Protocol for an Empirical Study on Software Architecture Design in Global Software Development

(original publication May, 2018)

Version update: 26th June, 2019

Lero Technical Report No. TR_2019_01

Outi Sievi-Korte, Ita Richardson, Sarah Beecham
contact: sarah.beecham@lero.ie

in collaboration with Tampere University, Finland

This work was supported, in part, by Science Foundation Ireland grant 13/RC/2094 and co-funded under the European Regional Development Fund through the Southern & Eastern Regional Operational Programme to Lero - the Irish Software Research Centre (www.lero.ie), and was also supported was also supported by the Academy of Finland.

Protocol for an Empirical Study on Software Architecture Design in Global Software Development

(companion to manuscript submitted to JSS[1])

May, 2018

Outi Sievi-Korte, Ita Richardson, Sarah Beecham
contact: sarah.beecham@lero.ie

Preamble

Building on our previous work in GSD and Architecture, e.g. [2-5], we conducted a systematic literature review (SLR) on the challenges and recommended practices for software architecting in global software development [6]. Although our SLR identified a set of recommendations, there remained several gaps in our knowledge. We therefore conducted an empirical study to try to broaden our understanding of the practices architects employ in a distributed setting.

The aim of this study is to gain new insights by interviewing software architects to explore these two research questions:

RQ1: What challenges do practitioners face when designing software architecture in GSD projects?
RQ2: What practices do software architects use to accommodate the distributed nature of development work?

This protocol documents the process we followed to (1) set up the empirical study, (2) conduct our interviews (including the set of questions asked), and (3) analyze the interview data.

1. STUDY SET UP

Sampling: Our population of interest, which defines our selection criteria, are **software architecture designers** working in a **globally distributed environment**. Of importance is that the architects design software for teams developing software across different physical and geographic locations. Our sample was opportunistic and purposive, as we reached out to architects known to us (the authors), in the first instance, who met our selection criteria. To widen our sample, we also asked our industry contacts (not directly involved in design) to connect us with colleagues who worked with software architecture design in a global software engineering (GSE) context. Our sample consisted of practitioners involved in a range of tasks to include involvement in making design decisions, prioritizing requirements and development work accordingly, and contributing to architectural artefacts, such as documentation.

Pilot Study: We ran a pilot interview with a colleague (not involved in the study) who had several years' experience in software architecture research and teaching. Following this, interview questions were modified and clarifications which might be required during the interview, were noted.

Permissions and ethics: All interviewees participated in the study voluntarily. They were asked for permission to allow the researchers to both record the interview and make notes. All interviewees were guaranteed anonymity and were given the option of proofing the manuscript before sending for publication in order to review the text and check anonymity. As a courtesy, we followed up with each participant once we finalized our draft, to double check they were happy with the text before publication. We received no requests for changes.

Recording and data storage: We conducted the interviews either face to face or via Skype. We recorded face-to-face interviews with a digital voice recorder, and recorded Skype interviews using Skype's own recording option. Additionally, the interviewer wrote hand written notes. All recordings were transcribed (and translated from Finnish to English if necessary) by a professional transcriber (and translator), hired outside of the university. The recordings were subsequently destroyed, and all interview transcripts were kept securely, on a secure server at Tampere University by the first author. The results were anonymized prior to sharing with the wider group of authors.
The semi-structured nature of the questions meant that the interview length varied from between 1 hour to over 2 hours.

2. CONDUCT THE STUDY

2.1 Interview themes

The interview questions were based on the themes found in our SLR[6] where we identified challenges and practices in the literature relating to architecting in GSD. The themes are:

2.1.1 Practices and design agreements (including well-defined interfaces)
2.1.2 Aligning software architecture, work structure and resources
2.1.3 Communication and lack of awareness between sites
2.1.4 Identifying dependencies
2.1.5 Creation of and sharing architectural artefacts
2.1.6 Quality assurance
2.1.7 Design compliance

2.2 The interview Script.

The following is the script used by the interviewer as a starting point for the semi-structured interview**.**

a. Context and purpose of study:

Please state whether you allow the recording of this interview and that notes are also created.

Tampere University of Technology has been working together with Lero – the Irish Software Research Centre on a project researching global software development. While a lot of research has been conducted in the general project management area of GSD, we have found very little guidance on solving architectural design issues in a distributed development context. We now look to you to help us explore this area.

In this interview, we will ask about your company's current architecting practices in the context of distributed development, and to discuss our current findings – we will ask you whether some suggested practices seem reasonable, how are they applied (if applied), and what is still missing.

