Software Architecture Challenges in Distributed Development Settings: An Experience Report

Tassio Vale[1], Taslim Arif[2], Laia Gasparin[3]

[1]Federal University of Recôncavo da Bahia
Rua Rui Barbosa, 710, Centro - Cruz das Almas, Bahia, Brazil

[2]Fraunhofer- IESE
Fraunhofer-Platz 1, 67663 Kaiserslautern, Germany

[3]VOMATEC Innovations GmbH
Riegelgrube 7, 55543 Bad Kreuznach, Germany

tassio.vale@ufrb.edu.br, taslim.arif@iese.fraunhofer.de,
laia.gasparin@vomatec-innovations.de

Abstract. *The RESCUER project proposes a system developed in a highly distributed setting of nine partners spread across the EU and Brazil. Regarding the software architecture activities in the project, failing to identify the key architectural challenges or identifying them at a very late stage of the project causes a lot of cost and effort. In this paper, we present a set of key architectural challenges identified during architecture iterations, and propose solution ideas to deal with it. Our experiences might benefit other organizations engaged in initiatives in these kinds of systems, since they can save valuable time and effort by discovering problems at a very early stage in the project.*

1. Introduction

Nowadays, most people use mobile devices and share their status and information about what is happening around them in real time. This phenomenon can help in an emergency situation, allowing a crowd with mobile devices to send detailed information to the command and control center. This information has high relevance for the operational forces because it is the key to what is happening in real time and originates from the place of the emergency.

The RESCUER system aims to develop a smart and interoperable information system that provides support in an emergency situation using crowdsourcing information. There are several challenges that we have experienced in architecting this system, such as: a) the user experience of mobile applications has to be excellent for diverse classes of users; b) in an emergency environment, the network is often interrupted and data collection therefore becomes difficult; c) the system has to be context-aware to make the analysis intelligent and visualization useful for the command and control center. On top of all these challenges, several development teams distributed throughout the EU and Brazil are developing our system. Such a distributed development setting raised the need for improved knowledge propagation on the technical level.

In this paper, we report a set of architectural challenges identified during the course of this project. We have also identified a set of solution concepts that could potentially help to tackle these challenges. We extend the report of the challenges and solution ideas from Vale et al. (2015), focusing on challenges concerning crowdsourcing systems.

Reported architectural challenges are the key for successful software development projects. Money and effort are saved if appropriate measures are taken to tackle these challenges. Building a crowd-based emergency management system involves a lot of functional and quality requirements. Moreover, distributed development settings create additional challenges during development and integration. We hope that our findings will make the architects of similar systems aware of all those traps that we learned to avoid over time in the hard way.

This paper is structured as follows: in Section 2, we describe the characteristics of the RESCUER project and part of its system architecture. Section 3 describes the architectural challenges and categorizes them according to the architectural viewpoints. In Section 4, we describe the high-level solution concepts and how we mapped them to our challenges. In chapter 5, we conclude our findings.

2. RESCUER Project

The RESCUER project aims to build a smart and interoperable computer-based solution to support emergency and crisis management, focusing on incidents in industrial areas and at large-scale events in Europe and Brazil. Such an infrastructure intends to provide faster and more accurate management in emergency and crisis situations by achieving: improved time to collect information regarding an emergency situation; decreased time and effort to analyze emergency data; improved and reliable information provided to different stakeholders within the shortest possible time; and context-aware interaction with different stakeholders; minimized effort among various workforces.

The goals and requirements defined in conjunction with the project stakeholders and partners guided the construction of the RESCUER architecture. We adopted the Fraunhofer ACES approach [Thorsten et al. 2011] as the software architecture construction process. It comprises three main activities: collecting requirements that need to be addressed by the system architecture; architecting the system; and documenting it through standard models.

The architecture document describes a set of architecturally significant requirements: development-related requirements, integration requirements, availability, robustness, scalability, reliability, performance, usability, security, safety, operability, upgradeability, auditability, and variability. Aiming to realize such requirements and to provide a better understanding for the software development teams, the RESCUER architecture was described in a set of perspectives. Regarding space constraints, we discuss in this paper only one of them: the sub-systems perspective.

2.1. Sub-Systems Perspective

The sub-systems perspective represents RESCUER as a set of components in order to decrease development complexity by assigning the construction of specific components to different teams, assuming they have the desired skills to do so. The RESCUER features are spread across five sub-systems: *Mobile Solution*, *Communication*

Infrastructure, *Data Analysis*, *Emergency Response Toolkit*, and *Integration Platform*. Figure 2 shows the interaction among stakeholders and sub-systems.

