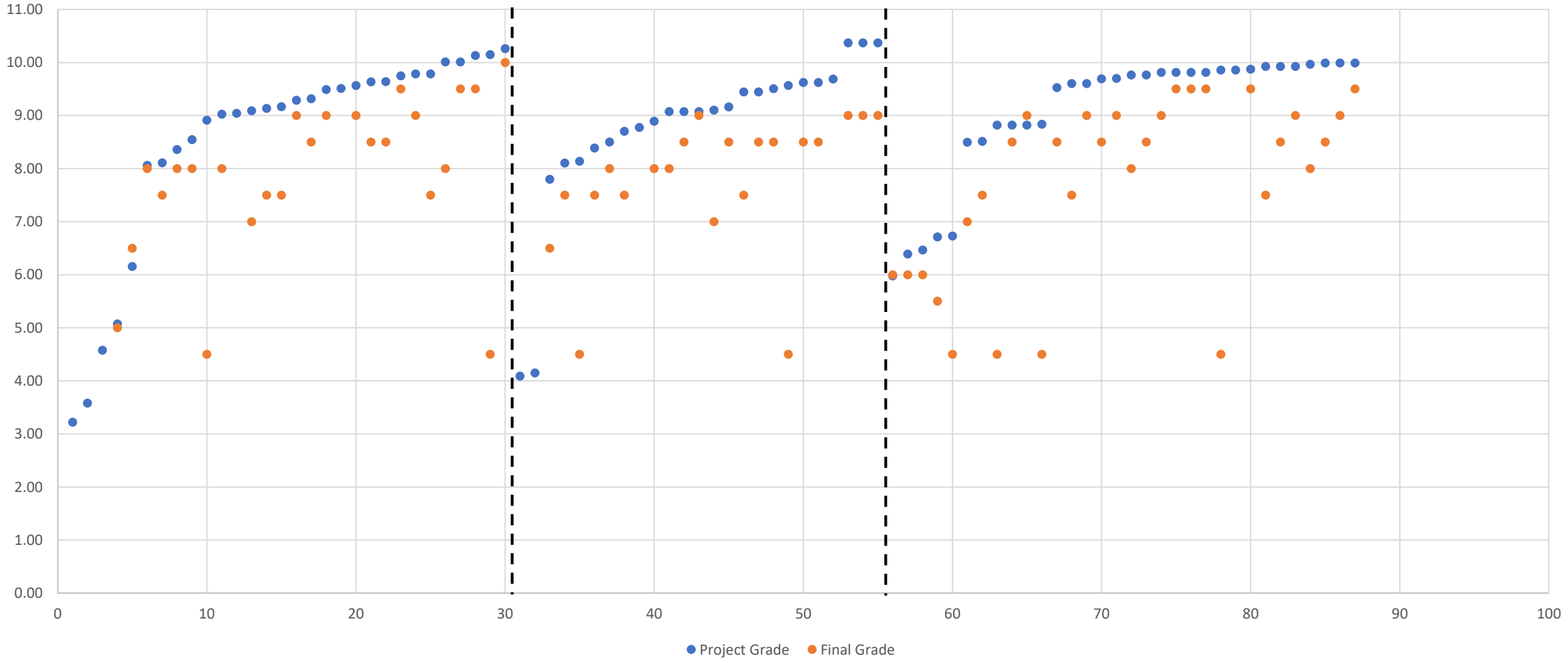
Sketch-based Code Merging

```
        <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
```

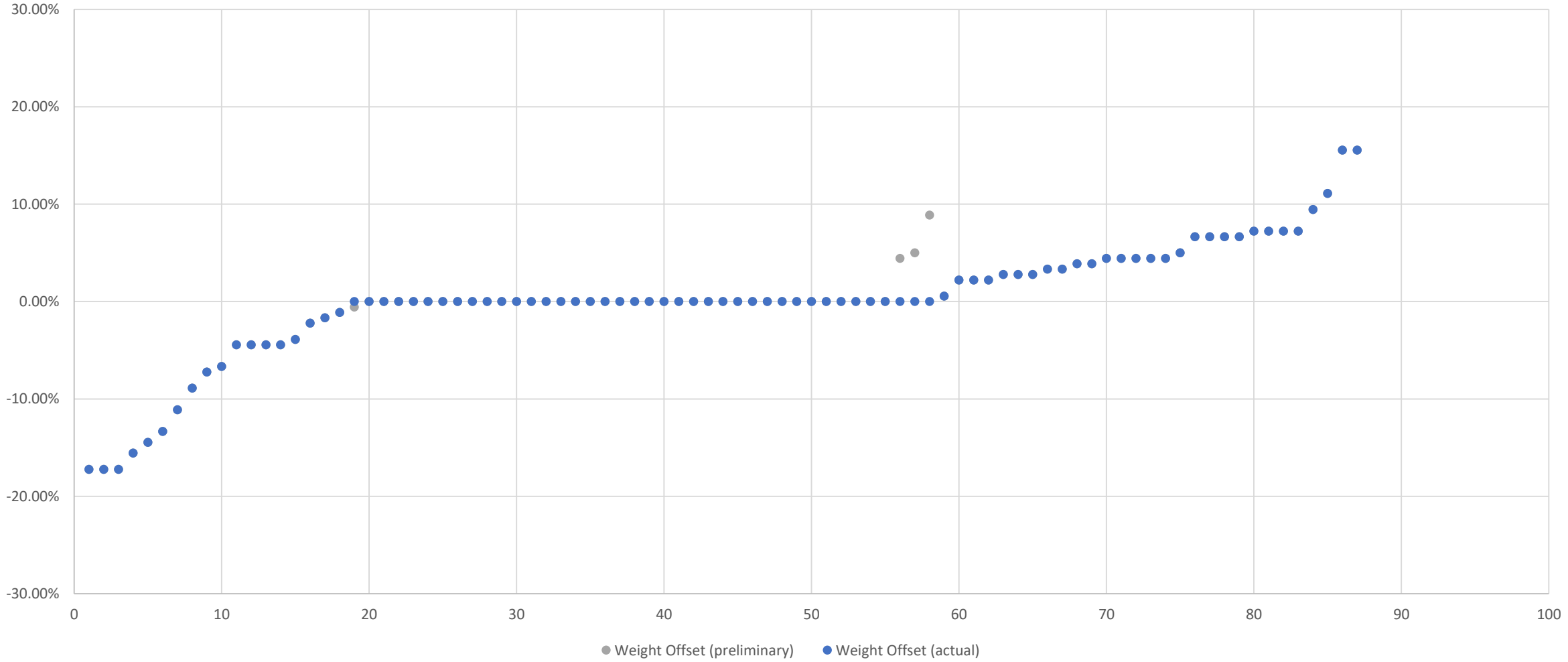
```
188                 <constraint firstAttribute=
                        "trailing" secondItem="LFs-rG-
                        Cv6"
                        secondAttribute="trailing"