All answers will be treated anonymously, and you will receive a copy of our manuscript draft to ensure that you are satisfied with how the material is presented and that your company is unidentifiable. All data will be stored securely at Tampere University of Technology.

b. Context of study and definitions of terms used:

To ensure you understand the concept of this research we explain:

We define ***GSD*** as software development that takes place in several locations that are in different countries. Meaning that the company has several sites, where teams or individuals from different sites are working on the same project. It might be that also teams themselves are distributed, and there are individuals belonging to the same development team but work in different locations.

Software architecture can be defined in many ways, what we are specifically looking at here is *architectural design* which is the process and practices that are required the (component) structure, design solutions, interface specifications and technological choices that in turn define the software at such a level that code can be written. Please bear this in mind when answering".

c. Interview questions. We divided questions into three broad categories of "Demographics", "Challenges" and "Practices". We based our questions on the themes found in our literature review, which acted as a driver for further questions and answers.

Demographics:

- Name of interviewee
- Name of company
- Company's field of work
- Company's size
- Company's size of IT/Software Development function?
- How many years have you worked in software engineering/IT?
- How many years have you been involved with globally distributed software development?
- How many sites have there been involved in your project(s)?
- How many different countries have the sites been located in?
    - What countries?

GSD context and processes used:

- What do you think is the most challenging aspect in GSD in general?
    - Are there times when due to time differences between sites that you cannot communicate with some of your project team (temporal distance)? (temporal dist)
        - If does this impact on your development process?
    - Does working in geographically distributed teams impact on your development process? (e.g. involve you in lots of travel)? (Geographic dist)
    - Do all members of your team share a common first language?
        - If not does difference in first language cause any problems in your team's communication? (Linguistic dist)
    - Do all members of your team share a common culture (western/eastern, high/low context, religious etc) (Cultural dist)
        - If not, how does working in a multi cultural team impact on your software development? (NB. This can be positive).
- What are the reasons for applying GSD?
- Are you applying agile methodologies?
    - If no, what development process do you apply?

GSD and architecture:

- What are the most important design decisions to consider in software development?
    - *Can be very wide and challenging to answer*
- What are the most important design decisions to include in distributed development – that might be different to collocated development?
    - Describe your (most important) current architecture design practices in GSD projects – for example, do you have practices in place that aim for structuring the architecture to consider the distributed teams and enable independent work for each site?
    - Are such practices known to everyone involved in the design?
        - How is awareness controlled?
        - How are design principles and practices recorded and their use enforced?
            - *If needed*: Are design principles recorded?

        - *If needed*: Is the use of design principles enforced?
        - *Notice level! Keep on architectural level, general company-wide design practices!*
- Who is in charge of the design process and decisions?
    - A team or one architect, where situated?
    - If distributed architects, how is the design process arranged?
- What are the architects' backgrounds and main drivers (technical, business)?
    - Are architects' background known?
    - If yes, is background considered in the design process?
- What principles are the design process and decisions based on? (Experience, literature, resources..)
- Is distribution of development/implementation work taken into account in the design process and decisions?
    - How?
- What are the most important responsibilities of an architect?
- How much effort is put into the design?
- If Agile is used, how does this work with distributed architecture design?

The remaining questions are based on THEMES derived from the SLR[6]. Where there are challenges and practices associated with the same theme, we keep them together (although, we do ask about challenges and practices separately under each theme). Some themes only have challenges currently.

Challenges: Insufficient/ambiguous practices
Practices: Well-defined interfaces, Common design agreements

- What is required from an interface to make it "well-defined"? What are the factors?
    - Is the term "well-defined" interface used in design practices?
    - Is everyone in agreement with what it takes to make an interface "well-defined"?
- How much instability is caused by the distributed nature of development?
    - How is stability enforced?
- Who defines prioritization?
    - Is prioritization of tasks considered?
    - Is there a dedicated person responsible for prioritization?
- *The theme was "ambiguous practices", usually the problem can be in lack of awareness.* What are the current knowledge management practices?
    - *If needed*: Are knowledge management practices applied?
- How are assumptions handled?
    - *If needed*: Are conflicting assumptions recognized as a problem?

Challenges: Information sharing between/lack of awareness in distributed team
Practices: Multiple views of architecture for different stakeholders, Architecture communicated throughout sites

- How are potential conflicts (e.g., in schedule) identified and handled? Schedule, release synchronization?
- Are there practices for identifying all relevant stakeholders?
- Are appropriate architecture views composed for each stakeholder's purposes?
  - Who decides on the view?
- Are all teams and their relevant members aware of the architectural design and decisions?
  - Who is responsible for awareness?
- Is understandability or architecture an issue?
  - How handled?

