Managing the Iterative Requirements Process in a Multi-National Project using an Issue Tracker

Christian R. Prause, Marius Scholten, Andreas Zimmermann, René Reiners, Markus Eisenhauer
Fraunhofer Institute for Applied Information Technology
Schloss Birlinghoven, 53754 Sankt Augustin, Germany

Abstract

Today’s software development is becoming more and more “international” every year. With involved stakeholders being distributed over several remote locations in the world, particularly the management of software requirements becomes complex and difficult to handle.

In this paper, we report on a web-based computer-aided requirements elicitation process that decreases the number of necessary face-to-face meetings and reduces communication overhead.

It is thus possible to distribute requirements elicitation efforts to partners or contractors that may be situated anywhere in the world.

1 Introduction

The paper presented here relies on the experience within a big integrated project: Hydra1 – Networked Embedded System Middleware for Heterogeneous Physical Devices in a Distributed Architecture. The nature of such projects is that all project work is highly distributed among different partners, working in different institutions and coming from different countries. As a result there is a strong need to implement a structured process to drive project management and system implementation.

We describe an approach for managing requirements across different, remote organizations. Software development in the Hydra project follows an iterative process. Four iterations span several years and involve more than 30 commercial developers and scientists from various European countries. The project result is a middleware for networking embedded devices. Target users of Hydra are developers, who then create end-user software. However, mastering the complex tasks of middleware development is time consuming problematic even for middleware experts [7].

The Hydra middleware will consist of several managers sometimes reflecting the specific expertise of a respective partner (e.g. network manager, security manager, ...). Efforts for creating these are divided among the different partners of the Hydra consortium. Although one project partner has the architecture lead — that is, this partner provides infrastructure, sets up a process and assigns work — actual requirements elicitation is done by each partner, which is an expert in its respective field. Not only is requirements elicitation distributed, but the entire process: requirements are gathered, refined, checked and finally implemented independently.

Quality is a major objective in Hydra, as it is increasingly demanded by users. Hydra is therefore strongly committed to a user-centered-design approach. We use an applied methodology for collaboratively eliciting requirements across far distances and national borders. Refined requirements are used within iterative development cycles to assure that the goal of a user-centered approach is met in all phases of the project. The requirements serve as a reference to measure if the development within the project is always in line with the functionalities and properties desired by them. The requirements derived are integrated and generalized to form a package of initial requirements specifications. These are fed into the architecture design specification.

This paper reports on the requirements engineering process and explains chosen methods. First, we motivate the usage of user-centered design approaches and present specific problems arising from user-centered design under the constraint of distributed requirements engineering (see section 2). These problems are addressed by a novel computer-aided software engineering process that combines web-based tools with a sophisticated requirements schema (see section 3). Next, we look at the requirements process of Hydra in more detail (see section 4), relate our process within the context of requirements engineering practice (see section 5), and report on the experiences with our approach (see section 6). Finally, we conclude this paper by summing up on our contribution (see section 7).

1http://www.hydra.eu.com
2 Distributed Management of User-Centered Requirements

In this section we give an overview of the user-centered requirements engineering process used in Hydra. We characterize ancillary conditions resulting from Hydra consortium organization, and describe the development environment with specific respect to the particularities of middleware development (see 2.1). Ultimately, we conclude this section by summing up on the problems that demanded for a new approach to managing our requirements (see 2.2).

2.1 The Hydra Requirements Process

The Hydra project is developing a middleware; hence users addressed by the Hydra software are developers. To avoid confusion, we name such users developer-users or simply users. We speak of end-users, if we refer to developer-users’ users.

In several steps, vision scenarios were created that illustrate Hydra functionality in various domains. End-user representatives validated whether these vision scenarios capture their goals, tasks and contexts of use. The validation was conducted by different partners in their respective countries. The derived vision scenarios are visions of future support and benefits of a system enhanced by the Hydra middleware for end-user and developer-user roles. They were developed with the IDON method [5].

In the IDON method a wide range of environmental factors, ambiguities and uncertainties are examined. This is to find out what roles they are likely to play for unfolding scenarios. In an initial phase environmental factors are collected, grouped together according to their degree of uncertainty, and it is decided, what their relative order in means of uncertainty and future impact is. Following IDON phases only consider factors with high uncertainty and direct impact on future developments. These factors are reformulated as either/or (so called flip/flop) questions and are grouped according to their interrelations and associations. Finally, the remaining factors are combined into four distinct possible futures. These futures are thus extrapolated from the thinking that was done by the group [5].

