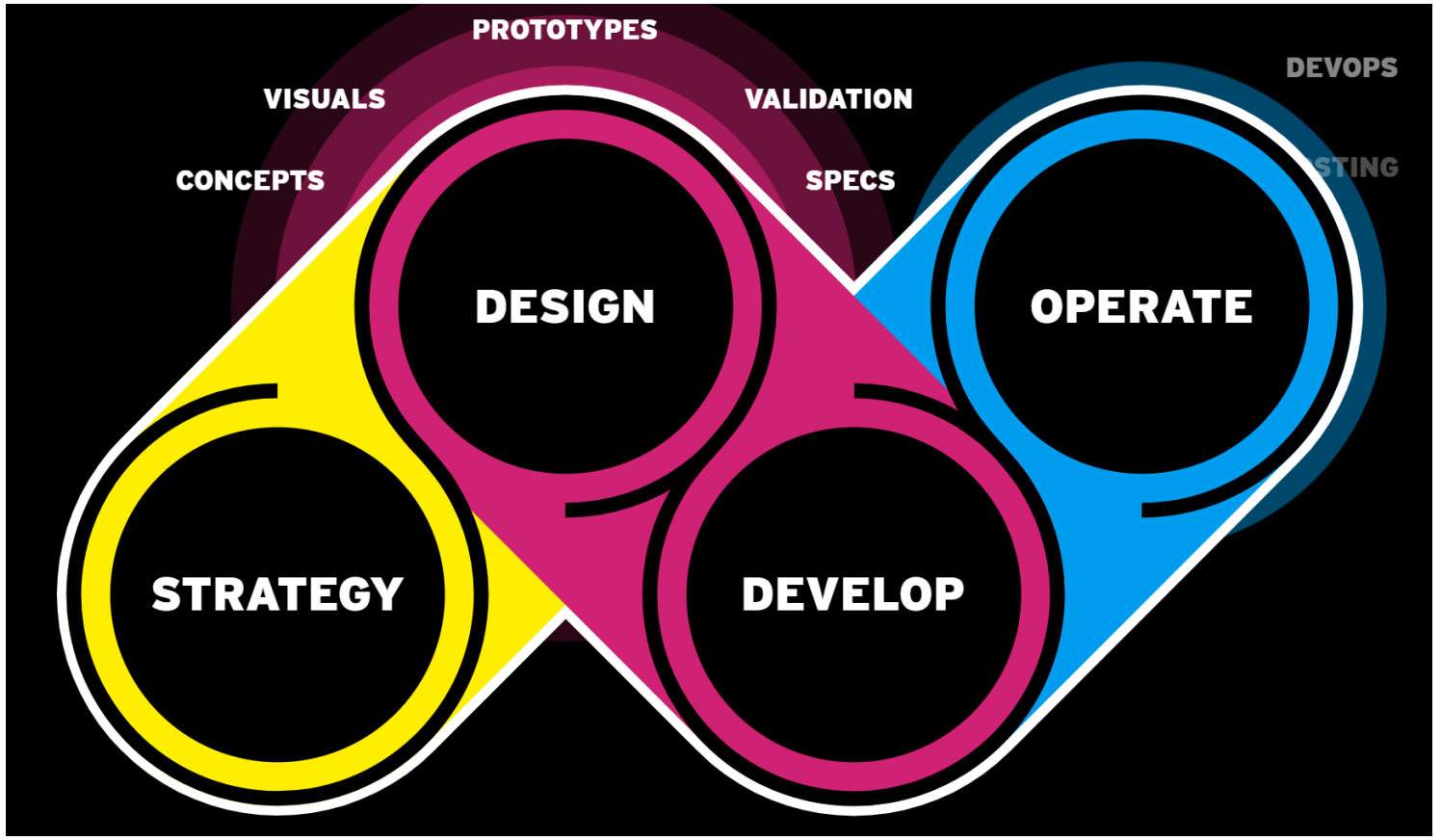
Collaborative Design: Guest Lecture via Zoom

Developing User-centered Products using Design Thinking and Collaborative Design

Stefan Schulz, ergosign

Monday, July 25th.

