13 Market-Driven Requirements Engineering for Software Products

Björn Regnell and Sjaak Brinkkemper

Abstract: An increasing part of software development is devoted to products that are offered to an open market with many customers. Market-driven development imposes special challenges for the requirements engineering process. This chapter provides an overview of the special characteristics of market-driven requirements engineering and describes the most important challenges of the area. Key elements of market-driven requirements engineering processes are presented together with a definition of process quality. Requirements state models and requirements repositories are also described and examples of typical solutions to progress tracking and data management are provided. The difficult problem of release planning is also discussed and an industrial example of a release planning process is given.

Keywords: Market-driven requirements engineering, Product software, Release planning, Requirements selection, Process quality, Process improvement.

13.1 Introduction

An increasing part of the software produced is aimed at being offered to an open marketplace rather than to one specific customer. This type of software development is often called market-driven and refers to the situation where the development costs of a generic product are divided among many buyers on an open market and where the potential profit is rewarded to the producer. Market-driven development is different from customer-specific development (also called bespoke development), where one single customer pays all development costs and the resulting product is specific to the needs and wishes of that one customer. This chapter explains the specific challenges of requirements engineering in a market-driven software development context, with focus on process issues and management concerns. It also describes some of the solutions provided by recent research in the area of Market-Driven Requirements Engineering (MDRE).

This chapter in particular, and MDRE in general, mainly takes the viewpoint of the developing organization and focuses on the producer's requirements engineering process, which is aimed at aligning the product content with the needs of the targeted market segments in order to create a profitable software product. There are a number of basic questions that need to be answered by an organization that is developing software products for an open market:

- *How to design and manage a MDRE process*? In order to maximize profit it is vital to outperform the competing software producers at requirements engineering. The developing organization needs to establish an efficient MDRE process

that defines how to work with the classical RE activities, such as elicitation, specification and validation, but in a market driven context.

- *How to design and manage a MDRE repository*? The requirements produced during classical RE are often stored in a document denoted "the specification". In MDRE, it is often more useful to store information in a repository that is dynamically evolving with past and recent data of varying type and level of abstraction, such as: potential and current customer profiles, current and previous release contents, up-to-date status of both candidate requirements and requirements under development.
- *How to make profitable release planning*? A key result of the MDRE process is the strategic decision of what to deliver when. This decision takes into account the strategic assets of the developing organization such as the competence of its engineers, its software architecture investments to date, its current customer base, and combines this with the overall business strategy of the company in order to form a list of adequately detailed requirements that are to be released to the market at a carefully selected point in time.

This chapter has many relations to other chapters of this book. Elicitation (Chap. 2) is a very important part of MDRE but its focus is shifted from acquisition of one particular customer's wishes to a combination of market analysis and generation of new ideas based on opportunities provided by new technology. Specification techniques from Chap. 3 can be utilized, but it is important to realize that in the MDRE situation the set of requirements rapidly may get very large and not all requirements can be specified in detail. Often natural language is the main way of describing the major part of the requirements, and how to deal with large repositories of textual requirements is further discussed in Chapter 10.

Prioritization (Chap. 4) is a key element of decision-making in MDRE, and decision support (Chap. 12) can help in making better re-lease plans. Although each requirement is treated as a separate element of the MDRE process, intricate dependencies among requirements (Chap. 5) make release planning (Sect. 13.5) and impact analysis (Chap. 6) increasingly complex. Requirements-based estimations in general become more uncertain as the overwhelming number or potential dependencies must be excluded from in-depth analysis for practical reasons.

It is recommended that the reader first get a basic knowledge of the state-of-the-art part of the book (in particular Chaps. 2, 4, 5 and 6) before reading this chapter. It is also recommended that Chap. 13 is studied in conjunction with Chaps. 10 and 12, to get a broad view of the challenges and tools within the MDRE area.

The chapter is organized as follows. Sect. 13.2 is devoted to an in-depth description of the context and concepts of MDRE and describes what is particular to the market-driven situation compared to the customer-specific situation. Sect. 13.3 describes the main elements of the MDRE process and discusses various issues in relation to that process, such as process quality and process capacity, and Sect. 13.4 describes MDRE data management and the relation between requirements refinement states and the use of a requirements repository. Section 13.5 provides details of the special nature of elicitation in the MDRE context. Section 13.6 de-

scribes road mapping and release planning as a vehicle for profitable products. Finally, Sect. 13.7 concludes the chapter.

13.2 Concepts and Context

This section introduces the MDRE context in more detail. Firstly, a number of concepts are defined in order to establish a basic terminology for different types of variants of MDRE. Secondly, a characterisation of the differences between customer-specific RE and MDRE is given. Finally, a number of important challenges in MDRE are discussed.

