Introduction to TReqs

Tutorial: Requirements Analysis and Decomposition for Distributed Systems based on Deep Learning at RE'23, Hannover

Eric Knauss knauss@computer.org

Chalmers | University of Gothenburg, Sweden

2023-09-04

Motivation: Need to support decentralized RM

Key idea and concept of TReqs

TReqs demonstrator

References

UNIVERSITY OF GOTHENBURG

CHALMERS UNIVERSITY OF TECHNOLOGY

TReqs Pitch

- *Objective:*
  - TReqs offers lightweight tooling to manage requirements in agile system development.
- *Philosophy:*
  - TReqs empowers agile teams to manage requirements together with changes of code and test.
  - This ensures transparency, consistency, scalability, and speed.
- *Offer:* The core of TReqs is available as open source. Based on this, we offer:
  - **Integrating** TReqs into a specific company's requirements strategy.
  - **Developing** a company's requirements strategy if needed.
  - **Adjusting** TReqs to match specific needs
  - **Training** for agile teams, product owners, system managers.
- *AI-based systems:*
  - Requirements, Architectural Decision, Scripts, Tests, Code evolve together
  - Support middle-out development

Motivation: Need to support decentralized RM

Our background

- We have worked with many large systems engineering companies (Knauss 2019)
  - Software center with Axis, Ericsson, Grundfos, Saab, Siemens, TetraPak, Volvo Cars, Volvo Trucks (Kasauli et al. 2020)
  - Vinnova FFI (NGEA) with many automotive companies in the region (Pelliccione et al. 2017)
- We find them struggle with similar challenges (Liebel et al. 2016), (Kasauli et al. 2020)
  - Continuous Integration and Deployment (Knauss et al. 2016)
  - RE for Scaled Agile System Development (Kasauli et al. 2020)
- There is no single solution for all challenges
  - But our treqs tool and concept is pretty promising for many challenges (Knauss et al. 2018)

How requirements engineering has evolved

Now

Breadth-First RE		
Iteration 1	Iteration 2	Iteration 3
Just-in-time RE	Just-in-time RE	Just-in-time RE
Design	Design	Design
Write tests	Write tests	Write tests
Write Code	Write Code	Write Code

Requirements are everybody's responsibility

Requirements Engineering:

a knowledge management problem

Why agile teams hate requirements (Knauss 2019)

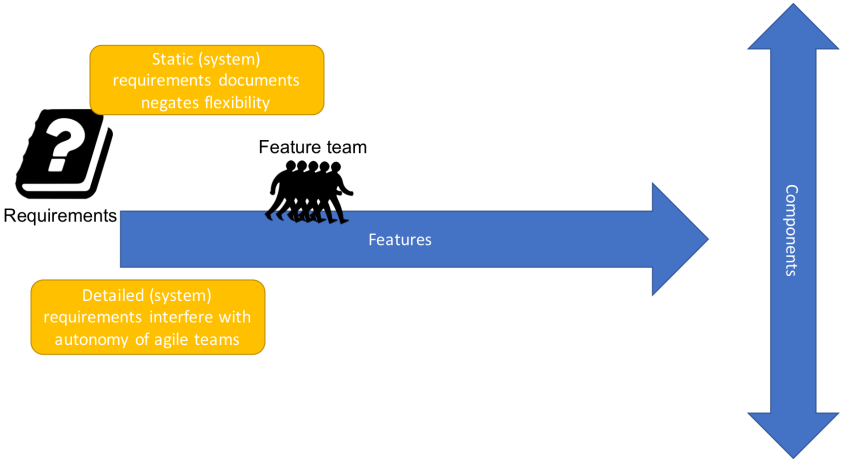

Figure 2: Static detailed requirements limit agility

Why agile teams hate requirements (Knauss 2019)