Challenges: Work structure should meet architecture, Cross-site modifications/work items, Matching available resources/organization to design
Practices: Architecture structure compliant with work structure, Work items chopped to manageable pieces, Leveraging local knowledge (technical skills, domain knowledge of localization proficiency)

- What organizational aspects are considered during design – are all necessary aspects covered?
  - *If needed*: Are organizational aspects considered during design?
- Does the software structure guide the organization or vice versa? How, in practice?
  - Does software structure guide the organization structure?
  - Does organization structure guide the software structure?
  - Are design decisions aligned to organization structure?
    - What kind of practices to do that?
- On what scale is work structure considered – per site, per team or per person?
  - What kind of practices are there to scale down work items – do they work?
    - *If needed:* Are there specific practices to scale down work items?
- Are the available resources considered during design?
  - If yes, do they act as a driving force for the design?
    - How? (Localization, technical skills, matching code to resources)
  - Is the distribution of resources considered? How?
- Are there working practices to handle misaligned interests?
  - What are they?
- Does the design and mapping to organization depend on the product under development?

Challenge: Need to identify correct and all dependencies
Practice: Analysis methods for coupling

- How is it verified that all dependencies are known? **State synchronization?**
    - *If needed:* Are there practices for identifying all dependencies in the software?
- How are interdependencies between decisions considered?
    - *If needed:* Are there practices for identifying interdependencies between design decisions?
    - *NB! If such practices don't exist, may be difficult to answer (but reveals a gap in the companies practices!) .*
- *Low coupling is considered essential for having independent components*. Are there mechanisms for enhancing low coupling? If so, what are they?

Challenge: Creation and sharing architectural artifacts
Practice: Single repository of architecture artifacts

- Where are the artifacts stored?
    - Is there a dedicated place for storing architectural artifacts?
        - If yes, is it known to everyone where that is?
        - If yes, does everyone have access to it?
- Who is responsible for the creation of architectural artifacts?
    - Is there a single person or team responsible for creating architectural artifacts?
    - If yes, is it known to everyone who that is?
- Are the artifacts understood by (representatives of) all teams? How is this ensured?
- How are they communicated throughout teams

Challenge: Quality assurance

- How is quality assurance handled in your company?
    - *If needed*: Are there quality assurance practices in place?
- Are design decisions delegated to teams?
    - If yes, are there quality management practices in place?
- Is there a dedicated person responsible for quality concerns? What is his main role?

Challenge: Design compliance (through changes)

- How is design compliance handled in your company? Where are the main concerns?
    - Particularly: architecture compliance with requirements
- Are possible changes in communication structure considered in the design structure? How?

Further questions

What are the facts that need to be considered in the design? [IF NOT APPARENT BASED ON INTERVIEW]

Did some other challenges or practices come to your mind based on the presented themes or questions that you would still like to discuss?

DATA ANALYSIS

Coding and creation of initial framework

When coding the transcripts, the codes ``practice" and ``challenge" were pre-determined, but new codes were generated based on the analysis of the transcripts and notes. We used a form of thematic analysis to derive a set of themes. Our lowest level theme we call 'a code'. Three researchers were involved. Researcher 1 derived the initial set of codes, conducted all the interviews, and made notes. Researchers 2 and 3 validated and questioned the coding at each stage of the process. Researcher 2 suggested aggregations, new themes, new labels for the themes. Where Researcher 1 and 2 disagreed with the coding, Researcher 3 acted as a moderator.
The thirteen experts interviewed created 187 pages of transcripts, and 452 quotes coded as challenges or practices. (Note there were 9 actual interviews since 2 of the 9 interviews involved three experts).

Figure 1 describes the analysis (all examples are related to the example quote given in Step 1):

Step 1: Researcher 1 associated each quote with one or more codes, resulting in 35 different codes. A quote varied in length, constituting a phrase, a sentence, or several sentences around one topic.

Example quote: "*We have virtual teams in projects. Even though we have these line teams, which we belong to and which have a*

Example memo: VIRTUAL TEAMS. Teams formed based on architecture. Level of team spirit? Again, feeling of belonging? Motivation? Commitment? On one hand, layers and certain functionality clearly divided between geographic sites, but then component development done by virtual teams? Modularity obviously key. Rel. Resources, Working culture, Arch. choices, Org. work allocation --> Further, supervisors allocate tasks to geographic teams, but teams themselves reform as virtual teams to fit the architecture. Double allocation?"

Step 2: Researcher 1 created a memo item for each quote.

Step3: Researcher 1 grouped memo items together, where there appeared to be an association.

Example grouping of memo items. Memo "Teams":

Item 1 "Team structure made based on architecture -> architecture guides organization (which has guided architecture to begin with..."

