Customer Involvement in Requirements Management: Lessons from Mass Market Software Development

Jaap Kabbedijk, Sjaak Brinkkemper, Slinger Jansen
Utrecht University
Department of Information and Computing Sciences
Padualaan 14, 3584 CH, Utrecht
{J.Kabbedijk, S.Brinkkemper, S.Jansen}@cs.uu.nl

Bas van der Veldt
AFAS ERP Software BV
Philipsstraat 9, 3830 AJ, Leusden
B.vdVeldt@afas.nl

Abstract

Product software vendors are regularly challenged with identifying and selecting the requirements of upcoming product releases. Companies struggle to involve their customers and have a hard time selecting the right requirements from the enormous number of candidate requirements. This paper presents a practical approach to overcome these problems by including customers in the requirements management process through different types of involvement. The presented case illustrates how software requirements are gathered from customers, by involving them on three different levels. Based on the industrial experiences we present six lessons from customer involvement in large scale requirements management.

1 Introduction: Customer involvement

Although customer involvement has been recognized as a major contributing factor to company success by most companies [11],in reality, the voice of the customer is frequently muted [15]. Organizing, selecting and prioritizing candidate requirements for large-scale commercial software products is a well known problem [14, 8]. This problem has been addressed and several solutions are offered for supporting the process of organizing, selecting and prioritizing the requirements for subsequent releases. Two practical solutions are (1) a tool that uses the Analytic Hierarchy Process (AHP) [6] and (2) a tool that applies linguistic engineering [8] to categorize and merge large amounts of requirements based on keywords. The results provided by these solutions are promising, but are still immature. There is a need for more adequate methods involving customers in the requirements management process, dealing with large numbers of requirements that result from this involvement and smoothly integrating the requirements with commercial software development [6].

This paper presents the results from an industrial study at AFAS ERP Software BV (referred to from now on as ErpComp). ErpComp is a software developer of ERP products, aimed at the Dutch and Belgian small to medium sized enterprise (SME) market. ErpComp has more than 10.000 customers, spread over fifty different market sectors. Between 15 and 20 thousand requirements for their software products are handled per year by ErpComp. The study was performed by the first three authors as scientific observers and by the last author, CEO of ErpComp, as an active participant providing industrial background and rationale for process implementation.

The reference framework for software product management by van de Weerd et al. [16] is used as the foundation for this work. The requirements management phase consists of the gathering, identification and organizing of the requirements for future software releases. The importance of requirements management is widely recognized by product software companies [2]. The method provided in this paper focuses on the steps of requirements gathering [16].

The vision of customer involvement within ErpComp takes on the challenge of unearthing how customer requirements are included in the requirements management processes of software vendors. The research method used is a case study, consisting of interviews and the studying of artefacts that were created as a result of Customer Participation Sessions (CPSs). The description of the role of moderator is based on the results of interviews with the moderator of the CPSs and with seperate product managers, who attend the CPSs and incorporate the results in their release planning processes.

The paper continues with a discussion on different methods that can be applied to involve customers (Section 2). Furthermore, the paper discusses the reduction of the number of development requests to a manageable number of requirements (Section 3), followed by the lessons learned (Section 4).