From the vision scenarios we derived technical scenarios that drew more concrete pictures of future work with our middleware. Technical scenarios were focused on the developer-user views. We created this kind of scenarios, because developer-users are the actual users of Hydra. Technical scenarios address technical questions referring to the middleware and its components. Derivation of scenarios was, again, done by several partners in different locations.

By focused user interviews, technical scenarios were evaluated and discussed by developer-user representatives. They assessed the functionality described in the technical scenarios (“Would they want to have such support?”), expressed their concerns (“Are there any issues that need to be taken care of?”) and named their requirements (“How would the system have to be in order to best serve its purpose?”).

In addition, the requirements gathering process was directed by focus groups located at various partners’ sites, in which a team of interviewers conducted discussions with 7 to 10 developer-users with different backgrounds. These focus groups provided valuable insight into the work of targeted developer-users and contributed to a worthwhile understanding of the current and imminent problems, limits, tasks and benefits of a future Hydra middleware.

All developer-user comments were documented and analyzed carefully. This exploration and analysis allowed for assessing a first set of requirements. During later iterations these requirements are reworked, checked and extended according to the experiences made with early prototypes of the Hydra middleware.

Early user requirements serve to guide the system specification and to evaluate design options. The integration in developing technology and solutions continues through several discussions with developers in teams of every partner. User-oriented development hence is a distributed and ongoing process: the next step is the validation of the derived and more specific technical scenarios, and a validation of requirements. An end-user evaluation of first prototypes, and a following review and elaboration of requirements are subsequent major steps.

The general approach to requirements gathering involves several activities in the Hydra project. It is important to underline that most of the following activities are performed in parallel and repeatedly at different partners’ sites:

- **Elicitation.** Discovering, extracting and learning needs of stakeholders. It includes a domain analysis that helps to identify problems and deficiencies in existing systems, opportunities and general objectives. Scenarios are part of this activity.

- **Modeling.** Creating models and requirements, looking for good understanding of them and trying to avoid incompleteness and inconsistency.

- **Negotiation and agreement.** To establish priorities and to determine the subset of requirements that will be included for the next phase.

- **Specification.** Requirements expressed in a more precise way, sometimes as a documentation of the external behavior of the system.

- **Verification/Validation.** Determining the consistency, completeness and suitability of the requirements. It could be done by means of static testing (using regular
reviews, walkthroughs or other techniques) and prototyping.

- **Evolution and management.** The requirements are modified to include corrections and to answer to environmental changes or new objectives. It is important to ensure that changes to requirements do not produce a large impact on other requirements. Requirement management means to face those modifications properly, to plan requirement identification and to ensure traceability (source, requirements and design traceability).

### 2.2 Summary of the Problem

A major problem that overshadows the user-centered design process for Hydra is how to combine all the various requirements from remote partners in a large project like Hydra without a core requirements team. As requirements are descriptions of how the system should behave, application domain information, constraints on the system’s operation, or specifications of a system’s property or attribute, an incomplete requirements analysis tends to lead to problems later in the system development. Notably, taking into account functional and non-functional characteristics during all development phases is of paramount importance when developing middleware software [6].

This is why it is important to always continue with the user-centered design process. Hence an initial set of requirements must steadily be overworked: new requirements arise or outdated ones disappear in the iterations of the project.

This, however, conflicts with the intentionally distributed requirements engineering process of Hydra: Every partner has his individual expertise in a specific domain like security, networking, embedded devices, semantic webservice or ontologies. Only experts of a respective domain can provide and maintain a set of high-quality and up-to-date requirements for this domain. But some of these experts work in places almost 3000km apart.

### 3 Volere and Jira

The partners involved in the Hydra project have different cultural backgrounds and interests, are separated by several national borders, and additionally, exhibit varying software engineering backgrounds. In order to guide and direct them to provide correct information about requirements that they need to elaborate, we identified two important prerequisites for the requirements specification process: an encompassing requirements schema and a centralized requirements repository.