11:30 am



Requirements Engineering

Lecture 9

Prof. Dr. Alexander Pretschner

Chair of Software & Systems Engineering
TUM Department of Informatics
Technical University of Munich

Orientation

Recap:

- Formulating different types of requirements
- Formalization
- Pros and Cons of Formalization

Coming up:

- How are Requirements handle in agile processes?
- How can we align all discussed techniques with agile development?

Even in agile contexts, requirements do not just appear!

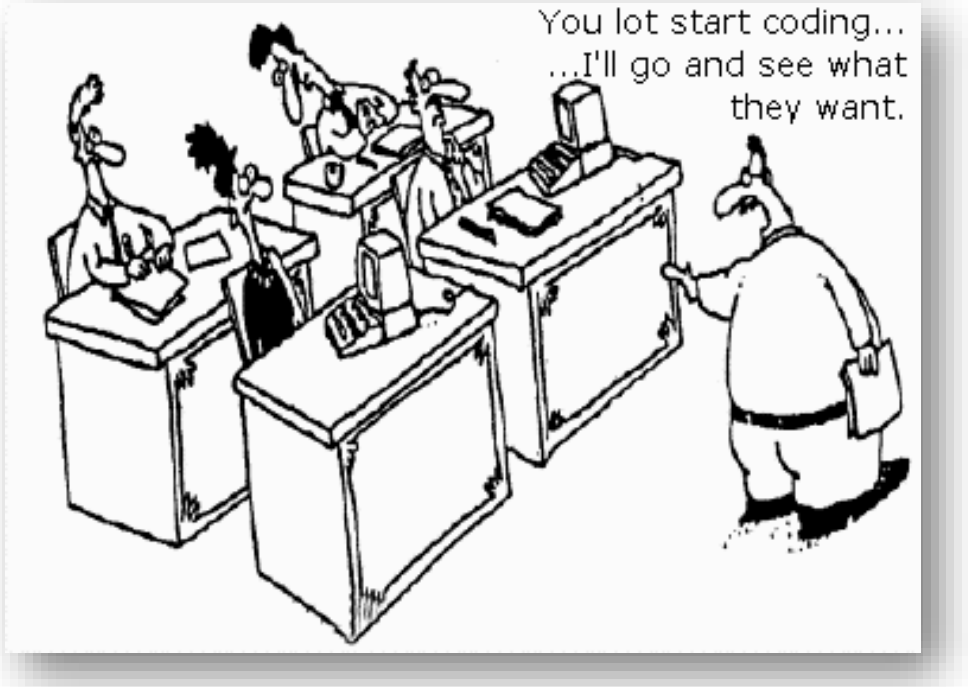
Remember:

Explicit versus implicit Requirements Engineering

Agile Development

$\neq$

No Requirements Engineering



Picture taken from: http://acceleratteddevelopment.ca/blog_images/You_lot_start_coding.gif

First thing to do: Product Vision

The product vision is a short and concise description of the product. It helps to unify the understanding of the product, makes goals explicit and helps assessing the business value of requirements.

FOR <target customer>
WHO <statement of need or opportunity>
THE <product name> **IS A** (product category)
THAT <key benefit>
UNLIKE <existing competitive product>
OUR PRODUCT <primary difference to alternatives>

Not to be confused with a slogan, which is targeted towards emotions and marketing.

Agile Manifesto

*“Individuals and interactions **over** processes and tools*
*Working software **over** comprehensive documentation*
*Customer collaboration **over** contract negotiation*
*Responding to change **over** following a plan*

That is, while there is value in the items on the right,
we value the items on the left more.”