                        id="w6V-ZV-c3M"/>
189             </constraints>
190         </view>
191         <connections>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

HBV501G Retrospective

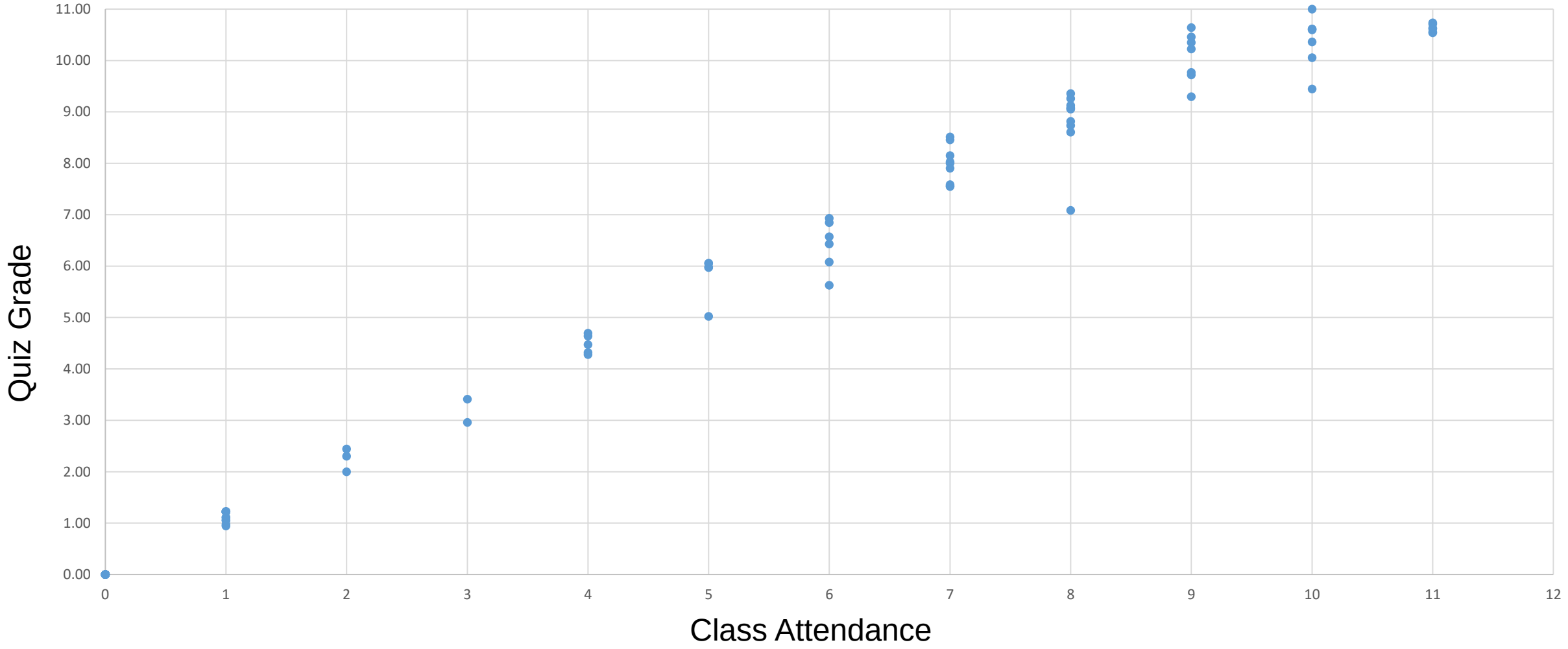
Project Grades (Grouped by Tutor)



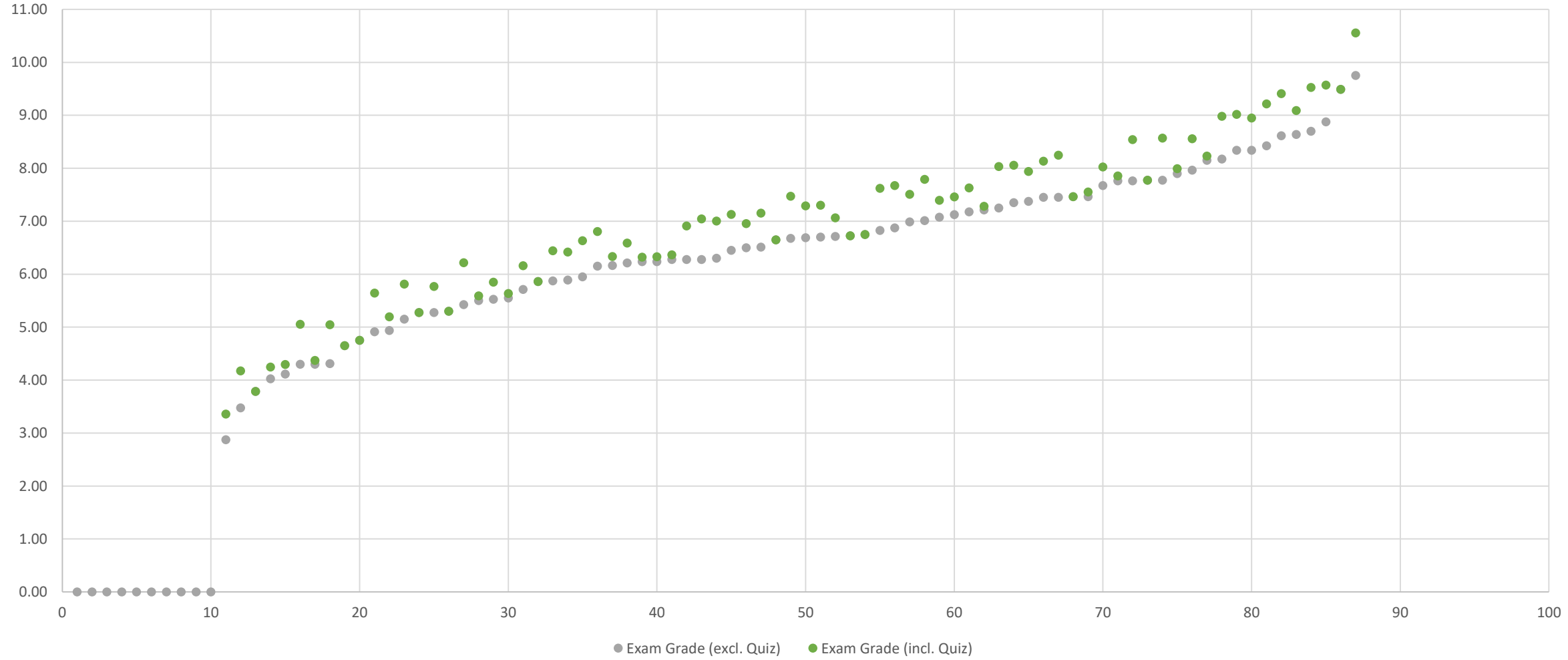
Project Weight Offsets



