Problems with Communication About Requirements in a Complex Program in a Large Organization

Agnete Røberg Horup
Mjølner Informatics A/S
Aarhus, Denmark
aho@mjolner.dk

Morten Jokumsen
Mjølner Informatics A/S
Aarhus, Denmark
mjo@mjolner.dk

Jens Bæk Jørgensen
Mjølner Informatics A/S
Aarhus, Denmark
jbj@mjolner.dk

Maja Due Kadenic
Aarhus University
Herning, Denmark
maja@btech.au.dk

Nina Wiborg Mølgaard
Mjølner Informatics A/S
Aarhus, Denmark
nwm@mjolner.dk

***Abstract*—This paper elucidates problems with communication about requirements in a complex program in a large organization. Our insights are rooted in experiences we, as industrial practitioners, have had in our extensive involvement in a specific program in which we delivered software consultancy services to an industrial client. Throughout this involvement, we encountered significant problems, which caused insufficient requirements engineering. Our efforts to communicate effectively about requirements proved to be inadequate. Although some problems could have been addressed with established requirements engineering techniques, we have not been able to convince key stakeholders to allow us to apply such techniques to the desired extent. The main contribution of this paper is five lessons about our insufficient understanding of key stakeholders' interests and about the impact both the program organization and the overall company organization had on the possibilities for and limitations of effective communication about requirements.**

Keywords—problem statement, requirements engineering, communication, stakeholders, business versus IT, cooperation

I. Introduction

Mjølner specializes in the development of tailored software solutions and provides software consultancy services to both Danish and international clients within a broad range of domains, e.g., manufacturing, utilities, media and finance. Mjølner was established in 1988 and has today approximately 250 employees.

This paper examines Mjølner's involvement in a recent program comprising several interrelated and interdependent projects for an industrial client. To safeguard confidentiality, the client remains anonymous. The client, which has thousands of employees and hundreds of thousands of customers, operates according to a subscription-based business model. Their primary revenue stream is generated through customers subscribing to various services offered by the company, entailing a monthly fee. The overarching objective of the considered program is to enhance digital channels facilitating interaction between customers and the client. These digital channels are accessible through web pages and apps.

Our purpose with this paper is to describe and analyze the program with a focus on the problems related to requirements and particularly shedding light on communication about requirements among stakeholders. The experiences and reflections may serve as beneficial insights and to gain a deeper understanding, which we, as industrial practitioners, can apply to ensure effective collaborations in future settings with similar characteristics.

Lauesen [1] classifies functional requirements as goal-level, domain-level, product-level, and design-level requirements. We have encountered problems at all four levels. Additionally, we have encountered problems with non-functional requirements.

Four of the authors of this paper are practitioners employed by Mjølner who have worked on the program with different roles (some with more than one role): user experience specialist, requirements specialist, architect, developer, project manager, product owner and Scrum master. All these roles work with requirements on different levels. The fifth author is from academia and has assisted with describing, understanding and analyzing the problems that the practitioners have encountered.

During our involvement in the program, we have continuously gathered written material that we have used in our analyses for this paper. One of the authors has maintained a diary to keep notes of main events and document various observations daily. We have also created slide decks for various meetings and presentations, and we have exchanged thousands of emails and chat messages that we have revisited, categorized and analyzed for this paper. This, together with our observations and recollections of personal experiences, is our foundation for writing this paper.

The paper is structured as follows: In Section II, we describe the anonymized client. Section III presents the

2332-6441/24/$31.00 ©2024 IEEE
DOI 10.1109/RE59067.2024.00036

program under consideration. In Section IV, we make some remarks about Mjølner's software development process and our approach to requirements engineering. Section V presents our involvement in the program. Section VI identifies and discusses the lessons learned. Related work is covered in Section VII and the conclusions are drawn in Section VIII.

II. THE CLIENT

The client company is a result of mergers of dozens of smaller companies, and its current organization is only a few years old. The company has a large workforce, consisting of thousands of employees, and follows a classical hierarchical organizational structure. The relevant upper-level management layers, crucial to the discussion in this paper, are depicted in Fig. 1.

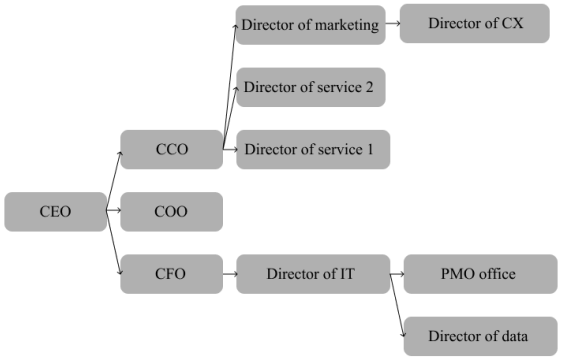

Fig. 1. Organizational diagram of the client's upper-level management layers

The top management comprises four individuals: the chief executive officer (CEO), the chief commercial officer (CCO), the chief operations officer (COO) and the chief financial officer (CFO).

The CCO has the overall responsibility for two important business divisions which provide the company's two main services to customers. The CCO also holds overall responsibility for the company's marketing, overseeing the company's director of marketing.

The company has a Customer Experience (CX) department dedicated to ensuring a positive experience for customers interacting with the company via digital channels. The director of CX reports to the director of marketing. CX is the main driver of the program under consideration in this paper. CX has the budget for the program and is responsible for the overall management.

The client company has an IT department, and this department is also a main stakeholder in the program we are considering. The director of IT reports to the company's CFO. Among other duties, the IT department is responsible for the company's data (we will elaborate this subject in Section VI.E) and they are responsible for the company's overall principles for systems and software architecture.

There is also a Project Management Office (PMO) which, among other things, keeps an overview of our client's ongoing projects and programs.

III. THE PROGRAM

The program under consideration started in early 2021. The program planned to deliver: (1) a new and improved web page, where the client could sell its services; (2) a new and improved self-service web solution where the client's existing customers could get an overview of their engagement, pay bills etc.; (3) a brand-new app providing various overviews to customers. The program organization is illustrated in Fig. 2.

