Classic Development Processes with Focus RE4DIST

Philipp-Henning Salla

2022-02-14 (Winter term 2021/2022)

Contents

1 Introduction

For a software project, a well structured beginning of the development process is strongly advised, if not mandatory, to lay a solid foundation for keeping the following development structured as well. Especially when developing as a team and/or the software to be developed, is considered as a distributed system, the topic of a structured beginning is even more important.

With that in mind, there are a few questions emerging probably immediately. Firstly, what is a good model to achieve a structured beginning of the development process, when regarding a distributed system? Secondly, considering that the speaking is of the beginning of the process, in what other models can the model, creating a solid foundation, be integrated in? And lastly, are the advantages of using a model for providing a structured beginning and having a structured beginning at all, that beneficial that considering a structured beginning as strongly advised or even mandatory is justified?

To answer those questions, the present work is going to describe a model and its steps to achieve a structured beginning of the software development process. This model is called "*Requirements Engineering for Distributed Systems (RE4DIST)*", developed by M. Heisel and R. Wirtz of the University of Duisburg-Essen in 2019 [1]. Following the description of RE4DIST, it will be set in comparison with other more general models for development processes and determining where RE4DIST could be included in said more general models. The last part of the present work includes a case study, in which the steps of RE4DIST are applied to the task of student project, I participated in, where originally no model-based development process was used and a structured beginning was not present at all.

To conclude, there is a short summary of the steps defined by RE4DIST as well as its position in a more general model. Furthermore the experience of the case study is used to underline the importance of a structured beginning of a software project, regarding a distributed system and a development as a team. The then following section will include the sources that were used to create the present work.

3 Requirements Engineering for Distributed Systems

Requirements Engineering for Distributed Systems is a model to distinguish the requirements for a software project in a distributed system and thus laying a structured foundation for the software's implementation. The model consists of several diagrams, mainly Problem Diagrams [2] and Context Diagrams, as well as variants of the Context Diagram, the Global Context Diagram and the Sub Context Diagram. All these Diagrams are used over the five steps of RE4DIST producing a set of diagrams modeling the requirements and its dependencies. In order to explain re4dist, more general and generic examples are used in this chapter first. Later, when applying re4dist to the practical project, more specific examples are shown.

The Global Context Diagram describes the general connections between components of the overall project. For example regarding the case study of the student project, which will be explained in more detail in chapter 5, the Global Context Diagram would contain the railway layout, Z21 controller, digital twin and the web application. The sub Context Diagram is more focused and describes the context for a certain subsystem, for example the digital twin. The Problem Diagrams then describes a requirement and its interaction with a certain subsystem or a certain connection between subsystems.

Each of the above diagrams consists of different domains [1]. Machine Domains, representing subsystems to be developed; Problem Domains referring to the interaction between subsystems; Remote Machines, describing machine domains that are part of a distributed system; and lastly a Distributed System which contains at least two machine domains. Connections between domains are described by interfaces whereas remote machine domains need to be connected by remote interfaces.

To ensure consistency in between each step and the correct usage of the method, the developers built a tool based on the Eclipse Modeling Framework (EMF) [1]. In this tool the user can enter and edit data required for the respective step. The consistency and input of correct data is checked by the tool mostly autonomously, using several validation conditions that have to be fulfilled to proceed to the next step. Since the tool is only a way to use the method of RE4DIST, this work will not go further into detail regarding the tool.

3.1 Step 1: Define global context and subsystems

The goal of the first step is, to get an understanding of the context the system to be developed is planned to operate in. To achieve this goal an informal description of the scenario is taken as input and will result in a context diagram which displays all problem domains. The context diagram is modeled using Unified Modeling Language (UML) while paying attention to additional rules.

Since existing systems do not need to be developed, these systems are displayed by causal domains. Taking the student project as an example, would be the Z21 [3] controller which provides track voltage, sends instructions to each train and connects to other modules. The connections between the distributed system and environmental domains is modeled by interfaces. Regarding subsystems, there must be at least two to consider the system as distributed. These subsystems are then represented as machine domains with an aggregation as connection to the distributed system.

The validation conditions provided by the developers require exactly one distributed

the occurance of errors right at the beginning of the project. The method of RE4DIST with its validation conditions and especially the usage of the built tool [1], provides a structured analysis and separation of problems for a development in different teams and is therefore well designed to be used in larger software projects developed using the v-model.

4.3 Spiral Model

A different approach is shown in the sprial model or meta model [7]. This model is not about completing an entire phase of development and then moving to the next, rather it cycles through different stages of planning, designing, implementing and testing multiple times [9]. The most significant advantage is that occouring bugs can be fixed until the next cycle through the phases is complete and thus another part of the software finishes. In addition to that changes of requirements during the ongoing development can more easily be implemented due to its cyclic design. This cyclic precedure will become problematic when attempting to use RE4DIST. Since RE4DIST in itself is built more like a waterfall model, one would have to run through at least one entire step of RE4DIST again when trying to implement a change that occured during one cycle. The reason is that the change of one requirement could impact other requirements and subsystems which then would need to be adapted, too.
With this in mind, RE4DIST's one-time run nature does not provide a solid support for a development along the spiral model by applying RE4DIST at the very beginning of the project, finish it and then move on to the rest of development using the results RE4DIST provides. Attempting to apply RE4DIST in every planning phase of the spiral model [9], not to analyze the requirements of the entire project, instead for the requirement that is planned to be developed next only. This could potentially cause errors regarding the dependencies between subsystems, especially when considering a distributed system, for which RE4DIST was developed. Due to the repeating testing phases, these errors would probably be detected and corrected, but still cause unnecessary workload and thus render RE4DIST for a usage in a spiral model as not optimal.

