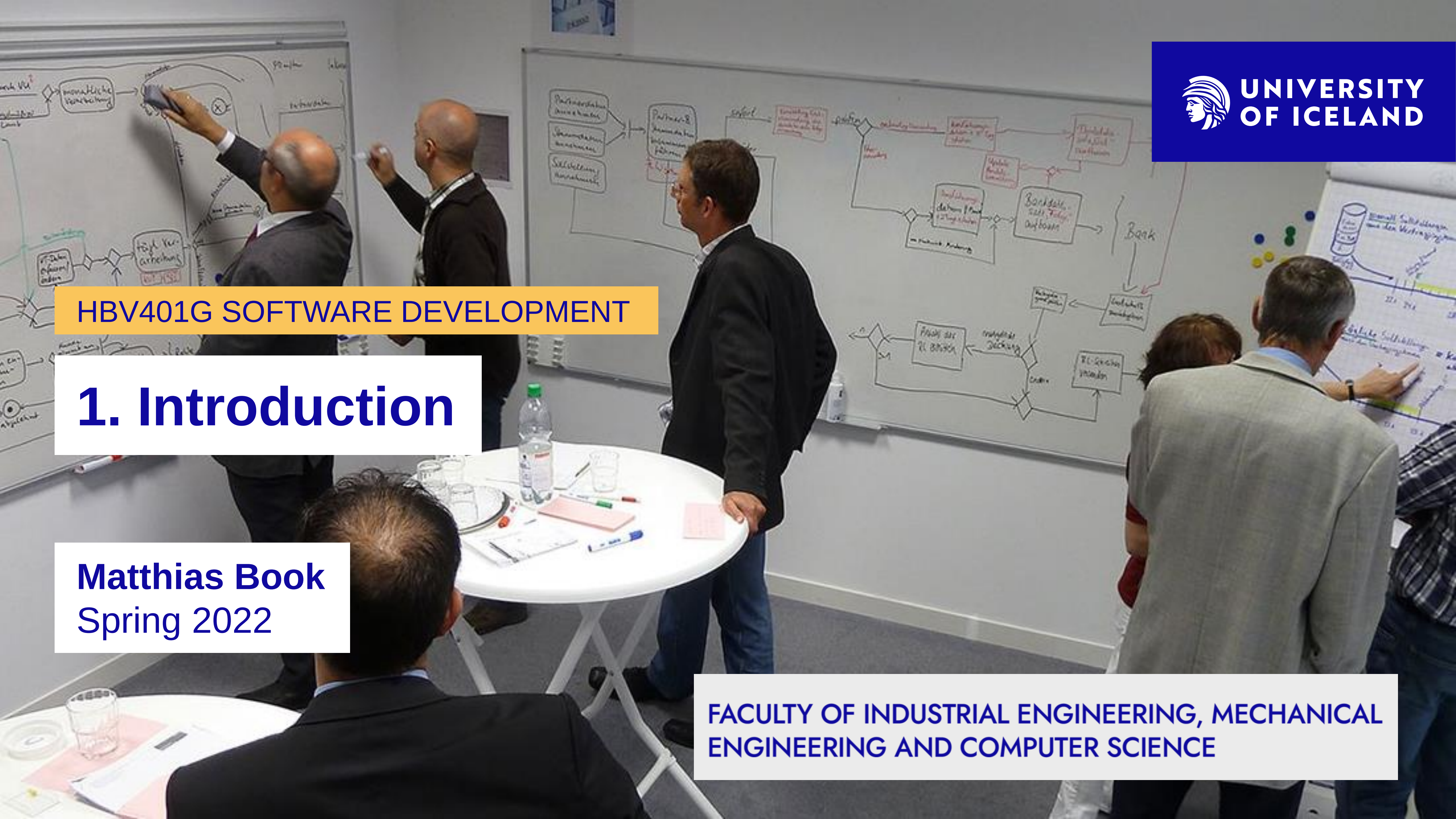


UNIVERSITY OF ICELAND

HBV401G SOFTWARE DEVELOPMENT

1. Introduction

Matthias Book
Spring 2022

FACULTY OF INDUSTRIAL ENGINEERING, MECHANICAL ENGINEERING AND COMPUTER SCIENCE

Privacy in Class Streams and Recordings

UNIVERSITY OF ICELAND

- While your microphone is on (unmuted), your **audio** will be broadcast to other participants and included in the class recording.
- While your camera is on, your **video** is visible to other participants in the Participant Gallery, and might be included in the class recording while you are speaking.
- While your camera is off, your **screen name** is visible to other participants in the Participant Gallery, but not visible in the class recording while you are speaking.
- **Chat messages** are visible to the addressed participants and may be saved by them. They are not part of the class recording.
- ***Privacy notice:*** *By activating your microphone and/or camera, you consent to your audio and/or video being stored as part of the class recording on Zoom, Panopto, Microsoft (Teams) and/or Instructure (Canvas) servers, and being broadcast to other participants live and when playing back the class recording. The teachers' class recordings are not available to the general public, but are only accessible to students enrolled in this course this semester. Participants may be able to record classes on their local computer, if given permission by the teacher.*

Doctor Who?

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

- **Contact information**
  - Office: Gróska 314
  - 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
  - Post-doctoral researcher and lecturer at Universities of Duisburg-Essen and Chemnitz
  - Research manager at software development company adesso SE in Dortmund
  - Teaching at University of Iceland since 2014

Research Interests

- **Multi-modal and sketch-based user interfaces**
  - Sketch-based software engineering: Understanding and manipulating software artifacts through sketches on code, models, user interfaces
  - Specifying and controlling user interactions with software systems through 2D and 3D gestures, voice commands etc.
- **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
- **Software engineering for high-performance computing**
  - Using software engineering methods and tools to efficiently develop scientific software

- MS & PhD projects on these topics available – contact me at book@hi.is

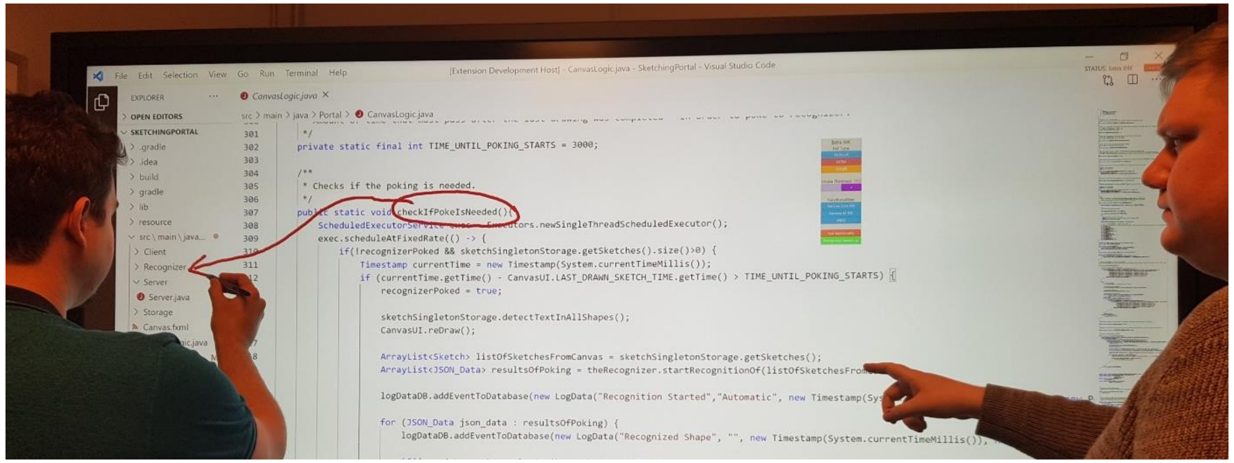
Research: Sketch-based Software Engineering

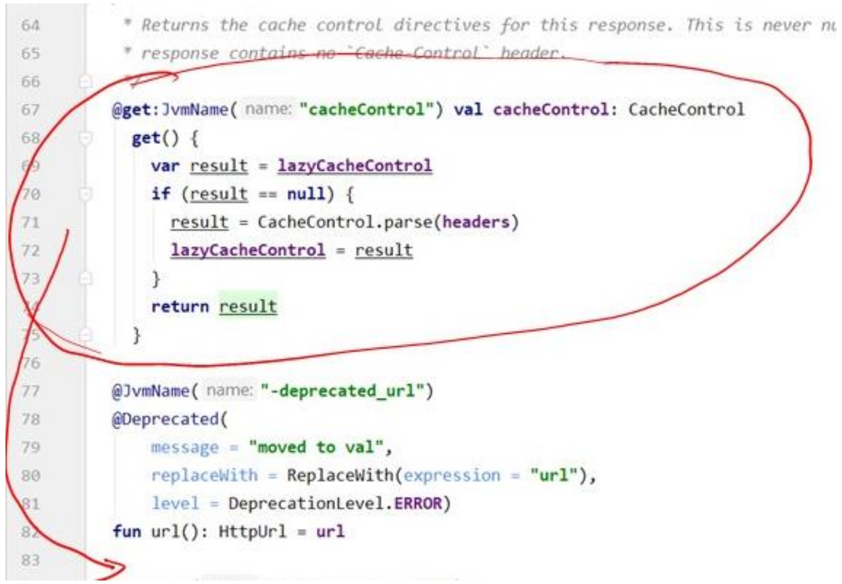
in cooperation with

supported by

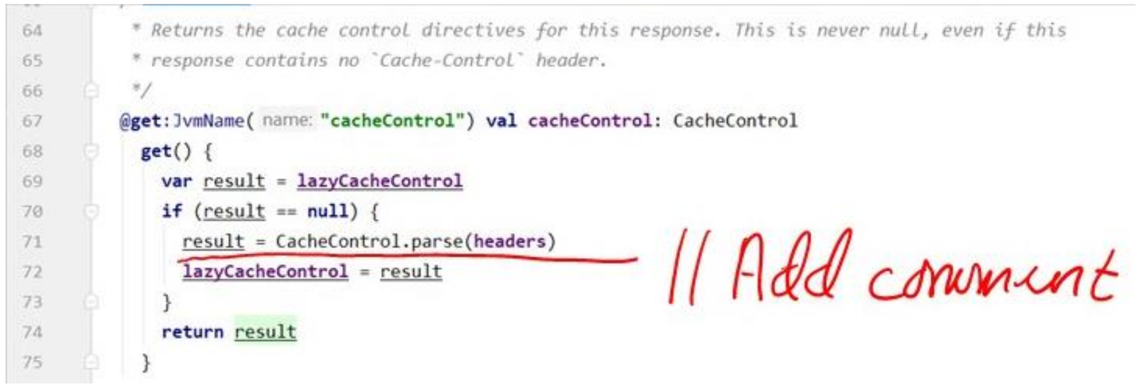
Icelandic Research Fund

Goal: Enable developers to sketch right on top of the source code in their IDE to quickly perform refactorings without using the mouse or keyboard



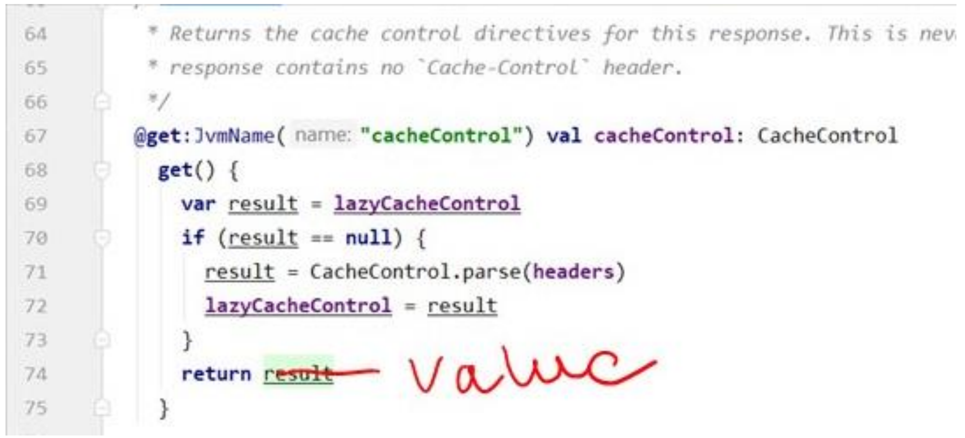


```
 * Returns the cache control directives for this response. This is never nu
 * response contains no `Cache-Control` header.

@get:JvmName( name: "cacheControl") val cacheControl: CacheControl
  get() {
    var result = lazyCacheControl
    if (result == null) {
      result = CacheControl.parse(headers)
      lazyCacheControl = result
    }
    return result
  }

@JvmName( name: "-deprecated_url")
@Deprecated(
    message = "moved to val",
    replaceWith = ReplaceWith(expression = "url"),
    level = DeprecationLevel.ERROR)
fun url(): HttpUrl = url
```



```
 * Returns the cache control directives for this response. This is never null, even if this
 * response contains no `Cache-Control` header.
 */
@get:JvmName( name: "cacheControl") val cacheControl: CacheControl
  get() {
    var result = lazyCacheControl
    if (result == null) {
      result = CacheControl.parse(headers)
      lazyCacheControl = result
    }
    return result
  }
```

// Add comment



```
 * Returns the cache control directives for this response. This is nev
 * response contains no `Cache-Control` header.
 */
@get:JvmName( name: "cacheControl") val cacheControl: CacheControl
  get() {
    var result = lazyCacheControl
    if (result == null) {
      result = CacheControl.parse(headers)
      lazyCacheControl = result
    }
    return ~~result~~
  }
```

value

Course Scope

- From programming-in-the-small…
  - **Individual developers** creating **compact modules** that solve **clearly defined problems**
- …to programming-in-the-large:
  - **Project teams** building **complex systems** that satisfy **vague customer requirements**

- Requires an expanded skill set beyond programming and algorithms…
  - ➢ **Lectures** on key software engineering concepts
    - Software process models
    - Requirements engineering
    - Object-oriented analysis and design
    - Test-driven development
- …and as much practical experience as possible
  - ➢ **Project teams** building and integrating software components

The Vasa

Photo © Hans Hammarskiöld,
Vasa Museum Stockholm, Sweden

An Ambitious Project and a Source of Pride …and a Disaster

- The warship Vasa, pride of the Royal Swedish Navy and built at the cost of 4% of the Swedish gross national product, was launched in Stockholm harbor on 10 August 1628.

- After sailing just a few hundred meters, a small gust of wind capsized her, and she sank to the bottom of the harbor with 53 souls on board, to remain there for the next 333 years.

Photo © Hans Hammarskiöld,
Vasa Museum Stockholm, Sweden

The Vasa
Reasons for Failure

- **Excessive schedule pressure**
  - The Vasa was completed under strong time constraints to meet a pressing need.
- **Changing needs**
  - Many changes to operational characteristics were made during construction of the ship (extended keel, additional gun deck, variations in armaments).
- **Lack of a documented project plan**
  - During a year-long transition in leadership, assistant had difficulty managing the project.
  - This resulted in poor supervision of the various groups working on the ship (400 people: shipwright and ship builder; hull, carvings, riggings, armaments, ballasting subcontractors).
  - There is no evidence that the new project manager (the former assistant) prepared any plans after the original shipwright died.