Fig. 2. Diagram of the program organization

The program featured a steering committee [2], chaired by the director of CX. The committee included a representative from the IT department and representatives from the two primary business divisions (service 1 and service 2), each represented at the vice president (VP) level; for each service, the vice president reports to the director of that service. However, organizational changes caused a discontinuity in IT representation, resulting in extended periods without expertise in software and software development within the steering committee.

The program's day-to-day management was a collaborative effort between two program managers – one from the PMO and another reporting to the director of marketing. Various organizational units, such as an architecture group and a core group were established to facilitate program coordination; the core group included several members from CX.

The program comprised eight development teams, each with approximately eight members. Some teams focused primarily on front-end development, concentrating on the creation of new customer-facing software clients. Other teams were predominantly back-end, dedicated to making data from numerous source systems accessible for the new software clients and managing data exchange between the source systems and the new customer-facing software clients.

The software teams were agile teams. Immediately before the summer of 2022, most of the team members had extensive training in Scrum [3], facilitated by an external consultancy

company. Several agile coaches were part of the program to guide teams in proper agile ways of working.

The communication between the software teams and the client's business divisions was facilitated by product owners, as defined by the Scrum framework. The teams held Scrum events, including regular sprint plannings, reviews and retrospectives. Typically, a team was comprised of a Scrum master, a lead developer, a tester, and several developers. The majority of the teams had product owners from CX.

The program relied on various data deliveries from the company's IT department, e.g., data about customers and their agreements from various systems in the client's full IT landscape. The establishment of this data exchange fell outside of the program's responsibilities. Consequently, Fig. 2 does not designate a specific entity responsible for data. Our conjecture is that this way to organize the program relative to its environment was the consequence of organizational politics rather than the result of a more rational decision about the optimal program structure.

IV. REMARKS ABOUT MJØLNER'S SOFTWARE DEVELOPMENT PROCESS AND APPROACH TO REQUIREMENTS ENGINEERING

Mjølner's developers are experienced in the agile approach to software development, in which the teams work according to the principles of Scrum.

Mjølner's software professionals are also experienced in various requirements engineering techniques. One such technique is EventStorming [4], a rapid, group modelling approach that forms part of domain-driven design. This method, originally created by Alberto Brandolini in 2012, is a workshop-style technique that brings together project stakeholders, including developers and non-technical users, to map and explore complex business domains. It is popular and in widespread use in the software industry today.

Another such technique is EventModelling [5], which describes systems by illustrating how information evolves over time. It omits transient details and focuses on what is durably stored and what the user can observe at any point in time. EventModelling represents a sequence of events linked together by user interactions and their related inputs (commands) and outputs (views). The method uses wireframes to document all these system interactions across different user types.

Both EventStorming and EventModelling aim to enable business stakeholders to describe business events that occur, or they want to occur, within the envisioned software system. A key component of these methods is that they allow business stakeholders to model a specific business process using their own words. The intention is that during these sessions, business stakeholders can provide enough detail to developers to gain a comprehensive understanding of what needs to be implemented and why.

By bringing business stakeholders and software professionals into the same room, the requirements engineering techniques have the advantage of being fast, straightforward, engaging and effective. They can result in a full behavioral model that can be quickly implemented as a proof of concept and then validated. One of the greatest values of these methods is the conversations they generate. These conversations catalyze mutual understanding, giving software professionals a better understanding of business processes, and business stakeholders a better understanding of technical solutions under consideration.

More generally, for projects and programs managed by Mjølner, three project members have leading and coordinating roles: A user experience specialist with a main responsibility for requirements engineering regarding functional requirements, an architect with a main responsibility for taking non-functional requirements such as performance, security, scalability, and maintainability into consideration and a project manager with a main responsibility for ensuring delivery within the agreed time and budget. These roles work very closely with each other and with other team members, including the developers who write the software. So, in summary, in Mjølner's projects, it is the user experience specialist and the architect who, together, are the main drivers of requirements engineering.

V. MJØLNER'S INVOLVEMENT

Mjølner's involvement with the client in the program started in July 2022, when the program had been running for more than a year.

We became extensively involved from the beginning, as developers from Mjølner joined all eight development teams, see Fig. 2. The developers adapted to the ways of working that had been established previously. A timeline of the main events is illustrated in Fig. 3.

Fig. 3. Timeline of the program's main events

The program suffered from substantial problems, e.g., many program participants experienced problems with lack of overview of the "big picture", insufficient quality assurance and low efficiency in their daily work. One of the authors, a project manager, served as a product owner (one of the few product owners not from CX) for several months on one of the

software teams, directly witnessing numerous problems in this capacity. Another author, an architect with experience from multiple projects and programs for this client, also gained insights into many of these problems.

Although many stakeholders of course were already aware that the program suffered from severe problems, we had not seen explicit descriptions of the problems, and it was our assessment that they had not been identified and presented clearly for the right stakeholders. Therefore, the project manager author reported his observations about the team in which he was product owner. He focused on the team level first, because there he had first-hand, reliable information, and supplemented with conjectures about similar problems on the program level; that was done in November 2022. There were problems with overall scope, detailed requirements, estimates, cost, schedule, quality, test, staffing, organization, communication, stakeholders and risks.

Among others, the head of the program steering committee, the director of CX, was present at this presentation. Immediately after, the problems in the program received more attention. More specifically, we started joint improvement initiatives between the client and Mjølner on the program level. The setting for this was a series of meetings with several representatives from CX, one representative from the IT department, the two program managers plus senior management from Mjølner and the project manager author from Mjølner. The representative from the IT department was only present at the first two meetings and did not participate in the following meetings. The reason for senior management from Mjølner to participate was that the considered client had very high priority; Mjølner really wanted to be successful. The presence of senior management might enable solutions to organizational problems and problems about staffing on various levels, and we expected that there would be a need for that (however, we did not succeed in solving these problems, as we will elaborate in Section VI).