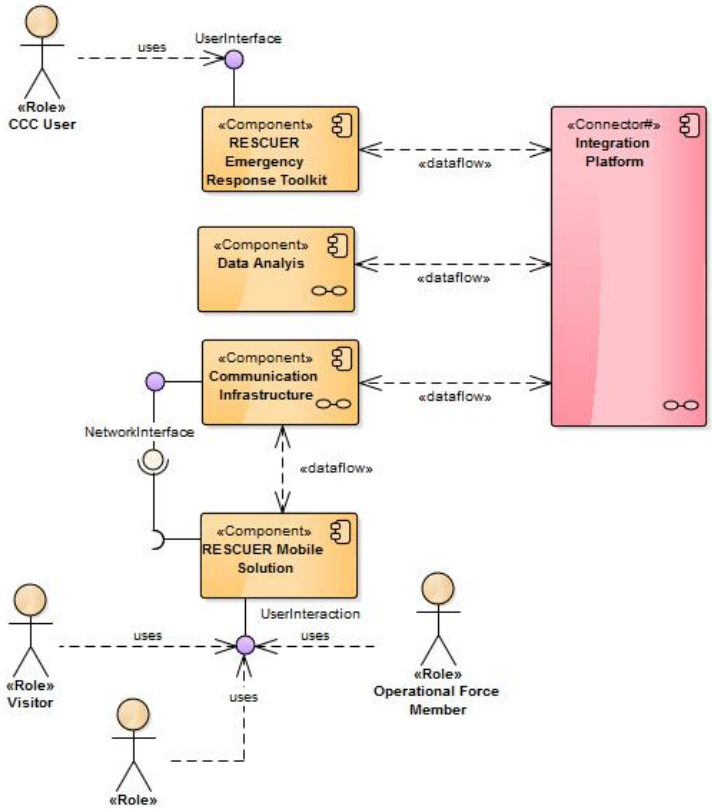

Figure 2. RESCUER sub-systems perspective

The *Mobile Solution* sub-system explores the use of mobile devices to gather information from the crowd in an emergency situation and to support follow-up interactions in an optimized and context-sensitive way. The resulting mobile application interacts appropriately to avoid cognitive overload for the users and to get people engaged in using the RESCUER system.

Aiming to support the information flow between the crowd and the command center, the *Communication Infrastructure* includes a server for receiving, synchronizing, organizing, and storing crowdsourced data from the users' mobile devices. In addition, this sub-system provides a solution for delivering messages to the users' mobile devices in a personalized, location- and situation-sensitive way. This sub-system is also responsible for providing peer-to-peer communication by using the built-in Wi-Fi capability of mobile devices if no Internet connection is available during the emergency.

Automatic data analysis is especially relevant for emergencies in large-event scenarios, where emergency reports from thousands of people are sent to the command center. The *Data Analysis* sub-system receives data to be fused and filtered in order to obtain an enriched collection of data about the emergency situation and thereby enable a more robust and efficient analysis.

The *Emergency Response Toolkit* supports decision-making, coordination of responses, and communication with stakeholders. This sub-system provides appropriate data visualization mechanisms through an intuitive, concise, but resourceful dashboard

with modern solutions to map an incident scenario. It also includes a semi-automated solution for communication with the community to provide timely, coordinated, and accurate information about the nature and status of an emergency situation.

In order to assure consistent and efficient interaction among the other sub-systems, the *Integration Platform* provides a communication protocol, storage, and technology to handle message (called topics) exchanges. In general, the sub-systems interact by publishing and subscribing appropriate topics to the *Integration Platform*, which enables normal system execution.

2.2. Distributed Development Setting

The RESCUER project fosters cooperation among companies, research institutions, and universities from Brazil, Germany, and Spain. In addition, this project has partners from industrial parks in Brazil and Austria to validate the proposed solution in a real-world scenario. Software development is being performed by the following members: MTM, DFKI, VOMATEC, Universidad Politécnica de Madrid (UPM), University of São Paulo (USP), and Federal University of Bahia (UFBA).

MTM is a Brazilian company responsible for implementing the *Mobile Solution*, which should run on both the Android and iOS mobile platforms. MTM needs to integrate the mobile solution with the two libraries provided by DFKI, namely the Ad-hoc P2P Library and the Sensor Recording Library. MTM is also responsible for building the BLOB Storage Service that is responsible for storing multimedia data.

DFKI is a German research institute responsible for developing five modules: Ad-hoc Network, Sensor Recorder, Sensor Data Receiver, Sensor Analysis, and SMS Receiver. All these solutions have to be integrated with the *Integration Platform*. In addition, DFKI might need to take care of the User-Interaction Data Receiver and Data Sender components. VOMATEC is a German company responsible for building the *Emergency Response Toolkit*, Combined Analysis Module, and *Integration Platform*.

UPM and USP are Spanish and Brazilian universities, respectively, developing the *Data Analysis* sub-system. UPM is a Spanish university responsible for building the Video Analysis Module interacting with the *Integration Platform*. USP is a Brazilian university responsible for building the Image Analysis Module, which communicates to the *Integration Platform*.