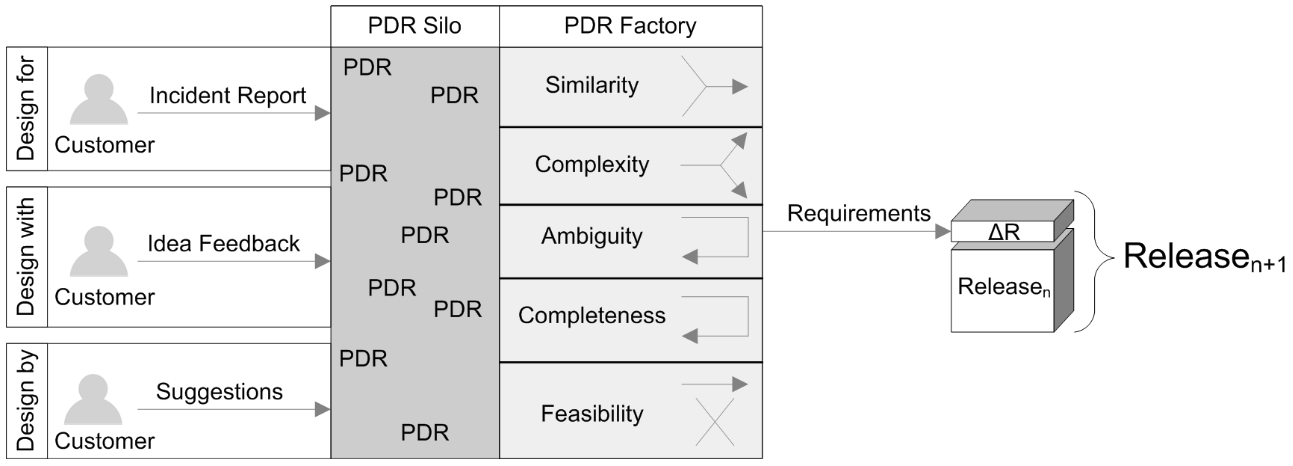

Figure 1: Customer Involvement Factory

2 Customer Involvement Methods

Three types of customer involvement are identified by Kaulio [7], where each involvement type is more intensive than its predecessor. These processes are arranged in a Customer Involvement Factory (fig. 1) with the involvement types at the left. The different types of analysis are discussed further in section 3. On the right side of the figure, the selected PDRs are formed into a release definition by the product manager, to be transformed into additional functionality to the existing product. The involvement types are:

- **Design for Customers:** A product development approach where products are designed on behalf of the customers. Data on users, general theories and models of customer behaviour are used as a knowledge base for the design. This approach often includes specific studies of customers, such as interviews and incident reports.
- **Design with Customers:** A product development approach focusing on the customer, utilizing needs and requirements similar to a 'design for' approach. In addition displays of alternative solutions are included. This enables the customers to react to different proposed design solutions (ideas).
- **Design by Customers:** A product development approach where customers are actively involved in the design of their own product by, for example providing their own suggestions or describing how they would design a certain part of the system.

Within ERPComp customers are involved particularly intensively during the specification phase of the software design process [12]. From each customer involvement method different types of feedback arise which all result in possible product development requests (PDRs), clearly defined tasks that can be assigned to a person or a team. Each PDR is annotated with a priority and a detailed description. Based on the annotated information, PDRs are organized and prioritized by a product manager. One of the main responsibilities of product managers is to translate these PDRs into usable requirements [8]. The usable requirements are needed to form a release definition that serves as a blueprint for the additional functionality that is added to the current software. The difference in functionality is referred to as ΔR. This ΔR, together with the current release (R_n), makes the future release (R_{n+1}). The prioritization of PDRs by product managers is by the most part informal. PDRs are analyzed in several ways (see section 3), but the majority of PDRs are observed briefly by the product manager when the PDR comes in. This ad hoc analysis has to be done considering the large number of PDRs ErpComp has to process every release.

ErpComp operates in a market of book keeping, payroll and warehousing products, with well established functionality and no major feature innovations. Therefore, it is relatively easy for the product managers to become experts on the market they operate in, the product and its customers. The success of the fast analysis conducted by the product managers on the incoming PDRs is due to their knowledge. If the product managers had insufficient knowledge on the relevant domain ErpComp operates in, they could not conduct the analysis in the fast way that is required for the large scale requirements management ErpComp has to deal with.

The types of involvement are mapped to the specific forms of involvement that occur within ErpComp, as can be seen in the different types of input in figure 1. Besides the three types of customer involvement, more involvement can be seen within ErpComp in other phases of the design process. During the prototyping phase for instance, beta testing is performed in a special usability laboratory within Erp-

Comp, to determine if the product does what it is designed to do in the customer environment [10]. Since the feedback from this type of involvement is received after the requirement management process, it is beyond the scope of this paper and will not be discussed. The feedback that evolves from the usability sessions are filed as PDRs and will be handled as such during a future release cycle. By putting these PDRs through the same scrutiny as other PDRs, they undergo similar profit-loss analysis as to that which all other suggestions for future releases do.