In December 2022, we proposed solutions to some of the identified problems. One element was to introduce explicit and classic project management to the program. Until this point in time, the Scrum roles of Scrum master and product owner were responsible for handling project management matters, yet their impact proved insufficient. To address this, we recommended adding software project managers to the program. These individuals would collaborate with the development teams, the core group, the architecture group and program management to implement improvements and get the program under control. This recommendation was based on our usual organization of programs managed by Mjølner as described in the end of Section IV; we reiterate that this program was managed by the client, not by Mjølner.

The other crucial element, and the central focus of this paper, was initiating the establishment and systematic application of requirements engineering. For this, we needed a user experience specialist and an architect, cf. Section IV, and this was the first involvement in the program of two of the authors of this paper.

We presented basic concepts and techniques as described in [1] for the client, i.e., the four levels of functional requirements - goal, domain, product and design - and the importance of early attention to non-functional requirements. We emphasized the importance of engaging the necessary stakeholders at the right time in the requirements engineering activities. We described how elicitation, specification and validation of requirements all are crucial activities. We emphasized that end-user expectations and technical feasibility must be aligned early, continuously and frequently. At the time, our perception was that we were effectively conveying important messages to our client (however, it became evident later that we were not doing so to a satisfactory extent, as we will elaborate in Section VI).

In January and February 2023, we applied our preferred requirements engineering techniques as presented in Section IV on a new feature that was to be developed. We had elicitation workshops with representatives from the business divisions responsible for service 1 and service 2 (see Fig. 1) and the development team which were going to develop the feature. The second workshop involved dedicated end-users exchanging knowledge about the domain, user and business needs, and pains and gains. Also, a shared understanding of dependencies between involved technical systems was obtained and documented.

We wrote a specification for a minimal viable product (MVP), which was the result of a tough prioritization, given the available budget and desired timeframe. We had a total list of 25-30 desired sub-features, and (only) 3-4 of these were selected for inclusion in the MVP. We validated the MVP specification with the representatives from the business divisions, and they agreed on the proposed scope. Our perception was that we were effectively communicating the intended message. Our own assessment was that we conducted valuable work and delivered a specification for an MVP, which was the result of an accomplished process with the involvement of the right stakeholders in the right activities. However, our main client stakeholder, CX, had a different opinion of the situation. Having observed our efforts, they expressed the view that it did not bring additional value. According to them, they had a comprehensive handle on what we referred to as "functional requirements". This was contradicting the message we got from many members of the development teams, when we, in March and April, had discussions with them about the status on their work with requirements.

At a difficult meeting in late March 2023, in the series of improvement meetings initiated in November 2022, we realized that important client stakeholders did not see the program under consideration as a software development program (with all the implications of this), but rather as a marketing initiative. This was a crucial realization, and we will elaborate and discuss it in more detail in Section VI.A. From this point, we were asked to stop our work with making general improvements to the work with functional requirements. CX decided that Mjølner, being software specialists, should concentrate on more technical subjects. In agreement with our client, we started to do a thorough architecture review.

The architecture review had two parts: (1) a so-called inside-out perspective and (2) a so-called outside-in perspective. The outside-in perspective starts with considering the users' perspective, and the inside-out starts with considering the software under development. Ideally, the two perspectives should of course be eventually aligned. The two perspectives and their relation can be seen as similar to the observation that in general, when we develop software, we are concerned with making an argument like "(A and S) implies

R", in which A describes assumptions about the environment, S is a specification of our software (product or design-level requirements) and R is domain-level requirements. Many authors have described this, see, e.g., Jackson [6] and Wieringa [7]. Wieringa refers to the implication above as the "software engineering argument".

Two of the authors were the main drivers of (1), and we applied the Architecture Tradeoff and Analysis Method (ATAM) from the Software Engineering Institute at Carnegie Mellon University [8], which has a high focus on non-functional requirements, in the ATAM terminology quality attributes or utility trees. The main drivers of (2) were two external consultants, not employed by Mjølner and with a close relations to CX, who were very experienced in the specialized domain of marketing technology and who have worked with numerous large companies and organizations on establishing modern marketing platforms.

The architecture review work was extensive and lasted from May 2023 well into the autumn 2023.

From early October 2023, we discussed new ways to organize the program with our client and considered a scaled agile approach. We considered introducing SAFe [9] and using concepts from SAFe as a framework for requirements engineering, e.g., discussing requirements at different levels represented as so-called epics, features and stories. Moreover, we worked on guiding which stakeholders should be involved in the processing of these during events such as product increment planning (PI planning) in the sense of SAFe and sprint planning in the sense of Scrum and SAFe. In this way, we again got the opportunity to discuss functional requirements with our client, but not as a subject in its own right as we had tried earlier, but as part of a general process improvement.

In November and December 2023, the efforts we invested in architecture review and program reorganization using SAFe were incorporated into concurrent work on a comprehensive digitalization strategy for our client, led by a prominent international consultancy company. In this way, our work was part of the foundation for more general improvements of our client's development projects and programs, but it was too late to have an effect on the program under consideration in this paper.

By the end of 2023, the considered program was seen as completed from a development and economic perspective, and it transitioned into operation, support and maintenance. In this new context, there is a substantial backlog of improvements that should be made, encompassing significant enhancements to both architecture and program organization.

VI. LESSONS LEARNED

Having described the context in which our work took place above, we will now identify and discuss lessons learned pertaining to our communication problems with stakeholders about requirements. There are five lessons; the first three lessons are related to communication problems we encountered with stakeholders within the program organization, whereas the last two lessons learned are related to stakeholders from the broader organization of the client's company.

A. Our communication was too technical too early and we were not sufficiently sensitive to the uncertainty caused by our presence

In our communication with the CX department, we did not start in the right way. Our choice of terminology, particularly in the discourse on "requirements" was inherently software-centric. We were aware that we should not discuss the technical details of software development, but the very term "requirement" hints an inside-out perspective (refer to Section V for the introduction of this term) that originates from the needs of software professionals, such as ourselves. What we failed to effectively communicate and establish was the crucial importance of early requirement definition and the necessary coordination and communication between business representatives and software professionals.