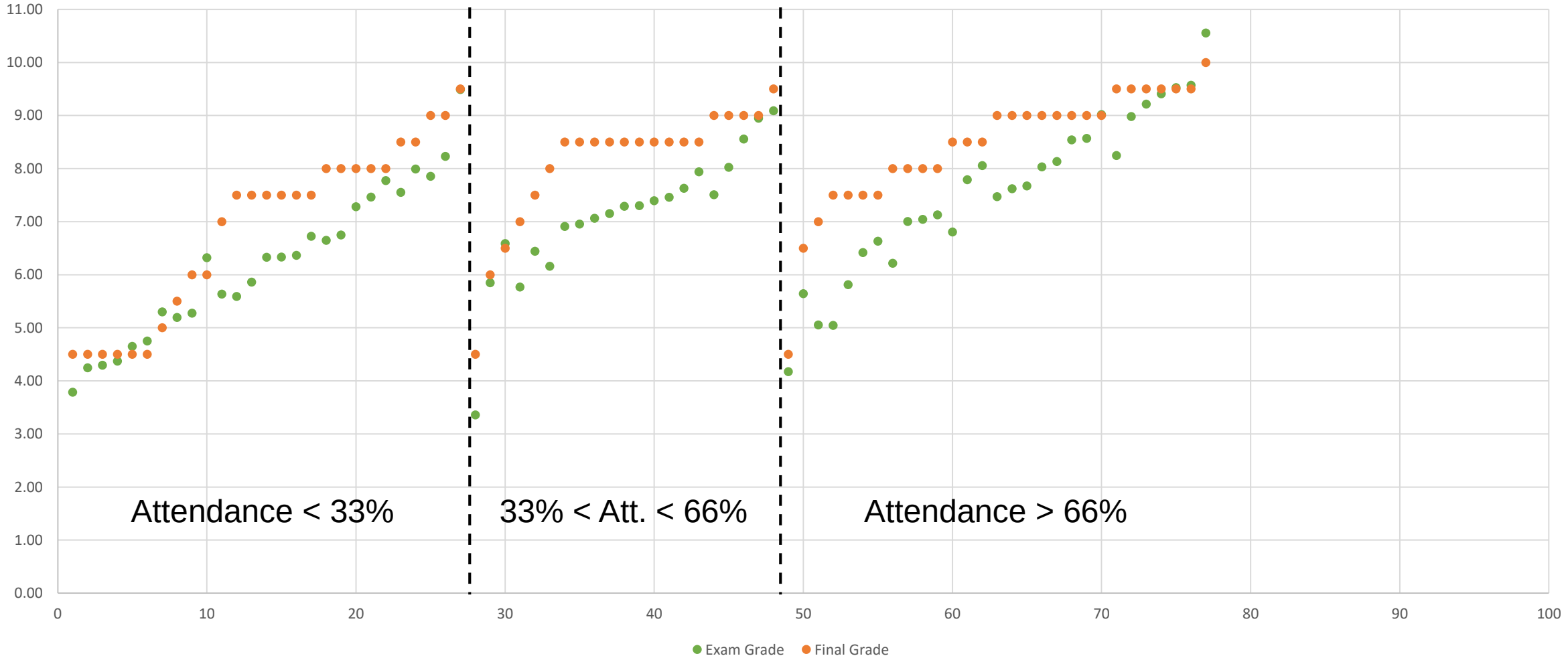
Quiz Grades by Class Attendance



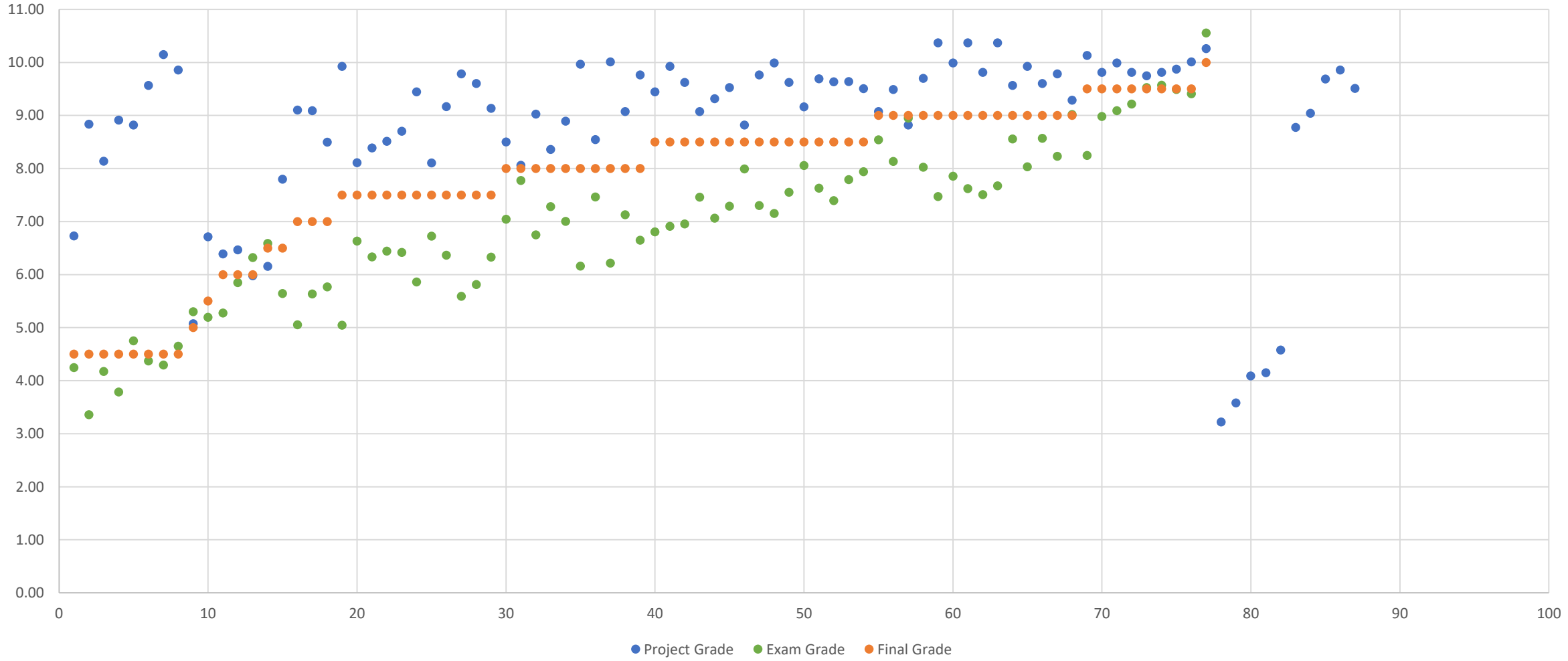
Exam Grades excl. and incl. Quiz Grades



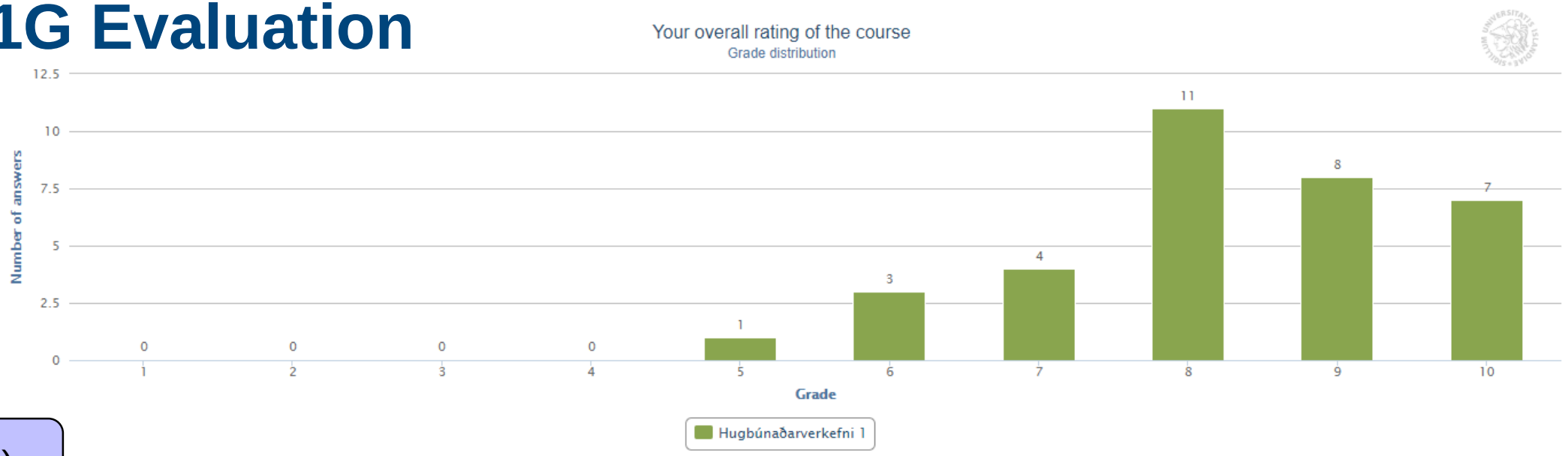
Final Grades (Grouped by Class Attendance)



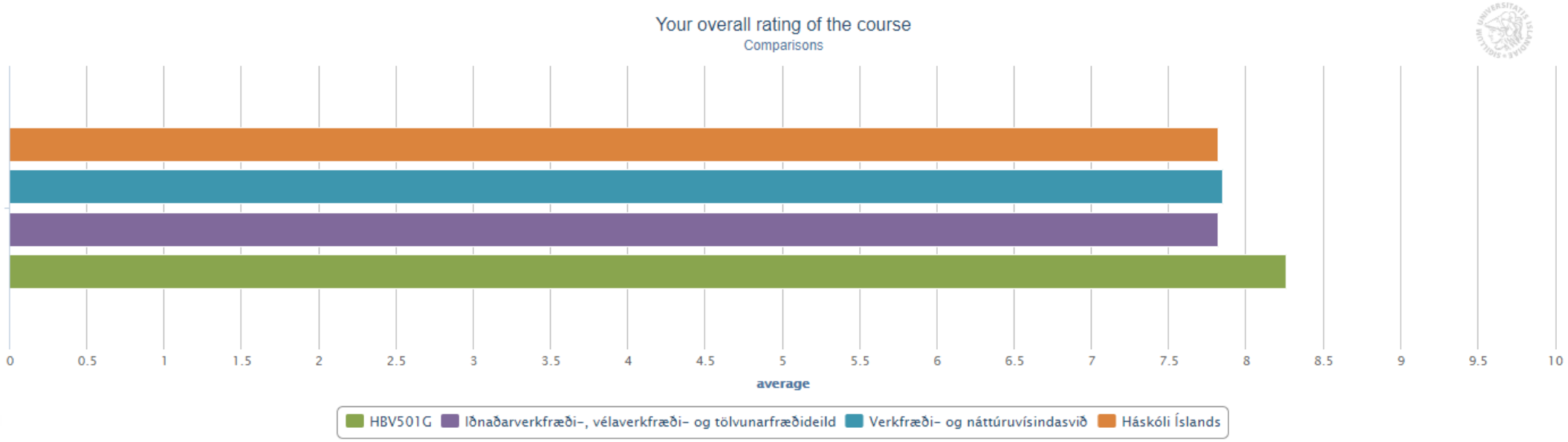
Final Grades



HBV501G Evaluation



Takk fyrir! :-)



Participation: 38/87

Your Feedback on HBV501G

- What I will do
- What you can do

- **Content-heavy lectures, full slides**
  - Slides are verbose for your benefit: exam prep, future reference
  - Attending classes nevertheless strongly encouraged
    - Higher level of cognitive involvement when attending live lecture than when watching a recording
    - Opportunity to ask questions (and quality of teaching increases the more questions are asked)
- **Workload can be frustrating if other team members don't contribute**
  - In case of problems with team members who aren't contributing, talk to them / your tutor / me early
  - Would you prefer assignment-level peer assessment of contributions?
