Experience in Managing Requirements Between Distributed Parties in a Research Project Context

Ken Pierce, Claire Ingram
School of Computing Science,
Newcastle University, Newcastle upon Tyne,
United Kingdom, NE1 7RU
Email: kenneth.pierce@ncl.ac.uk,
claire.ingram@ncl.ac.uk

Bert Bos
Chess iX
Lichtfabriekplein 1
2031 TE HAARLEM
The Netherlands
Email: bert.bos@chess-ix.com

Augusto Ribeiro
Department of Engineering,
Aarhus University
Finlandsgade 22, 8200, Aarhus N
Denmark
Email: ari@iha.dk

***Abstract*—This paper describes the experience of managing a requirements process between distributed parties with diverse interests in a research project context. We present some key 'lessons learned' from a new case study, the DESTECS project, and summarise lessons learned from previous experience reports. Key risks include obstacles imposed by the geographic distance; the different domain knowledge and working contexts of partners; and a risk that autonomous partners' goals do not always coincide. Our observations on a new case study broadly support a previous study, but we also propose some new lessons to learn, including the creation of a small, representative 'requirements authority' (RA); investing time in studying common concepts early in the project; and ensuring that expectations for requirements and for deliveries are made explicit.**

***Keywords*-Requirements, distributed software development, global software development, research consortium.**

I. Introduction

Requirements engineering has long been acknowledged to be a key element of a successful project (e.g. see Brooks [1]). Many innovative techniques have been proposed to improve the results of the requirements process, and of these many have demonstrated their effectiveness in a conventional setting, where requirements engineering (RE) and development is handled by a single, co-located organisation. However, RE can be more challenging in a project without centralised control, co-located teams, and a shared knowledge and working culture.

In this paper we present our experiences of RE activities in a European Commission funded research project (STReP) called DESTECS. Such projects explicitly require partners to come from multiple countries and consist of both academic and industrial partners. This setting challenges the assumptions of conventional RE techniques, because tasks are shared between multiple autonomous partners, subteams are geographically distant and partners have different goals, cultures and expectations. In addition, proposals for these projects are often created over short timescales and up against deadlines. Partners are mainly selected for their technical expertise, so RE may not receive sufficient attention during the proposal phase. It is unrealistic to expect that there will always be RE experts within such consortia.

While many of these are individual risks and problems that the RE community has long recognised, we believe that few case studies have been published of projects that experience all of these problems simultaneously in a research project setting. We hope to build on the small number of previous experience reports from similar projects and start the road towards a 'best practice' set of guidelines for future projects. In the remainder of this paper, related work is summarised in Section II. The case study is described in Section III. Lessons learned are described in Section IV, with reference to previous experience reports. Concluding remarks are made in Section V.

II. Related Work

There has not been much attention yet focused on the specific collection of challenges posed by the type of project we describe here. One exception is an experience report by Gürses et al. on RE from the Trusted Architecture for Securely Shared Services (TAS^3[†]) European research project, which is also a cross-border consortium of independent partners. We refer to this study in Section IV.

Although few researchers have concentrated on the specific collection of challenges described above, some attention has been paid to them individually. For example, in the past few years, Distributed Software Development (DSD) or Global Software Development (GSD) [2] has been the subject of much study. Introducing a physical distance between project members has been found to affect productivity negatively in the past [3], partly because activities such as co-ordinating delivery or updating partners becomes a challenge that must be actively worked on. RE activities in particular benefit from informal communications or 'corridor talk' — the ability to chat over coffee with users or developers; to get quick answers to simple questions; and to pick up on informal cues such as intonation of voice [4], [2]. Herbsleb also emphasises the importance of losing informal communication channels, which undermines awareness of how a team is progressing and makes it difficult to track small problems as they propagate throughout the project [5]. Verner et al. point out that geographic separation introduces significant complexity and risk, suggesting 'a 50% failure rate for distributed projects is not

[†]See http://vds1628.sivit.org/tas3/ for details. Note that the superscript '3' is part of the project's designation.

uncommon' [6]. A systematic review on collaborative software development identified the following challenges specific to requirements engineering that could be attributed to geographical distribution of stakeholders [4], [3]: diversity in customer culture and business; achieving appropriate participation of system users; lack of informal communication and diminished awareness of local working context; reduced level of trust; difficulty in managing conflict and achieving common understandings; ineffective decision-making meetings; and delays. A separate systematic review [3] found that there is currently a lack of successful approaches to overcome the problems of collaboration and social aspects of RE in distributed teams.