We did not adequately address, at an early stage and as a bridge to a subsequent discussion on requirements, the issues that were important ones, seen from the perspective of CX. These include personalized experience, customer convenience and digital go-to-market strategies. These topics adopt an outside-in perspective and are more fitting initial points of discussion, rather than immediately diving into detailed inquires as "please tell me about your goal-level requirements and your domain-level requirements". This type of inquiry is needed as a foundation for effective software development, but we probed it too early. There are some subtle differences about communication here that we do not fully understand; e.g., personalized experience and customer convenience are indeed fitting to categorize as goal-level requirements, but the very "requirements" angle caused us problems.

This realization came to the forefront during the architecture review, which we conducted in collaboration with cooperation partners possessing a background in marketing technology and extensive experience in communicating with organizational entities such as CX departments. Our partners' communication proved significantly more effective than ours. However, it did not explicitly address "requirements"; that would be our task, but we initiated this communication prematurely, as our stakeholders were not yet prepared for such discussions.

There are at least two reasons why we failed. The first is that we were eager to make progress and wanted to get requirements engineering established and thus requirements under control as soon as possible. The program had severe problems, and we were convinced that requirements engineering was one of the initiatives we could apply to make improvements. The second reason is that when we initiated our joint improvement initiatives between the client and Mjølner, in November 2022, it took several months before we realized that the CX department did not perceive the considered program as a software development program but as a marketing initiative, cf. the difficult meeting in March 2023 described in Section V. In a pure marketing initiative, "software requirements" are not on the agenda. It is a technical issue, not interesting for marketing specialists, and a subject that somebody else should care about.

We had assumed, and went too long before challenging this assumption, that our client's maturity regarding software development and software projects and programs was higher. We had not realized that our counterparts in discussions about requirements were much more marketing specialists than they were used to support software development. We did not

provide the necessary explanations in due time. In summary, we were operating in two distinct "worlds" without recognizing this divergence.

When we tried to establish and apply requirements engineering, we were, in retrospect, not sufficiently sensitive to the very uncertainty this attempt raised in the CX department. We were entering their territory. In their usual way of working, CX gathered requirements (again, our term rather than theirs) from the business divisions, and they passed these requirements on to the software teams; this is elaborated in Section VI.B.

We advocated a different, more iterative and direct approach. From our perspective, this change was not only evident but also necessary for improvement. The stakeholders in CX had a different perception. For them, such a shift might be seen as a threat to their organizational position, potentially resulting in a decrease in power and influence. We perceived opposition but struggled to effectively present our case. Unfortunately, we did not adequately recognize and address the fears, concerns and insecurities of our key stakeholders in a timely manner.

The lesson is that we need to allocate sufficient time to discuss our client's problems from their own perspective. While the top two layers in Lauesen's characterization [1] may be considered problem-oriented, an early and exclusive focus on requirements can be overly software-centric. The degree of being problem-oriented is context-dependent and requires a nuanced understanding. Additionally, building confidence with our main stakeholders is crucial. We should aim for them to view us not as a threat but as a valuable support, while we recognize that achieving this is challenging if we inadvertently actually do pose a threat.

B. *We did not succeed in explaining the need for direct communication between business divisions and software teams and for a program reorganization*

In our client's usual ways of working, CX was a main stakeholder in everything about requirements. They have done all communication about requirements with the business divisions on one side, and the development teams on the other side, as illustrated in Fig. 4.

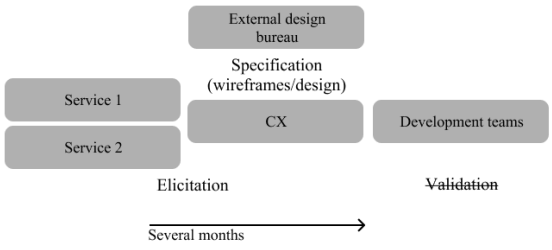

Fig. 4. Illustration of the work with requirements

Prior to our involvement, CX had held one workshop with one of the business divisions about a new large and complex feature to be developed in the program under consideration. As a result of that workshop, the CX department asked an external design bureau to design the feature, in the form of wireframes and interaction design [10]. The external design bureau did not participate in the workshop in which the first elicitation took place and had no written specifications documented. The external design bureau took some months to design what was requested from the CX department but without any communication with either the development teams or the business divisions during this time. When they were done, CX requested one of the development teams to implement the solution.

However, it turned out that the implementation of this new feature was not possible. Data to be presented on certain screens could not be made available, given the underlying IT landscape. What we observed here was a misalignment between the design-level requirements that were delivered by the external design bureau and the upper-level (goal and domain) requirements [1] that the business divisions had identified. Important information about purpose and context had not been passed on to the development team, which, thus, was not able to assess the technical feasibility of a proposed solution.

To us, this was not a surprise as it was a consequence of insufficient interaction between business stakeholders – who should gain value by the efforts of the development teams – and the actual development team. The fact that requirements passed through the CX department and the external design bureau caused distance between stakeholders who should coordinate closely, and it resulted in delays and misunderstandings.

Moreover, having an external design bureau do design without documenting the high-level requirements and the assumptions the design is based on in written form leaves gaps. Not all information of relevance for a development team can be read or understood by looking at interface design. More written specifications supplemented with oral and direct communication, workshops, etc. are also needed. A design should be created by involving business stakeholders as well as software professionals to ensure alignment between business wishes and technical feasibility.

More generally, the four levels of requirements [1] have not always been addressed at the right times and with the necessary stakeholders. Technical design-level requirements have sometimes been decided by the CX department or the design bureau, without consulting the development teams. As an example, CX instructed a development team that the web development technique of iFrames should be used for a feature, because CX had the impression that this would result in earlier delivery. In this way, the CX department was entering technical territory, and this should have been done with greater caution and a more open communication, as CX are marketing experts, not technical experts.