The encompassing requirements schema needs to be sufficiently rigid to force provision of vital information, but must also be flexible enough to capture the details. The centralized requirements repository should allow remote access to the database. As requirements are to be entirely developed in distributed teams, full access is a precondition. The tool used for remote access should be capable of representing and mapping the requirements schema that we want to use. Of course, it should not push project equipment costs over budget.

Therefore, we decided to base the requirements process on the Volere requirements mastering process and on the Jira\(^2\) bug-tracking tool. The following subsections introduce these two in more detail: the Volere schema requirements schema (see 3.1) and the Jira bug-tracker (see 3.2).

### 3.1 Volere: The Requirements Schema

The active involvement of users and a clear understanding of user and task requirements is a challenge in projects like Hydra for two reasons: first, the potential user groups are not known a priori, but need to be identified according to future scenarios; this needs to be revised; as visions evolve there may be various groups of potentially affected users. Second, the visions of the Hydra project are far-sighted and not close to users’ current experiences; therefore users may not be confident and precise about their needs concerning this future system.

ISO 13407 (“Human-centered design processes for interactive systems”) [1] standardizes goals for a human-centered design process. However, the standard does not prescribe specific methods to achieve these goals; they are to be chosen according to what state of the art is and what is appropriate under individual project circumstances. Based on practical experiences from other R&D projects, we have devised a scenario-based approach, combined with user interviews and expert analysis, based on the structure proposed by Robertson & Robertson for mastering requirements [9]. Their recommended Volere process ensures that all important aspects of requirements are carefully addressed and that the methods applied have proven their value in practical work. The first edition of the Volere Requirements Template was released in 1995.

The Volere template distinguishes between global constraints affecting the project, functional requirements and non-functional requirements, with a fine-grained distinction of different types. In addition, the categorizations of the Volere template require to define fit criteria, a rationale for each requirement and the evaluation of customer satisfaction and dissatisfaction if the requirement is implemented or not. All this has proven to be of great practical value. The philosophy of Robertson & Robertson is very much in line with ISO 13407 and allows a structured processing of

---

\(^2\)http://jira.atlassian.com
3.2 Jira: The Issue Tracker

Currently the most advanced way of dealing with bugs is to enter them into a bug tracking system [8]. Bug-trackers address the critically important task of tracking and managing issues and bugs that emerge during a project. Knowledge of encountered software errors cannot get lost and communication between users and developers of computer applications is enhanced, since the containment of a bug usually requires a sequence of questions and answers between users and developers. Such communication is then attached to the corresponding incident.

One such system, that is also highly customizable, is the Jira issue tracking system. Jira is a web-based application for managing software bugs, handling problems and for the administration of projects. It has been developed by Atlassian Software Systems. The primary application of Jira is as a bug-tracker in the field of software engineering addressing the tracing of error states and later the process of error correction. Jira allows for recording and documenting of errors and facilitates writing of reports about the status and features of a project. Using Jira’s application programming interfaces and plug-in architecture, one can add new functionality, manipulate data and even integrate Jira with legacy systems. Extension categories currently include version control integrations, workflow extensions, alternative clients, and more. Jira bases on Java EE and runs with just about any database and operating system.

For purposes other than tracking bugs, Jira enables the creation of workflows that map a specific issue’s life cycle according to specific business processes. Different input fields per issue tracked by Jira as well as entire issue management workflows can be re-defined. Furthermore, the reporting functionality of this tool can be configured as well: notifications can be sent via email or RSS feeds if an issue gets into a specific state. Automatically generated reports can be directly exported into project deliverables. Another important aspect consists of the ability of setting up fine-grained security levels and permissioning schemes for users, teams and projects roles. This allows to involve very different kinds of stakeholders in the requirements process.

4 Implementation of the Process

This section details the requirements engineering process of Hydra. First, it describes how we collected requirements in focus groups and how they were entered into the Volere schema. After that we explain the requirements process we implemented in Jira.

4.1 From focus groups to requirements

A focus group is a qualitative research method that uses groups of experts — developers in our case — to find out their attitude towards a new idea, product, service, or concept. Focus groups are an important tool for acquiring feedback regarding new ideas and for discussing or testing a new
product before it is made available to the public. A focus group establishes an interactive group setting, where participants (potential customers) are asked questions by one or two moderators. The participants are free and encouraged to discuss the questions with other group members.

