Hugbúnaðarverkefni 1 / Software Project 1

10. Value-Oriented Requirements & Design

HBV501G – Fall 2018

Matthias Book

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

In-Class Quiz Prep

- Please prepare a scrap of paper with the following information:

- ID: _____@hi.is Date: _____

- a) _____ e) _____
- b) _____ f) _____
- c) _____ g) _____
- d) _____ h) _____

- During class, I'll show you questions that you can answer very briefly
  - No elaboration necessary
- Hand in your scrap at the end of class

- All questions in a quiz weigh same
- All quizzes (ca. 10 throughout semester) have the same weight
  - Your worst 2 quizzes will be disregarded
- Overall quiz grade counts as optional question worth 7.5% on final exam

Assignment 4: Code Review – Schedule

- By **Sun 11 Nov,** make your **project artifacts** available to your partner team:
  - Your project vision and design model from Assignments 1 & 3 (incl. fixes of severe issues)
  - A current snapshot of your source code (does not need to be the finished product)
- Take **one week** to **review** the other team's code and **document** your findings:
  - Comment on clarity of design, quality of implementation, readability of code, tech choices
  - State what you like and make suggestions for improvements
- By **Sun 18 Nov,** submit your **review report** to Ugla
  - 1-2 pages in PDF
- On **Thu 22 Nov, discuss** your findings with the other team and your tutor.

- **Grading criteria:**
  - Quality of constructive feedback on other team's design and code (80%)
  - Design and technology issues identified in your own system (10%)
  - Coding style / clarity issues identified in your own system (10%)

Assignment 4: Code Review – Considerations

- **When reviewing your partner team's code, consider the following things:**
  - Clarity of the design
    - Is the code consistent with the design model?
    - Is it easy to trace the control flow through the code as a request is processed?
  - Quality of the implementation
    - [Is the system readily executable?]
    - Are there obvious bugs in areas that don't seem to be under construction anymore?
  - Readability of the code
    - Is the naming of classes, methods, attributes etc. descriptive and helpful?
    - Do the comments help you understand the code?
    - Can you tell which parts are done and which are still under construction?
  - Technology choices
    - Are the features of the programming language and application framework used appropriately?
- **What do you like? What would you improve?**

Assignment 4: Code Review – Partner Teams

('→' = 'receives and reviews code of')

- **08:30-09:30 Timeslot**
  - **Andri's teams**
    - 1 → 4
    - 4 → 1
    - 7 → 10
    - 10 → 7
  - **Daníel's teams**
    - 2 → 5
    - 5 → 8
    - 8 → 2
  - **Matthias' teams**
    - 3 → 6
    - 6 → 9
    - 9 → 3
- **09:30-10:30 Timeslot**
  - **Andri's teams**
    - 13 → 20
    - 20 → 13
  - **Daníel's teams**
    - 11 → 16
    - 16 → 11
  - **Matthias' teams**
    - 12 → 15
    - 15 → 19
    - 19 → 12
- **10:30-11:30 Timeslot**
  - **Andri's teams**
    - 23 → 26
    - 26 → 23
  - **Daníel's teams**
    - 24 → 27
    - 27 → 29
    - 29 → 24
  - **Matthias' teams**
    - 25 → 28
    - 28 → 30
    - 30 → 25

Assignment 4: Code Review – Partner Teams

- To find partner team members' e-mail addresses:
  - Talk to them in Thursday's consultations
  - Check Team Sign-up Doodle form (https://doodle.com/poll/v56843zigu2vkbub)
  - Ask your tutor

- **If you really want to change your partner team:**
  - Choose another team of the same tutor in the same timeslot
  - Rearrange reviewing relationships as needed and ensure all teams agree
  - Complete the reassignment and inform tutor (with teams in CC) by Fri 9 Nov

Quiz #7 Solution: Domain Models vs. Design Models

- Complete the following sentences with **(1)** "…in domain models", **(2)** "…in design models", or **(3)** "…in domain and design models":

a) Classes such as `EventListener` and `ArrayList` can be found… **(2)**

b) Failing to include an important aspect is more likely a risk… **(1)**

c) Methods are usually not defined for classes… **(1)**

d) Abstract classes can be used… **(3)**

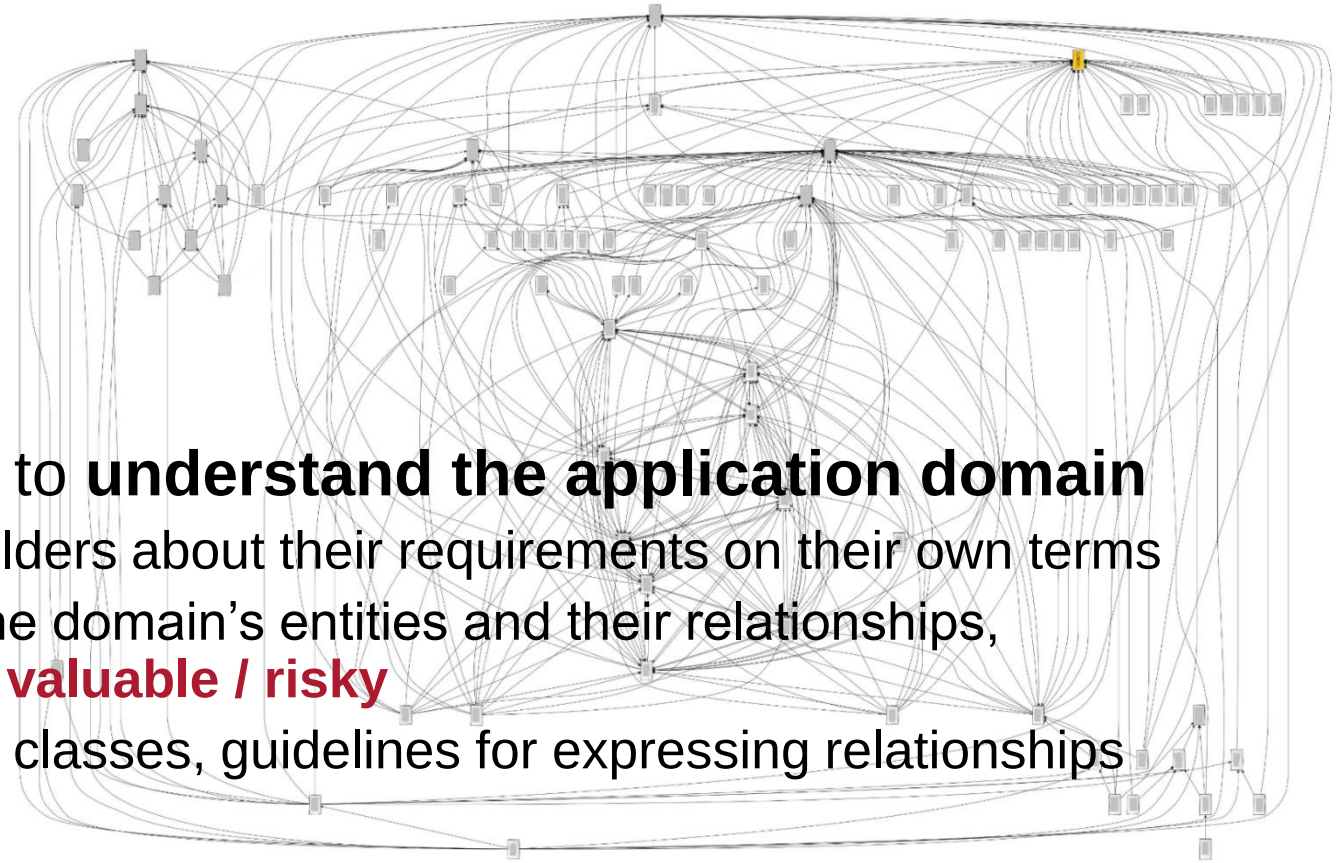
e) Discussions among developers revolve around aspects expressed… **(3)**

f) Classes such as `Customer` and `Contract` can be found… **(3)**

g) Modeling something in an inefficient way is more likely a risk… **(2)**

h) Associations are usually unidirectional… **(2)**

Recap: Domain Models vs. Design Models



Focus of this lecture

- We create **domain models** in order to **understand the application domain**
  - So we can talk to the business stakeholders about their requirements on their own terms
  - ➢ Key challenge is understanding all of the domain's entities and their relationships, i.e. **not missing anything important / valuable / risky**
  - Supported by: Strategies for identifying classes, guidelines for expressing relationships