R.E. Fairley, M.J. Willshire: Why the Vasa Sank: 10 Problems and Some Antidotes for Software Projects. IEEE Software, March/April 2003, pp. 18-25

The Vasa
Reasons for Failure

- **Lack of technical specifications**
  - The specifications were not revised as the operational requirements changed.
- **Excessive innovation**
  - No one in Sweden, including the shipwright, had ever built a ship having two gun decks.
- **Secondary innovations**
  - Many secondary innovations were added during construction of the Vasa to accommodate the longer keel, the additional gun deck, and other changes.
- **Requirements creep**
  - It seems that no one was aware of the degree to which the Vasa had evolved during the 2.5 years of construction.

R.E. Fairley, M.J. Willshire: Why the Vasa Sank: 10 Problems and Some Antidotes for Software Projects. IEEE Software, March/April 2003, pp. 18-25

The Vasa
Reasons for Failure

- **Lack of scientific methods**
  - There were no known methods for calculating center of gravity, stiffness, and the resulting stability relationships of the Vasa.
- **Ignoring the obvious**
  - The Vasa was launched after failing a stability test. Its inherent (and unfixable) instability should have been obvious from design parameters even earlier.
- **Communication breakdowns**
  - Results of the stability test were known to some but not communicated to others.
- **No lessons learned**
  - At subsequent hearings, nobody asked how or why the ship had become unstable, or why it was launched with known stability problems.

R.E. Fairley, M.J. Willshire: Why the Vasa Sank: 10 Problems and Some Antidotes for Software Projects. IEEE Software, March/April 2003, pp. 18-25

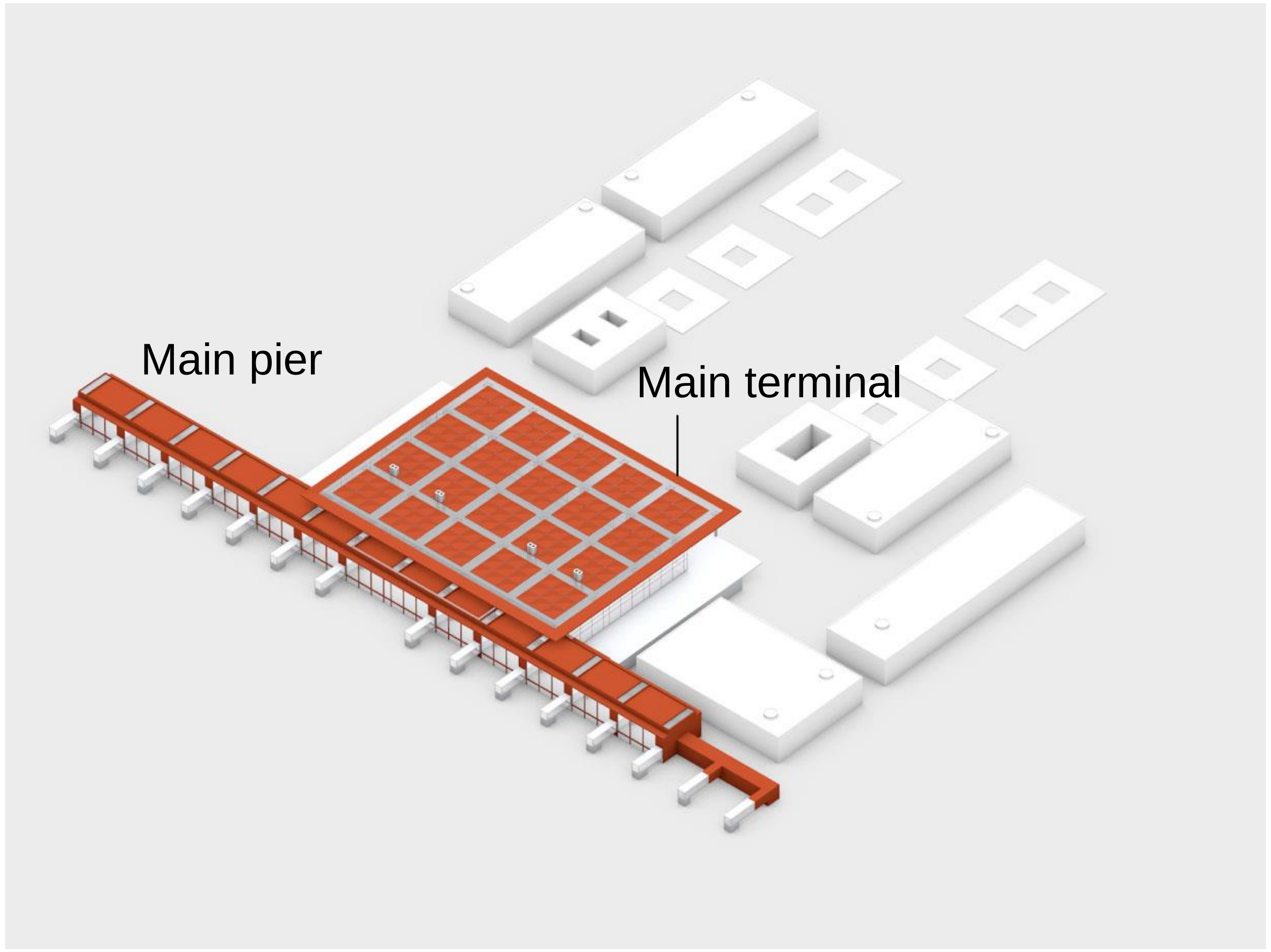
A Modern Example: Berlin Brandenburg Airport (BER)

360,000 m^2 – one of the largest buildings in Europe

Image: DPA

A Modern Example: Berlin Brandenburg Airport

- April 2007
  - Area: 220.000 m^2
  - Projected cost: 490 Mio. €
  - Scheduled opening: Oct 2011
- Construction permit
  for main terminal and pier

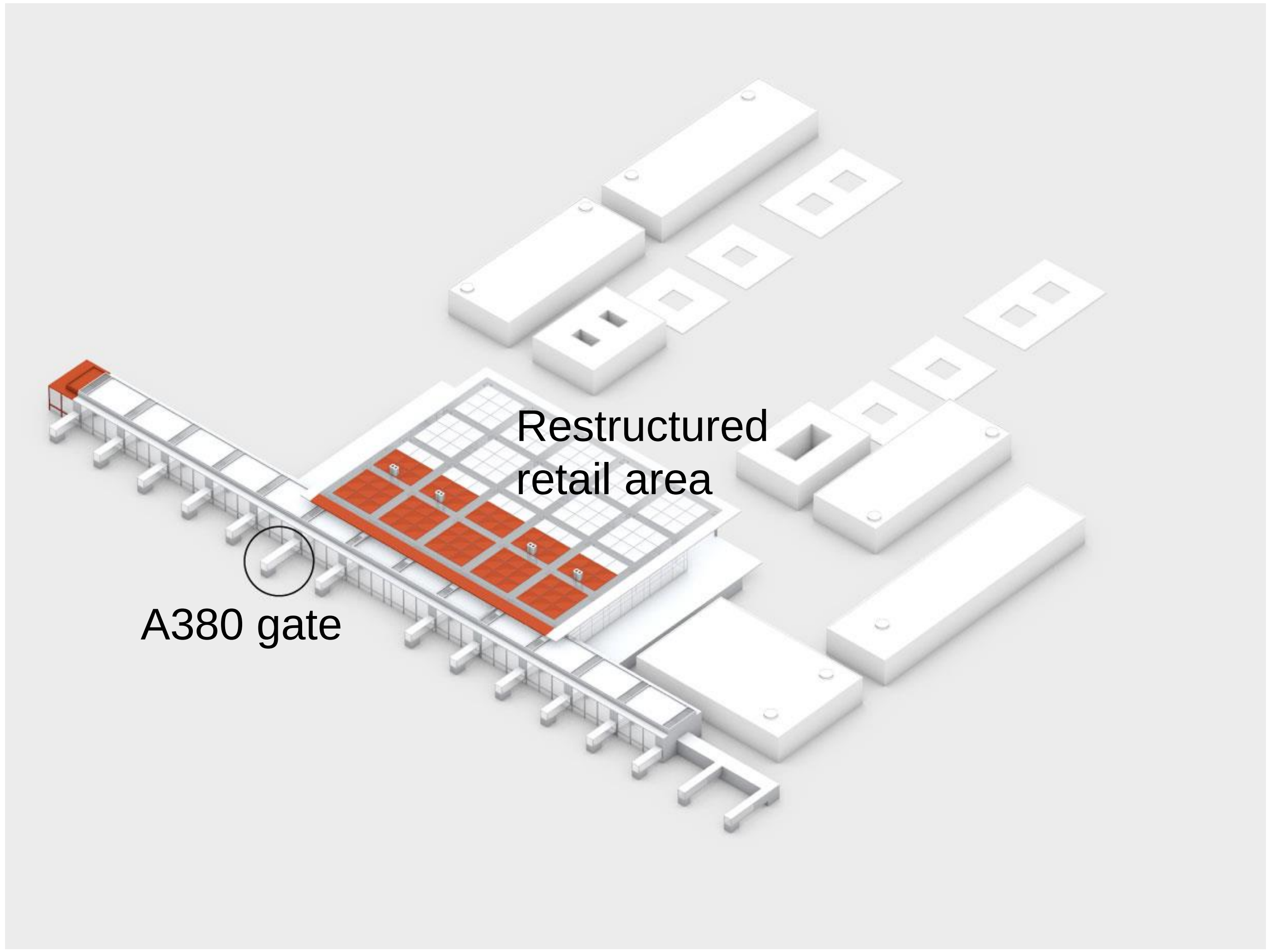


https://www.spiegel.de/wirtschaft/flughafen-berlin-brandenburg-wie-der-ber-zum-pannenflughafen-verkam-a-d007c890-1fd3-44f0-8f02-2d7c76f09e47

A Modern Example: Berlin Brandenburg Airport

- July 2007
  - Area: 230.000 m^2
  - Projected cost: 520 Mio. €
  - Scheduled opening: Oct 2011

- Additional A380 gate
- Restructured, larger airside retail area including luxury stores
- North extension of main pier



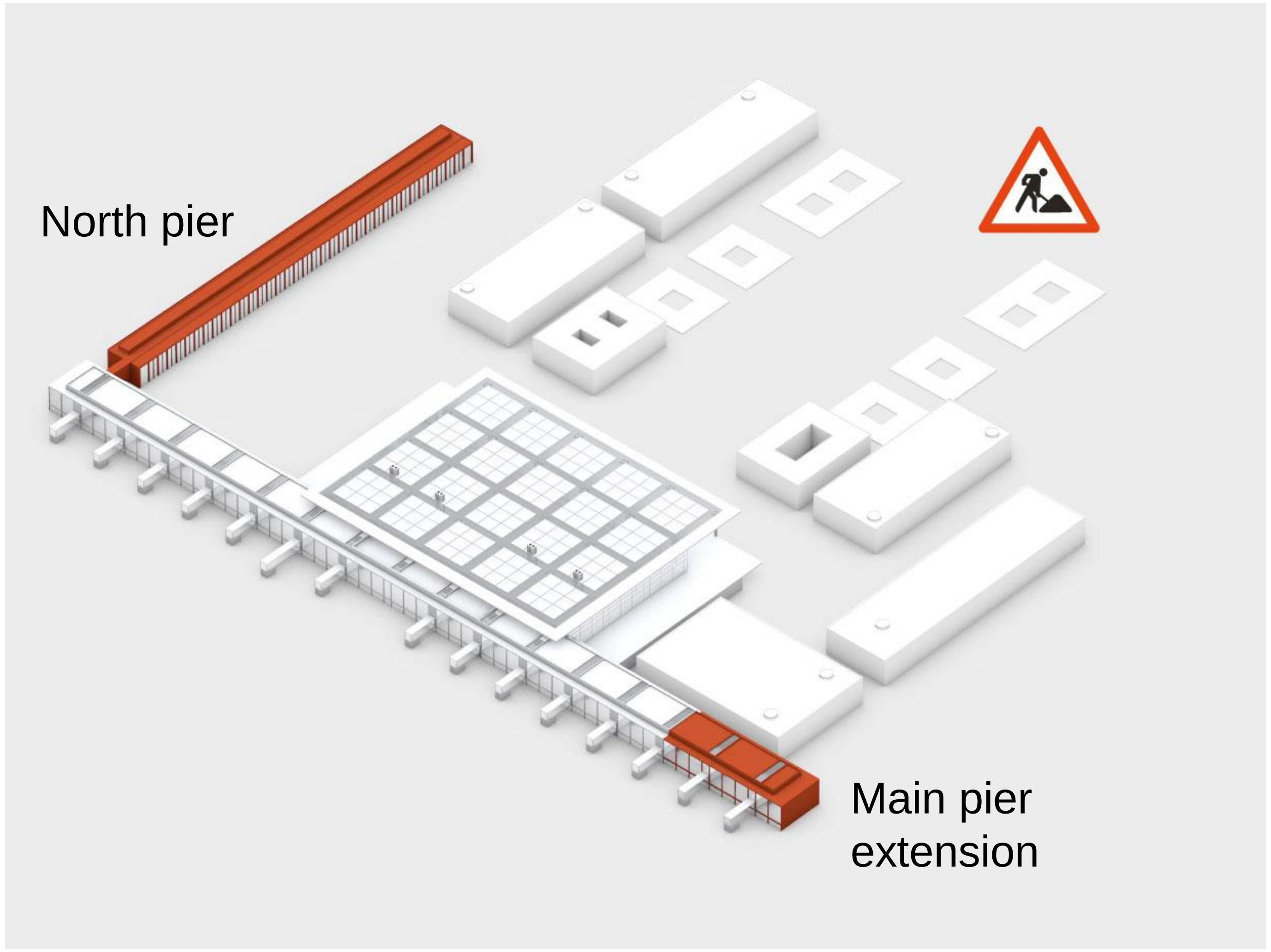
https://www.spiegel.de/wirtschaft/flughafen-berlin-brandenburg-wie-der-ber-zum-pannenflughafen-verkam-a-d007c890-1fd3-44f0-8f02-2d7c76f09e47

A Modern Example: Berlin Brandenburg Airport

- December 2007
  - Area: 278.000 m^2
  - Projected cost: 630 Mio. €
  - Scheduled opening: Oct 2011