2.1 Involvement through Incident Reports

The first instance where customers are involved in the gathering of requirements for new software releases occurs when they contact the customer support department. This happens when a customer has a problem concerning a software release. An Incident Report is generated immediately when the contacts takes place, containing the problem or complaint of the customer, along with the priority and the type of incident. Of these incidents approximately 85% are complaints or misunderstandings that can be taken care of by referring to the manual or providing a short explanation. Approximately 15%, however, are defects or shortcomings in the software. The 15% is transformed into PDRs to become future requirements.

2.2 Involvement through Idea Feedback

An alternative and more personal level of support is offered to customers through a Customer Participation Session. Once a year all customers, regardless of their contract size with ErpComp, are invited per market sector or software product type to come to the in-house theater in the ErpComp office complex for the CPSs (figure 2).

Figure 2: Customer Participation Session

During the sessions possible requirements for the future release of a specific software are presented. The customers are then asked to prioritize these candidate requirements. An idea is then briefly outlined/explained by a product manager, after which the customers are asked to submit their opinion through an electronic voting device. Electronic voting enables ErpComp to see the results of the vote immediately and decide on the spot whether an idea should be transformed into a PDR or rejected. This is done because they want *fast feedback* on the opinion of their customers regarding certain ideas. The ideas presented consist of possible requirements ErpComp thinks will enlarge their market. Another advantage of electronic voting is that participants are not influenced by the opinion of other customers. It is important that every person forms their own opinion based on their own experience and insight. Enabling customers to observe each others opinions increases the chance their decisions will be influenced by group (or peer) pressure, thereby polluting the social process [1].

2.3 Involvement through Suggestions

Besides voting on existing ideas, customers are also asked to provide their own suggestions in the theatre during the CPSs. This part of the CPS is bound to a strict protocol. A customer in the first row is asked to name one feature that needs to be changed or implemented. After he explains his suggestion, the product manager gives a response to the customer about his suggestion. Due to legal reasons or scheduled upcoming releases it is possible that certain suggestions can not be implemented. If the suggestion is a possible PDR, the audience is asked whether they agree with the suggestion by raising their hands. If the resulting count is in the majority, a customers need is identified, and the suggestion is instantly transofrmed into a PDR by the product manager. If only a few agree, the suggestion is rejected. This rejection is done on the spot. As a result the customer will know the suggestion is not widely accepted and not feel inclined to including the request in future releases of the software.

Since ErpComp operates in a stable market where no large innovations are needed, the danger of rejecting possible innovative suggestions that are shared by a restricted number of customers is small. When the voting is finished, the next customer can name his suggestion and the process is repeated, until everyone has had the chance to suggest one requirement. After this, the procedure starts again with the first customer for his possible second suggestion and continues until no new suggestions come up. Note, that the deliberate use of hand-voting in this type of customer involvement utilizes the social pressure discussed above, so some suggestions can be rejected on the spot. This is because the electronic voting is used on ideas that come from ErpComp itself and these ideas are always in line with the strategy of the company. These ideas are not succeptible to rejection through the mechanism of peer pressure.

The theatre-setting has several positive effects. First of all one is able to distinguish which requirements are supported by the majority or minority, or even single customers. Since ErpComp is a product software vendor and offers no customizations, they only want PDRs that are shared by the majority of the customers. The way of structuring CPSs as described above limits the resulting PDRs to relevant suggestions only. Secondly, the theatre enables a customer to see the underlying reason for his suggestion being rejected. He can immediately see that it is not rejected because of ignorance of the company, but because of the fact that no one else shares his opinion.