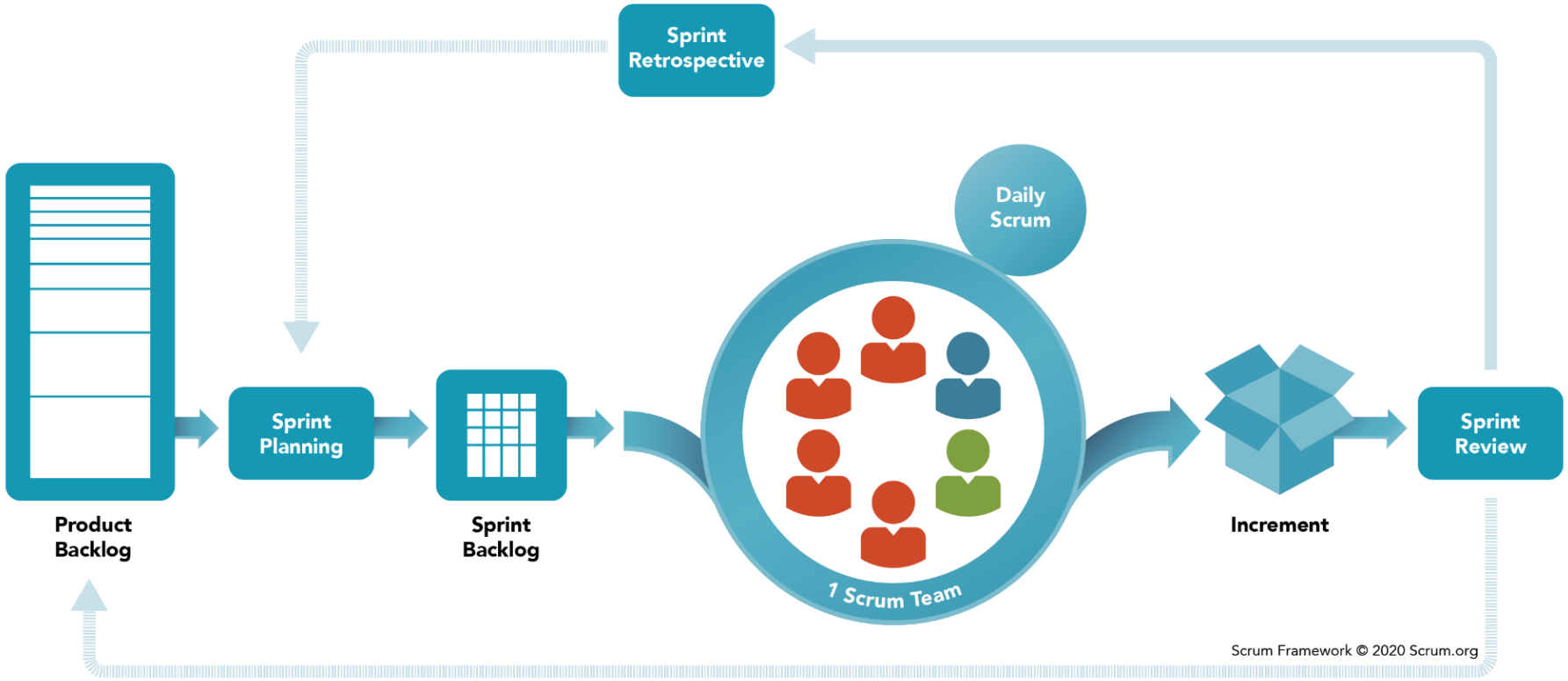
- The Agile Manifesto

Agile Development

Scrum

However, we will use Scrum as a running example

Scrum



In Scrum, requirements exist as Product Backlog items and Sprint Backlog items.

Typically, they are denoted as user stories.

Typical requirement format: User Stories

Purpose: Bridge between user, customer and developer

As a <role>, I want <functionality>, because <goal>.

Ron Jeffries' three C's:

- **Card**: A user story should fit on a card, ideally featuring a title and a summary
- **Conversation**: *"A user story is an invitation to a conversation."*
  The user story is just a summary of the user's need!
- **Confirmation**: Validate your understanding and set a definition of done!