In order to initiate the discussion within the focus group, we presented technical scenarios that transport the ideas of the envisioned software. The scenarios help capturing and illustrating features of the new system, its modes of usage and its benefits, and therefore, permit participants to imagine the new system. Information emerging from the conducted focus groups was documented by directly taking notes during a session. In some cases, when all details were to be captured, the participants were also videotaped. We decided to conduct focus groups, because they are a good tool to support the collection of ideas about how to optimize the use of a new technology. Furthermore, focus groups are low in cost, one can get results relatively quickly, and the sample size of a report can be increased easily by discussing with several people at once.

### 4.2 Entering Volere Requirements

The result of focus groups constitutes a set of statements that needed to be converted into requirements. Developer-user requirements are based on empirical data, i.e. on real developer-users’ comments on the vision depicted by scenarios. The discussions within focus groups were qualitative, explorative and meant to explore the context of use, as well as developer-users’ values and concerns. The condensation of the original user statements has been accomplished mainly by abstraction, i.e. by eliminating redundancy in clusters of equivalent statements and phrasing the essential meaning in one statement. This set of aggregated and more prescriptive statements has been entered into the Jira issue tracker using the Volere template as shown in figure 2.

The assignment of a suitable rationale and an appropriate classification has been done in several group discussions within the project team. At this step, the Volere template proved to be useful at this step, since the results needed to be documented in a way that can be communicated efficiently to developers. This initial set of user requirements is then reviewed during the user evaluation of first prototypes in order to gain the second, improved set of user requirements (see for example [10]).

### 4.3 Implementing the Volere Requirements Process in Jira

Jira allows issue-tracking according to customizable workflows. Each workflow can be connected to a permission scheme that allows different user groups to perform certain workflow transitions. Based on [9], we process Volere requirements according to the workflow depicted in figure 3.

As indicated by the coloring, the set of workflow steps can be clustered into three groups (The state Quality Check passed is omitted, see the explanation below):

- **Initial**: When a requirement is reported, it is in state Open. Open requirements can be edited or (re-)assigned (see below). To retain the possibility to modify requirements that already belong to the specification, the state Reopened is introduced. Reopened requirements differ from newly reported requirements only in having passed parts of the workflow before. That is, why both states, Open and Reopened, are considered to be initial states.

- **Quality check failed**: The quality check is a central instance in the Volere requirements process. If all fields in the Volere shell are filled in correctly and meaningful, the requirement passes the quality check. A requirement can fail the quality check for one of the following reasons:
  - **Incomplete**: A requirement could be filled in incompletely, mandatory fields could be missing.
  - **Ambiguous**: The meaning of a requirement’s content must be absolutely clear. If one or more
fields have ambiguous meanings, the requirement is transferred to the corresponding state.

- **Senseless**: A requirement is marked as senseless if its meaning is not understandable.

- **Resolutions**: If a requirement has passed the quality check, this does not imply that the requirement automatically belongs to the specification. Because requirements are created by multiple independent persons, duplicates can occur. If a requirement does not yield any additional value to the requirements that are already part of the specification, it is marked as **Duplicate**. If a requirement does not meet the software's concerns, if other requirements are already part of the specification or if it contradicts decisions that have already been made, it is marked as **Rejected**. The states **Duplicate**, **Rejected** and **Part of Specification** are called resolutions, because they typically are final states.

The state **Quality Check Passed** is not mentioned in the above classification, because it is a transient state. Marking a requirement accordingly without resolving it does not make sense.

Two different user groups are involved in the requirements process:

- **Reporters**: This group contains all project members. Reporters can use Jira’s web front end to create new requirements.

- **Assignees**: Each newly reported requirement is assigned to a single person – the assignee. The assignee is responsible for passing the requirement through the quality check.

The assignment of requirements work as follows: We extended the Volere shell with an extra field, that allows reporters to make a topical classification of their requirements. The members of the assignees-group then distribute incoming requirements among themselves, according to their expertise.

If a requirement has a rather poor quality, the assignee comments on the fields in question and sets the requirement’s state to **Ambiguous**, **Senseless** or **Incomplete** respectively. Jira features automatic email notification on workflow actions. The reporter is informed about the quality check failure and can update the adequate fields. After that the reporter sets the requirement’s state to **Reopened**, which indicates the assignee that he has to quality check the updated requirement again.