- We create **design models** in order to express **how our solution shall work**
  - So we can talk to fellow developers about what the best technical solutions would be
  - ➢ Key challenge is engineering a solution that enables efficient implementation, execution and maintenance, i.e. **creating an efficient design**
  - Supported by: Object-oriented design guidelines, design patterns

Large Software Project Challenges

- Typical observations in enterprise software projects:
  - **Poor overview** of business domain requirements and technical design rationales
  - **Insufficient understanding** of risks and uncertainties
  - **Negligence of value and effort drivers** in favor of more trivial, but better understood aspects
- Plan-driven process models don’t fix this
  - Reading a specification does not mean understanding it
  - **Most tricky aspects often buried** under heaps of trivia
- Agile process models encourage (and actually depend on) more interaction
  - but provide **little guidance for communicating** about what is really important in a project

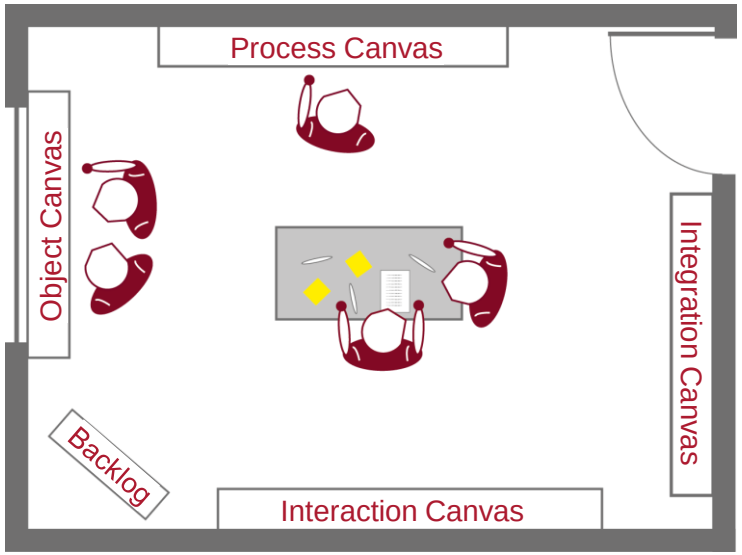
Communication Goals

- Support understanding and collaboration of all stakeholders
  - Achieve joint project ownership of Business and IT
- Make dependencies between processes, data and system landscape transparent
  - Create a basis for cooperation and agreement
- Focus on complexity, effort and value drivers
  - Put development focus on most important aspects
- Reveal uncertainties and misunderstandings
  - Create a basis for clarification

The Interaction Room

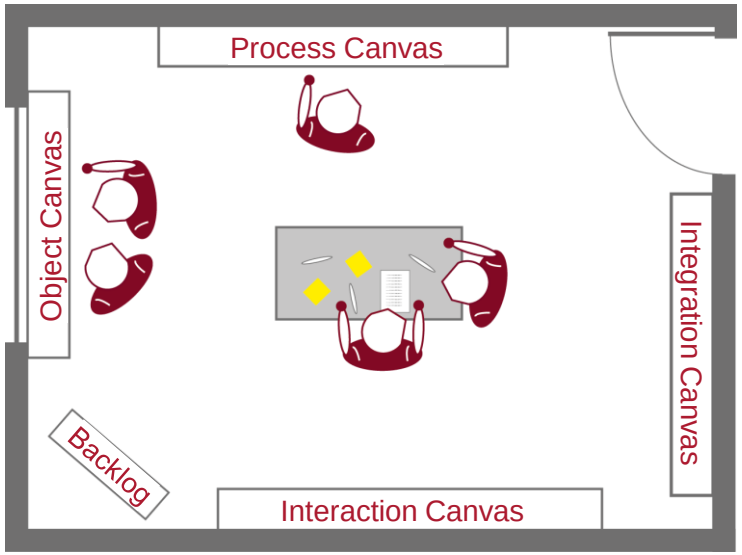
- Successful projects require
  - personal, focused discussion of most critical project aspects
  - early recognition of value and effort drivers
  - early elimination of risks and uncertainties

- The Interaction Room is
  - a **dedicated room** for the project team
    - where business *and* technical stakeholders feel at home
  - with **large whiteboards** on all walls
    - but without a classic conference table
  - to **visualize and discuss** key project aspects
    - instead of going over tedious documents



Interaction Room Canvases

- **Process canvas**
  - Visualizes business processes that the system needs to support
- **Object canvas**
  - Describes relationships between business and technical data structures
- **Integration canvas**
  - Illustrates interfaces and dependencies with external systems
- **Interaction canvas**
  - Sketches dialog flow and look & feel of key dialogs