2.4 Role of the moderator

The session moderators play an important role in the involvement of the customers during the CPSs. In this case, the moderators are the CEO and the appropriate product manager. Their task is to guide the sessions and make sure the sessions go according to protocol. Besides the guiding and direction of the session, the moderator has another role as well. Customers are often only willing to bring forth their suggestions if they feel that they will be taken seriously. It is the task of the company, more specifically that of the moderator, to assure the customers in this regard. ErpComp has built its in-house theatre with the primary purpose of improving the communication they have with their customers in addition to hosting the CPSs and knowledge sessions. These sessions are held to show the customers what ErpComp has done with their previous input and further involve the customer in the software design process. This is due to the level of personal interaction the moderator has with the customers and their suggestions.

3 From PDR to requirement

	Incident Reports	PDRs
2005	64,541	15,411
2006	62,981	12,913
2007	56,515	15,346
2008	68,570	17,904

Table 1: Number of incident reports and PDRs per year.

Since ErpComp has over 10,000 customers, the number of PDRs generated through incident reports, idea feedback and suggestions is significantly large. In 2008 ErpCom managed 68,570 incident reports and 17,904 PDRs for their software product (table 1). According to Regnell et al. [14] this can be characterized as very large-scale requirements management. These numbers require the three product managers within ErpComp to organize between 25 and 30 PDRs per day, each. All PDRs need to be reduced to a manageable number of requirements for the release definition.

It is a daunting task to reduce the large amount of information to a more appropriate number of requirements [4, 5]. Based on the quality gateway Natt och Dag et al. [9] described, analysis by the product manager within ErpComp takes place on three different levels. These types of analysis are extended by two additional levels, namely *complexity* and *feasibility* analysis (figure 1).

- **Similarity Analysis:** The similarity analysis method is performed in order to find requirements that may be *merged*, *grouped* or *linked*. For example, a requirement of hierarchical budget schemes in bookkeeping and of arbitrary groups in budget schemes may be merged or simply grouped together to make sure they are handled simultaneously during development.
- **Complexity Analysis:** The complexity analysis method is performed to determine whether descriptions of PDRs actually describe one requirement or a requirement or a function so complex that the PDR has to be *split* into multiple new PDRs.
- **Ambiguity Analysis:** The ambiguity analysis method is performed to determine whether the description of a PDR is clear enough to be transformed into a requirement. If the description is ambiguous or unclear, the PDR is *sent back* to the issuer. This issuer has the task contacting the customer who initiated the PDR for clarification and removing this ambiguity from the PDR.
- **Completeness Analysis:** The completeness analysis method is performed to ensure that enough information is included in the requirements to enable further refinement, such as prioritization, effort estimation and further requirements derivation. When the description of the PDR is incomplete, the PDR is *sent back* to the issuer, similar to as in the ambiguity analysis.
- **Feasibility:** Feasibility analysis is performed last and is not done for all PDRs due to this method being quite time intensive. PDRs are divided into three different classes. The first are PDRs that will not become requirements because they are outside the market scope of the company. The out-of-scope PDRs are easily identified by a product manager and are *rejected* immediately. The second class of PDRs consist of PDRs that will become requirements for certain. PDRs placed in this class are no harder to identify and are likewise immediately *accepted* by a product manager. The last class is significantly harder to identify and conatins the PDRs that have to the potential to be transformed into requirements.

For analytic purposes all questionable PDRs receive a score between 1 and 5 on certain predertermined criteria. Such criteria are for example if a PDR will enlarge the market, add a unique selling point to the product or will require a lot of experience to implement. These criteria all have a specific weighting on the end score. Some of which are even attributed a negative weight, such as requiring a lot of experience to implement. All weighted scores are added together and the product manager can use this end score to help him decide whether a PDR will be *accepted* or *rejected*. With this technique, all PDRs can be compared to each other and those that have the highest score can be accepted as requirements in the release definition. This analysis can be compared with the requirements triage technique that was described by Davis [3].