User Story Card

Story # <Number> | <Title>

Prio

As a <role>,
I want <functionality>,
because <goal>.

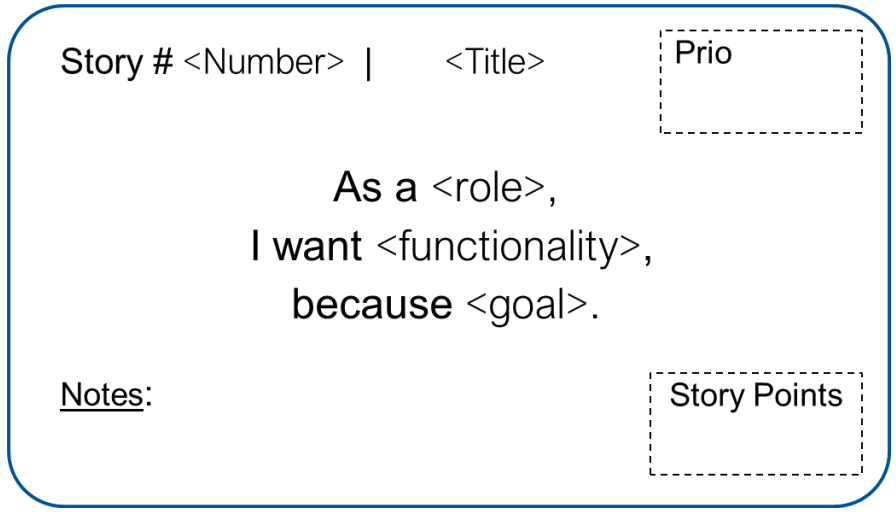
Notes:

Story Points

Elements:

- Story ID: unique identifier
- Title: optional
- Priority: prioritization
- Story Points: estimated effort
- Notes: additional information

Invest in User Stories!



I	independent	No additional info needed
N	negotiable	Customer is available for more details and discussion *
V	valuable	Value for the user (!)
E	estimable	Size must be estimable
S	small	Appropriate size, both in terms of effort and length of description *
T	testable	Acceptance criteria can be defined *

* Remember the three C‘s!

Discussion: Is this a good User Story?

“As a scooter renter, I want to see where available scooters are and my distance from them so I can decide which one to book.”

Example by Maria Chec : https://uxdesign.cc/user-stories-in-practice-part-2-8b731566f8ca

Refinement of the example

Independent: Do I have to be logged in? If yes, that functionality has to be implemented first!

Small: There are actually two functionalities described: showing location and showing distance

Value: Goal is not clear. Why does the user want to decide?

We can split this into two stories:

“As a scooter renter, I want to see the location of the available scooters to choose the one which is easiest to reach”

“As a scooter renter, I want to know the distance of the available Scooters to choose the closest one”

Identifying „bad“ user stories: Common mistakes

Role-related:

- Unspecific, faceless user: *“As a user, I ...”* Name the specific role! E.g. “frequent customer”
- Implementation detail without user value: *“As a developer, I want to use Tensorflow 2.4.1”*

Goal-related

- Not separating problem and solution: *“… I want a dropdown menu so that I can select an option”*
- Goal repeating the function: *“…, I want to edit my details so that I can change them.”*
- Mixing goal and function: *“… I want to log in, so that I can rent a scooter”*

Indicators that the story should be split:

- The word “and”: “*As a .. I want to do …. AND …. “*
- If people start using the ID of the user story instead of its title, the story might be to complicated

Scope of a user story

A User Story should have **appropriate size.**
This means it should fit into a sprint. The length of a sprint depends on the project.

Effort estimation is easier and more precise for small stories.

Smaller stories are easier to prioritize.

Just writing things into Jira or a User Story template does not make it a user story!

Definition of Done

Enables progress control, and guidelines for fulfillment.