A Pragmatic Approach to Software Modeling

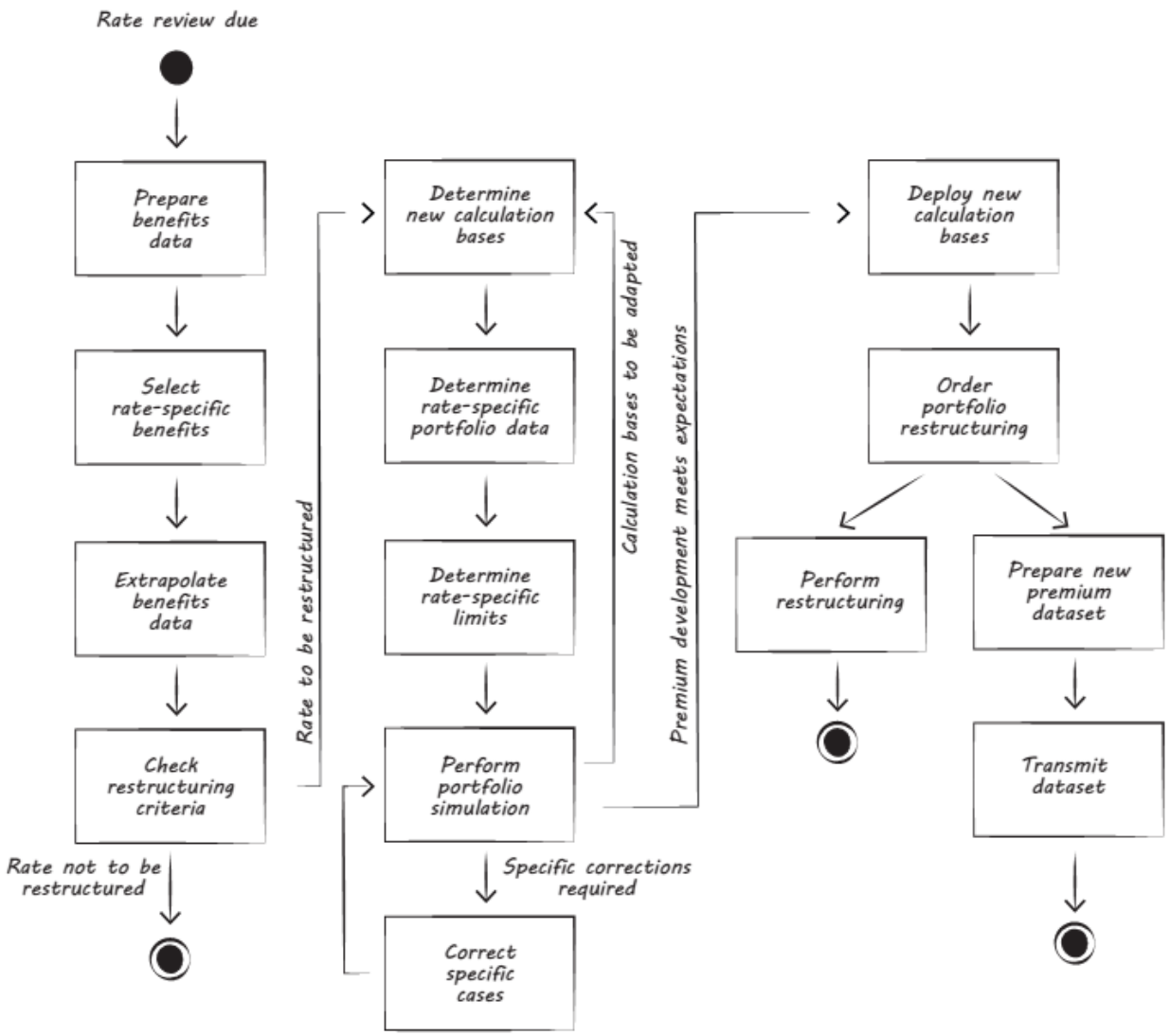
- **Informal, high-level sketches** of software models
  - sacrifice formality, consistency, completeness
  - in favor of focus, pragmatism, ease of use, interdisciplinary understanding, value-orientation

- Not a replacement for formal software specifications
  - Can be delegated to expert groups where necessary

- Informal sketches serve as **catalysts for the identification, understanding and discussion** of the most critical project aspects
  - High-level orientation about project goals, dependencies, conflicts
  - Early identification of value and complexity drivers, risks, uncertainties
  - Cross-departmental communication about priorities and trade-offs

Process Canvas

- **Purpose:**
  - Visualize business processes from the domain experts' perspective
- **Methodology:**
  - Sketch processes as agreement about them emerges from team discussion
  - Focus on main cases, no exotic exceptions
- **Notation:**
  - Loosely based on UML activity diagrams
  - Can be simplified if necessary

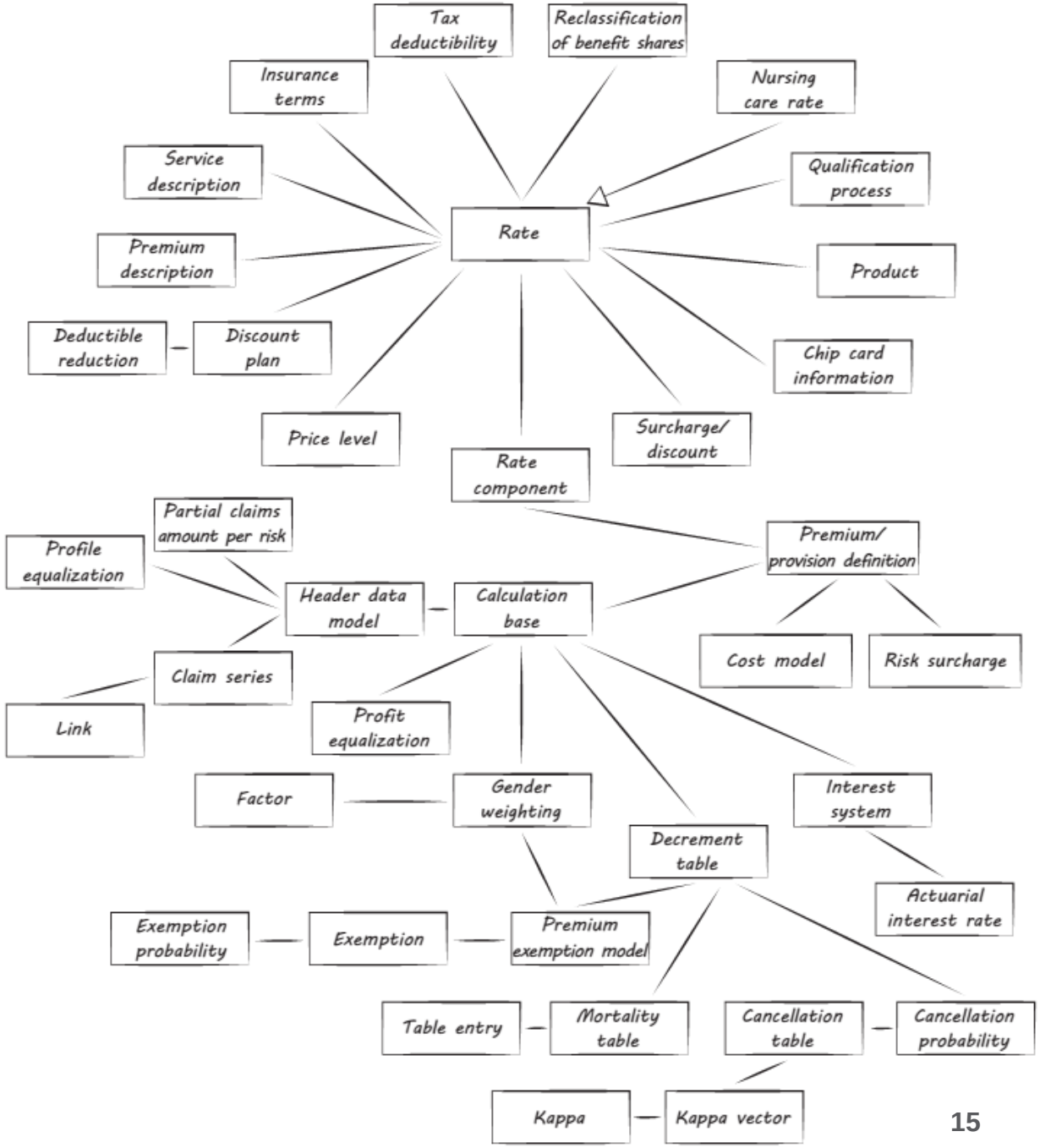


Object Canvas

- **Purpose:**
  - Visualize domain objects
    that are relevant for business processes
    and their relationships