13.2.1 Basic Concepts

Market-Driven Requirements Engineering (MDRE) covers the classical RE activities, such as elicitation, specification, and validation, adapted to the market-driven situation, where a software producer develops a product that is offered to an open market with many customers. MDRE also covers the specific activities needed in a market-driven context, such as release management and market analysis. MDRE is often conducted under the pressure of competition from other producers, and as the market and product evolve, the MDRE process enacted by a specific software developing organization also needs to be evolved in order to stay ahead of competition.

Of course, the buyer of a software product also has to do some careful requirements engineering in order to select the right product that matches the specific needs of that buyer. This selection process is out of direct control of the producer and a research area of its own (often called COTS selection, see e.g. [24, 18]) and is out of scope of this chapter. However, it is important for the producer to understand how potential buyers may think in their selection process. This type of information regarding customer priorities is subject to market analysis, as described in Sect. 13.4.

There are a number of variants of software products. Table 13.1 provides a classification and some examples of software products based on two dimensions: (1) the degree of customization and (2) the hardware/software content. The degree of customization is divided into three levels. A product is said to be *generic* if it is intended to be used as-is, out-of-the-box, perhaps with minor configurations that are possible to be done by the end-user. A product is said to be *customized* if the product is intended to be useful after it has been tailored to one specific customer's needs, e.g. through adding modules via an open application interface. A product is said to be *customer specific* if the entire product is developed with one particular customer's wishes in mind.

The hardware/software content is divided into three classes: *pure hardware* denotes products that are fixed through its hardware architecture and contains no software that can make the features of the product flexible; *embedded systems* im-

ply products consisting of both a hardware platform and accompanying embedded software; *pure software* denote a product that is completely comprised of software and sold independently of its hardware platform(s).

In Table 13.1, the types of software products that are market-driven include generic/customized and embedded systems/pure software and have shaded cells. The cells with thick frame are *product software* (pure generic/customized software).

The acronym COTS (Commercial Off-The-Shelf) is sometimes used to denote software product, but we have deliberately not used this term subsequently, as it is overloaded with many meanings, see e.g., [20].

Table 13.1 Examples of variants of hardware and software products

	Pure Hardware	*Embedded Systems (HW+SW)*	*Pure Software*
Generic	Note sticks	Mobile phone	Firewall
Customized	Office furniture	Customized car	Enterprise resource planning systems
Customer-Specific	Portrait painting	Military vehicle	Web Site

The distinction between market-driven and customer-specific development is not strict. For example, it is not uncommon that the developing organisation both sells a generic product to an open market and at the same time sells consultancy hours for customizing the product. Some new and costly parts in product evolution are often developed as a customer-specific feature that is paid by a specific client and later generalized and included in the generic product to get more revenue from the investment. In these cases, the software producer has to deal with both MDRE and bespoke RE, as well as generalisation of custom parts.

There are, of course, other aspects that affect the nature of the MDRE context, not represented in Table 13.1. One additional aspect is the *type of buyer*, which can be divided into enterprise versus consumer. Some products are sold to only one of these segments, whereas some products are sold to both types. MDRE for enterprise products may differ in many respects compared to MDRE for consumer products, e.g. with respect to usability issues, product image, type of marketing channels and number of customer relations that need to be maintained.

The level of *complexity of the user interface* is also a factor that affects the MDRE process. Some products are almost invisible, e.g. an embedded Automatic Braking System in a car that has a simple user interface including a pedal and a lamp, but the software itself is very complex. End-users of systems with complex user interfaces of, for example, desktop applications are probably more likely to give extensive feedback on user interface issues, whereas transparent embedded systems perhaps only render attention by end-users when they do not work as intended. This in-turn may have strong implications on the elicitation process and how to treat software usability in MDRE. (A case study in usability engineering in a market-driven context is presented in [23].)

13.2.2 Characteristics of MDRE

Empirical evidence from a number of case studies and surveys show that MDRE is different from the RE that is conducted in customer-specific projects in many ways [5, 6, 19, 26, 15, 25, 12]. The primary objective of market-driven development is to deliver the right product at the right time, while the bespoke situation often is focused on fulfillment of a contract and compliance to a requirements specification. In the MDRE case, success is determined by sales, market share, product reviews etc., while in the bespoke case, customer satisfaction and user acceptance is directly determining whether the project is a failure or not. The life cycle of a bespoke system is often viewed as divided into development first and then maintenance. There is often one major release, whereas market-driven development often is a long series of releases, and the product is undergoing continuous evolution rather than maintenance.