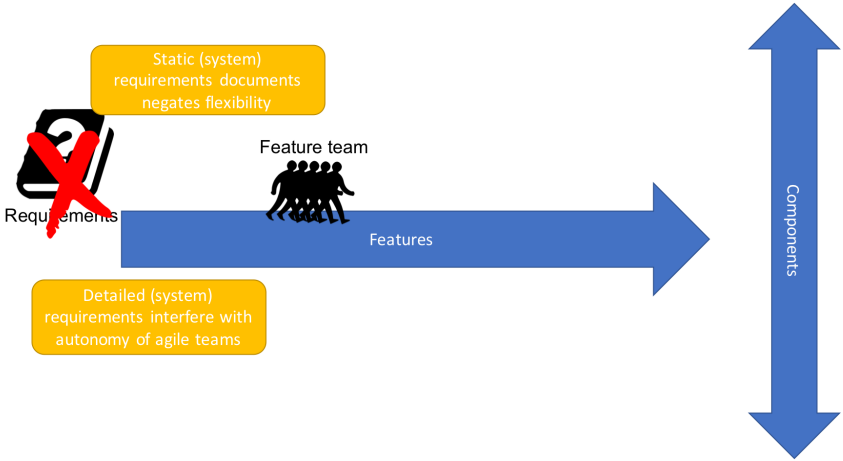

Figure 3: Easy solution, get rid of requirements

No requirements is not a good solution (Knauss 2019)

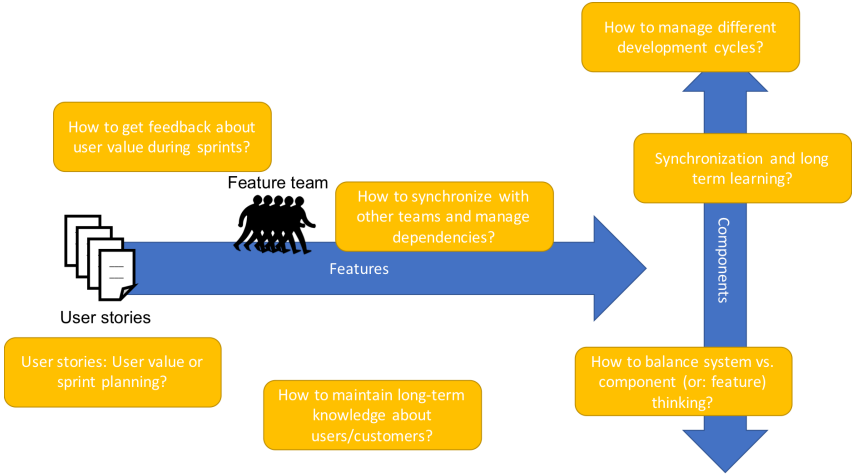

Figure 4: System engineering starts to fail if requirements are not managed

RE challenges frequently encountered in scaled-agile system development (Kasauli et al. 2020)

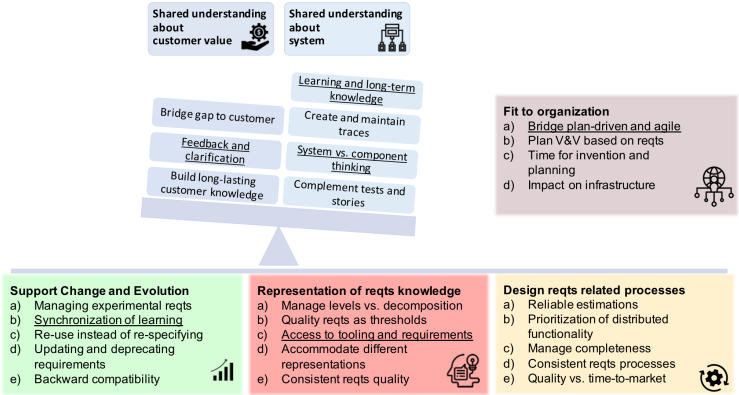

Figure 5: Common challenges in scaled agile system development

Context and Motivation

- Engineering complex systems increasingly done agile and continuous
- Agile (SW) teams will discover new or wish to update existing requirements
- Any system level role that owns requirements will become a bottleneck
- Agile teams avoid working with outdated requirements (don't read/don't update)
- System development looses a critical coordination mechanism

Additional needs for AI-based systems

- Support middle-out development
  - Iterate and make things work
  - Ensure that high-level system goals are achieved
- Manage infrastructure-as-code and requirements on infrastructure

Key idea and concept of TReqs

TReqs core idea

- Distinguish high-level customer/market/problem-focused requirements from system/solution-focused requirements
- Give agile (SW) teams ownership of system requirements in a scalable way
  - Bring the requirements into the tools that agile teams work with
  - Integrate reviews of requirements changes into the quality assurance workflows of agile teams
  - Derive reports for system level roles
  - Support DevOps vision of infrastructure-as-code
- Make explicit, how tweaks in data, scripts, code support system goals of **AI-based Systems**
- Result: scalability, speed, and real control through up-to-date requirements
  - Instead of: illusion of control by gatekeeping requirements that are not used