- **Not an in-depth introduction to web programming**
  - Course focus is on software engineering methods and architectural foundations
  - Take Vefforritun 1 & 2 for a more practical introduction to current web frameworks
  - HBV601G will be more Android-specific as we don't have another explicit course for this

Earlier HBV501G Feedback Applicable to HBV601G

- What I will do
- What you can do

- **Effort is concentrated towards end of course**
  - Schedule your project to spread development effort across whole semester
  - You can (and should) start prototyping even before your design documents are complete
  - I'll let you set your own assignment deadlines this semester

- **Assignment weights inconsistent with development effort**
  - Mostly a software engineering, not a programming class
    - You don't need to demonstrate that you can build software, but that you can do it methodically
  - Weighing the final (product) assignment more heavily would have several disadvantages:
    - Would incentivize even more focus on effort towards the end
    - Would place highest weight on assignment least indicative of learning outcomes
  - See implementation effort as opportunity to
    - learn and practice a new technology
    - gain experience in managing effort in a software project

HBV601G Syllabus

This Course in the Big Picture

- 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)**

- **Don’t skip HBV401G and HBV501G before taking HBV601G!**
  - Knowledge and project experience from both classes is expected
  - HBV402G (Software Development A) is not equivalent preparation
  - HBV501G and HBV601G are designed to be taken in sequence (continuous project)
  - You won’t have enough time this semester to accomplish required project work for HBV401G and HBV601G in parallel

Course Scope

- **Agile development in industrial context**
  - Agile processes
  - Agile requirements engineering
  - Agile software estimation
  - Agile contracts
- **Mobile software development**
  - Designing for mobile platforms
  - Developing Android apps
- **Brown-field software development**
  - Adapting and extending existing code

- 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 lets you practice **agile** methods in a new context

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

- 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

Unique Challenges of Mobile Software Development

- Example: Facebook's mobile device testing lab
  - "These are about 60 self-contained racks that house 32 phones each for testing new versions of Facebook's apps right on the device."

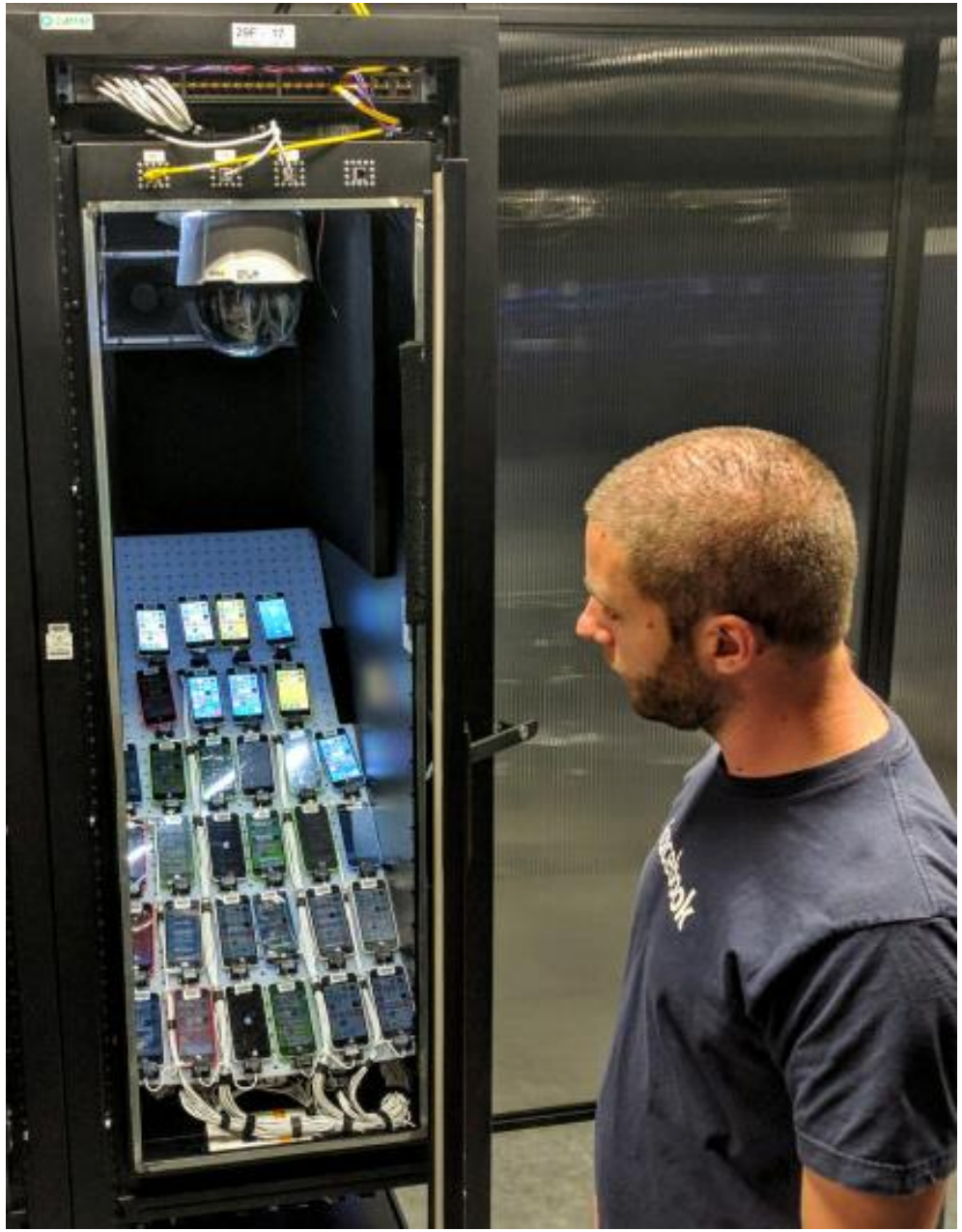


Photo: Frederic Lardinois: A look inside Facebook's data center. https://techcrunch.com/gallery/a-look-inside-facebooks-data-center/slide/26/

Course Schedule (tentative)

Wk	Thu AM: Lectures	Thu PM: Consultations	Sun: Assignments due
1	Introduction		
2	Requirements Elicitation	Sprint 1	
3	Software Estimation	Req. & Plan Draft	#1: Requirements & Project Plan (27 Jan)
4	Software Estimation	Req. & Plan Presentation	
5	Android Basics	Sprint 2	
6	Android Control Flow	Sprint 3	#2: OO Design Model (in Feb)
7	Android User Interfaces	Sprint 3	