- Additional north pier
- South extension of main pier

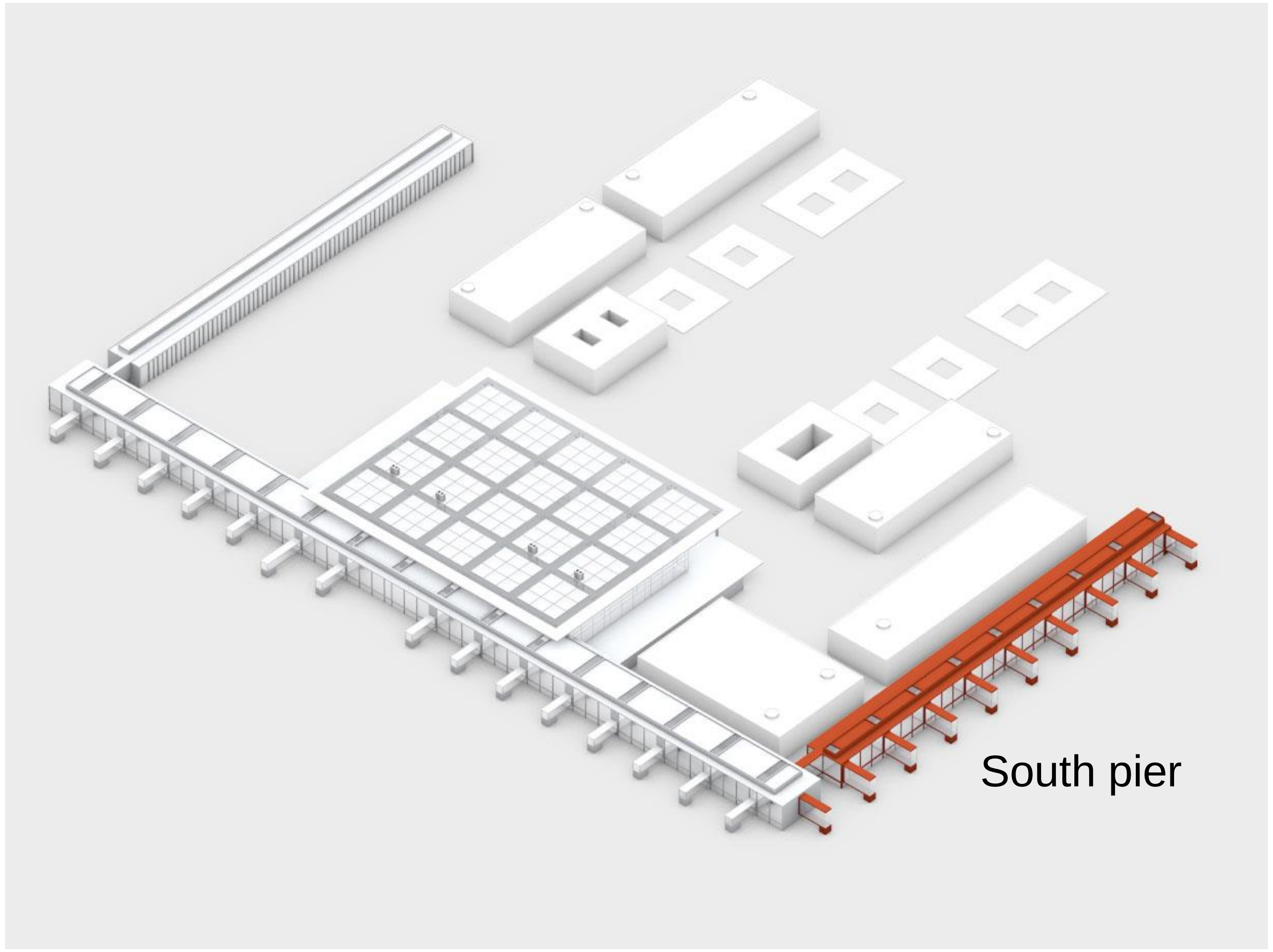
- Construction begins



https://www.spiegel.de/wirtschaft/flughafen-berlin-brandenburg-wie-der-ber-zum-pannenflughafen-verkam-a-d007c890-1fd3-44f0-8f02-2d7c76f09e47

A Modern Example: Berlin Brandenburg Airport

- October 2008
  - Area: 300.000 m^2
  - Projected cost: 720 Mio. €
  - Scheduled opening: Oct 2011
- Additional south pier

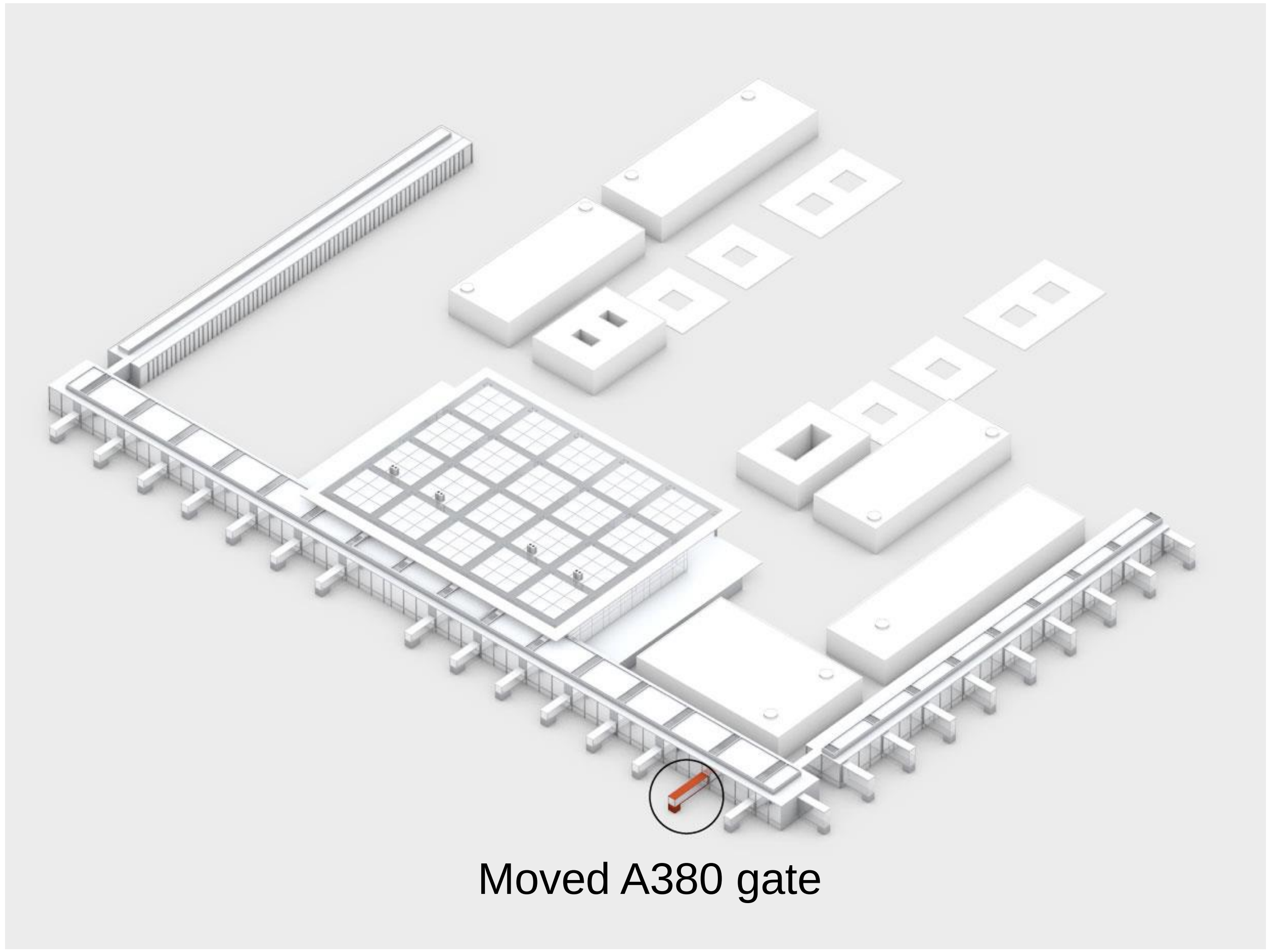


https://www.spiegel.de/wirtschaft/flughafen-berlin-brandenburg-wie-der-ber-zum-pannenflughafen-verkam-a-d007c890-1fd3-44f0-8f02-2d7c76f09e47

A Modern Example: Berlin Brandenburg Airport

- 2009
  - Area: 306.000 m^2
  - Projected cost: 800 Mio. €
  - Scheduled opening: Oct 2011

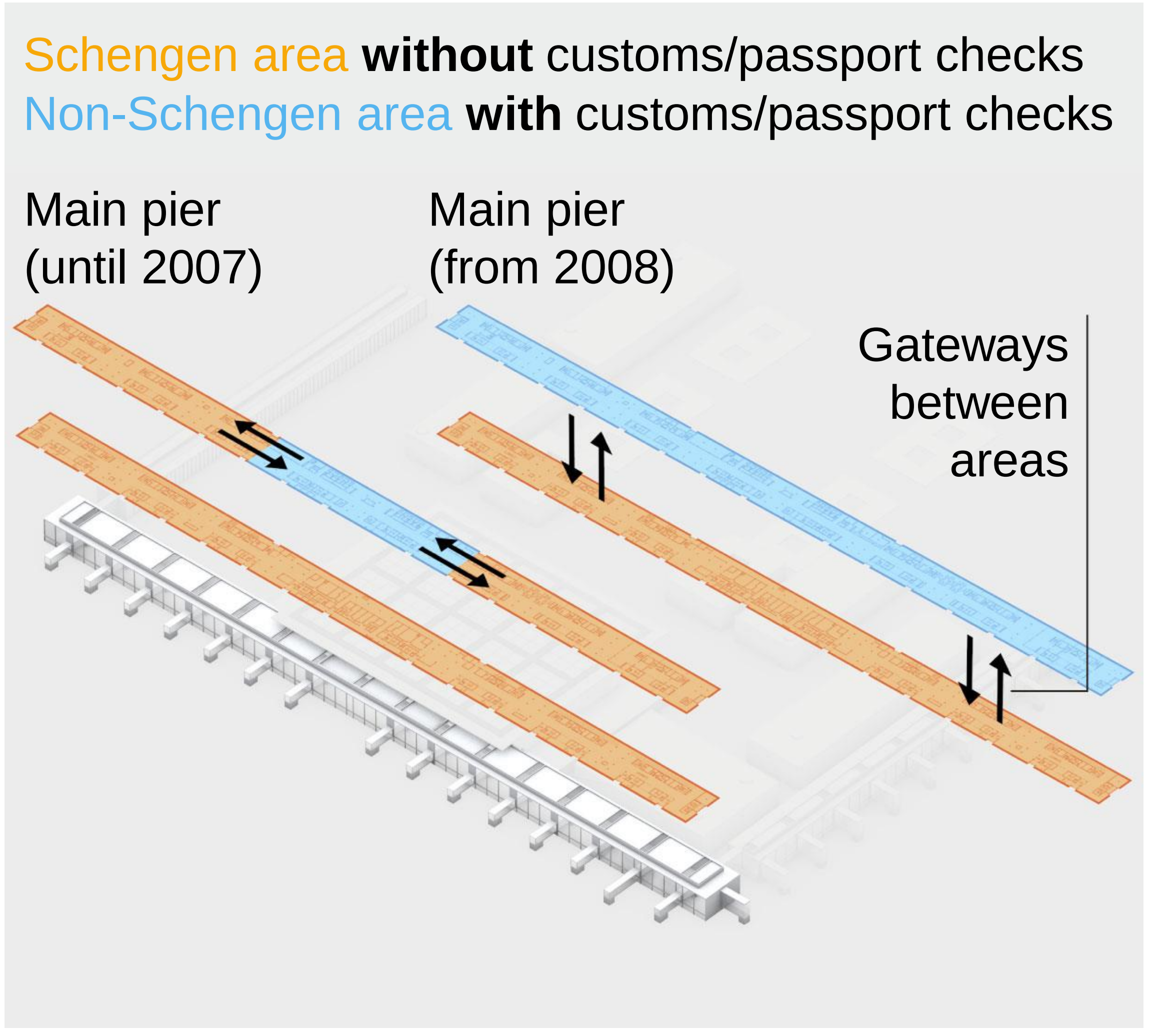
- A380 gate moved to south end of main pier in anticipation of Air Berlin's plans to fly the A380 from Berlin
  - Major restructuring of retail area, ventilation, power, plumbing etc.



https://www.spiegel.de/wirtschaft/flughafen-berlin-brandenburg-wie-der-ber-zum-pannenflughafen-verkam-a-d007c890-1fd3-44f0-8f02-2d7c76f09e47

A Modern Example: Berlin Brandenburg Airport

- 2009
  - Area: 306.000 m^2
  - Projected cost: 800 Mio. €
  - Scheduled opening: Oct 2011
- Extension of Non-Schengen area to whole length of upper main pier
  - Complete reorganization of passenger flows
  - Replanning of fire protection, escape routes
- Two-level jet bridges
- Design, certification and construction begin to overlap

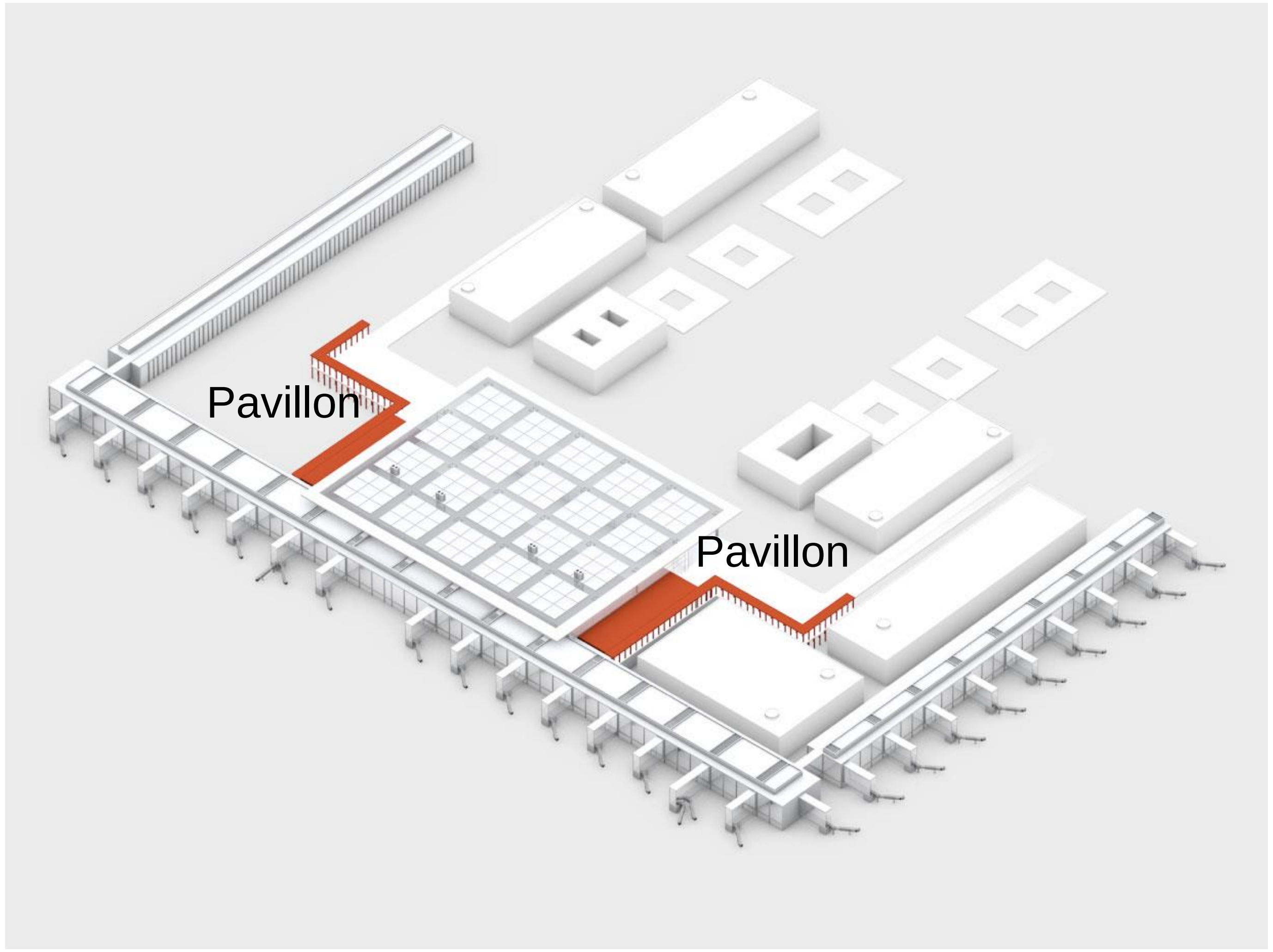

Schengen area **without** customs/passport checks
Non-Schengen area **with** customs/passport checks
Main pier (until 2007)
Main pier (from 2008)
Gateways between areas

https://www.spiegel.de/wirtschaft/flughafen-berlin-brandenburg-wie-der-ber-zum-pannenflughafen-verkam-a-d007c890-1fd3-44f0-8f02-2d7c76f09e47