There are some indications that distributed work practices are improving however, and some recent studies suggest that 'distance does not have as strong of an effect on distributed communication delay and task completion as seen in past research' [3]. Damian and Zowghi even suggest that locating the development team far from the requirements engineers and end users helps to remove them from distractions, which may prove beneficial [4]. Some studies have suggested that the presence of high quality collaborative tools may outweigh the disadvantages presented by distance. For example, an RE experiment using graduate students found that groups required to interact remotely using video conferencing tools produced more creative solutions than their peers who met face-to-face [7]. A similar experiment however found that face-to-face meetings were significantly more productive [8].

Treude et al. [3] cite a study which recommended that distributed teams focus on communicating the work that has to be done; ensuring there is active project management; using direct communication channels; and employing a single development environment; and conclude participants should accept some travelling. Treude et al. also cite a separate study based on three distributed projects from RTI International [3], with similar recommendations: communications tools should be easy to learn and use; teams should choose their own communication tools; and there should be shared file storage. They also concluded that travel is not always strictly necessary, however our experiences suggest that plenary meetings were helpful and that those involved in requirements validation benefited from working in the same location during intensive analysis sessions (see Sections III and IV).

It is worth noting here that previous research on distributed RE concentrates heavily on (a) organisations that outsource tasks to a team in another country where expert skills are cheaper, or more readily available; or (b) large multi-national corporations which owns sites in multiple countries. These scenarios differ from a research consortium. Departments from within a single organisation are likely to have similar training and tools as well as communication, decision-making and reporting structures. In scenarios where some tasks have been outsourced, we tend to see a client and vendor relationship, with the client relying on outsourced expertise still able to exercise preferences in communications, tools and processes. These common structures are not available in research consortia comprising academic and industrial partners.

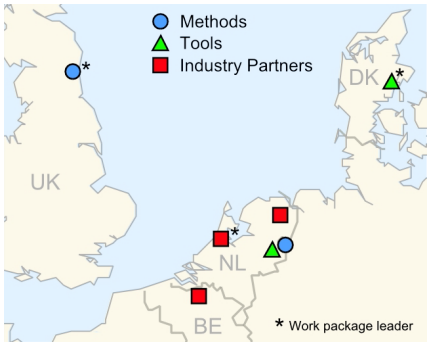

Figure 1. Map showing the distribution of DESTECS partners and their main roles in the project.

III. Case Study: The DESTECS Project

We present a case study of the DESTECS project[1]. This was a three-year project (2010–2012) that created tools and methods for designing fault-tolerant embedded systems, funded by the European Commission (STReP). Tools took the form of software to help in the design of such systems, and methods comprised guidance to help engineers using the approach. The creation of tools and methods was driven by the needs of three industrial partners, who provided case studies. The seven partners in DESTECS were spread across Europe, with work on tools, methods and industry case studies centered in different locations (see Fig. 1).

As is typical for a multi-partner European research project, the work in DESTECS was divided into work packages (WP1–5), laid down in a Description of Work (DoW) based on the proposal, which also detailed the expected contributions of each partner. Plenary 'all-hands' meetings were held every six months, with all partners presenting their progress, current state and upcoming plans. The approach adopted by DESTECS included an element of continuous process improvement, so steps were taken at various times to improve the way in which RE activities were conducted.

A. Initial ad-hoc eliciatation

While RE activities were considered during the proposal writing and therefore included in the DoW, responsibility was placed with the industry partners, who were not responsible for developing the tool or methodology. Some RE activities were also carried out by the methods work package, in the form of structured questionnaires. This led to a plethora of requirements-related material being produced in the first few months of the projects from two work packages (and points of view), but very few partners with a global oversight of the project. As found by Gürses et al. [9], due to differing levels of experience and expectations, these materials were inconsistent. Some requirements were too detailed or unrealistic, others were too vague, and some were duplicates. This initial approach is illustrated in Fig. 2 (left).

[1]Design Support and Tooling for Embedded Control Software. See http://destecs.org/ for details.

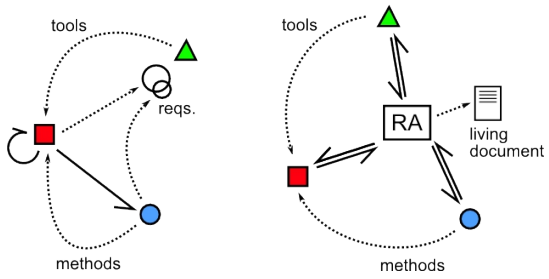

Figure 2. Diagram showing the initial, ad-hoc approach taken to requirements (left) and the introduction of the Requirements Authority (right). Dashed arrows are outputs and solid arrows are RE activities.