The Hydra requirements specification is rather complex. In order to decide, whether or not a requirement that passed the quality check should be added to the specification, rejected or marked as a duplicate in the specification, demands a detailed knowledge of the whole requirements engineering process. That's why only a small group of people deals with the resolution of requirements. This small group consists of selected experts from the various partners.

Figure 4 displays an example of a requirement that features a good quality and therefore, passed the quality check and became part of the specification.

5 Related Work

The standard ISO 13407 gives guidance on human-centered design activities throughout the life cycle of
Essential principles in this process are

- multi-disciplinary design,
- iteration of design solutions,
- appropriate allocation of function between developer-user and technology,
- and active involvement of users and a clear understanding of user and tasks requirements.

Multi-disciplinary design is given by the expertise in Hydra, which includes psychologists, computer scientists, usability engineers, and designers. The iteration of design solutions is implemented in the work plan, as a human-centered approach implies an iterative life cycle (four cycles are planned during project lifetime, aiming at validated prototype specifications and concepts of usage). A system is perceived as a socio-technical system, i.e., the novel technology is a fit between a technical system and its usage [4]. Scenarios are part of the system specification; they explicitly deal with the usage of a technical system, the context of use, and an appropriate allocation of function between the technical system and human users. Later, when a prototype is available, users try it and gain personal experience with it. Iterative cycles allow advancing from specification to prototypes, from experience and evaluation to improved specifications and enhanced prototypes.

Active involvement of users and a clear understanding of user and task requirements is a challenge in Hydra for three reasons: First, developer-user groups are only indirectly addressed by vision scenarios; the project has to focus on developer-users and at the same time not lose track on end-users. Second, potential end-user groups are not known a priori, but need to be identified according to vision scenarios; this needs to be revised as objectives of visions evolve. There may be various groups of potentially affected end-users. Third, the visions of Hydra are far-sighted and not close to developer-users’ current experiences; therefore developer-users may not be confident and precise about their needs concerning this future middleware.

One of the core tasks of user-centered design is to negotiate and facilitate the communication across well-known gaps between users and developers while acknowledging the different forms of expression and different requirements on each side. The literature has a lot of examples demonstrating that end-users have to bridge the large gap in understanding especially in projects that apply a waterfall model. Evolutionary or iterative approaches drastically reduce this gap [3].
Different phases of user-centered design are a cyclic process with no sharp start and end points: the context of the use phase with intensive user involvement continues for the whole duration of the process with an emphasis on the beginning. The requirements elicitation likewise extends well into the design proposal phase. There are four essential human-centered activities recommended by ISO 13407:

1. to understand and specify the context of use,
2. to specify the organizational and user requirements,
3. to produce design solutions, and
4. to evaluate design regarding requirements.

Evaluations of design proposals yield a richer understanding of the context of use and new or modified requirements and thus guide the evolutionary improvement of design. For users and designers the design proposals can be understood as design probes to explore the characteristics and usefulness of the proposed system. When a prototype is available, end-users can try it and gain personal experience with the system.

However, ISO 13407 does not prescribe specific methods to achieve these goals; they are to be chosen according to what state of the art is and what is appropriate under individual project circumstances. Based on practical experiences from other R&D projects, we have devised a scenario-based approach, combined with developer-user interviews and expert analysis, based on the structure proposed by Robertson & Robertson for mastering requirements [9].

The Volere process recommended by Robertson & Robertson ensures that all important aspects of requirements are carefully addressed and that the methods applied have proven their value in practical work. Several characteristics of Volere have proven to be of great value: The distinction between global constraints affecting the project, functional requirements and non-functional requirements — with a fine-grained distinction of different types — gives early hints for software architecture design. Categorizations of the Volere template with the need to define fit criteria and a rationale for each requirement capture valuable background information. And the evaluation of customer satisfaction and dissatisfaction if the requirement is implemented or not, help in prioritizing realization of requirements.

The philosophy of Robertson & Robertson is hence very much in line with ISO 13407 and allows a structured processing of the requirements assureing that they remain always applicable and testable.

Jira is originally a bug tracker, but has been re-purposed and is advertized as an issue tracker. An issue tracker not only allows tracking of errors in software, but also tracking of tasks, developments and improvements [2]. We further extend this view of Jira onto management of requirements for the Volere process.

