Requirements Engineering in Implementing IT Support for Scandinavian Healthcare Work Processes Using Outsourced Development in Egypt

Jens Bæk Jørgensen, Kim Henning Jönsson, Sofie Aaskov Nielsen, Christoffer Øland Skovgaard, Johan Rugager Vase
Mjølner Informatics A/S
Aarhus, Denmark
{jbj,khj,san,cos,jrv}@mjolner.dk

Mohamed Ghonemi, Rasha Adel Hassan
Crossworkers
Cairo, Egypt
{mgh,rah}@crossworkers.com

ABSTRACT

We have recently developed a new component for an existing healthcare system for Scandinavian users. The project setup included outsourced development in Egypt. In this experience report, we describe the project and the way we did requirements engineering. We identify and discuss a number of lessons learnt regarding requirements. Some of the lessons relate to the relatively long path from understanding and capturing of the needs of Scandinavian healthcare workers to providing software developers in Egypt the proper basis to do their work efficiently and with high quality. In our case, this path had four main constituents: (1) the Scandinavian healthcare domain; (2) a Scandinavian software company which was our customer; (3) the Danish software company Mjølner; (4) Mjølner's subcontractor Crossworkers in Egypt.

1 Introduction

Mjølner Informatics is a Danish software company and Crossworkers is an Egyptian-Danish company that provides outsourcing services for European companies with project teams based in Cairo, Egypt. This paper is about a project that Mjølner and Crossworkers did together. Our customer was a Scandinavian software company that delivers solutions to the healthcare sector. The project was to develop a new major component to an existing healthcare system which our customer delivers to a number of healthcare providers in different geographical regions in Scandinavia. With this organisation, requirements had to "travel a long way". Healthcare professionals in Scandinavia have ways of working that must be supported by the new component. Egyptian developers must have the right input to do their job properly.

This involves proper requirements engineering (RE) which is the focus point of this paper. Lauesen [1] classifies requirements as goal-level, domain-level, product-level and design-level requirements. This project dealt with requirements on the three latter levels and the interplay between requirements on different levels. In general, when we develop software, we are concerned with making an argument like "(A and S) imply R", where A are the assumptions about the environment, S is a specification of the software, in the form of either product- or design-level requirements, and R are domain-level requirements. Many authors have described this form of argument. Jackson made the foundational work with his explicit distinction and separation of "the machine" and the environment, see, e.g., [2]. Wieringa [3] refers to the implication above as "the system engineering argument". In the description of lessons learnt, we consider to which extent a number of key stakeholders had a need to make a system engineering argument, to which extent the way we organised this project supported this need, and how we can improve this situation the next time we do a similar project.

Our contribution is to present an experience report about a successful software development project which took advantage of outsourcing. The paper is written jointly by authors from Mjølner and Crossworkers. The involvement of both companies is crucial for us, because it has enabled us to analyse the project first-hand both from the perspective of the outsourcing consumer (Mjølner) and the outsourcing supplier (Crossworkers). The authors have had the roles of requirements engineer, project manager, architect, scrum masters (both Danish and Egyptian) and developers (both Danish and Egyptian as well).

2 Background

In this section, we describe the timeline, the organisation, the overall development process and the approach to requirements engineering.

Our customer had carried out a one-year requirements engineering process in 2017-2018 with representatives for the

ICSEW'20, May 23–29, 2020, Seoul, Republic of Korea

ACM ISBN 978-1-4503-7963-2/20/05…$15.00
https://doi.org/10.1145/3387940.3392239

future users of the new component, before our involvement in the project we describe in this paper. In March 2018, our customer decided that they wanted us to develop the new component. Our customer's RE work had resulted in a solution description containing the domain-level requirements in the form of descriptions of the primary workflows to be supported. We used the first two months mainly to acquire knowledge from our customer and to write the project foundation, describing the scope, process, assumptions, schedule, estimates, etc. for the project. In June 2018, we finished the project proposal and a contract was signed. The development project started in July 2018. We delivered the final component in May 2019, and shortly thereafter it was put into operation in several regions in Scandinavia.

The project organisation was as follows: Above project level, there was a steering committee. At project level, the Mjølner project team included a project manager, a requirement engineering team (RE team) and two Scrum teams. The RE team consisted of two requirements engineers, an architect and two of the Danish developers. Both Scrum teams had seven developers and one Scrum master. One of the Scrum teams was a Crossworkers team based in Egypt. We cooperated with a number of people from our customer, including a project manager, a product owner (PO), support teams with developers and architects, and a tester. The PO, who had a deep understanding of the domain-level requirements for the new component, was the main author of the mentioned solution description that was the foundation for this project.