8	Android Fragments	Sprint 4	
9	Android Storage	Sprint 4	
10	*Guest Lecture?*	Sprint 5	#3: Code Review (in Mar)
11	Android Networking	Sprint 5	
12	Android Threading	Sprint 6	
13	Software Architecture	Sprint 6	
14	Final Exam Prep	Final Presentations	#4: Final Product (14 Apr)

Course Structure

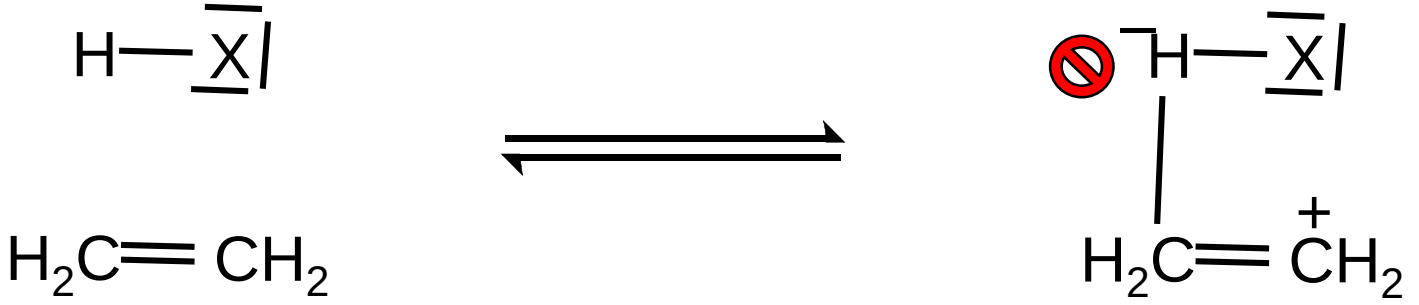
- **Lectures**
  - Thursdays 08:30-09:50, Askja 132
- **Team consultations**
  - Weekly meetings with tutors
    - to discuss scope, design, progress
    - to present assignment deliverables
  - Thursdays 13:20-16:30 (from 17 Jan), VR-II 152
    - Teams 1-10: 13:20-14:20
    - Teams 11-20: 14:25-15:25
    - Teams 21-30: 15:30-16:30
- **Team meetings**
  - Scheduled freely among team members
- **Project teams**
  - 3-4 students per team
  - Recommended (but not required) to work in last semester's teams
- **Project topics**
  - e.g. a service, community, game, etc.
  - Recommended (but not required) to continue working on same project
- **Basic architectural requirements**
  - Build a mobile app
  - Native Android (not browser-based)
  - Highly recommended to build on last semester's server-side code

In Case You Want to Change Teams

- Current team assignments are on Doodle (https://doodle.com/poll/v56843zigu2vkbub)
  - **No need to do anything if your team stays as it is**
  - To change your team 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.
- Notes:
  - Max. 4 members per team; 3-4 recommended
  - Your team number determines your consultation time slot (see previous slide)
  - If you can't find team members offline:
    - If you are early: Sign up for an empty team and wait for people to join
    - If you are late: Join a team that has only 1-2 members
    - If you can find only teams 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
- Make any changes by **next Monday (14 Jan)**
  - Formation of any new teams needs to be **finalized by next Thursday (17 Jan)** at the latest
  - So you can start working together in next week's consultations

Organic Chemistry Project Suggestion

- Educational app to practice understanding of electron bonds in molecules
  - In collaboration with Benjamín Ragnar Sveinbjörnsson, Asst. Prof. of Organic Chemistry
  - Contact me at book@hi.is if interested



Assignments

- 4 team assignments:
  1. Project plan & requirements: due Sun 27 Jan
  2. Design model: due on a Sun in Feb (scheduled by team)
  3. Code review: due on a Sun in Mar (scheduled by team)
  4. Final product: due Sun 14 April

- **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

Assignment 1: Project Plan and Requirements

- By **Sun 27 Jan**, submit a **project outline** in Ugla, containing:
  - Vision Statement (see template in Sect. 1.5 of RUP Vision & Scope document)
  - Choice of process model (agile or plan-driven)
  - Product backlog or brief-format use case document with effort estimates and priorities
  - Project schedule (dates for sprints/iterations, milestones, assignments 2 and 3)

- On **Thu 31 Jan**, present and **explain** your project outline to your tutor:
  - What influenced your choice of scope? What considerations shaped your planned schedule?