Item 2: "Teams are given a lot of power to decide -> though problems with communication, managing quality, interfaces, teams are TRUSTED Rel. Working culture, Interfaces, Comm. Quality"

Step 4: Researcher 1 discarded all quotes that did not address our research questions, i.e., were not labeled "practice" or "challenge".

Step 5: Researcher 1 reduced each of the retained quotes (of varying lengths) into one short phrase. The aim of this phrase was to capture and synthesize the underlying meaning. This step created the first abstraction level moving from raw quotes towards a framework theme.

Reducing quotes down to a short phrase – e.g., Assemble teams based on architecture design

Step 6: Researcher 1 merged newly formed practices and challenges (represented as a short phrase) to create higher-level themes. These higher-level themes were given a new label.

Example of merged practices, and the label created to represent the high level of abstraction.

Acquire skills based on architecture and align teams to architecture

- Select resources to match the architecture
- Acquire relevant skills and backgrounds for remote sites
- Collect resources to fit architectural decisions
- Assemble teams based on architecture design

Step 7: Researcher 1 repeated Step 6 (above) four times, until all the names and levels of abstraction were clear and discrete.

An example of the iterative process is shown in Figure 1. The codes created, the memo items that were associated with it, how the quote was worded as a practice and the steps leading to it ultimately belonging under the practice "Align organization and architecture".

By applying this process to each individual quote, we produced a multi-tier challenge-practice framework (Level 1: practices/challenges, Level 2: concerns, Level 3 and below: (aggregated themes), containing 26 individual concerns relating to practices and 23 individual concerns relating to challenges. Researcher 1 produced the initial framework.

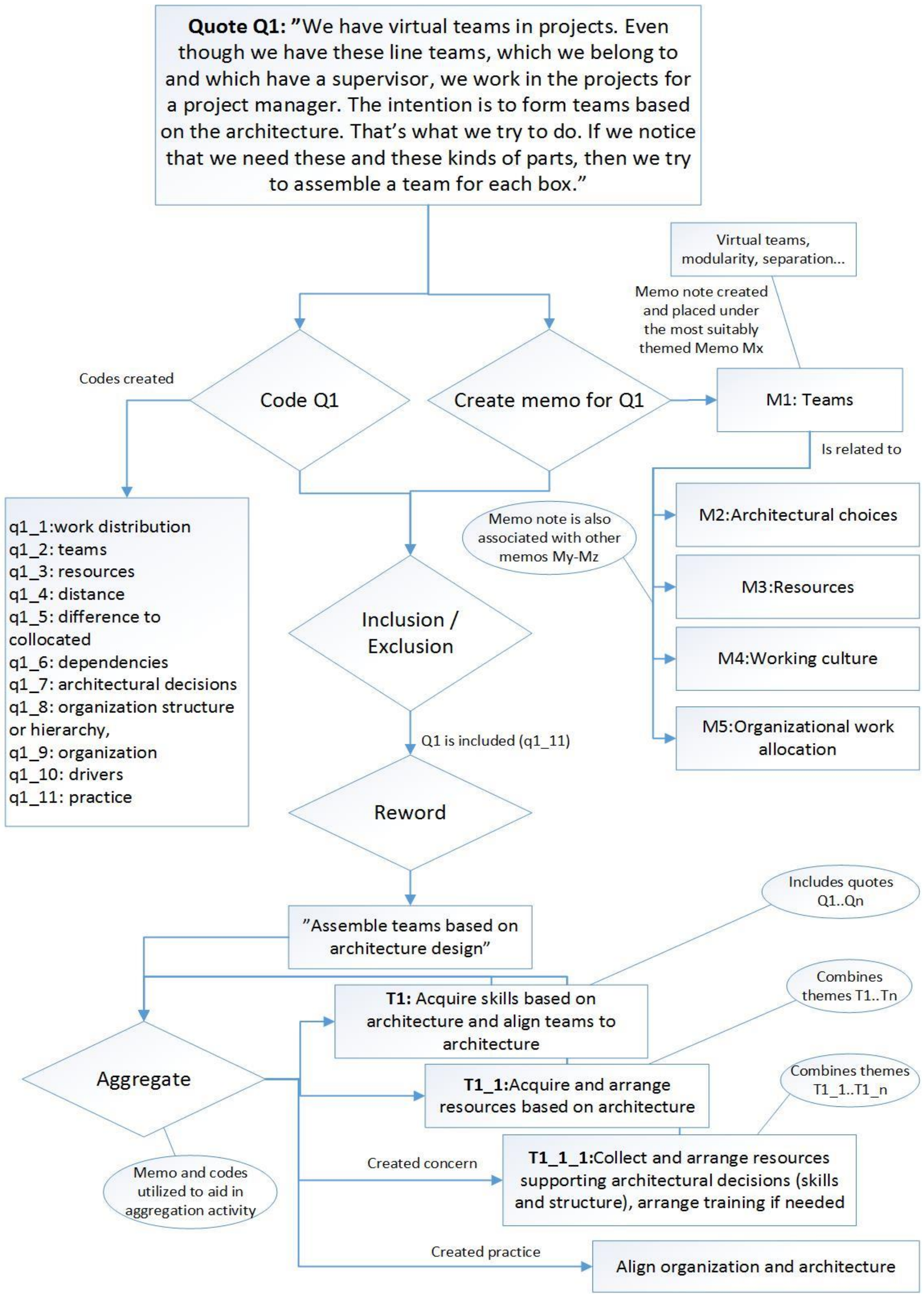
Quote Q1: "We have virtual teams in projects. Even though we have these line teams, which we belong to and which have a supervisor, we work in the projects for a project manager. The intention is to form teams based on the architecture. That's what we try to do. If we notice that we need these and these kinds of parts, then we try to assemble a team for each box."
Virtual teams, modularity, separation…
Memo note created and placed under the most suitably themed Memo Mx
Codes created
Code Q1
Create memo for Q1
M1: Teams
Is related to
q1_1:work distribution
q1_2: teams
q1_3: resources
q1_4: distance
q1_5: difference to collocated
q1_6: dependencies
q1_7: architectural decisions
q1_8: organization structure or hierarchy,
q1_9: organization
q1_10: drivers
q1_11: practice
Memo note is also associated with other memos My-Mz
M2:Architectural choices
M3:Resources
M4:Working culture
M5:Organizational work allocation
Inclusion / Exclusion
Q1 is included (q1_11)
Reword
"Assemble teams based on architecture design"
Includes quotes Q1..Qn
Combines themes T1..Tn
Combines themes T1_1..T1_n
T1: Acquire skills based on architecture and align teams to architecture
Aggregate
T1_1:Acquire and arrange resources based on architecture
Created concern
T1_1_1:Collect and arrange resources supporting architectural decisions (skills and structure), arrange training if needed
Memo and codes utilized to aid in aggregation activity
Created practice
Align organization and architecture