A Modern Example: Berlin Brandenburg Airport

- 2010
  - Area: 320.000 m^2
  - Projected cost: 1200 Mio. €
  - Scheduled opening: Jun 2012

- Regulations on security checks for liquids introduced
- Added extra pavillons for security checks



https://www.spiegel.de/wirtschaft/flughafen-berlin-brandenburg-wie-der-ber-zum-pannenflughafen-verkam-a-d007c890-1fd3-44f0-8f02-2d7c76f09e47

A Modern Example: Berlin Brandenburg Airport

- 8 May 2012
  - Area: 320.000 m^2
  - Accrued cost: 4500 Mio. €
  - Scheduled opening: unknown

- Transition from SXF to BER **cancelled ca. 3 weeks before planned grand opening** due to massive problems with fire safety system, baggage system and other key infrastructure
- No word of major issues at board meeting just two weeks earlier

Delayed Indefinitely

Unravelling Berlin's Airport Debacle

Berlin had hoped to gain worldwide recognition for its futuristic new international airport. But after the announcement that the facility won't open on time due to massive safety problems, the city is embarrassed and the hunt for who to blame has begun. By SPIEGEL Staff

15.05.2012, 18.06 Uhr

https://www.youtube.com/watch?v=IpUzuL9Cry0
https://www.spiegel.de/international/germany/how-berlin-s-new-international-airport-opening-was-delayed-a-833122.html

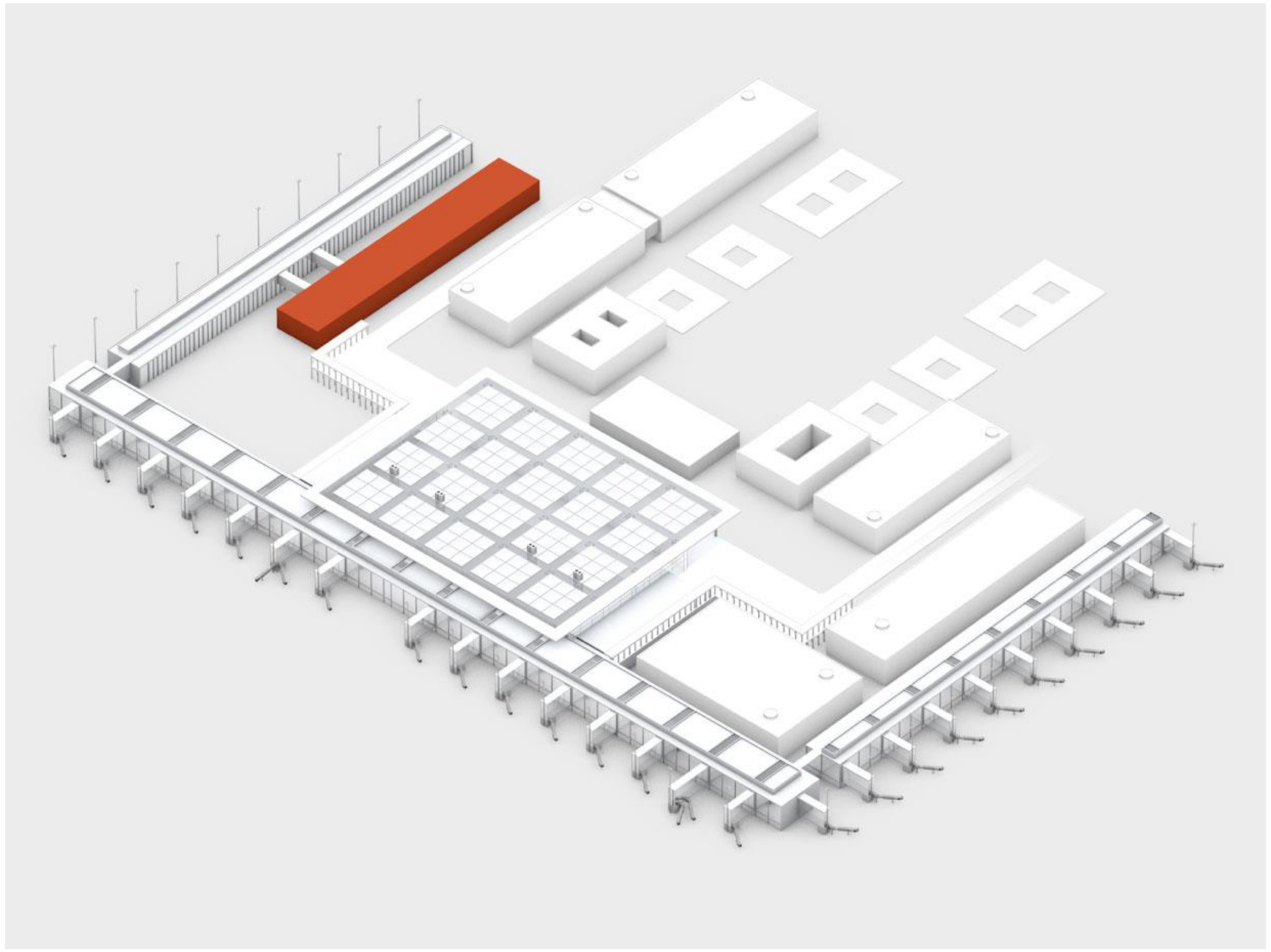
A Modern Example: Berlin Brandenburg Airport

- 2013
- 150,000 construction defects identified, airport deemed 56% complete. A few examples:
  - Fire protection and smoke extraction system misdesigned and misconstructed
    - Initial smoke extraction system (through basement) stretched laws of physics
    - Redesigned extraction system’s ventilators too heavy, threatening roof collapse
    - 600 fire protection walls had to be torn down and rebuilt as they weren’t fire-proof
    - Sprinkler system underdimensioned; pipes too thin to withstand water pressure
  - Data cables, high-voltage cables, uninsulated heating pipes all running alongside each other
  - Electric window motors do not work above 30°C; 80% of electric doors do not open
  - Various escalators too short by a few steps; gaps of up to 5 m in stairway handrails
  - All 5,000 terminal rooms mislabeled; >1,000 trees planted in wrong locations
  - Empty trains must regularly be run through underground railway station to provide ventilation
  - Terminal lights burning night and day due to incomplete building control software
- Cost for maintenance, security, energy etc. during construction: €10-20 million/month
  - Power consumption during construction higher than active airport TXL with 400 flights/day
- Countless changes of management, lawsuits between virtually all involved stakeholders

https://de.wikipedia.org/wiki/Bau_des_Flughafens_Berlin_Brandenburg

A Modern Example: Berlin Brandenburg Airport

- 2018
  - Area: 360.000 m^2
  - Projected cost: 7300 Mio. €
  - Scheduled opening: Oct 2020
- Terminal 2 commissioned 2018
  - Completed on schedule in 2019 by single subcontractor
- 750 display screens throughout terminal need to be replaced after 6 years of continuous operation without purpose



https://www.spiegel.de/wirtschaft/flughafen-berlin-brandenburg-wie-der-ber-zum-pannenflughafen-verkam-a-d007c890-1fd3-44f0-8f02-2d7c76f09e47

A Modern Example: Berlin Brandenburg Airport

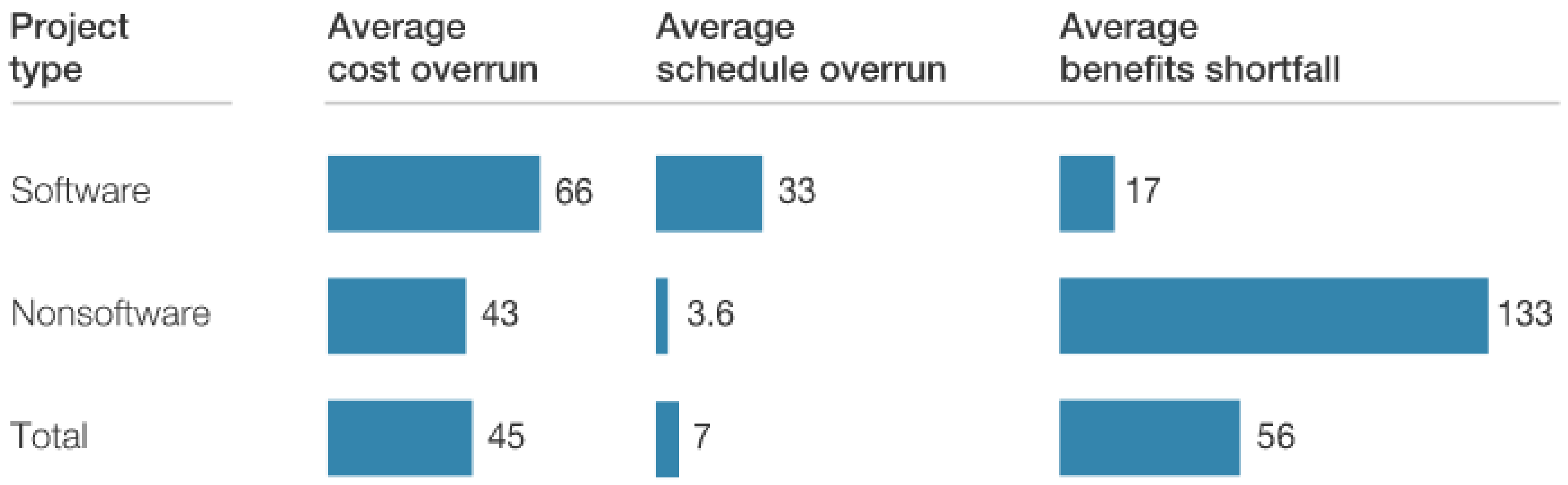
- 1991: planning begins for a new airport to replace Tegel (TXL) and Schönefeld (SXF)
- 1995: initial projections: budget: €770 million, opening: 2007
- 2008: construction begins; projected budget: €2.4 billion, projected opening: 2011
- 2010: opening delayed to 2012
- 2012: transition from SXF to BER cancelled 3 weeks before grand opening, cost so far: €4.5 billion
- 2013: 150.000+ construction defects identified, opening “before 2016 unlikely”
- 2014: opening “possibly 2018”, cost so far: €5.1 billion
- 2015: projected overall cost: €6 billion
- 2016: announcement of new opening date cancelled
- 2017: opening “maybe 2020”; projected cost: €6.5 billion
- 2018: partial opening scheduled for Oct 2020; projected cost: >€7 billion
- 2019: 11.500+ cabling defects remain
- 2020: Airport finally opens on 31 Oct, necessary expansion for €2.3 billion already planned
- 2021: Accountants deem airport technically bankrupt, will require massive state subsidies for >10y

} Istanbul Airport & Beijing Daxing Intl Airport construction time

https://en.wikipedia.org/wiki/Berlin_Brandenburg_Airport

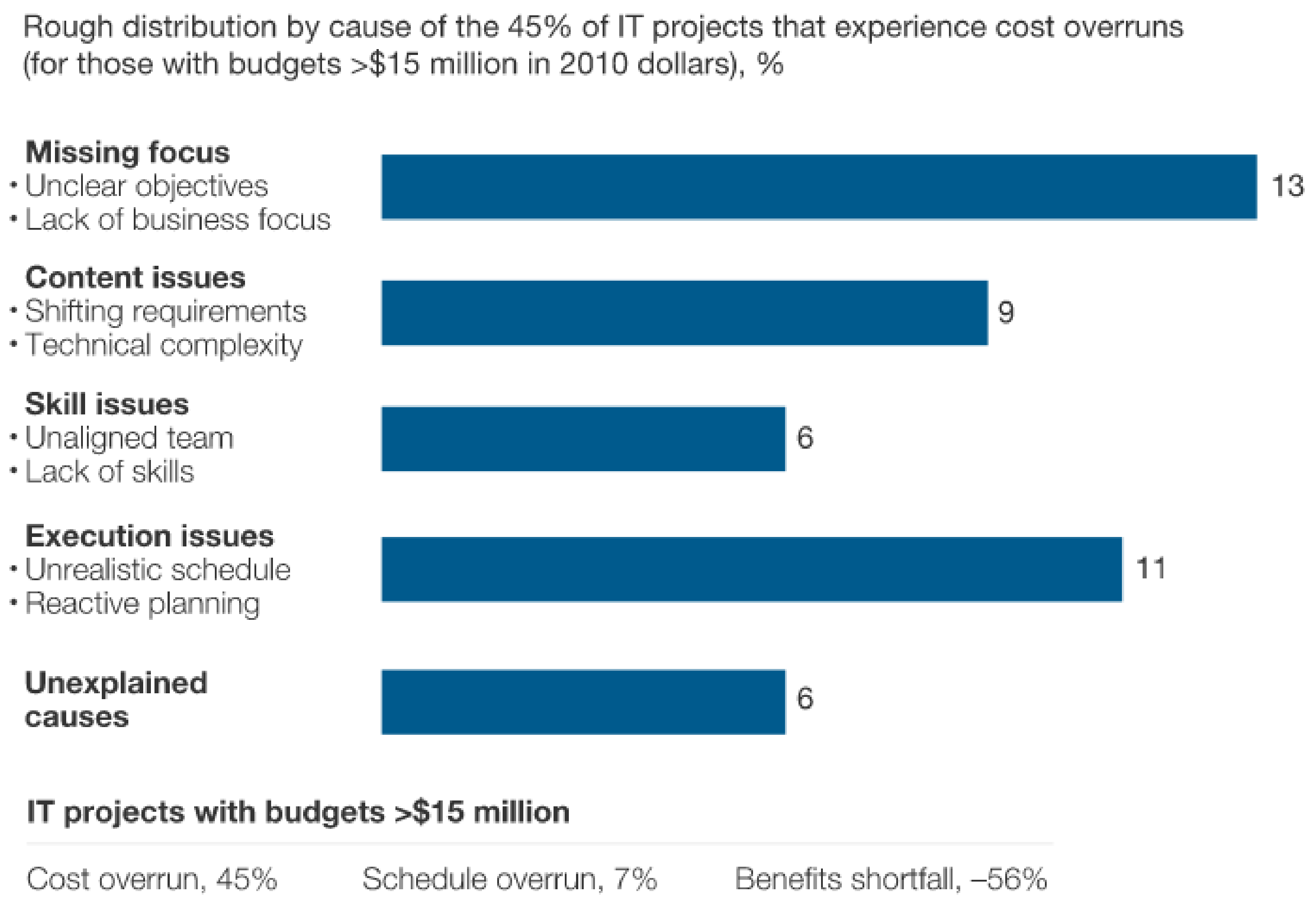
What about Software Engineering?