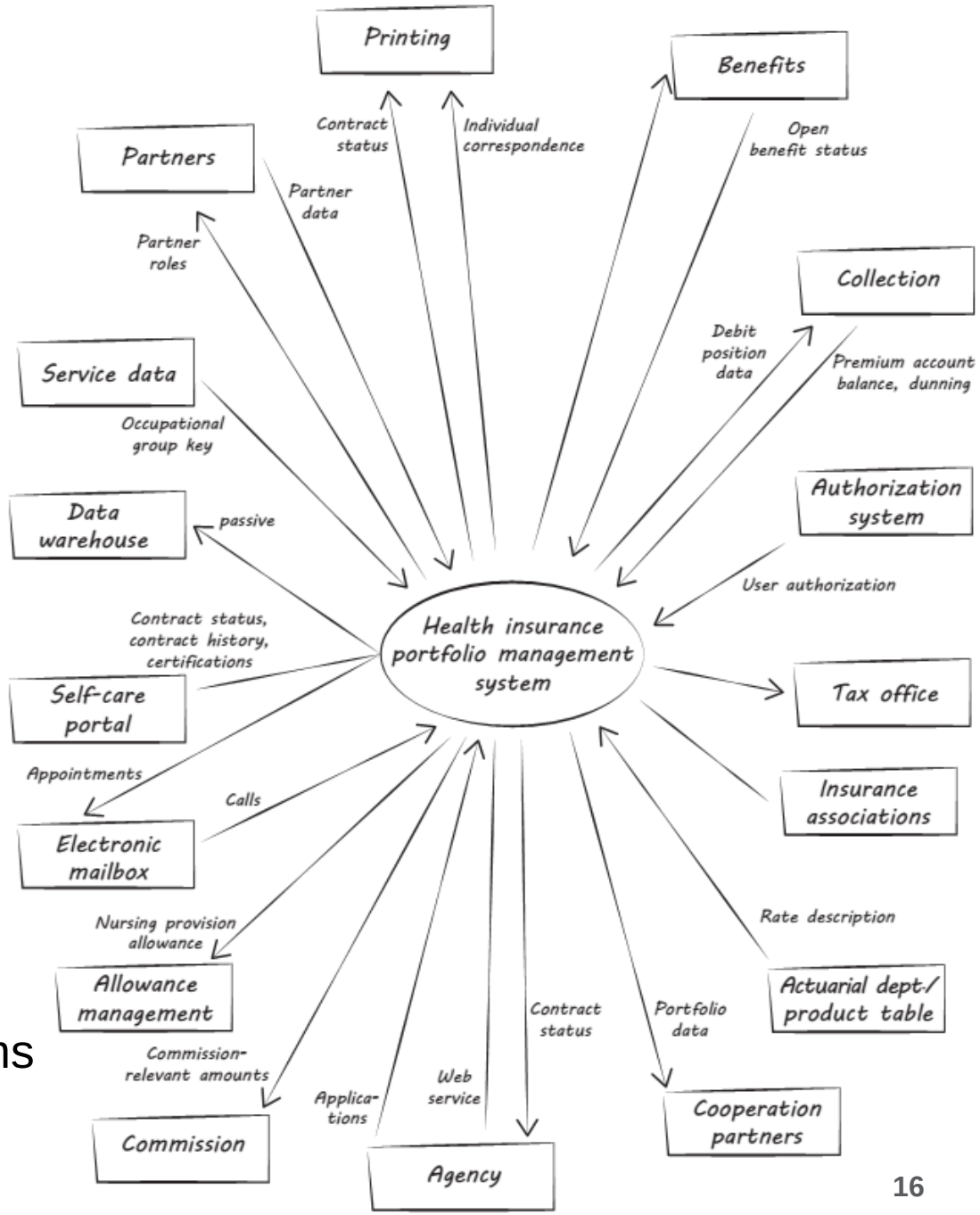
- **Methodology:**
  1. Record objects as they become obvious
     during process sketching
  2. Complete sketch with relationships
     and additional objects

- **Notation:**
  - Loosely based on UML class diagrams
  - Can be simplified if necessary

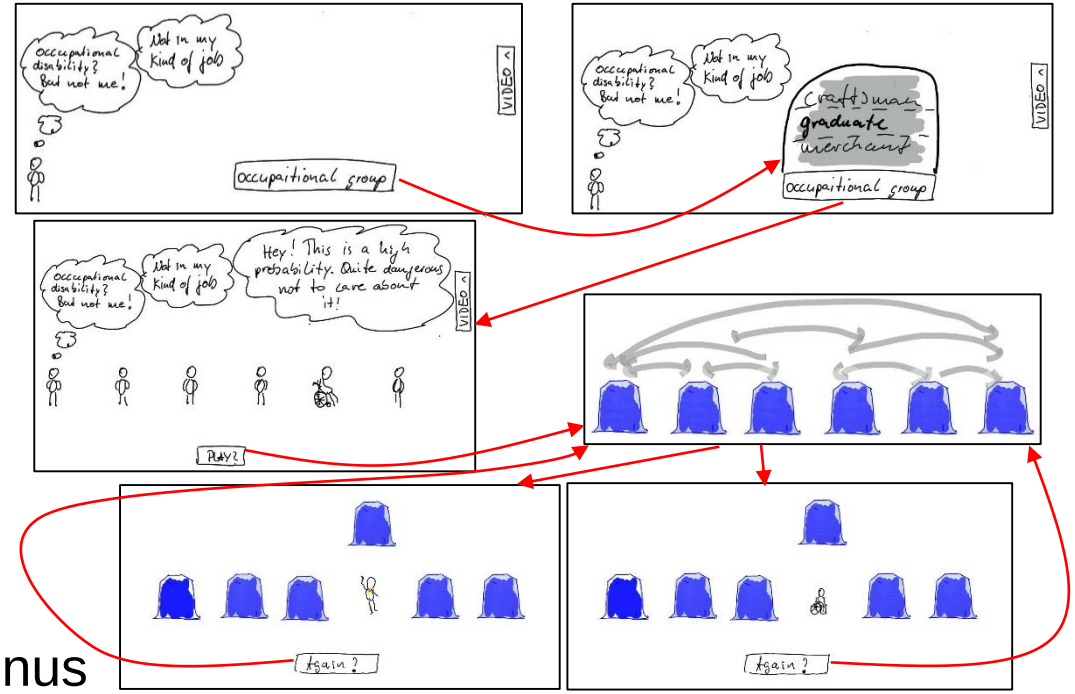


Integration Canvas

- **Purpose:**
  - Visualize interfaces and dependencies with surrounding systems
- **Methodology:**
  - Created system is located in the center of the sketch
  1. Record components as they become obvious during process and object modeling
  2. Complete sketch with relationships and additional components
- **Notation:**
  - Loosely based on UML communication diagrams
  - Can be simplified if necessary



Interaction Canvas



- Users think in dialogs, not abstract processes
  - Often more helpful perspective to elicit knowledge
- **Dialog flows**
  - (Conditional) transitions between dialogs
  - Separation of business-inspired navigation and menus
- **Storyboards**
  - Sketches of dialogs' look & feel
  - Interaction with and changes within a dialog
  - Usage of mobile device functions (touch, shake, etc.)