The development process was based on Scrum, with the constraint that due to the contract for this project, which was fixed-price, the product backlog was given up front in the form of the solution description, and it was agreed that all items in this backlog should be delivered at an agreed deadline. In the development process, requirements engineering work and design were one sprint ahead of the development. This ensured that at each sprint transition, the design that was necessary for the implementation work of the sprint to be started had been done in the sprint that was being concluded. In the four weeks period of a sprint, there was time for 2-3 iterations on the designs, i.e., the product-level and the design-level requirements. This involved that the RE team presented a first proposal of the design, the PO sometimes had dialogue with and got feedback from user representatives, the RE team updated the design and that the PO gave his final approval.

There was a lot of ad-hoc informal communication between the project participants. However, there were also a number of scheduled meetings, both with our customer and between Mjølner and the Egyptian team. Internally, we had the usual daily Scrum meetings that were held for each of the two teams. The languages used were Danish in the Danish team and Arabic in the Egyptian team. Between Mjølner and the Egyptian team, we had a weekly Scrum status meeting (an extended version of the daily Scrum) in English. Externally, we had regular weekly meetings with participation of relevant developers, the RE team and the PO; we refer to these as the "RE/PO meetings". These meetings were crucial requirements engineering activities in the project. Every month, we had sprint transitions. This included a sprint review where we presented the work we had done in the sprint being concluded for our customer, and where user representatives also participated. Moreover, it included sprint planning where we agreed on the contents of the sprint being started. In addition to the internal and external meetings we were involved in, the PO had weekly meetings with user representatives. For contractual reasons, we did not participate in these meetings, with a few exceptions. Because we had a fixed-price contract with our customer, direct communication with the user representatives would have incurred a risk of involving us in discussions and clarifications that it was our customer's responsibility to have – and subsequently digest and pass on to us in due time. At the sprint transition meetings, again for contractual reasons, we let our customer handle the direct communication with the user representatives. This is a subject we will return to in the section on lessons learnt. As a supplement to the regular, scheduled meetings, there were meetings which revolved around features. The Egyptian team was in direct contact with our customer at many meetings and on many occasions, in particular about technical matters. However, they did not participate in the weekly RE/PO meetings. These meetings were to a high extent about the domain and the default language was Scandinavian, so our initial assessment was that it was not a viable option to have the Egyptian team participate in these meetings. Again, this is a subject we will revisit in the section on lessons learnt.

The solution description was the starting point for the RE work that Mjølner was involved in. The solution description contained descriptions of the primary workflows to be supported. In the sense of Lauesen [1], these are domain-level requirements. Considering the system engineering argument, "(A and S) imply R", the domain-level requirements constitute the R's for the new component. Some, but not many, assumptions (A's) were also included in the solution description. While writing the project proposal, the RE team had meetings with the PO, clarifying the primary workflows. This increased the RE team's understanding of the R's. Between the customer meetings, the RE team focused on understanding the primary workflows, coming from domain-level requirements to product-level requirements (features). In the beginning of the development project, the RE team worked on refining the requirements, making more detailed descriptions of the R's and identifying corresponding product-level requirements and getting started on describing design-level requirements. This resulted a breakdown of the domain-level requirements into product-level requirements, i.e., features. Many deliverables were written in a Scandinavian language. Thus, they could be used by our customer in dialogues with user representatives. In the sense of the system engineering argument, drafts of S's were produced at this stage of the project. At the weekly RE/PO meetings, the RE team continuously considered the system engineering argument, in the sense that it, of course, was the purpose to make S's that were matching the R's that were currently considered. This was an iterative process where the design-level requirements were validated by the PO who at this stage used the A's that he was aware of (many of them were not available or, at least, hard to get for us) to make a system engineering argument. For the features that were to be developed in Egypt, after approval of feature designs by the

PO (approval of S's), Initial Feature Requirements Specifications (IFRS) were drafted in by the RE team. The purpose of the IFRSs was to ensure that the features were handed over to the developers in Egypt so they could start development. They had the form of quickly produced notes in English containing domain-level, product-level and design-level requirements through text, drawings and flow descriptions. The Egyptian team was hereafter responsible for creating a Feature Description Document (FDD) based on the IFRS and further clarifications with both the PO and relevant developers from our customer, in cooperation with the RE team. IFRSs did not go through iterations; FDDs typically did. When an FDD was ready, a kick-off meeting between our customer and us was held; this applied to all features, independent of whether they were going to be developed in Egypt or in Denmark. The PO participated, and again on this occasion had the opportunity to make the system engineering argument.