- McKinsey/Oxford University study of 5,400 IT projects across different industries
- Results for large IT projects (*initial* cost estimate > $15 million):
  - 50% ran massively over budget
  - 17% failed so spectacularly that they threatened the existence of the company
    - Budget overruns of 200-400%



M. Bloch, S. Blumberg, J. Laartz: Delivering large-scale IT projects on time, on budget, and on value. McKinsey-Oxford study on reference-class forecasting for IT projects, 2012

Causes of Software Project Troubles (2012)



M. Bloch, S. Blumberg, J. Laartz: Delivering large-scale IT projects on time, on budget, and on value. McKinsey-Oxford study on reference-class forecasting for IT projects, 2012

Causes of Software Project Troubles (1988)

- **Thin Spread of Application Domain Knowledge**
  - Projects in a new domain
  - Technology focus
- **Fluctuating and Conflicting Requirements**
  - Market
  - Different clients
  - Learning processes
  - Misunderstandings
- **Communication and Coordination Breakdowns**
  - Different goals and expectations
  - Incongruence between competence and responsibility
  - Surrender

B. Curtis, H. Krasner, N. Iscoe: A Field Study of the Software Design Process for Large Systems. Communic. ACM, Nov 1988, pp. 1268-1287

Definitions of “Software Engineering”

- “The establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines.”
  - P. Naur, B. Randell (Eds.): Software Engineering: Report of a Conference. NATO Scientific Affairs Division, 1969

Barton, Dijkstra, Wodon, Gill, Hume, Smith, Paul, Perlis, Randell at the NATO Software Engineering Conference, Garmisch, Germany, Oct 1968

- “The systematic approach to the development, operation, maintenance, and retirement of software.”
  - IEEE Standard Glossary of Software Engineering Terminology (IEEE Standard 729). IEEE Computer Society, 1983

Photo from Brian Randell’s home page: http://homepages.cs.ncl.ac.uk/brian.randell/NATO/

Definitions of “Software Engineer”

- “A software engineer must of course be a good programmer, be well-versed in data structures and algorithms, and be fluent in one or more programming languages. […] The software engineer must be familiar with several design approaches, be able to translate vague requirements and desires into precise specifications, and be able to converse with the user of a system in terms of the application rather than in ‘computerese’.”
    - C. Ghezzi, M. Jazayeri, D. Mandrioli: Fundamentals of Software Engineering. Prentice Hall, 1991

- “Software development is and always will be somewhat experimental. The actual software construction isn’t necessarily experimental, but its conception is. And this is where our focus ought to be.”
    - T. DeMarco: Software Engineering – An Idea Whose Time Has Come and Gone. IEEE Software, vol. 26, no. 4. IEEE Computer Society, 2009

A Software Engineer’s Most Important Skills

- Communication with people of different training, goals, expectations
- Communication on multiple levels of abstraction
- Creation and use of models and specifications
- Planning and coordinating work

Break

Course Schedule (tentative)

Week	Mon: Lectures	Wed: Team Consultations	Sun: Assignments due
1	Introduction, Software Processes		Team formation (16 Jan)
2	Requirements Elicitation	Project Topic Guidance	
3	Effort Estimation	#1a: User Stories Draft	#1a: User Stories (30 Jan)
4	Agile Project Planning	#1a: User Stories Presentation	
5	Modeling	#1b: Planning Poker	#1b: Product Backlog (13 Feb)
6	Object-Oriented Analysis	#1b: Backlog Presentation	
7	Object-Oriented Analysis/Design	#2: Domain Model Draft	#2: Domain Model (27 Feb)
8	Object-Oriented Design	#2: Domain Model Presentation	
9	Design Patterns	#3: Design Model Draft	#3: Design Model (13 Mar)
10	Software Testing	#3: Design Model Presentation	
11	Software Testing	#4: Test Cases Draft	#4: Test Cases (27 Mar)
12	Software Testing	#4: Test Cases Presentation	
13	Final Exam Preparation	#5: Final Product Draft	
14	*Spare lecture slot (if needed)*	EASTER BREAK	
15		#5: Final Product Presentation	#5: Final Product (24 Apr)

Course Structure and Online Venues

- **Project teams and clusters**
  - 3-4 students per team
    - Design and build a software component
  - 4 teams per cluster
    - Integrate components into a larger system
- **Team meetings**
  - Scheduled freely among team members
  - Commit to a fixed meeting time and establish team discipline early!
  - Recommended to use your team channel on the class Discord server for synchronous and asynchronous team communication
- **Lectures**
  - Mondays 10:00-12:20 on Zoom
    - https://eu01web.zoom.us/j/62847273071
    - Slides and recordings on Canvas
    - Camera usage discouraged
- **Team consultations with your tutor**
  - Wednesdays 15:00-18:15 on Discord (from 19 Jan)
    - https://discord.gg/4cCazJrWVa
    - not recorded by tutor
    - Camera usage encouraged
  - ~20-minute consultations per team
    - discuss questions, designs, assignments
    - present assignment deliverables

Online Class Participation Formats and Recommendations

- **Lecture participation** will be possible in several ways:
  - by attending the live stream on Zoom at the scheduled time
  - by watching the recording accessible on Canvas at your own pace

- Recommendations for staying on top of things:
  - **Stay aware of announcements**, assignments and course materials posted on Canvas
  - **Stick to the published schedule** and follow classes in real time as much as possible
    - Catching up with classes by binge-watching recordings later in the semester will not work!
  - **Use all available means of interaction** and communication
    - Ask questions during live-streamed classes on Zoom
    - Use Discord to post questions for classmates and teachers

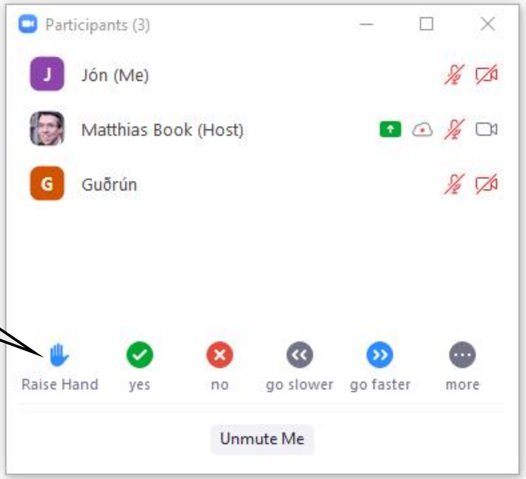
General Guidelines for Online Classes

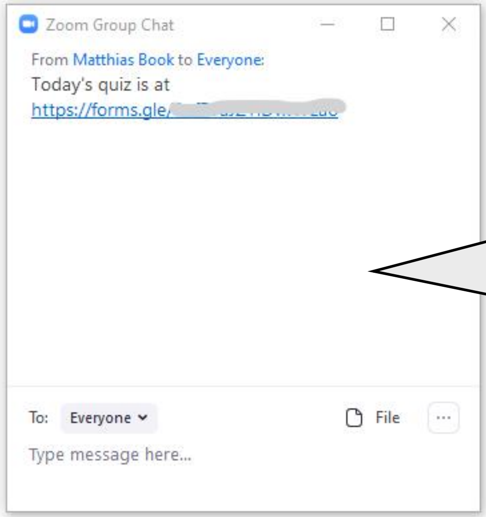
- **Class recordings**
  - Lectures will usually be recorded by the teacher and published on Canvas.
  - Consultations will NOT be recorded by the tutor.
- **Interaction protocol**
  - You are most welcome to ask questions at any time!
  - The preferred way is to simply unmute your microphone and interrupt the teacher
    - Chat messages, “raised hand” icons, emoji and other notifications are easily missed or even invisible to a teacher who is sharing their screen – do not rely on these.
  - Please mute your microphone otherwise to minimize noise for all participants.
- **Camera usage**
  - Camera usage is not required in any class, and discouraged in lectures.
  - Camera usage is encouraged in consultations as it facilitates more natural conversation, but is still your personal choice.

Participant Controls During Zoom Lectures

Recording indicator

Raise Hand (discouraged – easily missed by teacher while screensharing)





From Matthias Book to Everyone:
Today's quiz is at
https://forms.gle/[redacted]

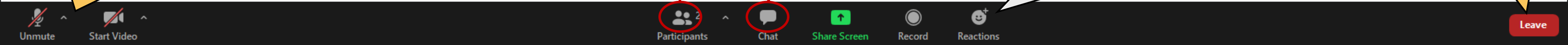
Chat messages (discouraged – likely invisible to teacher while screensharing)

Encouraged – if you have a question, feel free to just unmute and interrupt the teacher ☺

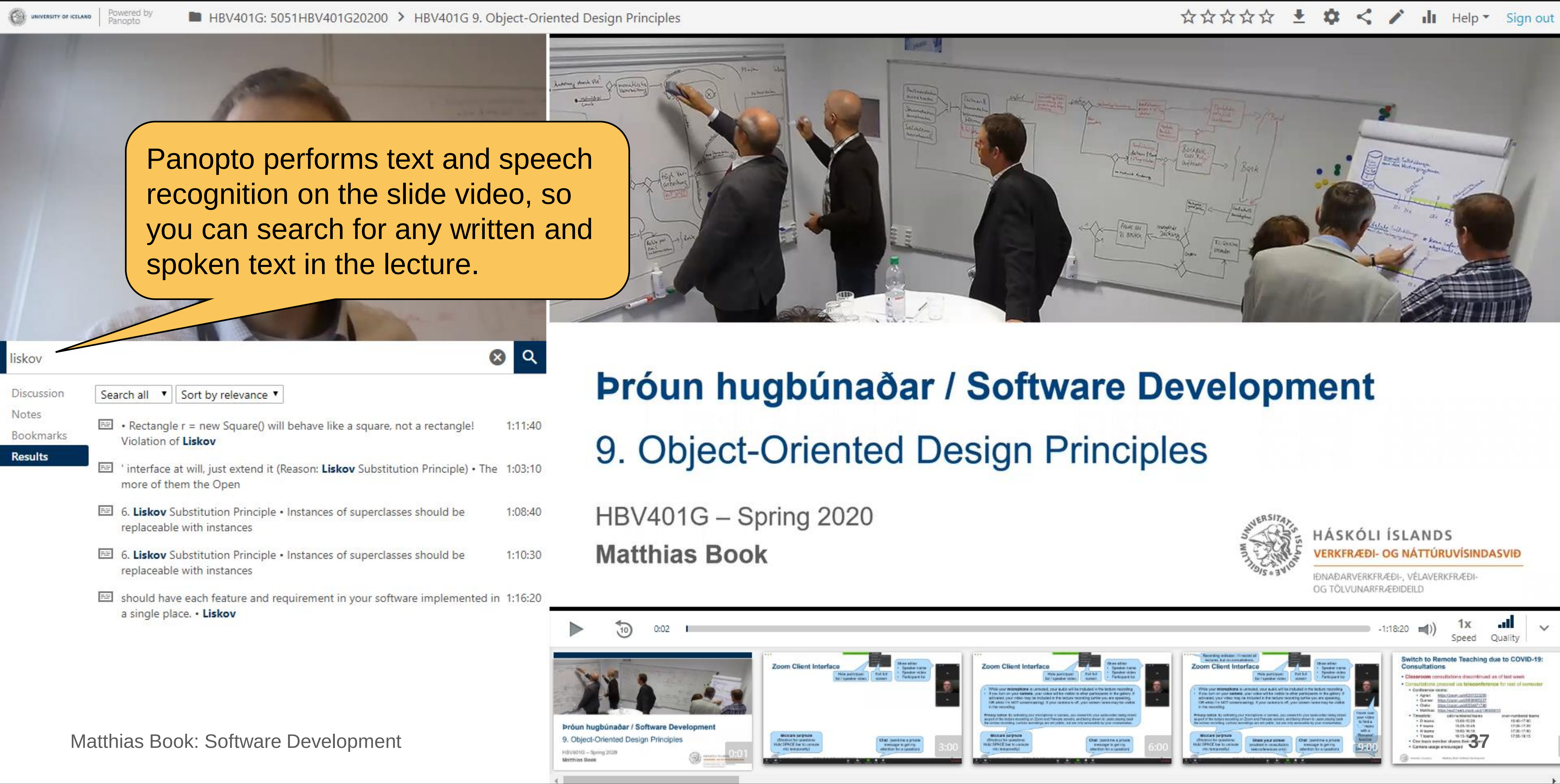
Mic/cam (un)mute, audio/video settings (Hold SPACE bar to unmute temporarily)

Reactions (discouraged – likely invisible to teacher while screensharing)

Leave session



Course Recordings (Panopto Video on Canvas)



HBV401G: 5051HBV401G20200 > HBV401G 9. Object-Oriented Design Principles

Panopto performs text and speech recognition on the slide video, so you can search for any written and spoken text in the lecture.