Model Annotations

- Highlight model elements that **merit particular consideration**:
  - **Value** annotations
  - **Effort** annotations
  - **Risk** annotations

- **Shift attention** from what is visible in models
  - to what is implied, what is assumed, what is unknown
  - i.e. those aspects that often make or break a project

Value Annotations

- **Business value:** Entities with central importance for the organization's core business
  - Example: Recommendation or user profiling algorithm

- **User value:** Entities with central importance for the user's value perception
  - Example: Key services or dialogs, e.g. user reviews

Effort Annotations: Quality Requirements

- **Reliability:** Desired availability and robustness requirements
  - Example: Avoiding duplicate entries after merging databases
- **Security:** Entities with particular requirements wrt. authentication, encryption, journaling etc.
  - Example: Preventing deletions of insurance claims records
- **High use:** Possibility of temporary high loads
  - Example: Deadline rushes, unforeseen events
- **Time constraint:** High-performance or real-time requirements
  - Example: Stock market transactions
- **Mobility:** Features/components that must be available anywhere
  - Example: Claim reporting tool for car insurance system

Effort Annotations: Quality Constraints

- **Usability:** Particularly complex dialog
  - Example: Complex insurance application form

- **Flexibility:** Components that will be exposed to change in the future (i.e. design for change)
  - Example: Foreseeable adaption to tax legislation

- **Manual task:** Process features that cannot be implemented in IT but require human expertise
  - Example: Coverage decisions by health insurance case managers

Effort Annotations: Domain & Technical Constraints

- **Policy constraint:** Rules limiting design options
  - Examples: Solvency II, ITIL, SEPA, internal Policies

- **Deprecation:** Component to be replaced/eliminated/migrated
  - Example: Legacy data for discontinued product

- **Invariability:** Component that must not be changed
  - Example: Key legacy system whose adaption requires too much effort or expertise

- **Need for improvement:** Component that must be improved
  - Example: Resolution of technical debt

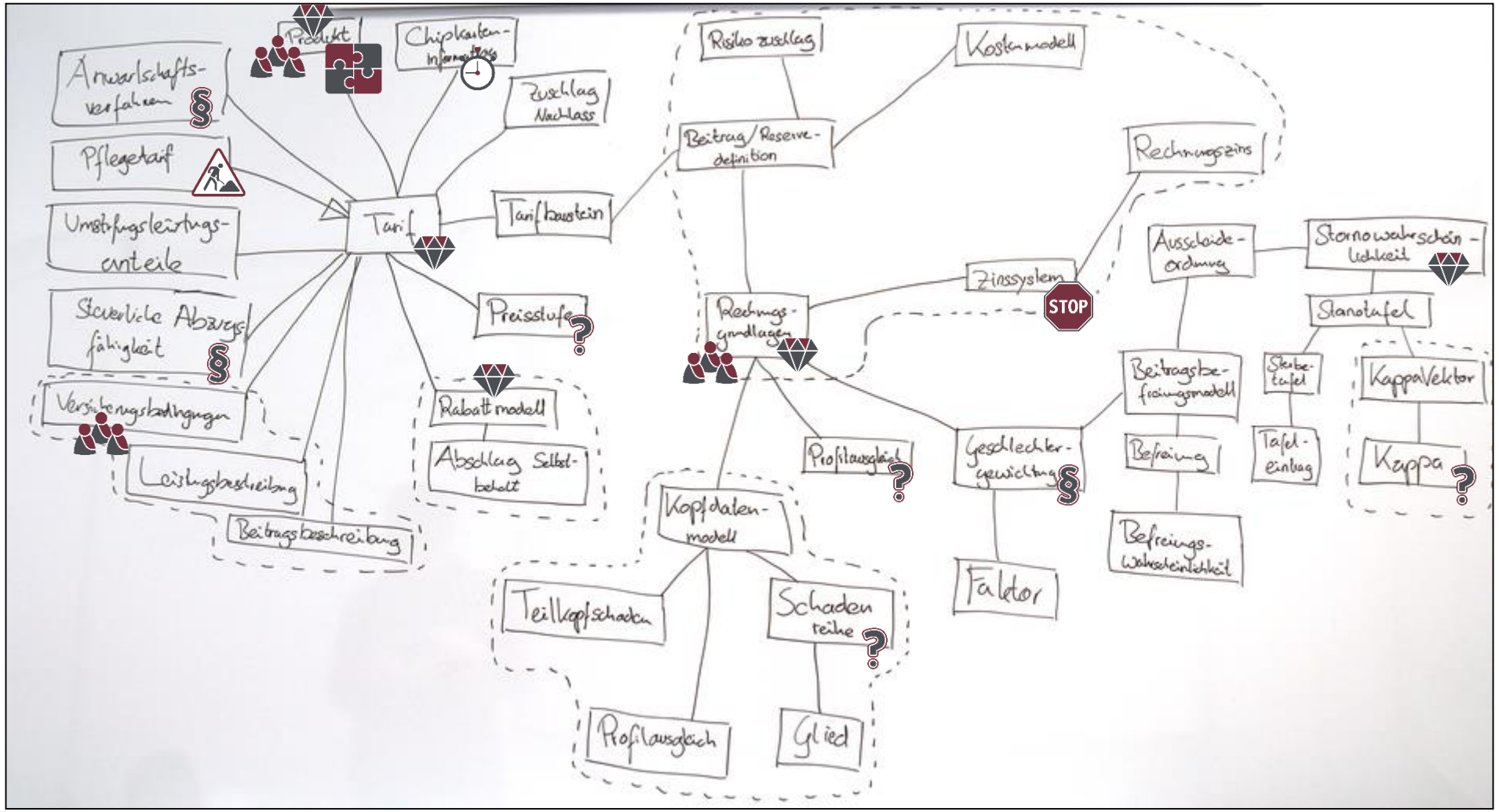
- **External resource:** Data/service provided by 3rd party
  - Example: Credit card validation service

Risk Annotations

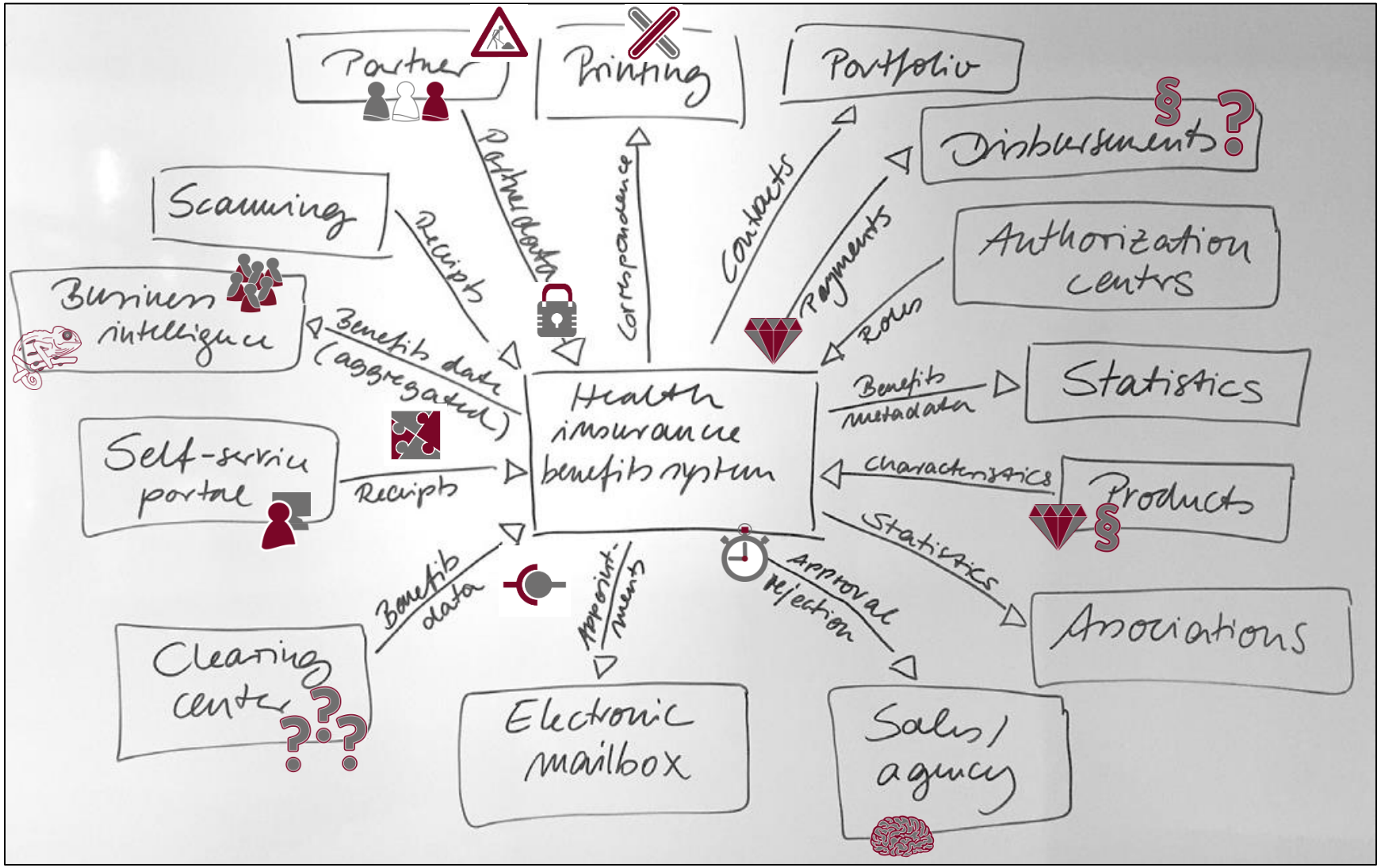
- **Uncertainty:** A stakeholder lacks the business/technical expertise required to design/implement a component
  - Example: Unfamiliarity with a certain protocol among some stakeholders