To rectify these problems, a 'requirements authority' (RA) group was created, which consisted of three members— one representative from each of the three work packages with a stake in the requirements process. The requirements authority was tasked with validation activities and subsequent management of all requirements for the duration of the project, as well as defining processes for proposing or amending requirements. The authority was empowered to make many RE decisions. This improved approach is illustrated in Fig. 2 (right).

The first task of the authority was to meet in person and validate the initial RE material, which involved checking for completeness, consistency, valid acceptance criteria, and duplication. Each requirement was marked as *accepted* or *rejected* by the authority. This first validation exercise was done face-to-face, which was felt to be a key part in its success. While the exercise was time-consuming, it was shortened by having a small, focused group that represented all stakeholders, meaning that discussions could be controlled and decisions reached more quickly. Gürses et al. [9] provides some insights for validation in projects larger than DESTECS.

Accepted requirements were then assigned a priority and an initial target delivery date (milestone). The 'MoSCoW' categorisation was chosen (*Must*, *Should*, *Could*, *Won't* [10]), which is relatively intuitive and easier to apply consistently. Requirements which represented a valid statement of industry need but which fell outside the project scope were designated as possible future work (using the *Won't* category).

Most requirements focus on features of the tools. However, as a research project, DESTECS has many other types of deliverables (e.g. on methodology). The RA therefore added requirements for other deliverables, to ensure they were visible to all partners and received appropriate planning attention.

B. Requirements driven development

After initial validation the RA presented the requirements to the whole consortium at the plenary meeting at the end of the first year. While this did achieve a consensus and ensured visibility of the RE process to the project as a whole, it was suboptimal. It led to time-consuming discussions over small details, which are best left to other communication media. In the later project plenary meetings, RE sessions focused on reporting progress towards completion of requirements and as a means to elaborate outstanding requirements. When input was required from the project as a whole, the most successful approach was to divide the consortium into smaller groups to consider a clearly scoped problem. For example, taking a vague or high-level requirement and discussing what concrete subrequirements would be required to fulfil it.

As an example of a requirements problem, one requirement stated that design space exploration should be supported by the DESTECS tool, but it was not clear how it would translate into actual tool features. It proved to be a question that could not be solved using email or chat. Instead it was solved at a plenary meeting, where concrete ideas could be brainstormed and commented by all partners on very quickly. In this case the project's wide geographic distribution introduced a delay in reaching a solution.

Partners were also asked to contribute towards prioritising outstanding requirements at plenary meetings. Some releases of the tool missed their deadlines, which revealed an important difference between academic and industrial partners. Industrial partners needed to schedule time in advance, and late delivery of releases would cause them to miss their allotted 'window' of time. Academic contributors were not required to allocate their time between projects in advance and could afford to be flexible. Therefore the requirements were employed as a tool for managing deliveries, with partners asked to agree on prioritising the next set of outstanding requirements. This approach was successful in marrying the different circumstances and needs of the academic and industrial partners. It was particularly effective for a decentralised consortium, as it allowed all parties to have their say and to 'buy into' development plans.

C. Requirements data and visibility

Requirements were stored throughout the life of DESTECS in an online tool bundled as part of the gforge[2] management tool. Elements stored for requirements included an identifier and descriptive name; a responsible work package; a 'MoSCoW' priority; and a target month for completion. Progress was also estimated at regular intervals (from 0% to 100%). In addition, each requirement had a list of 'updates' showing where it had changed. These messages gave rationale for changes and justified progress estimates. This provided some traceability over the history of each requirement and to actual deliverables. Our experience agrees with Herbsleb that GSD projects need to to preserve "project memory" [5].

Whilst the online requirements base was a useful, customisable tool, it was not an ideal way to present an overall picture of the project requirements or how they were progressing towards completion. To improve RE visibility, a 'living document' was created which could be automatically generated from the online data, but which presented the RE materials in a much more accessible way. For instance, coloured cells were used to indicate requirement completeness (from bright green for 100%, to white indicating no progress). Versions of this

[2]See http://gforge.org/gf/ for details.

living document were frozen and stored as progress updates every two months (coinciding with tool releases). An up-to-date copy of the document was presented to external reviewers for year-end reviews.