liskov

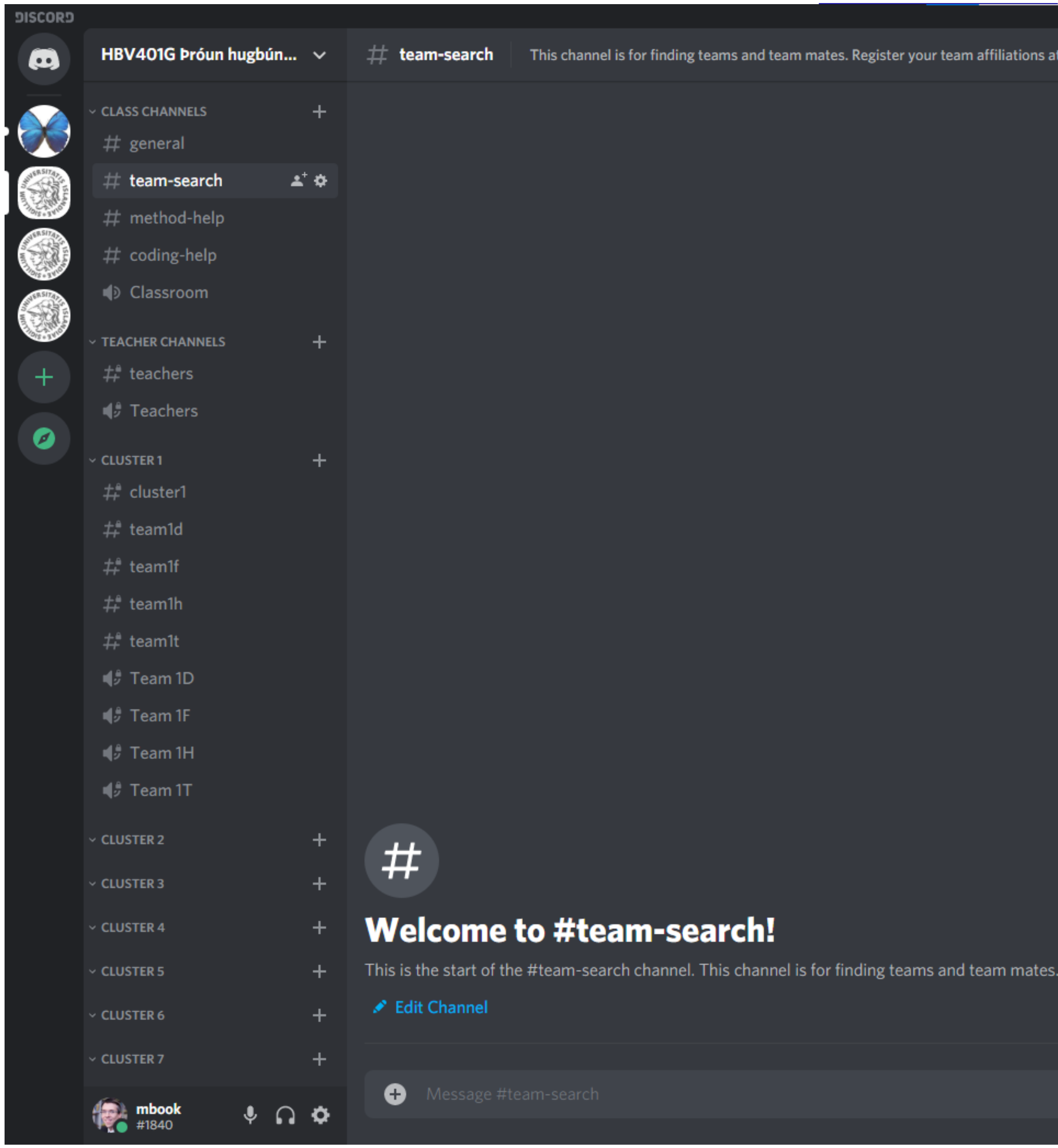
Search all ▾ Sort by relevance ▾

• Rectangle r = new Square() will behave like a square, not a rectangle! Violation of **Liskov**	1:11:40
' interface at will, just extend it (Reason: **Liskov** Substitution Principle) • The more of them the Open	1:03:10
6. **Liskov** Substitution Principle • Instances of superclasses should be replaceable with instances	1:08:40
6. **Liskov** Substitution Principle • Instances of superclasses should be replaceable with instances	1:10:30
should have each feature and requirement in your software implemented in a single place. • **Liskov**	1:16:20

Powered by Panopto
Help
Sign out
Discussion
Notes
Bookmarks
Results
Þróun hugbúnaðar / Software Development
9. Object-Oriented Design Principles
HBV401G – Spring 2020
Matthias Book
HÁSKÓLI ÍSLANDS
VERKFRÆÐI- OG NÁTTÚRUVÍSINDASVIÐ
IÐNAÐARVERKFRÆÐI-, VÉLAVERKFRÆÐI- OG TÖLVUNARFRÆÐIDEILD
0:02
-1:18:20
1x Speed
Quality
0:01
3:00
6:00
9:00

Course Support and Consultations on Discord

- **Text channels**
  - #general: Questions and announcements regarding course schedule, policies etc.
  - #team-search: Finding teams and teammates before signing up on Doodle
  - #method-help: Questions on course topics (Agile, UML, OO etc.)
  - #coding-help: Questions on programming your project (Java, IDEs, databases etc.)
  - #clusterX: Text channel for discussion within your cluster of teams
  - #teamXX: Text channel for discussion within your team
- **Voice channels**
  - Classroom: For plenary consultations
  - TeamXX: For discussions among your team and consultations with your tutor



team-search This channel is for finding teams and team mates. Register your team affiliations a

Welcome to #team-search!

This is the start of the #team-search channel. This channel is for finding teams and team mates.

Project Topic: Travel Portal

- **Vision:** A meta-search engine combining a variety of travel services

- **Team D: Day Tour Search**
  - Search engine for day tours offered in Iceland
  - Availability-based booking with hotel pickup option
- **Team F: Flight Search**
  - Search engine for flights involving airports in Iceland
  - Availability-based booking with seat reservation
- **Team H: Hotel Search**
  - Search engine for hotels in Iceland
  - Availability-based booking with room reservation
- **Team T: Trip Planner**
  - Meta-search engine for flights, hotels and day tours, based on traveler profiles
  - Suggestion and booking of matching combinations of travel services

The Software Engineering Challenge

- Projects are underspecified – you need to figure out requirements *and* solutions!
- Components built in the projects should work (rudimentarily) on their own…
  …and as part of an integrated system with other cluster components!

- This requires
  - Communication with the client, your team members, and other teams in the cluster
  - A joint vision of the overall project, and each component's responsibilities
  - An understanding of which aspects of a component are public, and which are private
  - A joint specification of the interfaces between communicating components
  - Test drivers that your component can rely on while other teams' components are still missing
  - Project management to ensure you deliver in time what other teams need and client wants

- These are typical challenges that any large software project is grappling with.
- Lectures and team consultations will show you how to deal with them.

Team Consultation Schedule

Timeslots	Teams			
Wed 15:00-15:20	1D	3D	5D	7D
Wed 15:25-15:45	1F	3F	5F	7F
Wed 15:50-16:10	1H	3H	5H	7H
Wed 16:15-16:35	1T	3T	5T	7T
Wed 16:40-17:00	2D	4D	6D	8D
Wed 17:05-17:25	2F	4F	6F	8F
Wed 17:30-17:50	2H	4H	6H	8H
Wed 17:55-18:15	2T	4T	6T	8T

- Team IDs: [Cluster][Component] (e.g. Team 4H: Cluster 4, Hotels component)
- Each team has about 20 minutes of consultation time with their tutor on Discord
- Use remaining time to collaborate with teammates and other teams in cluster!

Team Formation

- **Sign up in Canvas** (*People > Groups*)
  - Drag your name to the desired team (3-4 members per team)
    - Note: Your team number determines your project topic and consultation time slot (see prev. slide)
    - Ideally, form teams offline and then sign up all 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 few members
    - If you are looking for teams or team members: Use the *#team-search* channel on Discord
  - To change your group affiliation: Just drag your name to a new group
    - Resolve assignment conflicts by e-mail

- **When:** by this Sunday (16 Jan)
  - So you can start working on your project together in the first consultation next week

Assignments

- **5 team assignments**
  1. User stories (a) & product backlog (b)
  2. Domain model
  3. Design model
  4. Test cases
  5. Final product

- **Expected quality:** Assignment deliverables should be
  - of **realistic scope** for your project
  - 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

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 Canvas
  - 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 Canvas
  - Only the team member who will present must submit a document for the whole team
- A **presentation** of the deliverable must be given the following Wednesday
  - Given by one representative of the team (a different one for each assignment)
  - Based on the submitted document (don't prepare extra slides)
  - Taking around 5-10 minutes (plus some questions asked by the tutor)

Recommendation: Work towards a Friday 17:00 deadline and use the weekend as a buffer only if unavoidable!

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)
    - A presentation must be given by a sole student to count for their presentation grade.
      - Joint presentations are allowed but won’t be counted for anyone’s presentation grade.
    - If a student presents multiple assignments, their best presentation grade counts.
- The resulting project grade weighs 30-70% of the final course grade.

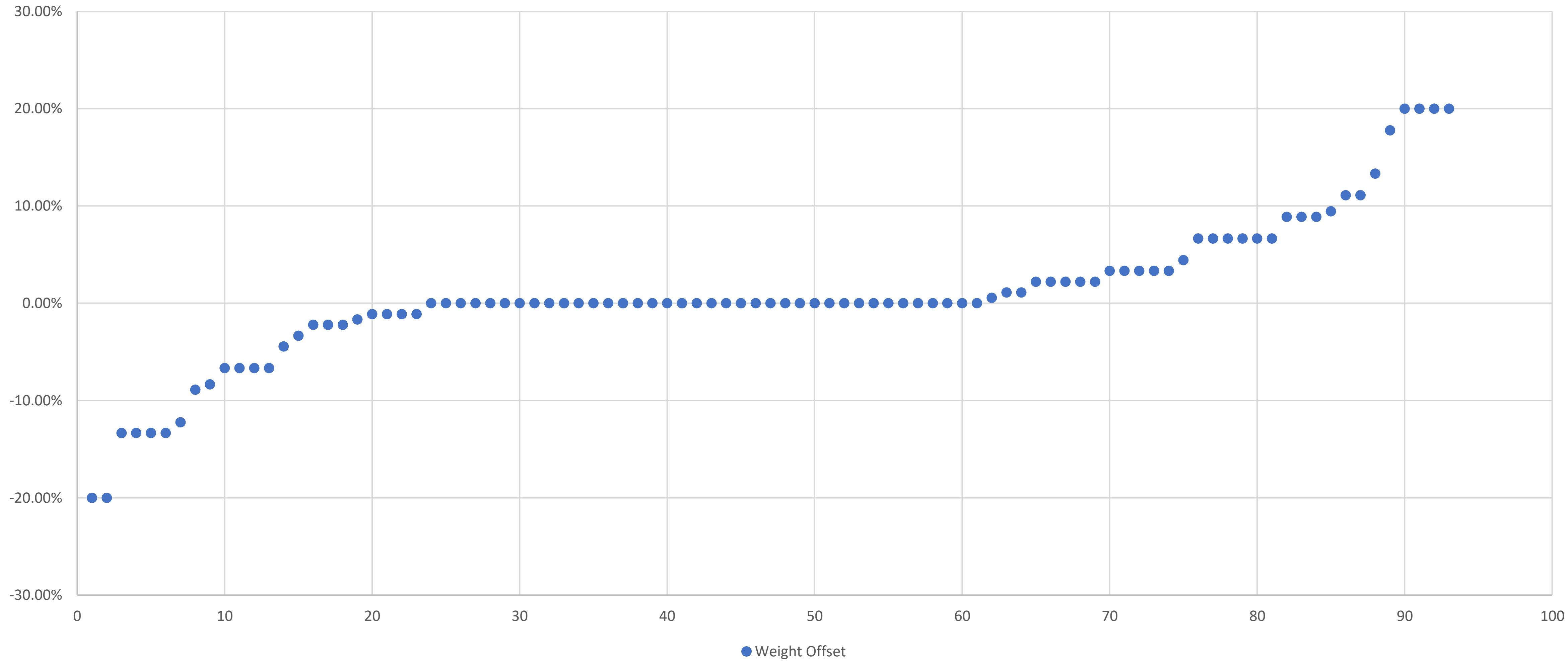
Grade Weights

- Assessment of team contribution
  - At the end of the semester, all team members assess how much each of their teammates 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

Example: HBV501G 2019 Project Weight Offsets



Final Exam

- **Date, Time & Location:** TBA (3 h)
- **Focus:** Understanding of software engineering concepts and methods
- **Scope:** Lecture slides
  - Note: The spoken part is relevant too!
- **Style:** Written digital exam
  - Likely on campus, but TBA later
- **Weight:** 30-70% of final course grade
- **Tools:**
  - One A4 sheet of your handwritten notes
  - Dictionary (in book form)
  - No electronic devices allowed
  - No collaboration with others allowed
- **Questions:**
  - 10 questions with same weight
  - Answers that exceed expectations can make up for deficiencies elsewhere
- **Answers:**
  - in English; in your own words
  - short paragraphs of whole sentences
  - possibly small code fragments / models

Optional In-Class Quizzes

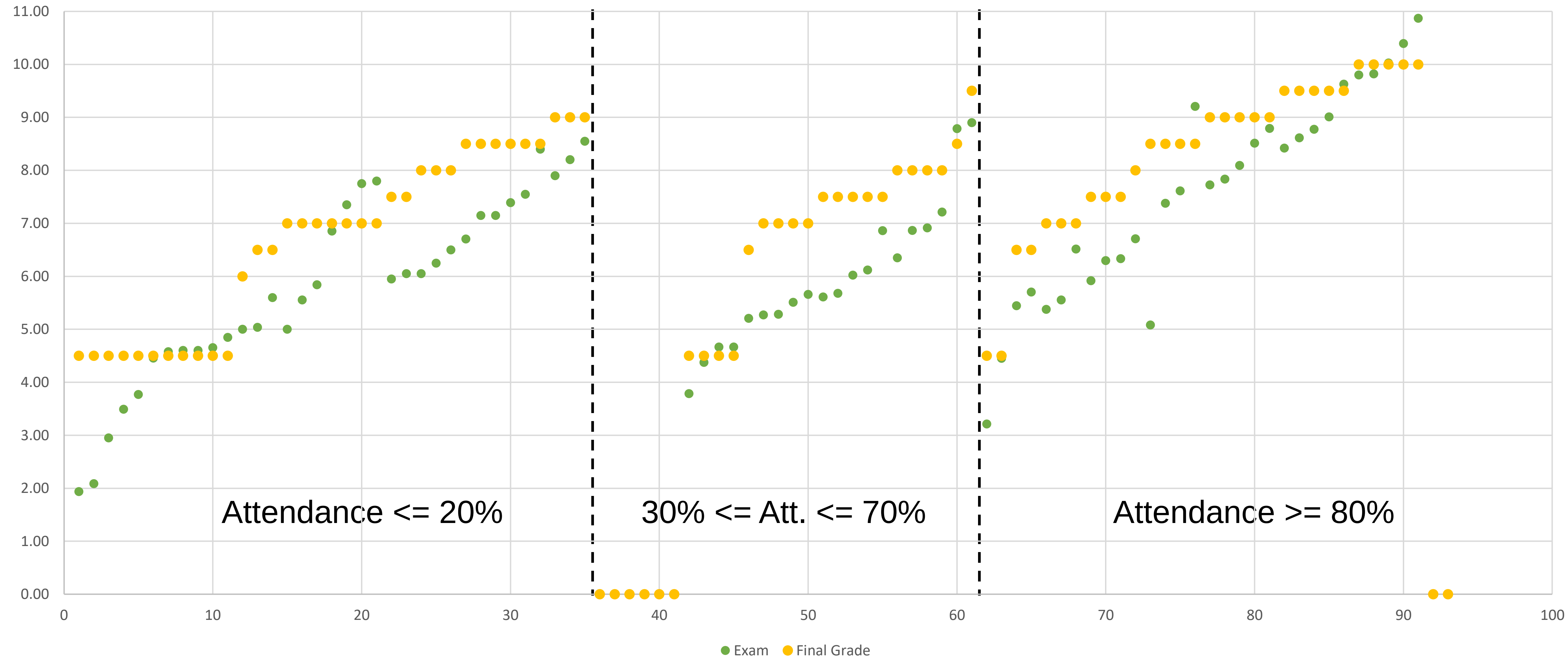
- To encourage class attendance: Small quizzes in most lectures
  - To be submitted online on the same day as the lecture
  - Graded on scale (0, 5…11), with 0 for any quizzes that are not handed in
    - two worst quiz grades will be ignored