The PDRs within ErpComp are the result of several different involvement occasions, such as incident reports and CPSs. Similarities amongst the PDRs may appear due to the high degree of customer involvment in all these occasions. Since issuers of PDRs have to fill in a description themselves, the descriptions all differ from each other in terminology, style and precision. This means some PDRs lack completeness in the way they are described, while others may be ambiguous. To cope with the large number of candidate requirements (PDRs) product managers perform the selection and reduction of the PDRs as daily routine.

4 Lessons in customer involvement

The usage of the customer involvement factory has several advantages for the requirements management, as well as for the software as a whole. The model helps to identify the needs of the customer in a manner that really involves the customers. Because of this the PDRs resulting from the different types of customer involvement are particulary valuable for product manager. They are functionalities customers actually want and therefore integrating these will improve the chance customers will stick to the software or perhaps even broaden the market. The process of analyzing, reducing and selecting the large number of PDRs is still a lot of work that is not yet automated, due to the high degree of difficulty involved [14]. However the analysis phase has become more structured with the customer involvement factory.

Besides the advantages in requirements management, the customer involvement described in the model is a great marketing tool that can significantly increase customer loyalty. Encouraged by the ability to contribute, customers indicate that they enjoy the sessions and are more likely to come back to future sessions as well as buy subsequent releases of the software. In addition, we want to present a few lessons to be learned from the use of the customer involvement factory.

1. *Organize CPSs in all phases of the software life cycle.* While the product is in the initial development phase [13], it is often very immature. Exposing the software to the customers at this point and inviting them to share their own comments and suggestions can result in a lot of criticism regarding the product. This is something that can dissuade a company from opening up to their customers, regardless of how potentially valuable the comments may be. It can be problematic to be exposed as such in the initial development phase of the software product, but the advantages from the resulting PDRs obtained from the CPSs outweigh the disadvantages, and as such this phase should not be omitted. While the software is in the maintenance phase [13], organizing CPs should not stop despite the software already having reached a mature stage. Customer comments and suggestions change over time, and therefore there is no limit placed as to when a customer is able to contribute to the PDRs. A third reason for being consequent in organizing the sessions, is the fact that it helps the customers trust the company, even if the software product is still quite immature.

2. *Responsible product manager should be present at the CPS.* Customers will only commit their time to improving the software product by submitting suggestions when they are assured their suggestions will be taken seriously. If a marketeer were present with the customers during the CPSs, the customers would be less inclined to share their suggestions. Instead, the presence of a responsible product manager with the ability to modify the product if and where needed will result in the creation of more relevant PDRs.

3. *There is no minimum number of participants for a PS, there is however a maximum.* Every customer is free to make suggestions about the current software, and therefore can directly contribute to the creation of PDRs. Since every PDR is a possible valuable requirement, there is no minimum to the number of customers that can participate in a CPS. Following this reasoning, the more customers that participate the better, but experience has shown that the productivity starts to decrease when the number of customers begins to exceed 50. This is due to the lengthened duration of the sessions in which customers being to lose interest and contribute less. The optimal number of participants for a CPS is determined to be between 30 and 40 customers. With this number of participants every customer can receive immediate feedback from the rest of the group regarding his/her suggestions. If a less than optimal number of customers are present during a CPS it is still conducted in the same manner, however, the effectiveness of the session is often below average. A solution to this is merging the CPS with another the following year to achieve the ideal number of customers.

4. *Give customers feedback afterwards about their suggested requirements.* Due to the CPs being held annually, a primary concern is ensuring the customers will

return the following year to once a gain give contribute to the next release in addition to continuing to use the software. This is done by giving customers feedback about their suggestions after the CPSs to ensure them that the company values and takes their contributions seriously. This feedback can be given directly after an informal session, placed on the website the following day, or after their suggestion is implemented in the software as a requirement.

5. *Focus on interactivity.* It is important to maintain the CPSs as interactive as possible. Customers are not invited to the sessions for talk about marketing, but to involve them in the requirements gathering process. Because of this, listening to the customers should take place as soon as possible during the sessions. Since the customers already are already aware of the company and software, a formal lenghty introduction is often unnecessary. Experience indicates that it is best to begin with customers sharing their suggestions within 15 minutes of the beginning of the CPS.