D. Common concepts

The DESTECS project brought together programmers, engineers, and formal methods experts. Whilst there is some overlap in these fields, there are some concepts and terms which have different meanings in each field (e.g. 'parameter' has different meanings for a programmer and for an engineer). This caused some confusion initially, with many partners unaware of the potential for misinterpretation. To tackle this problem, a 'concepts base' —a glossary of technical terms drawn from the relevant problem domains, providing definitions and meanings in a user-friendly narrative— was developed and updated throughout the project, and included in a deliverable. It served as a helpful reference for project partners and also as an accompaniment to user manuals and training materials. The development of a concepts base is also a useful tool during requirements elicitation and elaboration. If technical terms that are not yet fully understood by all groups are discovered, this suggests that more requirements elicitation is needed.

E. Communication media

Communication is clearly of crucial importance on a project like this. Because all partners are autonomous but co-dependent, it is important for political and practical reasons that everyone is informed of current issues and progress. DESTECS used the following tools:

- Teleconferences are difficult with more than a small number of people. However, teleconferences do have the advantage of speed; it is typically faster to update colleagues verbally than to write the information down.
- Instant chat was very successful in cases where more people needed to collaborate. Many work packages with distributed teams scheduled weekly 'net meetings' on an instant chat client. This is a relatively slow way to hold a discussion; there are frequent pauses whilst someone is typing a reply. However, if participants have different first languages, written discussions can be easier to follow. Also, it is possible to archive the text of the conversation and store it for future reference (e.g. as a reminder of exactly who agreed to complete specific work, and the dates that were agreed). Some previous studies of GSD where chat client usage yielded some positive results are summarised by Herbsleb [5].
- Subversion was used for managing code deployment. Many previous papers have indicated the importance of version control on a collaborative project (e.g. [2]).
- An online planning tool (gforge) was used that provided online document versioning, mailing lists, forums, blogs, and a wiki. The forums and blogs were not used by DESTECS, but a tracker tool packaged with the gforge environment was used to track requirements as well as bugs. The online wiki was used to help plan and manage face-to-face meetings collaboratively.

IV. Lessons Learned

This section lists lessons learned in DESTECS and summarises lessons learned from previous projects.

A. Lessons learned in DESTECS

The following techniques worked well in the DESTECS and have not been suggested in previous literature.

- Spend time on concepts work as soon as possible, both as part of an elicitation exercise and to ensure that finished requirements are unambiguous to all partners.
- Create a cross-work-package requirements authority, consisting of one person per stakeholding work package to make requirements decisions.
- Define a process for submitting new requirements or making changes.
- Restrict access to the requirements themselves to requirements authority members only.
- Employ a 'living' document which is updated frequently to reflect current progress and/or issues. Frequent frozen 'releases' or snapshots of this document form a useful regular update to the partners and an up-to-date frozen version may be submitted to the for external review.
- Use online tracking tools where possible. Scripts can easily be created to collect data from a requirements database and produce a formatted snapshot document minimize production effort for the authority.
- Record update messages or change history on the requirement each time a change is made to it. This provides useful traceability and accountability.
- Determine explicit expectations regarding features for the tools, frequency of deliveries and acceptability of beta versions with bugs, particularly for partners whose staff may be scheduling time in advance.
- Employ iterative, agile development techniques and use requirements to drive development. Ensure all partners collaborate in prioritising outstanding requirements, ensuring that all parties have a say in the project's direction and an awareness of future plans.
- Ensure that effort estimates are sufficient to 'guarantee' that promised features can be delivered on time.
- If necessary, create requirements for all major deliverable themes, to ensure that they remain visible and receive sufficient attention.
- Be prepared for the requirements authority to perform validation activities in the same geographical location—-face-to-face meeting time can be invaluable here for the authority to build trust.

B. Lessons learned from previous projects

DESTECS is not the only project with these characteristics to develop some lessons or guidelines based on experience. Gürses et al. also published a selection of lessons learned [9], based on the European research project TAS[3]. DESTECS

broadly corroborates observations from TAS3. In particular we agree that the RE process should drive the research and that scope and importance of RE activities must be realised from the onset of the project. The ability of partners to participate in RE activities should be assessed, appropriate methodologies selected and templates created that are tailored to the needs of the project. A common format for requirements should be defined and a collaborative environment should be employed to facilitate the communication among partners. Validation activities should be carried our regularly, checking consistency, completeness, and fulfillment of individual requirements.

Relevant lessons and guidelines can also be drawn from wider literature, including experiences of requirements engineers from systems of systems and distributed requirements engineering projects. Jiménez et al. [2], for example, present lessons learned based on a systematic review of literature on DSD. From their observations we agree that continuous process improvement is important. We identified problems in DESTECS and took measures to improve processes. In particular, we agree that "activities [should be] registered with information on pending issues, errors and people in charge" and that "provision [exists] for awareness in software development activities" [2]. We improved our approach in both these areas in response to problems with positive results. We also agree with their recommendation to use version control, and in hindsight we agree that training in RE activities should have had a higher priority early on.