Figure 1 Example of coding process

Inter-rater validation

The framework created by the researcher 1 was validated by the other two authors. The validation process was conducted separately for both challenges and practices:

1. Researcher 1 circulated three files to Researchers 2 and 3:
   (a) A "quote" excel file with 20 individual quotes each, so a total of 40 quotes were independently checked. (Researcher 1 and 2 were sent different quotes (10 for practices, and 10 for challenges). Since there were a total 264 quotes on practices and 188 on challenges, this represented ~10% of the total for challenges and 7,5% for practices. The quotes represented a stratified sample to ensure all themes that appeared as a result of the synthesis (ending up in the framework), were validated. The inter-raters all conducted their quote-to- code mappings independently, i.e., without seeing how the other raters mapped their quotes.
   (b) A set of "challenges and practices" pdf file (highest level of framework only) – initially Researcher 1 had identified 20 practices, and 20 challenges.
   (c) A "framework" excel file with the entire framework (Level 1: challenges/practices # 20, Level 2: concerns #26, Level 3: themes #46). (Level 1, is repeated in the pdf file- noted in (b) above).
2. Researchers 2 and 3 marked mapped quotes (given in the "quote" file) to codes (listed in the "framework" file), i.e., placing the quote against the practices, concern and theme given in the framework. They added comments as to why they chose a particular coding.
3. The first author received the marked and mapped quote files and compared the coding of quotes to her original coding. She marked on the quote files whether there was agreement or disagreement on the coding.
4. In case of disagreement, the Researcher 1 checked whether Researcher 2 or 3 had such reasoning behind the differing coding that she would be inclined to change her own coding to correspond. If she disagreed with the reasoning, she would mark her own original coding and give a note on her reasoning in the excel sheet, and send it to the validating author. If the validating author would then concur to the reasoning of the first author, an agreement was reached.
5. When two researchers could not agree, the third researcher acted as a moderator, and separately coded the quote. Based on the three codings, the authors discussed until agreement was reached.
6. The framework was re-drawn to accommodate the changes in coding resulting from Steps 4 and 5.
7. The inter-rater validation process was repeated (steps 1-6) with new sets of quotes and a new version of the framework was created when no major disagreements occurred in step 4.

The validation process resulted in two major revisions (step 6) to the framework. In these revisions, themes were re-worded, quotes were re-allocated under different existing themes, new themes were created, themes were merged, and some themes were deleted.