TReqs demonstrator

Getting started: TReqs demonstrator

- Assume that Alice and Bob want to create a function that translate latin numbers to roman numbers
- The start with a high-level requirement
  - (which they store in a markdown file for now, since those are nicely layed out and well integrated in their git based development environment (gitlab or github))

```
# Req-1: Convert arabic to roman

The function takes an integer i and returns a String that represents i as a roman number.
```

Basic workflow

- Alice wants to get started. Working in an agile way, she aims to use TestFirst

```
assertEquals("I", convert(1));
...
convert(int i) {return "I";}
...
# Req-1: Convert arabic to roman

The function `convert` takes an integer i and returns a String that represents i as a roman number.

## Req-1.1: Convert 1 to I

For i = 1, `convert` shall return `I`.
```

- Alice pushes these changes to her development branch and creates a merge request
- Bob reviews the change, sees that test, function, and requirement are updated and consistent
- The change is merged into the main branch
- Alice and bob continue with the number 2 to 10 in parallel

Create requirements

- Now, to support this scenario, we want to add some support.
  - Example, adding a link between the unit test and the Requirement 1
- If we want to offer tool support, we need to
  - define a unique ID (otherwise, Alice and Bob creating each a Requirement 1.1 at the same time will cause chaos)
  - make it easy for the tool to define where a requirement or test starts and ends
- With treqs create, we can conveniently create syntactically correct treqs elements, that are still rendered as markdown elements

```
<treqs-element id="b1cd3dba866a11ebbdfcc4b301c00591" type="requirement">
## Req-1.1: Convert 1 to I

For i = 1, `convert` shall return `I`.

<treqs-link type="parent" target="72f032c0866d11ebac03c4b301c00591" />
</treqs-element>
```

List requirements

- Now we have requirements in markdown files in a very useful form
  - They are easy to read for humans, either in the markdown file or in the rendered preview
  - They are easy to read for computers
- We can list requirements based on several criteria, which helps to keep an overview in large projects
  - Provide output as markdown table or (graphical) plantuml model

```
knauss$ treqs list
| UID | Type | Label | File | Line |
| :--- | :--- | :--- | :--- | :--- |
| 72f032c0866d11ebac03c4b301c00591 | requirement | # Req-1: Convert arabic to roman | requirements/system-requirements.md | 2 |
| b1cd3dba866a11ebbdfcc4b301c00591 | requirement | ## Req-1.1: Convert 1 to I | requirements/system-requirements.md | 9 |
| 54ed41e2867111eb91e5c4b301c00591 | unit-test |    * ## arabic2roman test | src/se/treqs/example/numconv/NumConverterTest.java | 2 |
```

Trace requirements

- Through the UID, we can also create and explore tracelinks from tests to requirements
- In javadoc or other comments just in front of automated tests, we can add tracelinks such as

```
/**
 * <treqs-element id="54ed41e2867111eb91e5c4b301c00591" type="unit-test">
 * ## arabic2roman test
 *
 * This test checks requirement Req-1
 * <treqs-link type="required-by" target="72f032c0866d11ebac03c4b301c00591" />
 * </treqs-element>
 */
@Test
public void arabic2roman() {
    assertEquals("I", conv.convert(1));
    assertEquals("II", conv.convert(2));
    assertEquals("III", conv.convert(3));
}
```

Check requirements

- Several checks are possible. Most important: are critical tracelinks set

```
knauss$ treqs check
| Error location | Error | File | Line |
| :--- | :--- | :--- | :--- |
| Element b1cd3dba866a11ebbdfcc4b301c00591 | Unrecognised link type parent within element of type requirement. | requirements/system-requireme
| Element 54ed41e2867111eb91e5c4b301c00591 | Element has an unrecognized type: unit-test | src/se/treqs/example/numconv/NumConverterTest.java
treqs check exited with failed checks.
```

Requirements as part of a commit

- By working in an agile, test-driven way, we now have commits to bundle changes of software, tests, and requirements.

```
knauss$ git status
On branch dev
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   requirements/system-requirements.md
    modified:   src/se/treqs/example/numconv/NumConverter.java
    modified:   src/se/treqs/example/numconv/NumConverterTest.java
```