V. Conclusions

Projects funded by the European Commission bring together autonomous academic and industrial partners, with wide geographic distributions, varying domains, and different experiences. Groups like this experience a particular set of problems for RE, exacerbated by the fact that proposals are often written over a short timescale, with partners selected for their technical expertise, and not necessarily for their experience with RE. In general, our experiences of this type of project have been very positive. Although a wide geographical distribution can present obvious obstacles, it is our experience that the learning and training opportunities created by collaborations between parties with very different backgrounds and skillsets more than outweigh this.

We present here a case study of a three-year European research project (STReP) called DESTECS, which supports the findings of a previous case study we know of [9]. We contribute our own lessons learned, with the view to contributing to a 'best practice' set of guidelines for future projects. We recommend that future projects carefully consider RE activities upfront, to ensure that there are no mismatched expectations. We also recommend that a small, representative group —a requirements authority (RA)— is created to validate requirements and control access to requirements data. Face-to-face contact between this group as they conduct initial validation activities is highly recommended. Larger numbers of people contribute most usefully when a specific, detailed elaboration task with a clear scope is presented for discussion (e.g. "suggest concrete sub-requirements for this high-level requirement" or "place these outstanding requirements in order of priority").

There are some factors that could affect the validity of our recommendations. First, our case study project was relatively small (7 partners). Larger projects may need to adapt our approach; e.g., a focused requirements authority may not feasible with a larger number of work packages (e.g. 10 or more). Forming a hierarchy within the requirements authority could mitigate such a problem. Second, the case study used an agile development methodology and a project using a different approach may not benefit from our observations. However the adoption of such an approach proved beneficial in our experience. Finally, while we believe our findings are generalisable to other projects in domain of computing science and engineering, conclusions drawn may not be valid in other domains. Despite these factors, we believe that our findings can contribute to the discourse on best practice for collaborative, distributed research projects.

Acknowledgments

The authors would like to thank their colleagues from the DESTECS project. The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreements no. 248134 (DESTECS) and no. 287829 (COMPASS). Additionally the work of Pierce is supported by the UK EPSRC Platform Grant on Trustworthy Ambient Systems.

References

[1] F. P. Brooks, Jr., "No silver bullet essence and accidents of software engineering," *Computer*, vol. 20, no. 4, pp. 10–19, Apr. 1987. [Online]. Available: http://dx.doi.org/10.1109/MC.1987.1663532

[2] M. Jiménez, M. Piattini, and A. Vizcaíno, "Challenges and improvements in distributed software development: A systematic review," *Advances in Software Engineering*, pp. 1–14, 2009.

[3] C. Treude, M.-A. Storey, and J. Weber, "Empirical studies on collaboration in softare development: A systematic literature review," Department of Computer Science, University of Victoria, Technical report, Dec 2009.

[4] D. E. Damian and D. Zowghi, "The impact of stakeholders' geographical distribution on managing requirements in a multi-site organization," in *Proceedings of the IEEE Joint International Conference on Requirements Engineering (RE'02)*, 2002.

[5] J. D. Herbsleb, "Global software engineering: The future of socio-technical coordination," in *2007 Future of Software Engineering*, ser. FOSE '07, 2007, pp. 188–198. [Online]. Available: http://dx.doi.org/10.1109/FOSE.2007.11

[6] J. M. Verner, O. P. Bereton, B. A. Kitchenham, and M. Turner, "Systematic literature reviews in global software development: A tertiary study," in *Proceedings of the International Conference on Evaluation and Assessment in Software Engineering (EASE) 2012*, 2012.

[7] R. Ocker, S. R. Hiltz, M. Turoff, and J. Fjermestad, "Computer support for distributed asynchronous software design teams: Experimental results on creativity and quality," in *Proceedings of the 28th Annual Hawaii International Conference on System Sciences*, ser. HICSS, 1995.

[8] H. P. Andres, "A comparison of face-to-face and virtual software development teams," *Team Performance Management: An International Journal*, vol. 8, pp. 39–48, 2002.

[9] S. Gürses, M. Seguran, and N. Zannone, "Requirements engineering within a large-scale security-oriented research project: lessons learned," *Requirements Engineering*, vol. 16, pp. 1–24, 2011.

[10] D. Clegg and R. Barker, *Case Method Fast-Track: A Rad Approach*. Boston, MA: Addison-Wesley Longman Publishing Co., Inc., 1994.