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

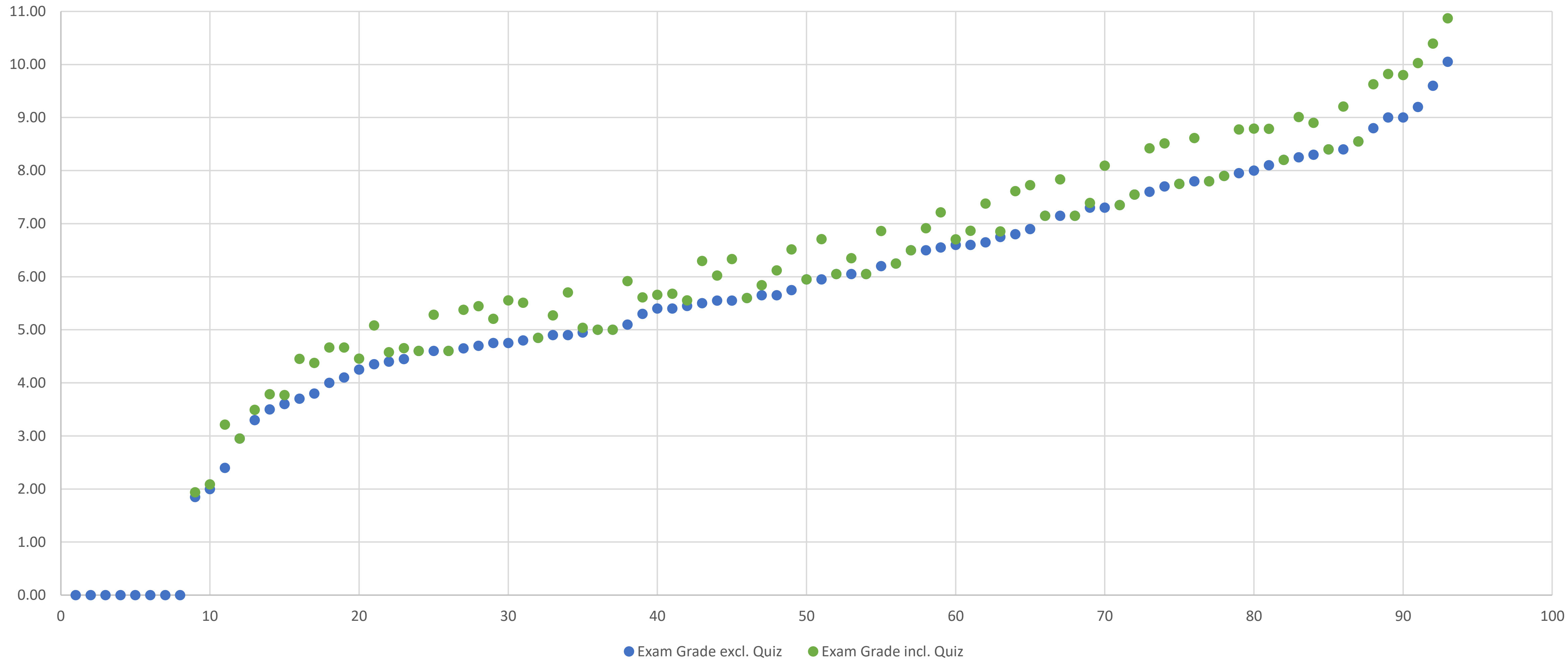
➢ If you don’t participate in any quizzes, you can still get top marks on the exam

➢ 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

Example: HBV501G 2019 Exam and Final Course Grades (Grouped by Class Attendance)



Example: HBV501G 2019 Exam Grades excl. and incl. Quiz Grades



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 course grade is rounded to nearest half point
  - In case of a passed exam, final course grade is capped at 10.0
  - In case of a failed exam, final course 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, and re-taken next year if failed again
  - No need (and not possible) to redo a passed project
    - Project grade remains valid for only one year though

Learning from Criticism / Managing Expectations
(based on previous years' course feedback)

- 🟥 What I will do
- 🟩 What you can do

- **Uncertainty about what is expected in assignments**
  - Assignments' open-ended nature reflects actual project challenges:
    You will rarely face precisely stated, closed questions in your professional career
  - 🟩 Read slides carefully and ask for clarification in class or consultations
  - 🟩 Discuss drafts of your assignments with tutor early to get feedback before submission
  - 🟥 I'll provide examples of well-done assignments from previous semesters as examples
- **Uncertainty about what to do / long waiting times in consultations**
  - Consultations are not just about checking in with your tutor
  - 🟩 Use the time to discuss open issues with your team members
- **Little emphasis on programming (time- and teaching-wise)**
  - Teaching programming skills is not a focus of the course
  - Focus is on practicing software engineering techniques (requirements, design, planning etc.)
  - 🟩 But you're encouraged to begin implementing prototypes early to learn about technology

Learning from Criticism / Managing Expectations
(based on previous year’s course feedback)

- **People who did little work still got similar grades as people who contributed a lot**
  - Flexible project/exam grade weights should provide more incentive to contribute
  - In case of problems with team members who aren’t contributing,
    talk to them / your tutor / me <u>early</u>

- **Recorded lectures are nice to have**
  - I’ll make all class recordings available on Canvas, in addition to the lecture slides
    - Caution: No quality or availability guarantees!
  - Attending live lectures on Zoom is **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)
    - Virtually impossible to effectively catch up on several missed classes using only recordings
  - Use recordings only as a fallback when you occasionally can’t join the live lecture

Course Support

- **Questions and feedback** welcome anytime! Preferably:
  - during the lectures on Zoom (just interrupt me anytime)
  - during the team consultations on Discord (just ask your tutor)
  - anytime on the Discord channels (just ask the @Teachers and/or your fellow students)
- **See Canvas for**
  - Lecture slides and recordings
  - Important course announcements
- **Teaching Staff**
  - Matthias Book (book@hi.is)
  - 4 TAs (TBA)

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

- **Agile software development**
  - ❖Dan Pilone, Russ Miles: Head First Software Development. O'Reilly, 2008
  - Mike Cohn: Agile Estimating and Planning. Prentice Hall, 2005
- **Requirements engineering**
  - Dean Leffingwell: Agile Software Requirements. Addison-Wesley, 2011
  - Karl Wiegers, Joy Beatty: Software Requirements. Microsoft Press, 2013
- **Object-oriented analysis and design**
  - Craig Larman: Applying UML and Patterns. Prentice Hall, 2004
  - Russ Miles, Kim Hamilton: Learning UML 2.0. O'Reilly, 2006
  - Eric Freeman, Bert Bates: Head First Design Patterns. O'Reilly, 2004
- **Programming in the large**
  - Joshua Bloch: Effective Java. Addison-Wesley, 2008
  - Steve McConnell: Code Complete. Microsoft Press, 2004

Optional Pre-Class Reading

Lecture	Prior Reading Suggestion
Introduction	
Software Process Models	Head First Software Development, Ch. 1
Requirements Elicitation	Head First Software Development, 1st half of Ch. 2
Effort Estimation	Head First Software Development, 2nd half of Ch. 2; Ch. 3
Agile Project Planning	Head First Software Development, beginning & end of Ch. 4 & 9, beginning of Ch. 10
Modeling	
Object-Oriented Analysis	Learning UML 2.0, Ch. 4 & 5
Object-Oriented Analysis/Design	Learning UML 2.0, Ch.7; Head First Software Development, Ch. 8
Object-Oriented Design	
Software Testing	Head First Software Development, Ch. 7 & 8
Software Testing	Head First Software Development, Ch. 8
Design Patterns	Head First Design Patters, Ch. 1.24-30; 5.1-5.6; 4; 7.1-14; 11
Final Exam Prep	

Let’s get started!

Software Process Models

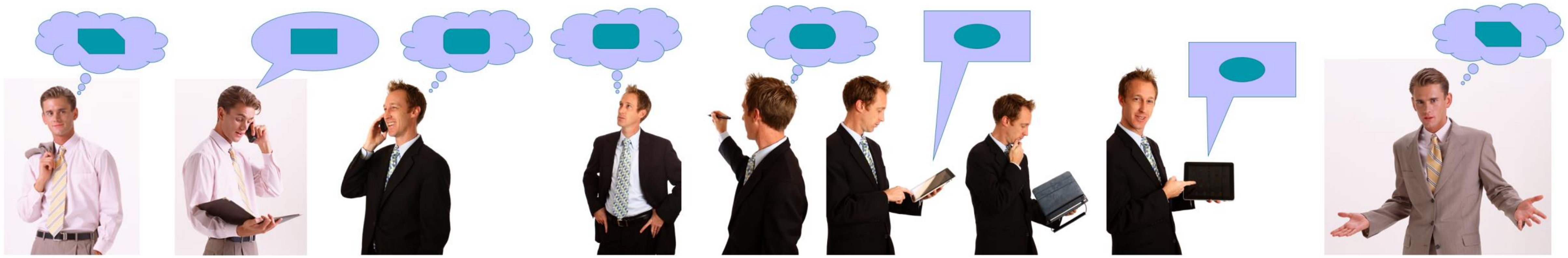
It All Starts With an Idea

A Typical Software Process

A Lose-Lose Situation

No
payment
No
product

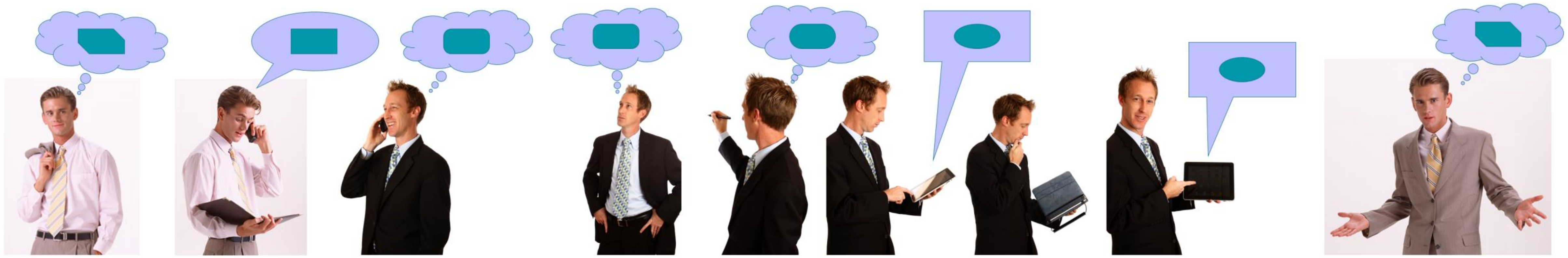
The Client is King

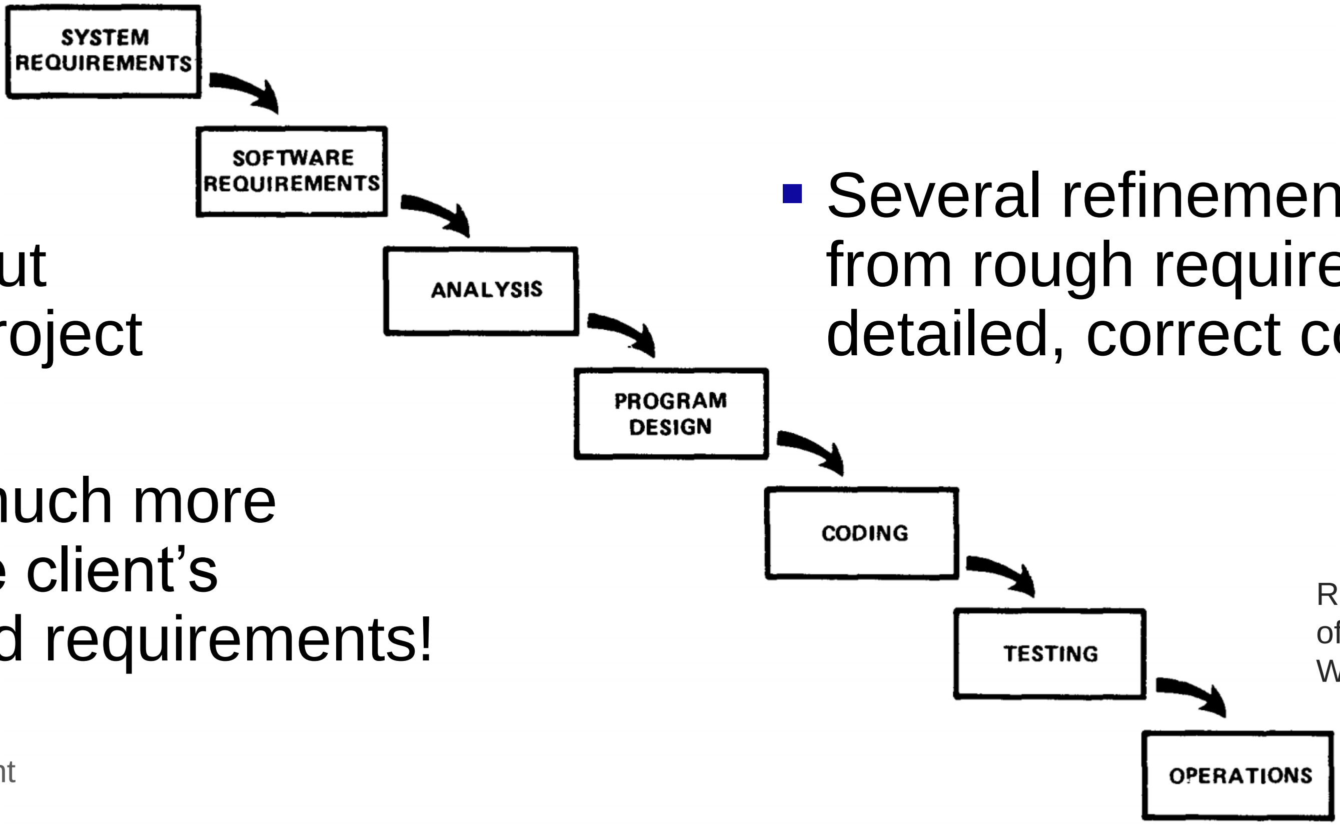


- **If the client isn't happy, you've built the wrong software.**
- No matter…
  - how awkwardly the client described it
  - how much better you like your solution
- It's your job to understand the client.