In MDRE requirements elicitation is often devoted to innovation of new requirements combined with market analysis, whereas customer-specific elicitation is focusing on collecting information regarding one organizations wishes through, e.g., interviews with the known users. In MDRE, some of the features to be released may be confidential and the eventual users unknown, so elicitation cannot always rely on interviews with customers and end-users as the main source of information. Requirements specifications in the MDRE case are often less formal compared to the bespoke case, and natural language text is the dominating way of documenting the results of MDRE. (See also Chap. 15 on elicitation issues in web-based information systems.)

While much effort in bespoke RE is devoted to negotiation and conflict resolution (see Chap. 7), the MDRE case is more focused on prioritization, cost-estimation and release planning, and these activities are all conducted by the developing organization [5]. An example of a case study in market-driven prioritization is available in [28] and Chapter 4 includes an in-depth account of prioritization techniques.

In the bespoke case, validation can be made continuously through the contacts between the customer and the developers, but in the market-driven case validation is often delayed until a late stage in the development, e.g. at expositions during fairs or during beta tests with selected key customers.

Some of the most important characteristics of a typical MDRE context are summarized subsequently.

- The developing organization makes all decisions but also takes all risks.
- There is a continuous flow of requirements throughout the product lifetime.
- The requirements volume is potentially very large and continuously growing.
- A majority of the requirements are informally described.
- The product is evolving continuously and delivered in multiple releases.
- Release planning focuses on time-to-market and return-on-investment.

13.2.3 Challenges in MDRE

In a survey on market-driven requirements engineering [15], a number of challenges were identified. The study results are based on interviews with employees at five different companies of varying size and maturity. The purpose of the study is to provide insights into the special RE challenges in market-driven software development. Subsequently follows a short explanation of the most salient challenges found. For more details see [15].

- *Balancing market pull and technology push.* It is necessary to find a good trade-off between requirements corresponding to perceived user needs and new, inventive ones that may provide a competitive advantage through groundbreaking technology. Finding a good balance between technology-driven and needs-driven requirements may be a delicate challenge.
- *Chasm between marketing and development.* In some companies it can be observed that there is a gap between marketing and developers concerning the views on requirements engineering. Better communication and collaboration between these groups are needed, in order to increase the requirements quality and thereby the quality of the final product.
- *Organizational instability and market turbulence.* Companies without a defined process take a significant risk if key persons leave the organization, since they lack the necessary documentation and structure. In times of downsizing or rapid expansion it is very difficult to install a repeatable process.
- *Simple tools for basic needs.* Some companies requested simple and easy-to-use techniques for basic activities. For these companies it was a challenge to find solutions that are not too complex.
- *Requirements dependencies.* Dependencies among requirements make release planning difficult. Some companies treat dependencies in a basic way by bundling related requirements, but efficient ways of managing at least the most important dependencies are needed. (See further Chap. 5.) Different types of dependencies are reported in the case study by Carlshamre et al. [7].
- *Cost-value-estimation and release planning.* Release planning relies on accurate estimates; underestimation of cost may result in an exceeded deadline while over-estimation of cost may exclude valuable requirements; over- or underestimation of value may result in a product that is badly aligned with actual market needs and thus make the development investment a losing business.
- *Overloaded Requirements Management.* Requirements suggestions from developers and customers are essential. It is a challenge to prevent the requirements repository from being flooded with requirements and how to maintain throughput at times when the number of arriving requirements peak.

The challenges stated above reveal intrinsically difficult problems and it is unlikely that the challenges can be met by a single, simple solution. The key issue for a market-driven company is to continuously improve in managing these challenges in such a way that it stays ahead of competitors.

13.3 The MDRE Process

This section provides a definition of MDRE process quality in terms of decision outcomes in requirements selection. Process capacity and the importance of having a screening function is also discussed.

As described in Sect. 13.1, requirements are continuously generated during the entire lifetime of the product. The software is released in a series of releases as a result of product evolution, where new features are added and existing features are improved according to the advancement of the targeted market. In general, the MDRE process can be seen as a way of synchronizing the work with the continuous flow of candidate requirements and the work with the discrete release events. This synchronisation should enable all parts of development from RE to V&V to work in concert towards the same goals. The main vehicles for communicating these goals are the strategic roadmap together with the release plan of the product.

When designing an MDRE process for a specific company, it is important to realize that there are many situational factors that determine what the best concrete process implementation is. Such factors include: type of development process, type of distribution channels, price and licensing policy, type of market, what is the distinguishing customer value, product complexity, nature of competition, customer behaviour, requirements on product flexibility and adaptability, user interface complexity, predictions on sales, sales channels, etc. It is obvious that the maturity of the developing organization's development process with the competence of the developers, as well as the maturity of the market with customers' knowledge of how to apply technology for their own benefit, are major determining factors of what is most important to get right in the MDRE process. A further discussion on maturity issues in MDRE is provided in [16].