- **Complexity:** Particular design/implementation challenges
  - Example: Expert knowledge required to fine-tune reinsurance underwriting heuristics

Example: Annotated Object Canvas



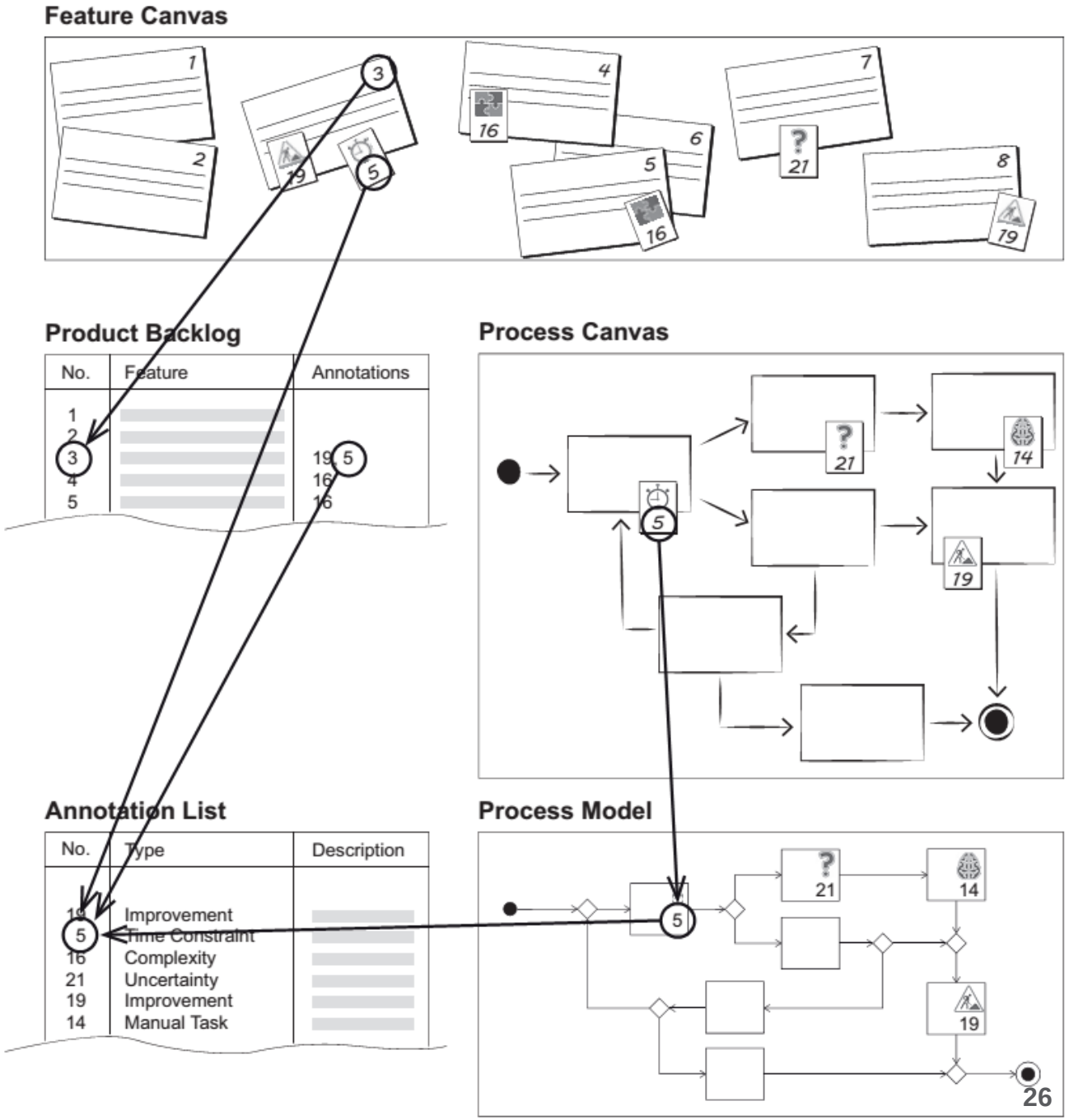
Example: Annotated Integration Canvas



Annotation Traceability

- For each annotation, team records
  - what exactly is the highlighted issue
  - what impact it has on the business
  - how complex it will be to handle

- Knowledge that is usually tacit
  - is elicited early
  - and explicitly attached to artifacts for later consideration in construction