4.4 Findings On Integratability

Because of RE4DIST's detailed and in-depth analysis, its strength lies in the one-time run before the design and implementation of the software. Therefore fully planned and sequential models in a more general context, provide a good plattform to use RE4DIST on. As seen in the basic waterfall model and v-model, the clear delimitation of phases allows RE4DIST when applied, to take over an entire step and build the foundation for the following design and implementation. When moving away from models with clearly defined phases to an iterative approach, the integrability starts to decrease. Depending on the type of iterative approach, RE4DIST can still be used in some cases. This applies to iterative models whose beginning is nevertheless clearly planned, structured and, above all, run through once only.
Upon reaching fully iterative or cyclic models like the spiral model, RE4DIST becomes very hard to integrate in. Due to possibly frequently adapting requirements and the processing of only a few or one requirement at a time, changes in dependencies between each requirement or subsystem become hard to track. The change of a requirement that potentially impacts others, requires RE4DIST to be applied all over again, if not for the entire project at least for smaller chunks, which could then lead to an increase of workload and error potential.

5.5 What If

After the creation of all diagrams proposed in RE4DIST one can imagine a *what if* scenario. In other words, how would the start of the project have gone if RE4DIST had been used from the beginning. The first thing that becomes obvious is time. Creating the diagrams would have taken a lot of time at the beginning, especially if none can be combined together or omitted altogether. This was only possible because of the simplification for the purpose of the example. Furthermore, each team member, or at least those who would have created the diagrams, would have had to be well familiar with the method of RE4DIST in order to minimize the loss of time. On the other hand, the creation of the diagrams could have clarified the priorities, so that, for example, it would have not been investigated how the braking and acceleration can be modeled right at the beginning of the project, while there is not yet the possibility to communicate with the train at all. Furthermore, the diagrams could have been seen as individual tasks to be completed which would have prevented some team members from becoming idle. However, it is important to keep in mind that RE4DIST is not optimal for projects with changing requirements, as seen in chapter 4. In the beginning the project could have been based on the waterfall model in which RE4DIST could have been integrated well, but the implementation phase would have resulted in changes of requirements, because often an implementation had to be changed fundamentally for a problem. An example for this is the selection of a route for a train and its control. At first these were only connected via a web socket, but then other modules had to be interposed to enable other functions. Nevertheless, the creation of diagrams using the method of RE4DIST would have provided a basic structure to start with, which could then have been built upon, providing a clearer structure throughout the project.

6 Summary

In conclusion, the questions posed at the beginning can now be answered. RE4DIST is ceratinly a good model for bringing structure to the start of a software project, especially regarding distributed systems. Although it has some limitations in terms of applicability and integrability bacause it is not optimal used in projects with constant changes in requirements as well as more general models that are based on an iterative idea. The comparison with some well known, more general, models has pointed out the differences. RE4DIST is best used in straight and less flexible models such as the waterfall model or the v-model. Since RE4DIST only effects the analysis phase of the development process it is applied at the beginning of the development and replaces the analysis phase. Regarding the importance of a structured project start, the student project has shown that it can undoubtedly not be a bad choice to sacrifice time at the beginning to bring structure into the development, for example by means of a model like RE4DIST. Nevertheless, it has also been shown that one should not sacrifice too much time, especially if the model to be applied is not known to the team. The student project ultimately evolved to use more agile development processes towards the end to stay structured. This could have been achieved using RE4DIST, too, but at the cost of time and a more inflexible development process. Due to the fact that a good structure also supports the work in the team, it can be concluded that structuring the developement, for example by means of a model, is highly recommended, especially at the beginning when the team may not yet know each other or the exact specifications are unclear. However there is no question of such being mandatory, because as happened in the project, the unstructured, bumpy start resulted in a solid project. Even if this was certainly not the most efficient way to do it.

7 References

[1] Heisel, M. and Wirtz, R. RE4DIST: Model-based Elicitation of Functional Requirements for Distributed Systems. *14th International Conference on Software Technologies - ICSOFT* (2019), pages 71-81. SciTePress.

[2] Jackson, M. Problem Frames: Analyzing and Structuring Software Development Problems (2001). Addison-Wesley/ACM Press via google.de/books.

[3] Z21 is an innovation of ROCO and FLEISCHMANN, www.z21.eu/de/z21-system/allgemein. Last accessed january 8th 2022.

[4] Diagram created with draw.io, www.app.diagrams.net.

[5] Attack Vectors from the *Common Vulnerability Scoring System CVSS*, FIRST.org (2015).

[6] ChuuChuUDP student project. www.vs.uni-due.de/en/teaching/ws21/projekt. Last accessed january 8th 2022.

[7] Chandra, V. Comparison between Various Software Development Methodologies. *International Journal of Computer Applications* (2015). www.academia.edu

[8] Cho, J. Issues And Challenges Of Agile Software Development With Scrum (2008). www.iacis.org/iis

[9] Munassar, A. and Mohammed, N. and Govardhan, A. A Comparison Between Five Models Of Software Engineering (2010), *International Journal of Computer Science Issues - IJCSI, Vol.7 Issue 5*, pages 94-101, citeseerx.ist.psu.edu.

[10] Raspberry Pi Foundation, www.raspberrypi.com/products/raspberry-pi-4-model-b. Last accessed january 22nd 2022.