The Fraunhofer Institute for Experimental Software Engineering (Fraunhofer IESE) develops innovative methods and solutions for the development of high-quality, complex information systems and embedded systems. Fraunhofer is responsible for dealing with infrastructure tasks such as requirements engineering, software architecture description, system integration and user interface design.

UFBA is a Brazilian university responsible for building connectors to social media and external legacy systems of the workforces. This university also provides support for developing several modules in the *Emergency Response Toolkit* and *Integration Platform* sub-systems. UFBA is also handling infrastructure tasks.

3. Architectural Challenges

Considering the distributed development scenario abovementioned, the software architecture team faced a set of challenges concerning architecture

specification/dissemination and the feasibility of design decisions. The challenges are classified as deployment viewpoint, system performance and context-awareness challenges.

We extend the report of the challenges and solution ideas from Vale et al. (2015), focusing on challenges concerning distributed software development and context-awareness. At the end of this Section, we relate the architectural challenges and the solution ideas applied to mitigate them.

3.1. Deployment Viewpoint Challenges

Deployment viewpoint challenges are related to the construction of a running environment to test and operate the RESCUER system. It involves design decisions such as component deployment by the project partners and the integration of the individual components to generate a unique and transparent system of systems.

- AC01 – *Commercial deployment*: this project intends to develop an experimental crowdsourcing system, bringing several skills together to deliver a proof of concept for the emergency and crisis management domain. RESCUER provides innovative features such as crowdsourcing information gathering, image analysis and video analysis for this specific domain. In this context, it is reasonable starting with moderate expectations from the audience.

  However, the expectation of potential users (professionals from industrial parks in Camaçari-Bahia-Brazil and Linz-Austria) is to use RESCUER in a real-world scenario since its first release. For architectural matters it presents some challenges concerning how to adapt RESCUER and operate it in very different contexts, considering different requirements, laws and regulations. As consequence, the software architecture has to cover several aspects: realizing the variability between different real-world scenarios by achieving component adaptability and negotiating it with the project partners, dealing with different priorities of architectural significant requirements (e.g. availability, performance and robustness) and providing a minimal infrastructure to deploy each RESCUER component.

  Currently, the experimental scenario does not consider any context variations. This is a consequence of our iterative development approach, since the first increments of the architectural views do not fully address commercial expectations. Additionally, many system components face state-of-the-art issues during implementation. As result, RESCUER architects provide documentation with limited features (smaller scope of functionalities) from the expected commercial product;

- AC02 – *Test deployment*: the project is performing an integration taskforce to set up a suitable environment for system and integration testing. To achieve it, the taskforce members (including the architecture team) must negotiate with RESCUER partners to provide the software deliverable and specify the requirements to install it in a controlled setting. The architecture team has to extract the needed information from each partner and document it properly.

  Currently, the architects face the challenge of extracting from the development teams their understanding about the software requirements as well as their

architectural knowledge, aiming to verify whether they have a suitable component for system integration and testing. In case of requirements or architectural misunderstanding from the development teams, the component integration and testing might be compromised.

3.2. System Performance Challenges

Performance challenges are related to architectural significant requirements describing any aspect that impacts on the overall system performance.

- AC03 – *Performance from data extraction to visualization*: the time required for gathering crowdsourcing information from the mobile applications, performing data analysis and visualizing the results has to meet the maximum response time of 0.5 seconds. Despite the architecture team has proposed a set of design decisions to address this significant requirement, evaluating it still remains a challenge.

  Architecturally, such an action involves components from different partners around the world. Measuring performance of individual components is trivial, however, evaluating the performance of the end-to-end communication of this scenario implies a consistent system testing environment as well as suitable metrics for this context. Consequently, the architecture has to provide metrics support the configuration of a testing environment in a proper way;

- AC04 – *Scalability*: fostering the loose coupling of architectural components to keep an independent work for the distributed teams comes at a price. There is an architecture significant requirement stating RESCUER must address scalability for 100-200 test users. The partners are spending effort to meet a set of quality attributes considering the components they are developing.

  However, it does not guarantee the RESCUER system achieves the required scalability automatically. The architecture team must provide proper environment and evaluation strategies for this scenario. Once the results do not meet the significant requirements, the architects have to adapt the components description if necessary;

- AC05 – *Extensibility towards new components*: the design decisions addressing independent components in a distributed development setting deal with high coupling. As new requirements arise, RESCUER becomes more complex and their components tend to incorporate more features. It turns fine-grained components into coarse-grained ones.

  In this scenario, the maintainability is compromised and designing smaller/extensible components is essential. In addition, procedures and technologies to provide a comprehensive integration are essential.