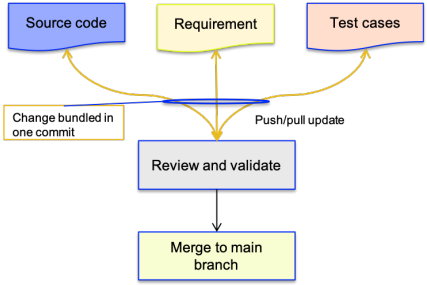

Figure 6: TReqs workflow

Review requirements

rashida @rashida started a thread on commit b5a536a2 1 month ago
Last updated by **rashida** 1 month ago

Toggle thread

README.md

```
4  4
5  5   ## Mission: Parser of roman numbers
6  6
7     - Key idea: Write a (java) class with a method int convert(String) that takes a roman number such as
        "III" and gives the arabic number (here: "3").
```

rashida @rashida · 1 month ago

Maintainer

Should we then edit this statement to take arabic and parse Roman numeral? It then becomes the same as the first requirement.

eric.knauss @eric.knauss · 1 month ago

Can you please check again? I think I have un

requirements/system-requirem

```
5  5
6  6   ### [requirements id
7  7   The system shall pro
       times `I` (e.g. 1 --
   8  +
   9  + ### [requirements id
  10  + The system shall pro
```

rashida @rashida started a thread on commit 4b55ff0a 1 month ago

Toggle thread

src/se/treqs/example/numconv/NumConverter.java

```
1  + package se.treqs.example.numconv;
2  +
3  + public class NumConverter {
4  +
5  +   public String convert(int i) {
6  +     return "III";
```

rashida @rashida · 1 month ago

Maintainer

Does it then return only III?

Break down requirements and keep them consistent with implementation.

Modeling support

- Write UML models as text
- Use github to version control and merge
- Use plantuml to convert to png
- Use treqs to create png references
- TReqs can be extended to allow clickable links in models and tracing of model elements
- Works great at scale (Liebel and Knauss 2023)

Req-1: Convert arabic to roman

The function `convert` takes an integer i and returns a String that represents i as a roman number.

```
@startuml convert-example-use-case
:User: - (convert arabic to roman)
@enduml
```

Req-1.1: Convert 1 to I

References

References I

Kasauli, Rashidah, Eric Knauss, Jennifer Horkoff, Grischa Liebel, and Francisco Gomes de Oliveira Neto. 2020. "Requirements Engineering Challenges and Practices in Large-Scale Agile System Development." *Systems and Software* 172. https://doi.org/10.1016/j.jss.2020.110851.

Knauss, Eric. 2019. "The Missing Requirements Perspective in Large-Scale Agile System Development." *IEEE Software* 36 (3): 9–13. https://doi.org/10.1109/MS.2019.2896875.

Knauss, Eric, Grischa Liebel, Jennifer Horkoff, Rebekka Wohlrab, Rashidah Kasauli, Filip Lange, and Pierre Gildert. 2018. "T-Reqs: Tool Support for Managing Requirements in Large-Scale Agile System Development." In *Proceedings of 26th IEEE International Requirements Engineering Conference (RE'18)*, 502–3. Banff, Canada. https://doi.org/10.1109/re.2018.00073.

Knauss, Eric, Patrizio Pelliccione, Rogardt Heldal, Magnus Ågren, Sofia Hellman, and Daniel Maniette. 2016. "Continuous Integration Beyond the Team: A Tooling Perspective on Challenges in the Automotive Industry." In *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '16)*. Ciudad Real, Spain: ACM. https://doi.org/10.1145/2961111.2962639.

Liebel, Grischa, and Eric Knauss. 2023. "Aspects of Modelling Requirements in Very-Large Agile Systems Engineering." *Systems and Software*. https://arxiv.org/pdf/2209.01993.

Liebel, Grischa, Matthias Tichy, Eric Knauss, Oscar Ljungkrantz, and Gerald Stieglbauer. 2016. "Organisation and Communication Problems in Automotive Requirements Engineering." *Requirements Engineering Journal (REEN)*. https://doi.org/10.1007/s00766-016-0261-7.

References II

Pelliccione, Patrizio, Eric Knauss, Rogardt Heldal, S. Magnus Ågren, Mallozzi Piergiuseppe, Anders Alminger, and Daniel Borgentun. 2017. "Automotive Architecture Framework: The Experience of Volvo Cars." *Journal of Systems Architecture*. https://doi.org/10.1016/j.sysarc.2017.02.005.