3 Lessons Learnt

3.1 The Egyptian Team Should Have had More Direct Contact with the PO

It is more important for developers to do verification than validation in their daily implementation work. Validation is checking whether we are building the right product and verification is checking whether we are building the product right, i.e., according to its specification. The former, in essence, is doing the system engineering argument. By the very nature of the task, developers are regularly inspecting S's and comparing their current implementations with the S's that specify what should be implemented.

However, often questions about S's that arise naturally as implementation goes along have dependencies to the A's and the R's that are brought together via the system engineering argument. When this has happened for developers on the Danish team, questions have often been answered promptly because of physical co-location at Mjølner's office in Denmark with the Mjølner RE team. Additionally, many misunderstandings and doubts have quickly been discovered and addressed. Moreover, the Danish developers have had easier access to direct communication with the PO, in particular by participating in the RE/PO meetings when it was useful. When questions emerged in the Egyptian team, it was more difficult to get answers because of the physical distance, in spite of extensive communication via Skype and similar means.

Some of the A's were be related to how the Scandinavian societies are organised in general. Egyptians have no knowledge of this which makes it even more difficult for them to assess whether they are on the right track. On the weekly meetings between the Mjølner RE team and the PO, A's and R's were discussed and elaborated. As mentioned previously, the Egyptian team did not participate in these meetings. The reasons for this choice were that these meetings were to a high extent about the domain and they were held in Scandinavian.

A lesson learnt is that we did not have the best organisation. We believe that the entire process would have been more efficient if some members of the Egyptian team had participated directly in the regular RE/PO meetings – from some point in time. The RE team had to gather and interpret information and pass the relevant portion on to the Egyptian team with risks of delays and distortion, and with time overhead. We relied to a too high extent on written and quickly produced communication about requirements in the IFRSs, and we were not always sufficiently careful checking the understanding with the Egyptian team before they started the implementation.

Thus, our hypothesis is that it would have improved the project, if the Egyptian team had been in more direct contact with our customer. To put this hypothesis into perspective, we believe that it would not have been viable to put the Egyptian team in direct contact with the user representatives. Many users of the new component are not able to or comfortable with speaking or writing English. Hence, there would have been no common language to communicate in, and the cultural distance between Scandinavian healthcare workers and Egyptian developers would probably have been too big – and much bigger than between the PO and Egyptian developers who all are IT professionals.

3.2 The RE Team Should Have had More Direct Contact with the User Representatives

Members of the RE team have continuously been considering the system engineering argument, in the process of creating S's. These have, of course, been created to be in accordance with the considered R's, and also the A's that have been known.

The PO also had a very high need to do the system engineering argument. A main problem is that he has been very busy for long periods in the project. Because of this, there have sometimes been delays in clarifications and in approval of our S's and in conveying of relevant A's. The system engineering argument has not always been done at the right time, and this has caused rework because we have sent not finally approved S's to the Egyptian team in order to fully utilise our total development capacity and with the goal of meeting the project deadlines and keeping the project budget (a similar remark applies to the work done in Denmark; there has also been rework here for the same reason).

The user representatives for the component also had a very high need to do the system engineering argument. In this project, as described earlier, the direct communication with and getting feedback from the user representatives have been the responsibility of our customer, and there has been some inertia there which has resulted in late feedback that has caused rework in the project. If, e.g., the user representatives only do the system engineering argument when a feature has been implemented and delivered, there is obviously a high risk for a mismatch between domain-level requirements and the now implemented design-level requirements.

We believe that it would have been an improvement if the RE team had had more direct contact with the user representatives. This could have alleviated the fact that the PO was busy; it could have extended the entire project organisation's RE/PO capacity. As an

additional benefit, it would have brought the RE team closer to the domain. Consequently, Mjølner would have had a deeper and quicker understanding of R's and A's which would have enabled making system engineering arguments easier and earlier and would in particular have made it possible to serve the Egyptian development team better. In essence, this would have reduced the length of the communication path.