A story is either done or not done. There is no intermediate state.

Every story has its own definition of done (Three C‘s: Card, Conversation, **Confirmation**).

Generic Requirements that apply to all user stories can be summarized in a **general DoD**.
Common Examples: *Update documentation, tested with specific coverage criterium, UI texts added in both German and English, non-functional requirements, …*

If a non-functional requirement is no longer fulfilled, the story is not yet done!

Non-functional Requirements

Non-functional Requirements typically are added to the Definition of Done.
The DoD defines how they are verified.

If the NFR can be refined into functional requirements, the NFR is implicitly part of the user story:

As a <role>, I want <functionality>, because <goal>

“As an app user, I want my password to be stored encrypted to avoid theft of my data.”

How to get User Stories?

Primary Source: Customer/User and Product Owner!

But, sometimes, there is no customer or user we could ask!

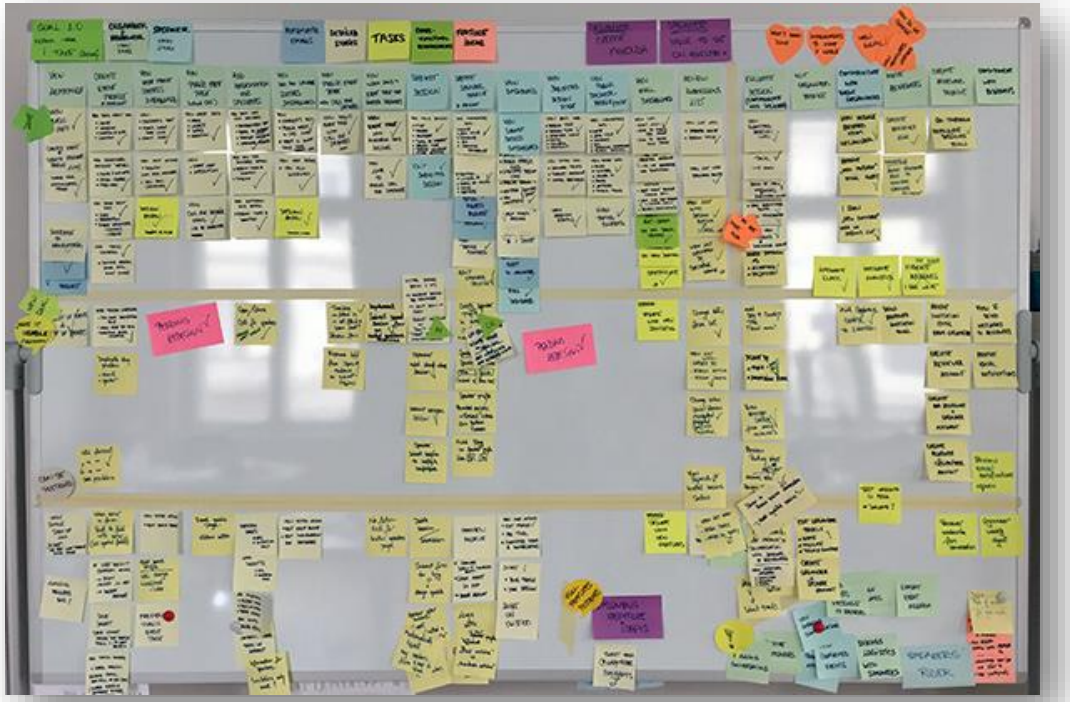
Secondary Sources:

- Find early users as soon as possible, e.g. by offering special conditions
- Create your own user! => Personas
- Sources for Requirements as discussed in previous lectures, e.g:
  - Design Thinking & Elicitation Techniques
  - Competitors, other Stakeholders, Workshops,
  - Laws, Literature, Standards, …

Managing User Stories

The current state is illustrated using a Scrum Board, Kanban Board, or similar technique.

Notably, the Story Board is often physical and offline.