Furthermore, as depicted in Fig. 4, requirements were elicited and specified but not validated. This led to significant re-work as assumptions made during the initial elicitation phase were misunderstood. To avoid mistakes like these, several actions should be taken, including involvement of end-users to discover a potential mismatch between what the business thinks is needed and what the end-users actually need. In addition, validating requirements from all three perspectives - 1) end-users' needs, 2) business' needs and 3) technical feasibility – is crucial. To avoid rework, alignment should occur concurrently with the design process, ensuring the design remains current until the time it is finalized. This approach was implemented during our second workshop held in January and February 2023 about the MVP for a new feature, where we demonstrated our requirements engineering techniques, cf. Section V.

The lesson learned is that we should to a higher extent have argued the inadequacy of the current way to organize the design/requirements work and the insufficient bridging to software development, and we should clearly have argued that a program re-organization was needed.

C. *We did not argue proper domain modelling, global overview of dependencies, focus on non-functional requirements and resolution of basic architectural inadequacies sufficiently early*

Foundational principles of requirements engineering have not been followed by our client. This has ramifications for other crucial sub-disciplines in software engineering such as architecture and testing.

To illustrate this, we consider one of the development teams in the program which was working on backend services to support the new app (one of the authors was on this team). Close to the planned launch of the app, it was discovered that there were two different and contradictory views of the domain in which the client's services were used, in two systems that the app was dependent on. This was critical.

The first system owned all the customer data. In this system, important customer information was stored, which included the customers' addresses. Here, one customer could have one or more addresses (e.g., the year-long residence and a country house) while at the same time, one address could be attached to one or more customers. In short, there was a many-to-many relationship between customers and addresses. The second system relied on data from the first system. It was therefore essential that these two systems had the same view of customers and their addresses. But the second system only had a one-to-many relationship; one customer could have many addresses, but one address could have only one customer. This obstructing problem was not noticed until the beta testing of the app, when testers found this problem.

A problem like this should certainly have been identified earlier, and it would have been identified with proper requirements engineering, with emphasis on an agreed and consolidated domain model across a program - or even better, across the whole client company. With requirements engineering in place from the beginning, there would have been a foundation with a domain model. Describing not only the system to be developed (the "machine"), but also the environment is crucial, as observed by Jackson and many others decades ago [6].

The CX department was acting as a bridge between the business divisions and the development teams, cf. Fig. 4. Since CX is a marketing department, they were not aware of the importance of a domain model.

The absence of such a model was one of the major findings in the architecture review. At the same time, a global overview of dependencies between the work done by different development teams did not exist. Such an overview could be the responsibility of a program architect, but architect responsibilities in the program were, in general, too distributed, and not with clear roles and responsibilities. This was a situation which many program participants could see the consequences of in their daily work, and one of the conclusions of the architecture review was that this problem should be resolved as soon as possible.

The lesson is that we should have communicated clearly that a domain model and a global overview of dependencies is necessary, and without it, problems like the one described above will keep occurring. It should be ensured that all four requirement levels [1] are described and aligned, so that the developed technical solutions are in accordance with the needs in the real world. We should also have insisted on earlier attention to non-functional requirements and resolution of basic architectural inadequacies. It was evident for many program participants, including ourselves, that there were severe problems. We should have made these arguments for the right stakeholders, but we were not able to summon them in our client's organization. The right stakeholders are obviously not CX, who have focus on marketing; non-functional requirements and software architectures are more proper subjects for discussion primarily between software professionals, of course with the remark that it is important for all stakeholders that a system of good quality is developed.

We did make these arguments, and, we believe, with conviction, as part of the architecture analysis, cf. Section V, in which, e.g., non-functional requirements like performance, security, scalability and maintainability were put in focus and prioritized. However, it was too late to have the desired effect in the program, as the architecture review was done after we had been on the program for about one year, and close to the completion in the end of 2023.

D. *We did not argue the need for a steady input of high-level requirements from the organization to the program sufficiently well*

For an efficient program, it is crucial that there is proper input of high-level requirements from the organization to the program, with a fixed allocation of development capacity for a given period of time. This was not always the case.

As we have described earlier, the CX department provided the requirements to the development teams. CX was therefore in charge of the requirement elicitation, e.g., in the form of the workshop with one of the business divisions that we have discussed earlier. However, CX was dependent on the rest of the organization, not the least the business divisions, where the high-level requirements naturally have their origin.

There were bottlenecks in the system. Some of the development teams experienced periods with many new requirements, which all had a tight deadline. At other times there would be very few requirements to work on. In the busy periods, the development teams might ease on the software quality to reach deadlines, while in the quiet periods, the development teams would have time to fix technical debt and the like.

Our client's top management often communicate to the client organization that the company is very busy and must act fast to preserve and expand its market position – this happens, e.g., at quarterly so-called town hall meetings for all employees on Teams. The overall recognition is that digitalization should happen now and fast, otherwise, our client's competitors will do it earlier and, ultimately, win the market.

For this to be operationalized efficiently, a company organization must be in place that ensures that this very high-level, goal-level requirement, results in well-agreed so-called epics, features and stories to work on during any given sprint planning, where the backlog for the next sprint is defined, for

any development team in the considered program (and other programs and projects as well).

This has not always been the case. On many occasions, there was a lack of well-refined requirements seen from a developer's point of view. And there were disagreements between the product owners from CX and the developers on the development team. From what we have experienced, and as described previously, the CX department would think that the requirements provided for the development teams were sufficiently refined and ready for development, while the development teams would disagree.

The lesson learned is that there is a need to restructure the client's organization in a way that makes it possible to maintain a steady input of high-level requirements, and to foresee the requirements before a potential tight deadline shows up. The development teams need time to analyze and investigate how problems are best solved. We have, however, not been good enough at communicating this.