6 Lessons Learnt

This section documents the outcome and experiences gained from the application of the Jira bug-tracker to the requirements engineering process of the Hydra project. Throughout the course of this phase new questions and ideas arose, which might trigger a continuation of the project with specialized foci on these new issues. For the next iteration of the developments in the Hydra project, these lessons learnt will have considerable impact on the next phase of the requirement engineering process.

The lessons learnt documented by this section base on experiences of 24 people involved in the requirement engineering process. 21 of these people reported requirements, while 10 are responsible for quality checking and the final resolution of requirements. Currently the Jira database contains 449 requirements in total. Out of these 291 have been made “part of specification”, 57 are duplicates, 54 were rejected and 47 are pending for final resolution. Reasons for not having finally resolved a requirement are either because the requirement has been added recently (22 requirements) or just passed the quality check (11), because it was returned for quality checking after corrections (10), or it was declared to not fulfill quality standards by the assignee (3 “ambiguous” and 1 does not make sense).

As a general result, the overall experiences are throughout positive, since the combination of Volere and Jira fulfilled all our requirements. The following subsections explain the three major lessons learnt in more detail.

6.1 Enhanced Communication

We configured the reporting functionality provided by Jira in a way that email notifications are sent to people in their specific role within the requirements gathering process, if a requirement changes its description or state in the life cycle. These email notifications initiated fruitful discussions between the reporter of the requirement and the assignee as the person responsible for the quality of the requirement. Furthermore, the partners involved with the Hydra requirements engineering process extensively made use of Jira’s functionality to submit comments to requirements. In the course of such ongoing discussions, the quality of requirement descriptions increased significantly.

6.2 Managing Requirements

The long term application of Jira for the requirements engineering process in the Hydra project demonstrated that
its functionality allows for an easy managing of requirements. Particularly, the filtering and search mechanisms have proven to be very valuable in supporting the work with requirements. Mostly used search profiles concern the identification of requirements in a critical state within their life cycle, the listing of requirements associated with one specific assignee and the filtering of requirements with regard to the individual work packages of the project. The second valuable feature of Jira consists in the automated generation of reports based on the requirements, which make up a substantial part of current project deliverables. Considering an amount of 449 requirements, automated documentation significantly aids the work with requirements.

6.3 Insufficient Requirement Quality

Even though, Jira supports and facilitates the distributed handling of requirements, the initial quality of requirements has been insufficient. Involved persons quickly learnt how to use the tool, however, the initial description of the requirements have been incomplete and inadequate. Particularly, the specification of the fit criteria of requirements has often been neglected, and critical states of requirements have been ignored. Of course, insufficient quality of some requirements cannot be attributed to Jira, however, its built-in functionality could be adapted, in order to encourage people to produce high-quality requirement descriptions.

7 Conclusions and Future Work

Hydra is a multi-national research project that implements a human-centered development process according to norm ISO 13407. Developing a highly innovative and user-friendly technology raised the need to choose appropriate software engineering methods. Specifically, the Hydra consortium had to implement a sophisticated requirements engineering process. The major challenge was to combine the iterative requirements elicitation of a user-centered design process (as presented in section 2) with the distributed management of requirements by different experts.

Our solution to this problem involves the Volere requirements schema and the Jira issue tracker (see section 3). Volere ensures that information on requirements is consistent to an agreed schema. Jira manages the requirements in a central place and provides access to all the involved parties across Europe.

Requirements are based on empirical data, i.e. on real developers’ comment on the vision depicted by the Hydra project. We conducted qualitative discussions within focus groups to explore the context of use, user values and their concerns. Original statements were aggregated to a traceable set of requirements utilizing Volere templates and a Volere process that was implemented in Jira (see section 4).

As an aspect of future work, we plan a closer integration of the management of requirements with the remaining steps of the software engineering process. Currently, we realized a linkage between requirements and the Subversion repository, which manages the versioning of the developed software modules: If the comment of a committed source code file contains the identifier of a requirement stored in Jira, this comment is directly linked with the description of this requirement.

The Hydra requirements engineering process supports remote collaboration of domain experts and enables quality project results.

Acknowledgment

The Hydra project is co-funded by the EU.

References