- **Clients care about three things:**
  - Will it do what I need? **(Quality)**
  - How much will it cost? **(Budget)**
  - How long will it take? **(Schedule)**
- Big Bang approach puts all the risk to satisfy these expectations on your shoulders.

Big Bang Approach





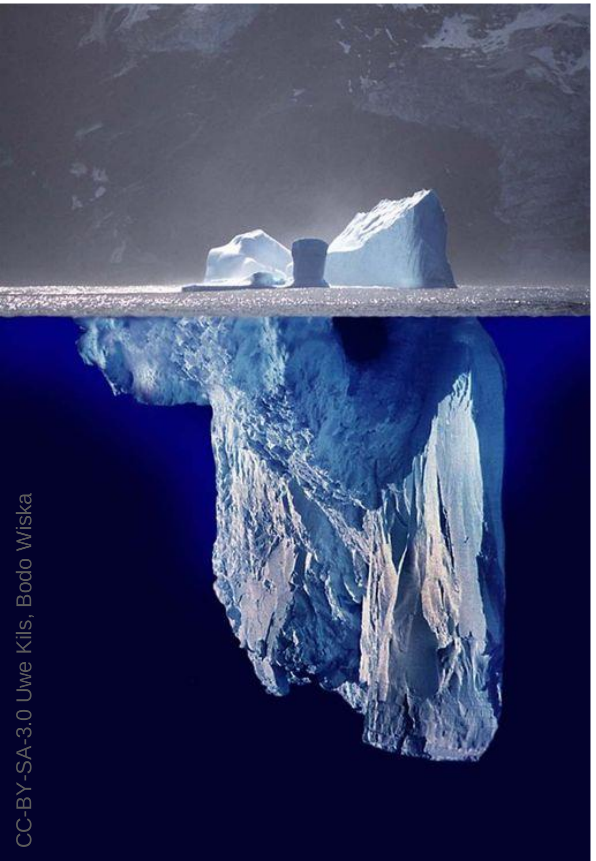
- Developer works without client input throughout the project
- But software is much more complex than the client's initially expressed requirements!

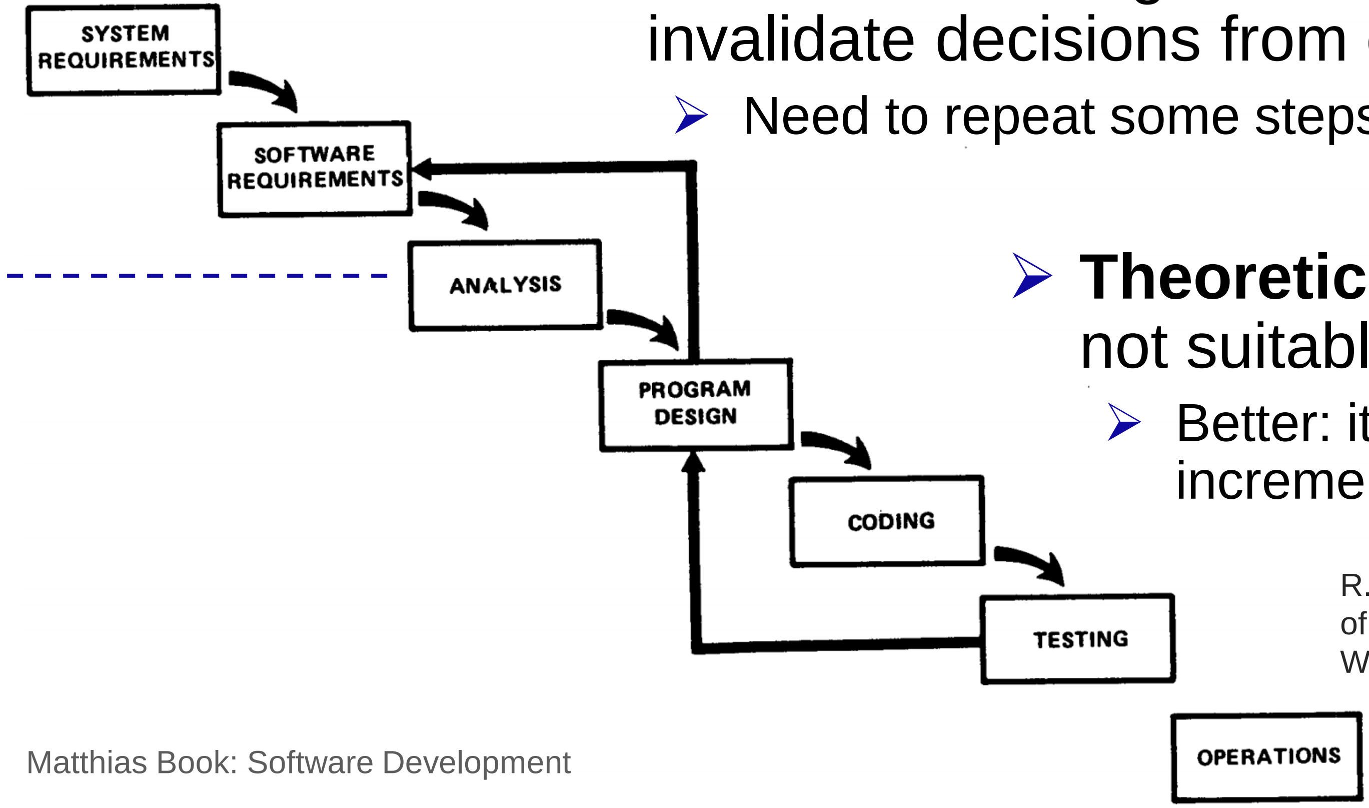
- Several refinement steps required from rough requirements to detailed, correct code

R. Winston: Managing the Development of Large Software Systems, Proc. IEEE WESCON 26, pp. 1–9 (1970)

The Waterfall Model

- Comprises typical steps that can be found in any software process model
- **Problem #1:** Virtually impossible for a developer to get right without client input
  - ➢ Need frequent feedback from client





- **Problem #2:** Insights of one phase may invalidate decisions from earlier phases
  - ➢ Need to repeat some steps

➢ **Theoretical model,** not suitable for practice
  - ➢ Better: iterative-incremental model

R. Winston: Managing the Development of Large Software Systems, Proc. IEEE WESCON 26, pp. 1–9 (1970)

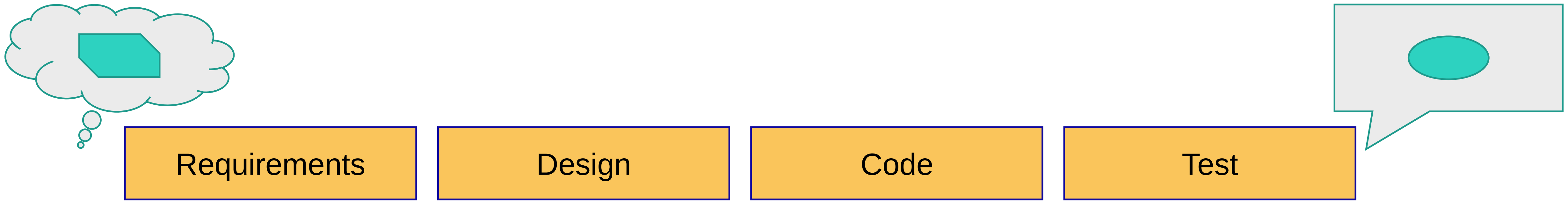
Iterative Development

- Split the project into **iterations**
- Each iteration is a mini-project…
  - Understand requirements
  - Design, code and test software
  - Roll out and get feedback
- …and produces quality software
  - Does the right thing
  - Does that thing right

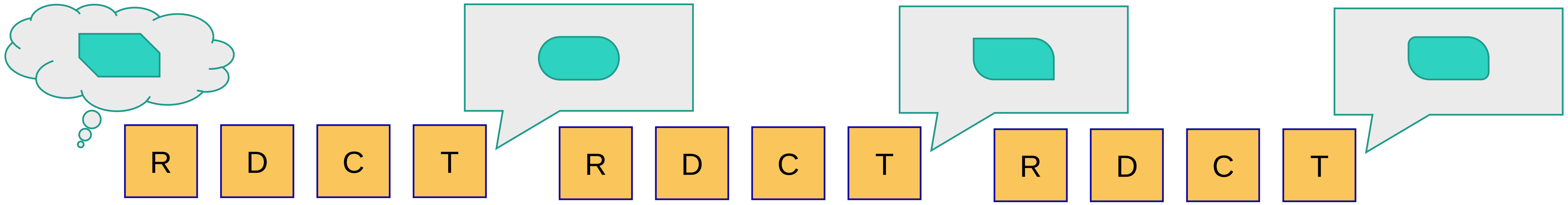
- **Always get feedback from client**
  - If you are not sure what the client wants
  - …and also if you think you are sure!
    - Because you might be wrong
      - Misunderstandings
      - Incorrect assumptions
    - Because client needs may have changed
      - Market/technology evolution
      - Budget/schedule/priority changes

Each Iteration is a Mini-Project

- **Big-Bang Approach**



- **Iterative Development**



Things to Consider in Software Project Planning

- How long should an iteration take?
- Who should be involved in an iteration?
- What activities are performed in an iteration?
- How much can be accomplished in one iteration?
- How can I leverage iterations to deal with uncertainty and change?
- How do I make sure I'm still on the right track in terms of budget and schedule?

- A **software process model** provides guidelines for…
  - Activities to execute
  - Artifacts to create
  - Roles to involve
  - Checks to perform
- …in building a quality software product.

Software Process Models: Two Philosophies

Focus of this course

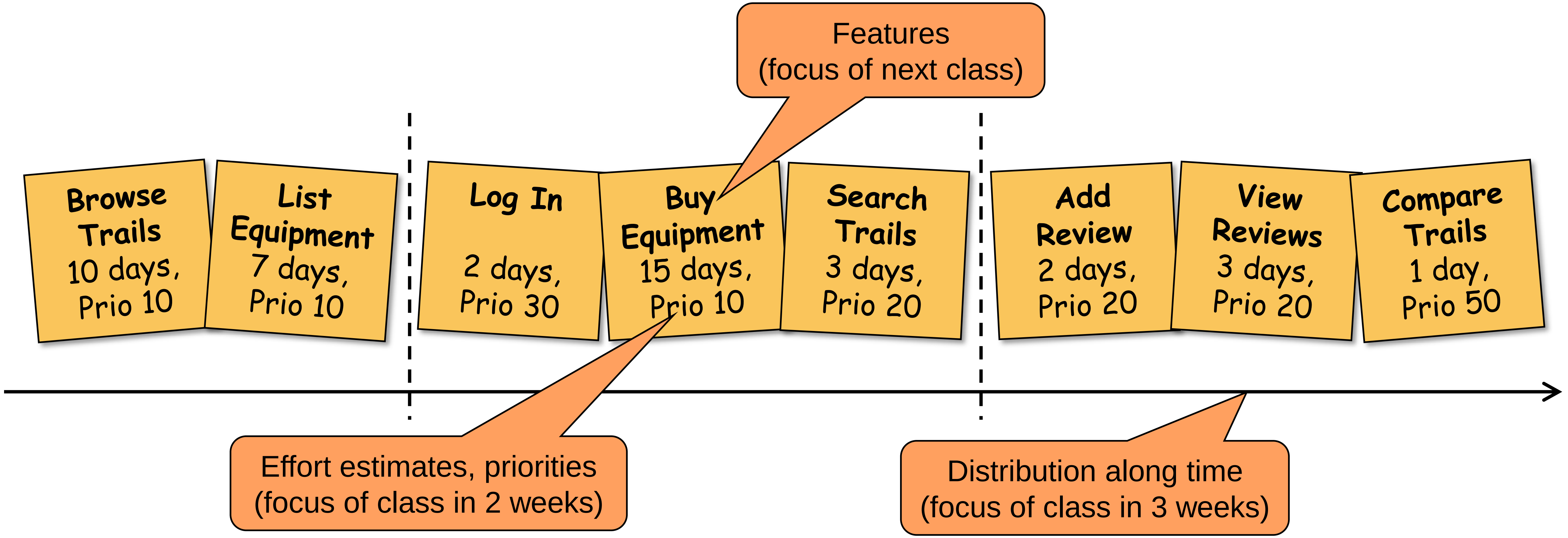
Plan-Driven Models

- Attempt to understand and specify large parts of the system in detail at the beginning of the project
- Developers are assigned tasks
- Fixed schedule for component deliveries throughout project
- Small flexibility for midway changes
- Higher (perceived) control of project progress and outcome
- e.g. Spiral Model, RUP, V Model

Agile Models

- Start with a rough project vision, refine through many prototyping and customer feedback cycles
- Developers pick their own tasks
- Component deliveries can be re-prioritized and rescheduled anytime
- No fixed specifications or plans
- Lower (perceived) control of project progress and outcome
- e.g. XP, Scrum, Kanban

Planning an Iterative Software Project


Features
(focus of next class)
Browse Trails 10 days, Prio 10
List Equipment 7 days, Prio 10
Log In 2 days, Prio 30
Buy Equipment 15 days, Prio 10
Search Trails 3 days, Prio 20
Add Review 2 days, Prio 20
View Reviews 3 days, Prio 20
Compare Trails 1 day, Prio 50
Effort estimates, priorities
(focus of class in 2 weeks)
Distribution along time
(focus of class in 3 weeks)

- **Plan-driven models** try to figure this out in the beginning, and stick to it
- **Agile models** write and revise the plan for the next few iterations as they go

Conclusion & Outlook

- Successful development of complex software systems hinges most importantly on **communication** among all stakeholders – especially with the client
- **Iterative-incremental** software development processes encourage communication and are more tolerant of changing requirements
- **Software process models** provide guidelines for structuring the activities in a software project
  - Plan-driven models emphasize detailed specifications and fixed plans
  - Agile models emphasize feedback loops, room for changes, joint discovery of best solution
- **Focus of this course:**
  - Overview of all activities in a software process
  - Management of agile software projects

Quiz #1: Software Process Models

- **Indicate which of the following statements apply to plan-driven models and which ones apply to agile models:**

Answer in Canvas
(Assignments > Quizzes > Quiz #1)
by tonight

a) Attempt to understand and specify large parts of the system in detail at the beginning of the project

b) Component deliveries can be re-prioritized and rescheduled anytime

c) Developers are assigned tasks

d) Developers pick their own tasks

e) Fixed schedule for component deliveries throughout project

f) No fixed specifications or plans

g) Small flexibility for midway changes

h) Start with a rough project vision, refine through many prototyping and customer feedback cycles

Gangi þér vel!

book@hi.is