E. *We did not argue overall company reorganization and tighter connection between cooperating units and programs sufficiently well*

The program considered in this paper (in this section consistently referred to as "our program", as we also discuss other programs here) was not the only program our client was running. Several programs were active as part of a consolidation of a fragmented IT landscape, which was a natural consequence of all the company mergers.

Our program had many direct dependencies to other programs. Without deliveries from these programs, our program would not be able to be a full success. Particularly, two of these other programs were essential: (1) development of a system that would handle all the customers and their products and subscriptions and (2) development of a system that would hold information about existing legacy systems in which a customer has records. These programs were seen as IT programs and were the responsibility of the IT organization, see Fig. 1. Without these systems, our program would not have access to the data needed to fulfil the requirements for the digital channels.

Ideally, these programs would be aligned and be in close contact with each other. But in this organization, each program would be steered according to deadlines defined by the business divisions – with low or no alignment between the different programs. This approach failed multiple times when dependent programs missed their deadlines or had higher prioritized tasks and goals. The effect on our program was that multiple temporary solutions were developed to meet the deadlines. It also had a significant impact on the ability to test the systems. Test of the features developed by the development team required extensive coordination between the different programs. The complexity of these tasks was, in our assessment, not fully understood by key stakeholders within our client.

One might anticipate that problems of this nature would be addressed and managed by the steering committee in charge of coordination. However, in our program, that was not the case. While the problem was acknowledged, the steering committee considered it beyond their influence and did not, as we saw it, perceive it as part of their responsibility. We can only speculate about the reason, but it might be that they were already too busy with their own responsibilities and took the position that higher-level coordination should be dealt with by higher-level management in the company organization. As previously noted, this program was identified as a marketing initiative rather than an IT project. The lowest-ranked employee to tie marketing and IT together is the client's CEO, cf. Fig. 1. To us, this seems to be a problematic organization.

Unfortunately, we were unable to effectively communicate these problems to the right stakeholders, despite the fact that the problems regularly impeded the progress of our development teams. Internally, we frequently discussed the issue, but we struggled to identify the right channels for expressing our concerns and observations. At this point in time, the trust between our client and us had already been challenged due to misunderstandings. We never recovered from the fact that we did not succeed in our communication about the need for a significantly improved approach to requirements engineering.

The lesson learned is that we did not argue overall company reorganization and tighter connection between cooperating units and programs sufficiently well. Alternatively, we should have more readily accepted these organizational characteristics as given and attempted to navigate around them, rather than trying and hoping to directly contribute to their resolution.

VII. RELATED WORK

In this section, we explore the literature on communication challenges related to discrepancies in perceptions among project stakeholders and roles, as well as the human, cognitive and organizational aspects in requirements engineering relevant to the lessons learned in section VI.

Our objective is to comprehensively enrich our understanding of this critical domain and contextualize this study within the broader landscape of existing and future research in the field. Communication is the dynamic process of exchanging information, ideas and emotions between individuals or entities, which fosters understanding and connection [11]. It is fundamental in managing relationships with project stakeholders, as it facilitates the exchange of information, expectations and feedback [12]. Yet, communication gaps between organizational roles within development affect requirements [13].

Requirements engineering is recognized as the foundation of software development [14] encompassing a systemic and integrated process of analysis, elicitation, specification, validation, assessment, negotiation, prioritization and evolution. Obtaining high-quality requirements is critical, and difficult, with numerous challenges associated with these efforts [15,16]. Common challenges encompass the selection of elicitation techniques, traceability and prioritization of requirements, requirement change management, and communication gaps with customers and other stakeholders [16]. Notably, Ambreen, Ikram, Usman and Niazi [14] identify requirements communication among emerging topics within the empirical research of requirements engineering. Communication problems and barriers in requirements engineering are also observed outside the IT industry [17].

The intense communication effort involves a broad range of stakeholders with diverse skills, backgrounds and statuses in order to surpass the semantic gap that these stakeholders (such as users and developers) inevitably foster [18].

Therefore, achieving effective communication is consistently challenging and remains a recurring problem in the elicitation of requirements. A primary cause of communication problems is that the nature of system design and development is inherently a behavioral process. Human and organizational elements significantly influence the design in this context [18]. Communication problems hinder the establishment of shared understanding among stakeholders, such as users and developers. Poor communication may arise from challenges in articulation, involving the ability to express information effectively, as well as from misunderstandings characterized by divergent interpretations of the same piece of information, and conflicts [18].

Kasauli, Knauss, Horkoff, Liebel and Neto [19] identify six areas of challenges through a multiple-case study. These areas encompass the following: 1) to build and maintain shared understanding of customer value, 2) to support change and evolution, 3) to build and maintain shared understanding about the system, 4) to make a suitable representation of requirements knowledge, 5) to take process aspects into account and 6) to consider organizational aspects. The authors emphasize that regardless of the overall development methodology (plan-driven or agile implementation), similar challenges persist in all companies. Indirectly, some of these challenges can be related to communication problems according to Coughlan and Macredie [18]. Particularly, the ability to establish a shared understanding of value as well as of the system among stakeholders.

The literature reviewed here underscores the widespread presence of communication challenges within requirements engineering. Our empirical data and experiences are supported by existing literature, affirming that communication problems are indeed a well-recognized aspect of this domain. By elucidating how these insights contribute to a more comprehensive understanding of this critical area, our study is positioned within the broader landscape of existing and future research in the field.

In this context, elicitation of requirements is also associated with the perspectives of stakeholders [20], which include customers, requirements engineers, analysts, domain experts and developers [21]. Gathering requirements is a knowledge-intensive task, wherein stakeholders require diverse information for understanding or negotiating the requirements and need tools to support the information overload.

Burnay, Jureta and Faulkner [22] emphasize the importance of conducting interviews with stakeholders during the requirements elicitation phase. Direct communication with stakeholders provides invaluable information through verbal and nonverbal communication. Misunderstanding stakeholders can result in the specification of a wrong system or in lack of compliance. The diversity among stakeholders entails various expectations and perceptions of the systems-to-be. Yet, different assumptions are not the problem itself; it is when these assumptions remain implicit. The authors conclude that stakeholders tend to share information about what they want and their expectations to the system, but the engineers should not expect the stakeholders to know what information on requirements is relevant for designing the system-to-be.