We illustrate how the framework changed as a result of the validation process via examples given in Figures 2 and 3.

Example 1: During initial creation of the framework, quote Q2 is first rephrased into a challenge Q2-L0, the lowest level in our multi-tiered framework, as shown in Figure 2. Other quotes are combined with Q2 and aggregated under a common theme, L1_T1, which is aggregated with other themes under a higher-level theme (not shown in the Figure). At the same time, L1_T2 and L1_T3 are aggregated under a higher level theme (L2_T2), as shown.

During validation process, the authors disagreed on the coding of quote Q2. As a result, the quote was first re-allocated under a different theme (L1_T2 instead of L1_T1) and then also rephrased to better portray the message of the original interview quote. Both L1_T1 and L1_T2 were also rephrased. All parent themes (L2_T2 and L3_T3) were further rephrased, and finally a new abstraction level was created with the theme L4_T2. Finally, other L0-level themes previously under L1_T1 were re-allocated under different themes. While L1_T1 initially combined six L0-level themes/quotes, after the validation process, L1_T1 only combined two sub-themes.

Example 2: In example 2, given in Figure 3, we have illustrated how initially a level 3 theme L3_T8 “Take advantage of agile methods” was created, alongside with L3_T1 “Agree on project management and collaboration practices tailored to distributed development”, L3_T2 “Engage developers across sites in architectural work” and five other level 3 themes. As a result of the first validation round, L3_T2 was sent down on the hierarchy level and placed under L3_T1 to gain consistant granularity and level of abstraction throughout the framework. When the framework was re-validated, L3_T8 was similarly also sent down one level and placed under L3_T1.

References

1. Sievi-Korte, O., I. Richardson, and S. Beecham, *Software Architecture Design in Global Software Development : an empirical study.* Journal of Systems and Software, 2019. **In revision**.
2. Ali, N., S. Beecham, and I. Mistrik. *Architectural Knowledge Management in Global Software Development: A Review*. in *KNOWing Workshop co-located at International Conference on Global Software Engineering*. 2010. Princeton, New Jersey, U.S.A.
3. Beecham, S., et al. *Crafting a global teaming model for architectural knowledge*. in *Global Software Engineering (ICGSE), 2010 5th IEEE International Conference on*. 2010. IEEE.
4. Noll, J., S. Beecham, and I. Richardson, *Global software development and collaboration: barriers and solutions.* ACM Inroads, 2010. **1**(3): p. 66-78.
5. Noll, J., et al. *A Global Teaming Model for Global Software Development Governance: A Case Study*. in *Global Software Engineering (ICGSE), 2016 IEEE 11th International Conference on*. 2016. IEEE.
6. Sievi-Korte, O., S. Beecham, and I. Richardson, *Challenges and recommended practices for software architecting in global software development.* Information and Software Technology, 2019. **106**: p. 234-253.