Sprint Planning Meeting:

The whole team refines the user stories and discusses their prioritization and estimated effort. Then, they select which user stories to implement in the sprint based on the available time and expected value.

Note: in Scrum, effort estimation is not used for planning, but for decision making!

Effort Estimation

Planning Poker:

1

Secret voting:
Everyone chooses a card with a number describing the estimated effort.
Only certain values are allowed! (e.g Fibonacci-Numbers or 2^n)

Estimations are simultaneously revealed

2

Subsequent discussion:
First, team member with the lowest estimate explains her reasons, then the member with the highest estimate.

Other popular techniques: T-Shirt sizes, Story Points, Person Days, …

The main driver for agile processes is **Change.**

What are the consequences for RE?

Embrace Change!

User Stories are not set in stone, they are always **negotiable**

Incorporate user feedback

This includes: Establishing a mechanism to collect and evaluate this feedback!

The Product owner is responsible to mirror the target group‘s feedback in the meetings.

Accept imperfection:

Until the story is considered for the next sprint, accept to have a high-level, not perfectly detailed state for the time being.

Backlog refinement

Regularly adapt your requirements to change and new user needs

Mitigating the negative effects of Change

Change requires re-planning: Do not plan what you cannot plan in the first place.
Sprint Planning: only plan in detail what you can achieve in the next sprint!
During a sprint: the requirements that were agreed upon are fixed and must not change.
⇒ Choose the length of sprints according to the expected change!

Change requires the team to discuss everything again: No, the responsibility is pinned to one role only. The product owner identifies the need for changes and adapts the backlog items accordingly. Already refined items in the sprint backlog are not changed.

Change can make everything we already did obsolete: Yes, but to save time, the PO has the authority to call an „abnormal sprint termination“, e.g. if a security patch is needed immediately, or the sprint goal is no longer valid

Change will lead to wrong decisions: Must empirically manage the process, must set up a learning culture!

Refinement of User Stories

Sometimes called „grooming“

Elaborate on the details of a user story before it is added to the sprint backlog:

- Why do we need it? What is the business value?
- Should we split the story?
- What are acceptance criteria for the story?
- Considering the acceptance criteria, is the story technologically feasible?
- Effort estimation

Add the story to the Sprint Backlog or send it back to the Product Backlog

Agile Requirements Engineering: Summary

- Do not plan what you cannot plan!
- Learn to live with an imperfect state.
  Refinement and grooming will take place **at the right moment** (before the sprint), but do not waste time before that. By that, effort caused by change is mitigated.
- Get rapid custom and user feedback – PO must make sure to collect the user feedback!
- If there is no user or customer yet, try to find one as soon as possible.
- **INVEST** in user stories. Not everything that looks like a user story is a good and useful user story.

Developers take Responsibility

In agile processes like Scrum, the developers are responsible for prioritization of requirements, selecting the sprint backlog items, and **commit to the product vision and the sprint goal**

Conflicts between contrasting goals are resolved by discussion (at least in theory)

Empowerment of the developers leads to ethical responsibility!

Everything is negotiable.

No excuses such as “I didn't know what it will be used for”.