While literature emphasizes the importance of thinking about stakeholders, our empirical evidence and experiences deviate. In the light of the knowledge gained through the literature, we acknowledge that we did not sufficiently and explicitly consider the project stakeholder management aspect. This realization opens avenues for further research, particularly in empirical studies, to explore and understand the dynamics of project stakeholder management within requirements engineering more comprehensively.

In a literature review, Davey and Parker [23] identify several challenges that are related to communication. The human aspects of requirements engineering pose challenges to straightforward communication between consultants and clients. Cognitive limitations inherent in individuals hinder effective communication. Additionally, diverse cultural and background factors contribute to the absence of a common language, e.g., technical professionals may struggle to grasp business concepts, while business professionals may face difficulty understanding IT concepts.

Human language often proves inadequate for describing technological solutions, e.g., numerous terms commonly employed in the real world, such as 'user friendliness' and 'reliability,' lack precise technical definitions. Requirements undergo changes as the project progresses and clients gain insights into the possibilities as the project unfolds [23]. Given the inherently dynamic nature of business, requirements are subject to modifications throughout the project's lifespan. Additionally, individuals may alter their preferences and perspectives on what they initially desired. The client may not be able to articulate what the business needs, where certain requirements remain tacit, meaning they are understood by the client but not explicitly stated. Some clients may be unwilling to assist with the project, where a client representative may have conflicting interests with other project members or the project's goals. Additionally, clients might perceive the new system as a component of power struggles within the organization [23].

With this knowledge in mind, our observations underscore the persistent challenge of cognitive limitations and language gaps between business and IT professionals. This dynamic demands attention from organizations, urging them to find common ground to bridge communication deficits between these two domains. Moving forward, future research should investigate deeper into the cognitive barriers hindering effective communication between business and IT professionals in requirements engineering. Additionally, exploring the role of mediators who possess a comprehensive understanding of both technical and business aspects could offer valuable insights into enhancing communication and collaboration within requirements engineering processes.

In summary, our review highlights persistent communication challenges in requirements engineering. It underscores the need for effective communication strategies, particularly in areas like shared understanding, system representation and organizational aspects. Requirements engineering emerges as a knowledge-intensive process shaped by stakeholder perspectives, which highlights the human and cognitive factors that contribute to communication challenges.

VIII. CONCLUSIONS

Communication between IT professionals and other stakeholders is a recognized and complex challenge. We continuously attempt to improve in this area, as illustrated in the successful collaboration and communication detailed in

[24]. However, our current study reveals a contrasting scenario within the program under consideration in this paper, in which we have encountered unresolved problems in communication and collaboration. Our understanding of our stakeholders and our client's organization was inadequate. It is noteworthy that this understanding has evolved throughout our involvement in the program and was not fully developed in the early stages of our involvement.

The lessons learned discussed in Sections VI.A, VI.B and VI.C are within the context of the considered program, whereas the lessons of Sections VI.D and VI.E provide insights related to the organizational structure of our client.

During our involvement in the program, and prior to conducting the analysis for this paper, we were not fully aware of this distinction between program-level problems and problems rooted in the organization of our client. If we had been, our focus and efforts might have been prioritized differently. It is important to note that requesting organizational changes often exceeds the scope of influence of a software team or a software company, even with the software company represented by senior management, which was the case for us. In retrospect, perhaps we should have more readily accepted these organizational characteristics as discussed in the end of Section VI.E. Our efforts would probably have been better concentrated on addressing program-level problems. It is important to emphasize that Mjølner is not a management consultancy company; it is a software company. Clients seeking advice on organizational changes, typically coupled with strategy, are better served by a management consultancy company.

As outlined in Section V, during the timeframe under consideration, our client engaged with a prominent international management consultancy company to develop a new digitalization strategy. While we were somewhat involved in this initiative, we should have actively sought greater influence to contribute towards shaping a new and improved company organization.

Additionally, our approach to the problems with the program organization (Section VI.B) and the overall company organization should have been different. Specifically, in terms of the program organization, we failed to explicitly communicate to our client the distance between the business divisions and the development teams, and that having the CX department between them was a fundamental problem. Similarly, in the context of overall company reorganization, we held back from clearly and unambiguously stating the problems we saw. This reluctance was rooted in our concern about how such feedback might be received, fearing that such communication could worsen our situation within the program.

The communication problems we encountered were in fact with different stakeholders and we had an awareness of the presence of the various stakeholders, which is evident throughout our reflections in the five lessons learned. Stakeholders differ in terms of their power, influence and interests [25] and they can impact the execution or termination of a project [26,27]. Nonetheless, our stakeholder awareness was not explicated sufficiently through a structured stakeholder mapping process, such as positioning the stakeholders within a 2x2 matrix based on their influencing ability (the capacity to collaborate or pose a threat to the program) [28,29,30]. This would have given us the opportunity to align the stakeholder landscape and management of those by adapting our communication approach about requirements better with respect to the stakeholders.

Our communication approach and how we met the client and addressed requirements did not properly take into account the client's digital maturity level [31], as discussed in Section VI.A. Indirectly, we assumed a higher digital maturity level of our client when considering the ambitious scope of the program. However, throughout the course of the program, we realized that our client was at a lower digital maturity level, while we, in fact, communicated about requirements fitted for a higher digital maturity level. Inevitably, this left a gap that fostered misalignment and misunderstanding about requirements in our communication approach.

This paper serves as an extensive retrospective analysis. By reflecting on the program under consideration, we have identified areas for improvement. The lessons learned hold immediate value for us and will influence our approach in future projects and programs. We are actively implementing these insights with other clients. A more general (and obvious) lesson is that it is the difficult projects that you can learn the most from. The paper [32] is about a project, also difficult, we did at Mjølner about 10 years ago. The writing of [32] and the lessons identified there helped us to improve our ways of working with requirements which we have benefited from since.