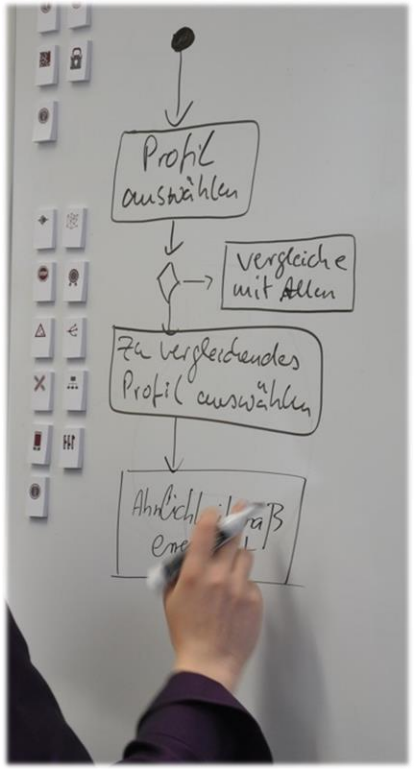
Interaction Room Workflow

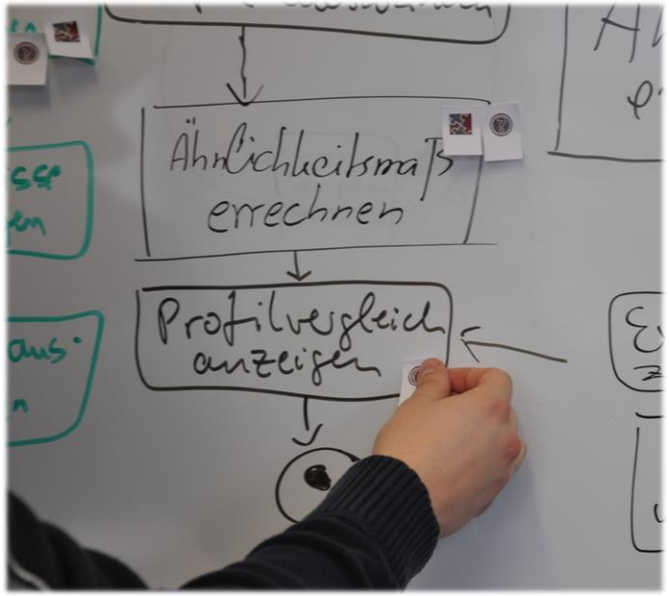
1. Elicit high-level requirements

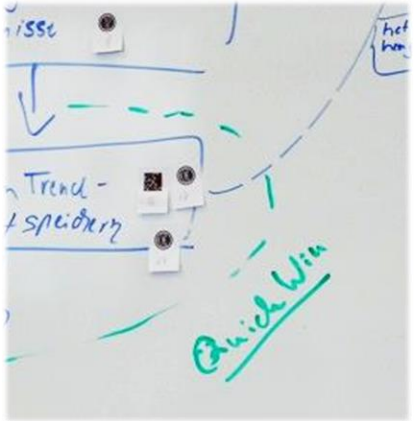
2. Sketch canvases
   - Process canvas
   - Object canvas
   - Integration canvas
   - Interaction canvas

3. Annotate canvases
   - Several rounds with 5-7 annotations each
   - Final round for uncertainty annotation

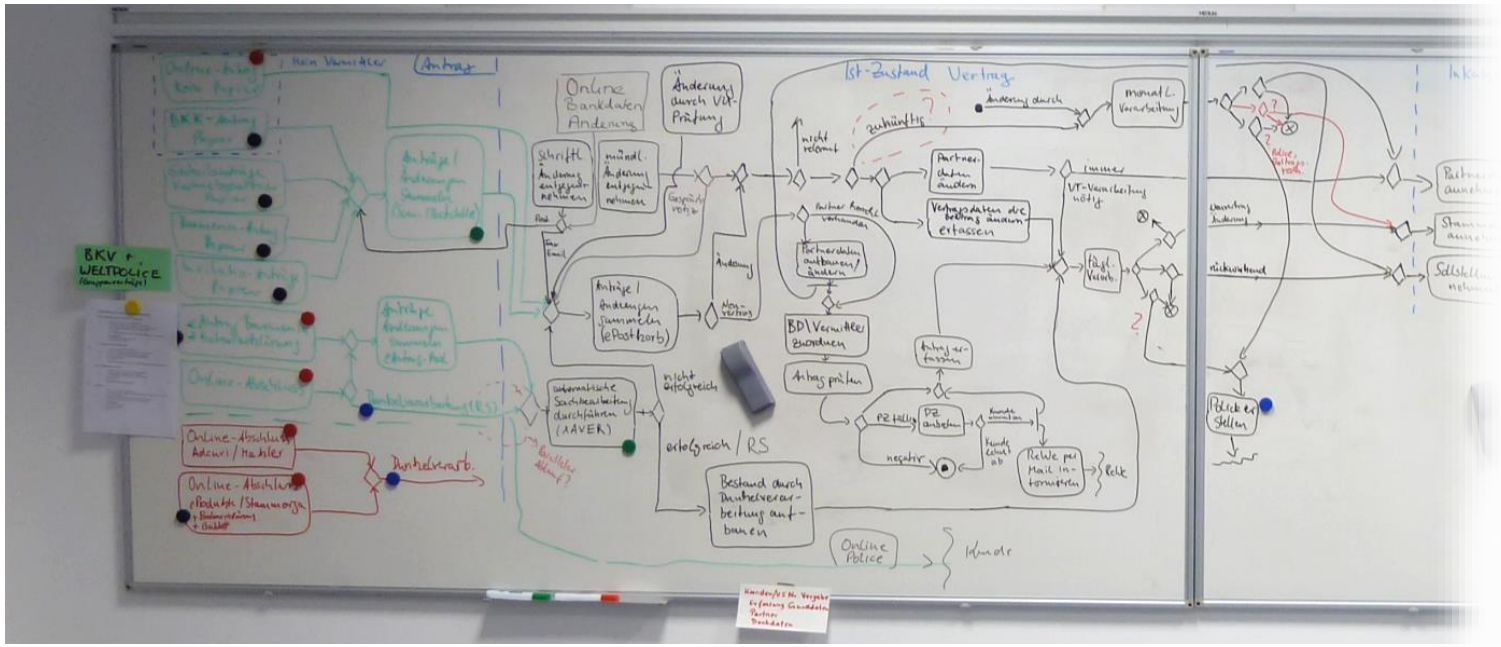
4. Plan sprint(s)