- **Grading criteria:**
  - Vision statement is clear and plausible (25% of this assignment's grade)
  - Product backlog/use case doc describes features clearly and with realistic scope (50%)
  - Project schedule is clear, realistic, and specifies dates for assignments 2 and 3 (25%)

Assignment 2: Design Model

- By the deadline specified in your project plan, submit a **design model** in Ugla:
  - UML class diagram of your system (detail level reflecting state of implementation/planning)
  - Suitable UML behavioral diagrams to show
    - User navigation in your app
    - Control flow between key components of your app

- On the Thursday after submission, present and **explain** the model to your tutor:
  - How will your system work? What influenced your design choices? What is still unknown?

- **Grading criteria** (25% of this assignment's grade each):
  - System structure is plausible, consistent with requirements and behavioral diagrams
  - User navigation is plausible and shown in a suitable diagram
  - Control flow is is plausible and shown in a suitable diagram
  - UML diagrams are clean and syntactically correct

Assignment 3: Code Review

- Find a team you would like to pair up with, and agree on a mutual Code Review date
- By the deadline specified in both of your project plans,
  make the **project artifacts** you created so far available to each other:
  - Your project outline and design model from previous assignments (incl. fixes of identified issues)
  - A current snapshot of your source code
- Take 1 week to **review** the other team's code and briefly **document** your findings:
  - Examine how they structured their system. Do outline and design model help to understand it?
  - Make suggestions for improvement in object-oriented design, technology use and coding style
- On the Thursday 1 week later, **discuss** your findings with other team and tutor:
  - What did you like? Which questions did you run into? What would you recommend to change?
- **Grading criteria:**
  - Quality of constructive feedback on other team's code (70% of this assignment's grade)
  - Design and technology issues identified in your code (15%)
  - Coding style / clarity issues identified in your code (15%)

Assignment 4: Final Product

- On **Thu 11 Apr**, **demonstrate** and **explain** your product to your classmates:
  1. **Product:** What does your product do? Demonstrate the key features of your system.
  2. **Architecture:** How does your product work? Explain architecture & key design decisions.
  3. **Process:** How did you build the product? Relate and interpret challenges you faced.

  (Precise format and logistics of presentations to be announced later in semester)

- On **Sun 13 Apr**, submit your **final product** in Ugla, including:
  - Complete source code and installation instructions
  - Slides of your final presentation

- **Grading criteria:**
  - Final product satisfies requirements defined in plan, and running smoothly (75%)
  - Critical retrospective given of chosen process, architecture and technology (25%)

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** (also not for the assignment 2 & 3 deadlines in your project plan!)
  - Undefined grace period until whenever tutors happen to 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 **artifacts** 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 **20%** of project grade
- Over the course of the semester, each team member must lead the **presentation** of at least one assignment to the 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 ("5th assignment", weighs **20%** 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

- Peer 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. a blank A4 sheet 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; in your own words
  - 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 as usual (0…11), with 0 for any quizzes that are not handed in
    - two worst quiz grades 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

Grading Scheme

- All contributing factors are graded on a scale of 0…11
  - Grading criteria for assignment deliverables and individual presentations
  - Questions in 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

Course Support

- **Questions and feedback** welcome anytime! Preferably:
  - in class
  - in team consultations
  - in Piazza: https://piazza.com/hi.is/spring2019/hbv601g (self sign-up)

- **See Ugla for**
  - Slides (available after each lecture)
  - Recordings (provided on a best-effort basis; no quality/availability guarantees)
  - Important course announcements

- **Teaching staff**
  - Matthias Book (book@hi.is)
  - Andri Valur Guðjohnsen (avg4@hi.is)
  - Daníel Páll Jóhannsson (dpj@hi.is)
  - Sigurður Gauti Samúelsson (sgs31@hi.is)

Suggested Literature

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

- **Requirements Engineering**
  - Dean Leffingwell: Agile Software Requirements. Addison Wesley, 2011
  - Karl Wiegers, Joy Beatty: Software Requirements. Microsoft Press, 2013
- **Software Estimation and Planning**
  - Mike Cohn: Agile Estimating and Planning. Prentice Hall, 2005
  - Steve McConnell: Software Estimation – Demystifying the Black Art. Microsoft Press, 2006
- **Android Development**
  - ❖B. Phillips, C. Stewart, K. Marsicano: Android Programming (3rd ed.). Big Nerd Ranch, 2016
- **Object-oriented Analysis and Design**
  - E. Gamma, R. Helm, R. Johnson, J. Vlissides: Design Patterns. Addison-Wesley, 1995
  - Eric Freeman, Bert Bates: Head First Design Patterns. O'Reilly, 2004
- **Software Architecture**
  - L. Bass, P. Clements, R. Kazman: Software Architecture in Practice. Addison-Wesley, 2013
  - Martin Fowler: Patterns of Enterprise Application Architecture. Addison-Wesley, 2003

Let’s get started!

book@hi.is

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

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

Recap: Ways of Structuring that Social Learning Process

Plan-driven Iterative Development

- Gain understanding through communication and specification
- Risk management through planning
- Aspiration for stable structures
- Optimization by spec refinement
- Work on most risky features first
- Increments are partial systems
- Stable overall target vision
- Well-defined roles and responsibilities
- Discipline required to follow plans

Agile Iterative Development

- Gain understanding through communication and feedback
- Risk management through flexibility
- Acceptance of fluid structures
- Optimization by code refactoring
- Work on most valuable features first
- Increments should be viable products
- Open overall target vision
- Self-organizing teams
- Discipline required to utilize freedom