Unfortunately, it is not straightforward to make such an arrangement contractually, with fixed price and fixed requirements on one side, and ongoing clarifications on the other side. This was a severe drawback of the contract for this project. It might have been helpful to make an additional agreement about more RE work by Mjølner, as assistance to the PO, but with our customer still ultimately contractually responsible for getting the clarifications and for consequences of delays etc.

3.3 Concepts, Languages and Translations Should Have Been Addressed More Carefully

Obviously, there have been difficulties in the project due to different languages. The users' native language and our customer's native language is a Scandinavian language which is understood by the Danish team from Mjølner. Our Egyptian team's native language is Arabic. Communication between Danes and Egyptians has been in English. Our customer also uses English as its corporate language. Thus, we have had three languages involved in this project: Scandinavian, English and Arabic.

In general, in software development projects, it is a crucial project activity to establish a common terminology which can be difficult even if only one language is involved; see, e.g. [2]. Three languages involved obviously amplify this problem.

As the users of the component are not comfortable with speaking or writing in English, meetings between our customer and the user representatives had to be in Scandinavian. This also applied for written material exchanged between our customer and the user representatives. Moreover, the component is a business system supporting work processes in a specialised domain with lots of concepts to understand and to give proper designations.

3.4 Differences in Cultures and Societies Must be Taken into Account

Denmark and Egypt are countries with very different cultures and differently organised societies. We had fewer problems related to this than we had expected. We managed to make an organisation as one co-operating team consisting of software professionals from Denmark and Egypt, and we faced challenges and found solutions together, in a very good cooperative spirit.

Of course, there have been a number of mundane differences to consider. Examples are that the work week is Monday – Friday in Denmark while it is Sunday – Thursday in Egypt, that most Danes have three weeks summer vacation in July, that Egyptians work in the Christmas period but have reduced hours during the Ramadan period and that during Egyptian team members' visit to Denmark, halal food should be an option for the Muslim guests. Most of these differences have been managed with proper project planning.

Assumptions (A's) can be about a broad range of issues, including issues about Scandinavian welfare societies where citizens are granted free healthcare paid for by the government, and where there are laws, rules and regulations for which services should be offered to which citizens, and under which circumstances. In Egypt, healthcare is organised and provided in a different way. Because of this, Danish software professionals may have a better "feeling" for some A's than Egyptian developers when developing IT support for work processes to be carried out in Scandinavian societies.

4 Discussion and Conclusions

It may be considered if some of the problems described in the lessons could be solved with proper, well-known and well-established agile software development principles, with frequent user involvement, continuous prioritisation of requirements, embracing change, etc.

However, due to contractual conditions, it was not possible for us to be full-fledged agile, or to "grow" the new module, instead of basing it on a written-down overall requirements specification created before the development project began. These were the given circumstances that we had to accept and deal with in the best possible way. We had a fixed-price contract with our customer and our customer had fixed-price contracts with the healthcare providers. Let us review our degree of agility, based on the Agile Manifesto's four statements: (1) Individuals and interactions over processes and tools; (2) Working software over comprehensive documentation; (3) Customer collaboration over contract negotiation; (4) Responding to change over following a plan. We have been compliant or reasonably compliant with all but (4): It has been a central goal in the project, both for Mjølner and for our customer, to follow the plan and keep the number of changes low. This is due to the contracts with fixed price, fixed scope and fixed deadlines. Regarding (3), there has been good collaboration below the steering committee level. But given that we had a fixed-price contract, there have been discussions about what was within scope of the contract and what was not and thus should be seen as a change request to be paid for separately. This is only natural, and we have managed to settle everything during negotiations.

This paper can be seen as an extended retrospective. We have completed a project and with this paper, we have reflected on how we can improve. The lessons have an immediate value for us, because based on them, we will do things differently in similar projects in the future. We are already applying the lessons within the Mjølner/Crossworkers cooperation to be even more efficient together in ongoing projects.

REFERENCES

[1] S. Lauesen, Software Requirements - Styles and Techniques, Addison Wesley, 2004.

[2] M. Jackson, Software Requirements & Specifications – a lexicon of practice, principles and prejudices, Addison Wesley, 1995.

[3] R. J. Wieringa, Design Methods for Reactive Systems – Yourdon, Statemate, and the UML, Morgan Kaufmann, 2003.