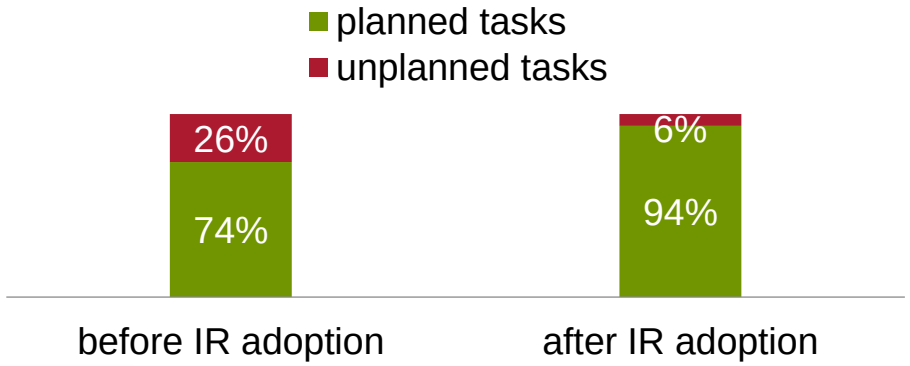


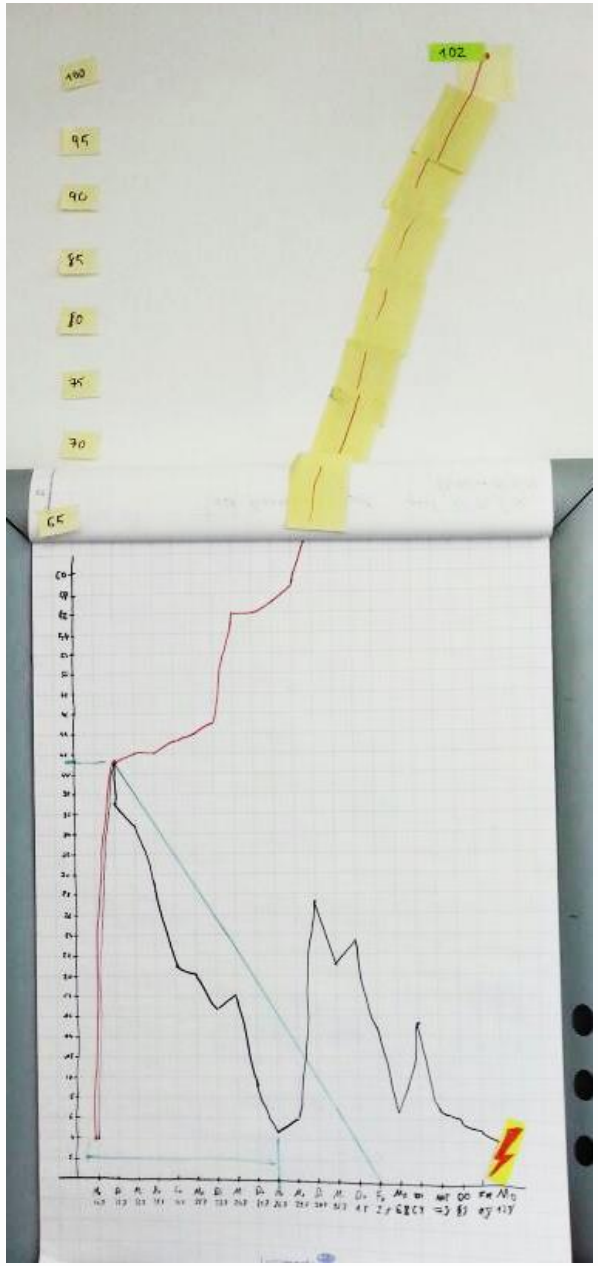




Industry Project Example



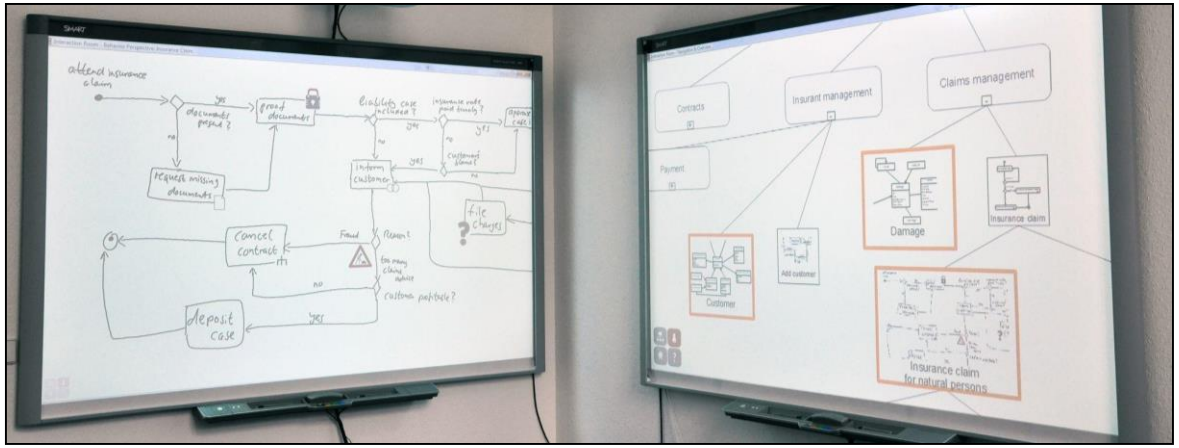




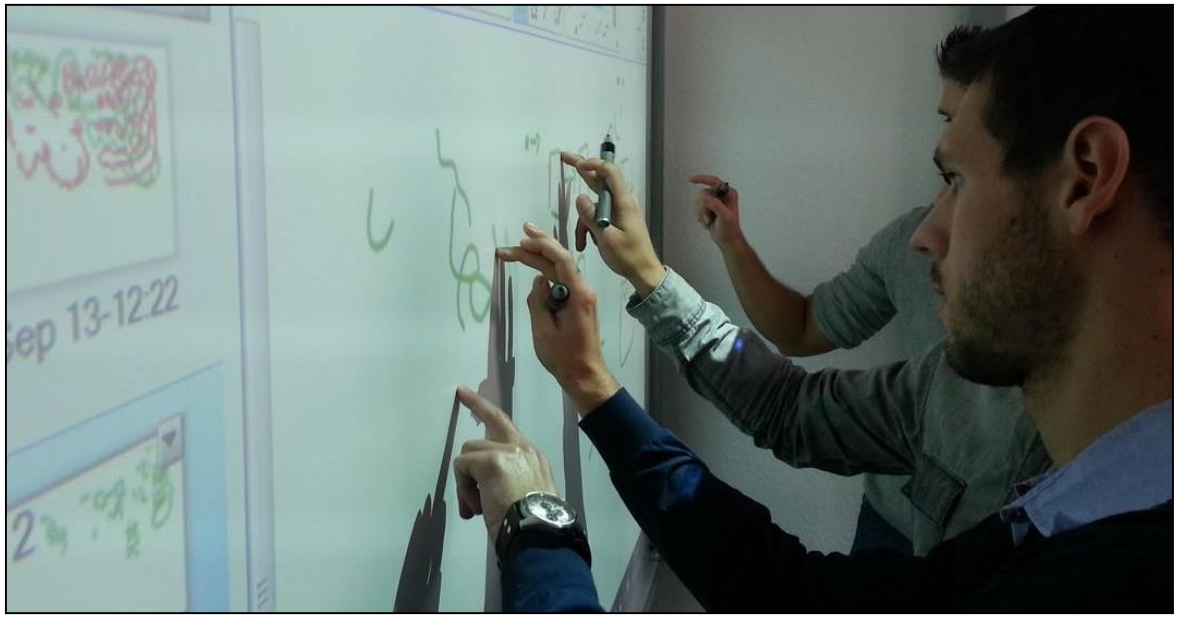
Ongoing Research: Augmented Interaction Room (AugIR)

(→ BSc/MSc Project Opportunities :-)

- Emulates and augments whiteboard through intuitive **drawing features**
  - Infinite canvas
  - Natural distinction of pen / eraser / lasso
  - Horizontal and vertical model navigation



- Automatic generation of **trace links** between sketches
  - based on recognized handwritten labels



- Enables **annotation** of sketches
  - Facilitates traceability of annotation information across artifacts

In-Class Quiz #8: Interaction Room Concepts

- **What is the purpose (1-4) of the following canvases (a-d)?**

a) Integration canvas
b) Interaction canvas
c) Object canvas
d) Process canvas

1. Describes relationships between business and technical data structures
2. Illustrates interfaces and dependencies with external systems
3. Sketches dialog flow and look & feel of key dialogs
4. Visualizes business processes that the system needs to support

- **What is the meaning (5-8) of the following annotation symbols (e-h)?**

e) 

f) 

g) 

h) 

5. Business value
6. External resource
7. Security
8. Uncertainty

Further Reading

- Book, M., Gruhn, V., Striemer, R.: Tamed Agility – Pragmatic Contracting and Collaboration in Agile Software Projects, Springer Intl. Publishing 2016. http://www.springer.com/gp/book/9783319414768

- Grapenthin, S., Poggel, S., Book, M., & Gruhn, V. (2015): Improving task breakdown comprehensiveness in agile projects with an Interaction Room. Information and Software Technology, 67, 254–264. http://doi.org/10.1016/j.infsof.2015.07.008

- Book, M., Grapenthin, S., Gruhn, V.: Highlighting Value and Effort Drivers Early in Business and System Models. Proc. 13th Intl. Conf. on Intelligent Software Methodologies, Tools and Techniques (SoMeT 2014) – Revised Selected Papers. Communications in Computer and Information Science (CCIS) 513, Springer 2015, pp. 211-222. http://dx.doi.org/10.3233/978-1-61499-434-3-530

- Book, M., Grapenthin, S., Gruhn, V.: Seeing the Forest and the Trees: Focusing Team Interaction on Value and Effort Drivers. Proc. 20th Intl. Symp. on the Foundations of Software Engineering (ACM SIGSOFT 2012 / FSE-20) – New Ideas Track, ACM 2012, no. 30. http://dx.doi.org/10.1145/2393596.2393630