6. *Actively manage PDRs.* Due to the large amount of PDRs handled, it is important to accurately and consistantly manage them. This is done by having a product manager analyze the PDRs as part of a daily routine. This has several advantages, such as being able to instantly merge similar PDRs when they come in. The same holds for PDRs that are too complex and must be split up. Not doing this on a daily basis can result in a counterproductive accumulation which will make subsequent analysis increasingly difficult.

5 Conclusion and Outlook

Different types of customer involvement benefit a software vendor's requirements management process. PDRs are easily acquired by inviting customers to CPSs. Furthermore, PDRs can be maintained easily by using analysis methods, such as establishing completeness, ambiguity and similarity. The model has proven to work well for ErpComp in gathering and organizing PDRs, reducing them to an acceptable number of requirements and creating a release definition for ErpComp. We believe that the method, when used in a similar context, improves the selection process of the right requirements for a product software company that has to deal with large numbers of requirements.

Acknowledgements

We want to thank Mohamed Amri and Ariaan Hoogendijk of AFAS ERP Software for their time, knowledge and expertise. Also we want to thank Kevin Vlaanderen for his suggestions and help in writing this paper.

References

[1] S. E. Asch. Opinions and social pressure. *Scientific American*, 193:31–35, 1955.

[2] P. Carlshamre and B. Regnell. Requirements lifecycle management and release planning in market-driven requirements engineering processes. In *International Workshop on the Requirements Engineering Process: Innovative Techniques, Models, and Tools to support the RE Process*, 2000.

[3] A. M. Davis. The art of requirements triage. *IEEE Computer*, 36:42–49, 2003.

[4] A. Edmunds and A. Morris. The problem of information overload in business organisations: a review of the literature. *International Journal of Information Management*, 20:17–28, 2000.

[5] M. S. Feather, S. L. Cornford, and M. Gibbel. Scalable mechanisms for requirements interaction management. In *Proceedings of the 4th Int. Conf. on Requirements Engineering*, 2000.

[6] J. Karlsson, S. Oisson, and K. Ryan. Improved practical support for large-scale requirements prioritising. *Requirements Engineering*, 2:51–60, 1997.

[7] M. A. Kaulio. Customer, consumer and user involvement in product development: A framework and a review of selected methods. *Total Quality Management*, 9:141–149, 1998.

[8] J. Natt och Dag, V. Gervasi, S. Brinkkemper, and B. Regnell. A linguistic-engineering approach to large- scale requirements management. *IEEE Software*, 22:32–39, 2005.

[9] J. Natt och Dag, B. Regnell, P. Carlshamre, M. Andersson, and J. Karlsson. Evaluating automated support for requirements similarity analysis in market-driven development. In *Proceedings of the 7th Int. Workshop on Requirements Engineering: Foundation for Software Quality*, 2001.

[10] J. Nielsen. *Usability Engineering*. Morgan Kaufmann, 1994.

[11] T. C. Powell. Total quality management as competitive advantage: A review and empiracal study. *Strategic Management Journal*, 16:15–37, 1995.

[12] R. S. Pressman. *Software Engineering: A Practitioner's Approach*. McGraw-Hill, 2005.

[13] V. T. Rajlich and K. H. Bennett. A staged model for the software life cycle. *Computer*, July:66–71, 2000.

[14] B. Regnell, R. Berntsson Svensson, and K. Wnuk. Can we beat the complexity of very large-scale requirements engineering? In *Proceedings of the 14th Requirements Engineering: Foundations of Software Quality Conference*, 2008.

[15] M. R. Solomon. *Conquering Consumerspace: Marketing Strategies for a Branded World*, chapter 5, pages 122–126. AMACOM, 2003.

[16] I. van de Weerd, S. Brinkkemper, R. Nieuwenhuis, J. Versendaal, and L. Bijlsma. Towards a reference framework for software product management. In *Proceedings of the 14th IEEE International Requirements Engineering Conference*, 2006.