In essence, this paper functions as a Problem Statement, inherently speculative. While the outcomes of different actions in the program remain unknown, we believe that prioritizing communication on requirements and the importance of proper requirements engineering, as highlighted in the lessons, would likely have alleviated some of the problems we have encountered.

A final note: This paper focusses on problems in the program under discussion. In spite of these problems, the program did complete by the end of 2023, from a development and economic perspective; workarounds were found, and useful features have been delivered and serve a purpose for our client's customers.

References

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

[2] PRINCE2 – Managing Successful Projects with PRINCE2, Axelos, 2017.

[3] K. Schwaber, J. Sutherland, "The Scrum Guide – The Definitive Guide to Scrum: The Rules of the Game," https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf.

[4] P. Rayner, The Eventstorming Handbook – Unlocking Creativity, Collaboration, and Communication for Your Teams, leanpub.com, 2022.

[5] Event Modeling: https://eventmodeling.org/

[6] M. Jackson, Problem Frames – Analyzing and Structuring Software Development Problems, Addison Wesley, 2001.

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

[8] P. Clements, R. Kazman, M. Klein, Evaluating Software Architectures – Methods and Case Studies, Addison Wesley, 2002.

[9] SAFe – Scaled Agile Framework https://scaledagileframework.com/

[10] H. Sharp, Y. Rogers, J. Preece, Interaction Design, John Wiley & Sons, 2007.

[11] J. Keyton, Communication and Organizational Culture: A Key to Understanding Work Experiences. Sage Publications, 2010.

[12] J.T. Karlsen, "Project Stakeholder Management," Engineering Management Journal 14.4 (2002): 19-24.

[13] E. Bjarnason, K. Wnuk, B. Regnell. "Requirements are Slipping Through the Gaps—A Case Study on Causes & Effects of Communication Gaps in Large-scale Software Development,", 19th IEEE International Requirements Engineering Conference (RE11), Trento, Italy,2011

[14] T. Ambreen, I. Naveed, U. Muhammad, N. Mahmood, "Empirical Research in Requirements Engineering: Trends and Opportunities," Requirements Engineering 23, no. 1 (2018/03/01 2018): 63-95.

[15] A. van Lamsweerde, "Requirements Engineering in the Year 00: A Research Perspective," Proceedings of the 22nd international conference on Software engineering, Limerick, Ireland, Association for Computing Machinery, 2000.

[16] T. Shah, S.V. Patel. "A Review of Requirement Engineering Issues and Challenges in Various Software Development Methods," International Journal of Computer Applications 99, no. 15 (2014).

[17] G. Liebel, M. Tichy, E. Knauss, O. Ljungkrantz, G. Stieglbauer. "Organisation and Communication Problems in Automotive Requirements Engineering," Requirements Engineering 23 (2018).

[18] J. Coughlan, R.D. Macredie. "Effective Communication in Requirements Elicitation: A Comparison of Methodologies," Requirements Engineering 7, no. 2 (2002/06/01, 2002).

[19] R. Kasauli, E. Knauss, J. Horkoff, G. Liebel, F. G. de Oliveira Neto, "Requirements Engineering Challenges and Practices in Large-Scale Agile System Development," Journal of Systems and Software 172 (2021/02/01/, 2021): 110851.

[20] N. Ali, R. Lai. "Requirements Engineering in Global Software Development: A Survey Study from the Perspectives of Stakeholders," J. Softw. 13, no. 10 (2018).

[21] W. Maalej, Z. Kurtanović, A. Felfernig. "What Stakeholders Need to Know About Requirements," 4th IEEE International Workshop on Empirical Requirements Engineering (EmpiRE) at 22nd IEEE International Requirements Engineering Conference (RE14), Karlskrona, Sweden, 2014.

[22] C. Burnay, I. J. Jureta, S. Faulkner. "What Stakeholders Will or Will Not Say: A Theoretical and Empirical Study of Topic Importance in Requirements Engineering Elicitation Interviews," Information Systems 46 (2014/12/01/ 2014).

[23] B. Davey, K.R Parker. "Requirements Elicitation Problems: A Literature Analysis," Issues in Informing Science and Information Technology 12 (2015).

[24] J.B. Jørgensen, H.L. Christensen, S.T. Hansen, B.B. Nyeng, “Effective communication about software in a traditional industrial company,” 44th International Conference on Software Engineering (ICSE2022), 5th International Workshop on Software-Intensive Business, Pittsburg, Pennsylvania, USA. IEEE, 2022.

[25] PMI. To the Project Management Body of Knowledge (Pmbok® Guide)–Fifth Edition, 2013.

[26] P. Eskerod, M. Huemann, C. Ringhofer. "Stakeholder Inclusiveness: Enriching Project Management with General Stakeholder Theory," Project Management Journal 46, no. 6 (2015).

[27] P. Eskerod, A.L. Jepsen, Project Stakeholder Management, Routledge, 2016.

[28] R.E. Freeman, Strategic Management: A Stakeholder Approach, Cambridge University Press, 2010.

[29] M.J. Polonsky, D. Scott. "An Empirical Examination of the Stakeholder Strategy Matrix," European Journal of Marketing 39, no. 9/10 (2005).

[30] G.T. Savage, T.W. Nix, C.J. Whitehead, J.D. Blair, "Strategies for Assessing and Managing Organizational Stakeholders," Academy of Management Perspectives 5, no. 2 (1991).

[31] G. C. Kane, D. Palmer, A. N. Phillips, D. Kiron, and N. Buckley, “Achieving Digital Maturity” MIT Sloan Management Review and Deloitte University Press, July 2017.

[32] M. Holmegaard, J.B. Jørgensen, M.S. Loft, M.S. Stissing, "Requirements Problems in the Development of a New User Interface for Healthcare Equipment," 23rd IEEE International Requirements Engineering Conference (RE15), Ottawa, Ontario, Canada, 2015.