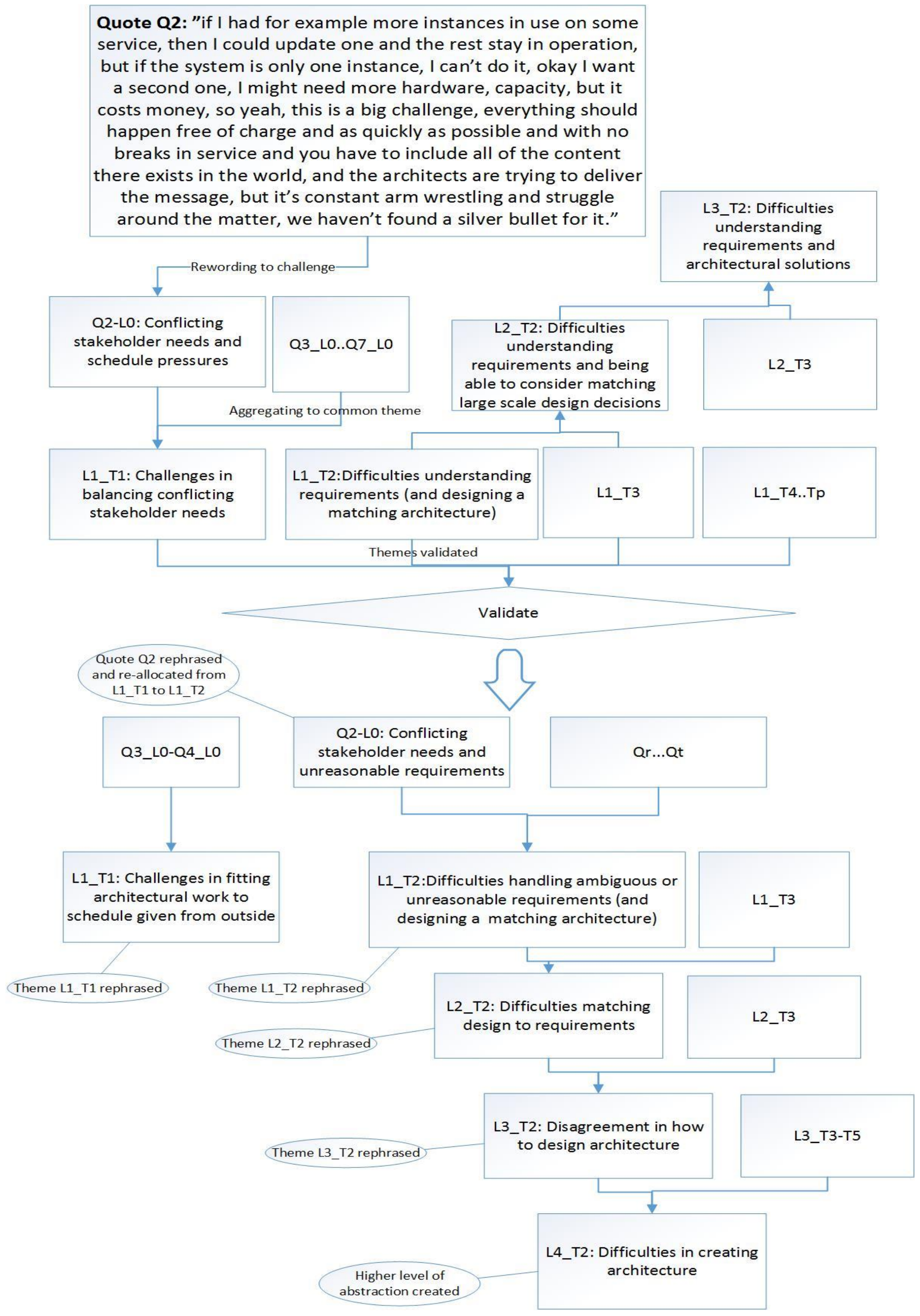
Quote Q2: "if I had for example more instances in use on some service, then I could update one and the rest stay in operation, but if the system is only one instance, I can't do it, okay I want a second one, I might need more hardware, capacity, but it costs money, so yeah, this is a big challenge, everything should happen free of charge and as quickly as possible and with no breaks in service and you have to include all of the content there exists in the world, and the architects are trying to deliver the message, but it's constant arm wrestling and struggle around the matter, we haven't found a silver bullet for it."
Rewording to challenge
Q2-L0: Conflicting stakeholder needs and schedule pressures
Q3_L0..Q7_L0
Aggregating to common theme
L3_T2: Difficulties understanding requirements and architectural solutions
L2_T2: Difficulties understanding requirements and being able to consider matching large scale design decisions
L2_T3
L1_T1: Challenges in balancing conflicting stakeholder needs
L1_T2:Difficulties understanding requirements (and designing a matching architecture)
L1_T3
L1_T4..Tp
Themes validated
Validate
Quote Q2 rephrased and re-allocated from L1_T1 to L1_T2
Q3_L0-Q4_L0
Q2-L0: Conflicting stakeholder needs and unreasonable requirements
Qr...Qt
L1_T1: Challenges in fitting architectural work to schedule given from outside
L1_T2:Difficulties handling ambiguous or unreasonable requirements (and designing a matching architecture)
L1_T3
Theme L1_T1 rephrased
Theme L1_T2 rephrased
L2_T2: Difficulties matching design to requirements
L2_T3
Theme L2_T2 rephrased
L3_T2: Disagreement in how to design architecture
L3_T3-T5
Theme L3_T2 rephrased
L4_T2: Difficulties in creating architecture
Higher level of abstraction created