3.3. Context-Awareness Challenges

Depending on the type of incident (e.g. fire, explosion or gas leakage), the source of information (e.g. firefighter, affected person or civilian), regulations/laws of the affected area and other aspects, an emergency and crisis management system must behave differently to provide consistent information and combat the incident. Aiming to adapt the system according to several possibilities of an emergency and crisis scenario, the

architects concluded RESCUER has to address context-awareness. The context-aware architectural challenges are discussed next.

- AC06 – *Identification of contexts*: understanding the variations of context faced by the RESCUER system is necessary. It will demand requirements elicitation with the potential users (professionals from industrial parks in Brazil and Austria), and the architecture should accommodate it in a self-adaptive system. The current architecture does not support context-aware features. It requires refactoring and all partners must be aware of eventual modifications;
- AC07 – *Context-aware data analysis*: this is the core feature to turn RESCUER into a context-aware system. RESCUER must be able to get raw data (text, images and videos) and generate proper information considering the current context of an incident. It involves a well-defined structure of knowledge about the variations of the context, a reasoning engine to process data and provide context-aware information, and a database to store previous experiences that might help to understand future contexts;
- AC08 – *Context-aware visualization*: the architecture specification describes usability as significant requirement. Visualization mechanisms for context-aware systems is still a research gap, however, RESCUER must adopt existing visualization metaphors and techniques to provide a consistent visualization for mobile and command center users. Considering context-aware visualization is a research topic, the architects must describe a flexible component able to incorporate greater changes requested by the new requirements that can arise for this task.

4. Solution Ideas

Facing the challenges previously presented, the architecture team is incorporating six preliminary solution ideas into the project:

- *S01 (related to AC02, AC03 and AC04) – DevOps toolchain*: incorporating DevOps [Bass et al. 2015] concepts would greatly help in our scenario. It would facilitate continuous integration, continuous testing, and continuous delivery with appropriate tool support. In addition, such tools have features to improve the distributed software development, making the activities and results sharing easier;
- *S02 (related to AC01, AC03, AC04 and AC06) – Continuous feedback from potential end-users*: as the domain is not yet well understood, it is not possible to elicit all requirements in the first attempt. Therefore, it is important to continuously show the prototypes or partially implemented system to the end-users. Refining the requirements and making the system acceptable to the community is not possible without continuous feedback;
- *S03 (related to AC01, AC02, AC03 and AC04) – Plan for creating development/test environment*: it is important for distributed development projects to plan for IT provisioning support. The architects are creating a schedule for system integration and testing, defining which features each component must deliver and the related test cases. Consequently, the

development teams must manage their tasks to achieve the integration goals. It is essential architects to perform such plan, since they have an overall understanding of the RESCUER system. Individual development environments provided by each team are not enough for doing sound integration and acceptance testing;

- S04 *(related to AC05) – Decouple components*: every component needs to be decoupled as much as possible from the rest of the system. A component should provide specific services but it should not be aware of how the service will be consumed;
- S05 *(related to AC03 and AC04) – Continuous testing*: to learn about the current status of the development, it is important to set up an environment and a process for continuous integration. This will make pain-point explicit to everyone;
- S06 *(related to AC06, AC07 and AC08) – Context interpreter component*: this kind of system needs to use modern analysis and visualization techniques. To do an effective and efficient analysis, it is important to know the context of the current state of the system. It is therefore important to build a context interpreter that keeps track of the context and supports all other components with the context information whenever necessary. The whole system should be built around this new component.

5. Conclusion

In this paper, we presented the challenges we experiences in the RESCUER project. Understanding architectural challenges is the key for making any software-intensive system successful. The architectural challenges drive the architecture and it is located at the center of any system development. We classified the challenges according to architectural viewpoints, and we also presented high-level solution concepts, relating them corresponding to each challenge.

We assume this work can benefit other organizations to build such a system in a distributed setting, preparing themselves with respect to the challenges and solutions described here and thus greatly reduce the cost and effort for development. In addition, the architectural challenges can be addressed by further research. As a future activity, we will continue refining the solution concepts and applying them in the RESCUER setting, striving it to obtain evidence with respect to our ideas.

References

Vale T., Arif T., Pedraza L., Vieira V. (2015) “Architecting Crowdsourcing Systems: Challenges and Solution Ideas from the RESCUER Project”. Anais do Primeiro Workshop sobre Sistemas de Crowdsourcing. Belo Horizonte, Brazil.

Bass L., Weber I., Zhu. L. (2015) DevOps - A Software Architect’s Perspective. Pearson Education, Inc.

Thorsten K., Jens K., Matthias, N. (2011) “Architecture-centric software and systems engineering. Fraunhofer ACES: Architecture-Centric Engineering Solutions, IESE-Report, 079.11/E, 2011”, http://publica.fraunhofer.de/dokumente/N-186361.html.