Remember Lecture 6

EDAP: Ethical Deliberation for agile software processes

Phase I: Descriptive system analysis

Phase II: Descriptive value analysis

Phase III: Technical Analysis

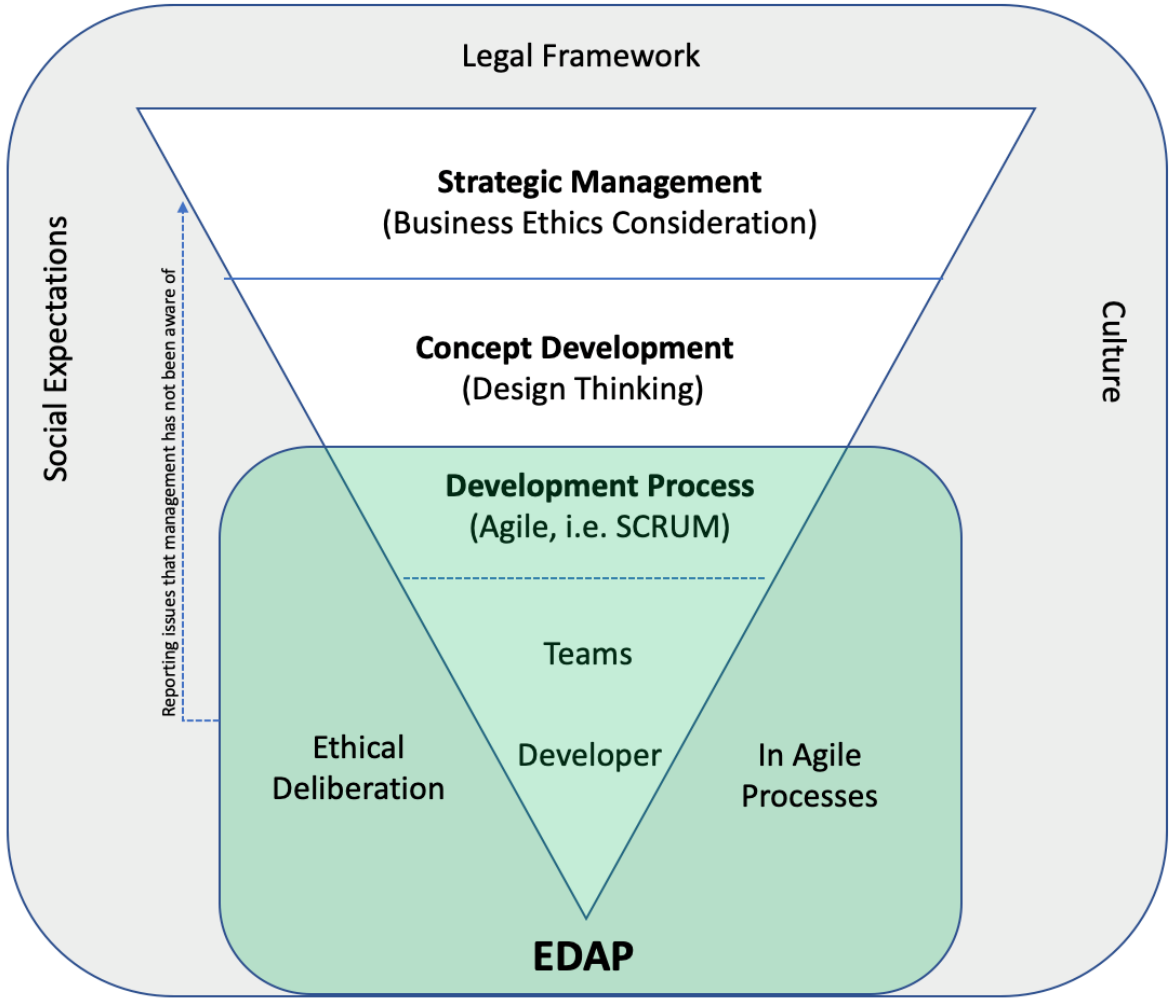
Phase IV: Value or goal conflicts

Phase V: Ethical System Review

Phase VI: Judgment Phase (Coherence)

Phase VII: Technical feasibility

Phase VIII: Verification



Discussion: RE in large-scale agile processes

How can we apply these concepts to large-scale agile processes like SAFe or Scrum of Scrum?

Is ethical deliberation still possible?
Are our lessons learned for RE still valid?

Outline and Outlook

Terms and Definitions
Context-specific nature of SE and RE
Quality models for requirements
Engineering Models
Stakeholder and Requirements Elicitation
Goals and Goal-oriented RE
Non-functional Requirements
Functional Requirements
Formalization

Agile Processes

Requirements Management and Quality Assurance
Trends in Research