Figure 2 Example 1 on validation process and framework changes

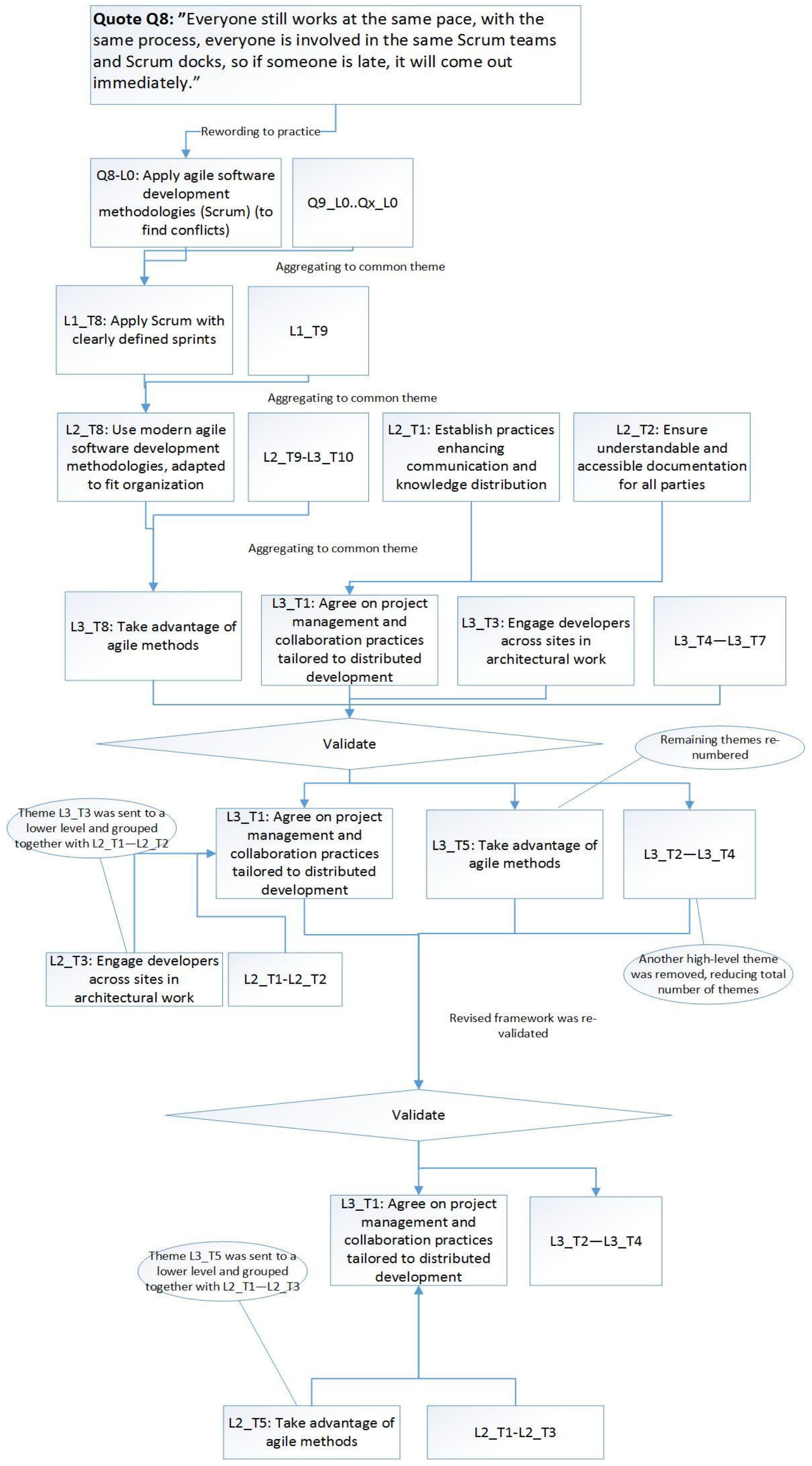
Quote Q8: "Everyone still works at the same pace, with the same process, everyone is involved in the same Scrum teams and Scrum docks, so if someone is late, it will come out immediately."
Rewording to practice
Q8-L0: Apply agile software development methodologies (Scrum) (to find conflicts)
Q9_L0..Qx_L0
Aggregating to common theme
L1_T8: Apply Scrum with clearly defined sprints
L1_T9
Aggregating to common theme
L2_T8: Use modern agile software development methodologies, adapted to fit organization
L2_T9-L3_T10
L2_T1: Establish practices enhancing communication and knowledge distribution
L2_T2: Ensure understandable and accessible documentation for all parties
Aggregating to common theme
L3_T8: Take advantage of agile methods
L3_T1: Agree on project management and collaboration practices tailored to distributed development
L3_T3: Engage developers across sites in architectural work
L3_T4—L3_T7
Validate
Remaining themes re-numbered
Theme L3_T3 was sent to a lower level and grouped together with L2_T1—L2_T2
L3_T1: Agree on project management and collaboration practices tailored to distributed development
L3_T5: Take advantage of agile methods
L3_T2—L3_T4
L2_T3: Engage developers across sites in architectural work
L2_T1-L2_T2
Another high-level theme was removed, reducing total number of themes
Revised framework was re-validated
Validate
L3_T1: Agree on project management and collaboration practices tailored to distributed development
L3_T2—L3_T4
Theme L3_T5 was sent to a lower level and grouped together with L2_T1—L2_T3
L2_T5: Take advantage of agile methods
L2_T1-L2_T3

Figure 3 Example 2 on validation process and framework changes