13.3.1 Process Quality

When designing a MDRE process that is adapted to a specific organisation's needs, it may be valuable to define criteria for process success and thus to have a concrete notion of process quality. Of course, the process quality is intimately related to the quality of the artefacts that are produced during the process, and MDRE processes typically generate requirements descriptions in various forms. However, a major process quality issue in MDRE is the quality of decisions that are made about produced artefacts. One way of capturing decision quality is by referring to the ratio of correct requirement selection decisions that are made during the recurring release planning activity, as in the *alfa/beta model* of MDRE selection quality [29], where the decision outcomes are divided into four cases, as described in Table 13.2.

An *alfa requirement* is a requirement that has such a high inherent quality that it ideally should be *selected*. The alfa requirements are thus the "golden grains" among all candidates that the MDRE process should bring forward. "High quality" can, for example, be interpreted as the actual added profit that the requirement is

contributing with if included in the product. Correspondingly, *beta requirements* are those that ideally should be *rejected*, as they are of inherently low quality.

Table 13.2 Decision outcomes in requirements selection

		Decision	
		Selected	*Rejected*
Requirements Quality	*alfa*	*A* Correct selection ratio	*B* Incorrect selection ratio
	beta	*C* Incorrect selection ratio	*D* Correct selection ratio

In Table 13.2, the ratios of the different decision outcomes can be used to define metrics that can characterize the product and decision quality [29]. The *product quality* Q_p can be defined as $Q_p=A/(A+C)$, meaning the share of selected (and thus implemented) alfa requirements of the total selected requirements. The *decision quality* Q_d can be defined as $Q_d=(A+D)/(A+B+C+D)$, representing the share of correct decision in relation to the total number of decisions.

The main challenge of the MDRE process is to find and select alfa requirements, while rejecting beta requirements, and thus maximizing *A* and *D* while minimizing *B* and *C*. However, the problem is that it is not easy to know if a requirement is actually an alfa or a beta requirement, as the cost-benefit trade-off is very difficult. Estimations of both cost and value are inherently error prone and dependent on difficult forecasting of market and technology advancements as well as guesses about actions of competitors. Only post factum, when a product has been out on the market for a longer period, it is possible to say with some degree of certainty if it was a correct decision or not to select or reject a specific requirement [17]. Nevertheless, it is the quality of this uncertain decision-making that determines winners and losers on a software product marketplace.

The elicitation sub process of MDRE (see further Sect. 13.4) has a major impact on the process quality as it influences the fraction of incoming alfa requirements. The better the elicitation process is, the higher the share of alfa requirements, and thus representing an effective elicitation process that make the golden grains come forward. The *golden grain ratio*, defined as the number of issued alfa requirements divided by the total number of issued requirements, can thus be used for characterizing the outcome of the elicitation process.

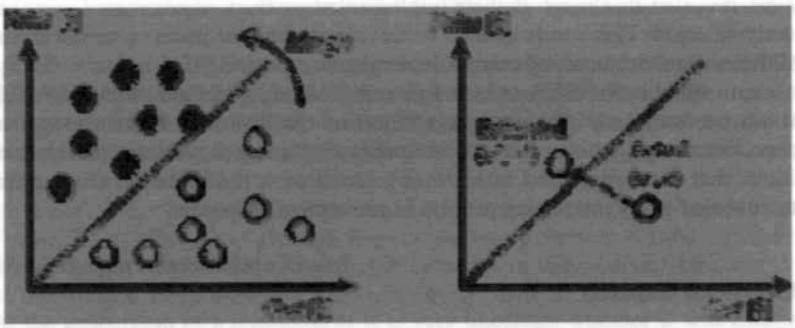

Fig. 13.1(a) Cost-value diagram with alfa-requirements (filled) and beta-requirements (empty)

Fig. 13.1(b) Estimated values are differing from actual values causing wrong selection decision

Figure 13.1 illustrates alfa and beta requirements using a cost-value diagram [13]. In Fig. 13.1 (a) the alfa requirements can be seen as those requirements that have values that are larger than their costs (filled circles in the figure). This means that they are above the *margin* line. If a higher margin of say 20% is requested, then the slope of the margin line is increased to the proportional factor of 1.2, which in turn increases the demand for a requirement to be of alfa type. It should be noted though, that the actual cost and value of a requirement is generally unknown. Furthermore, he decision-making is only based on uncertain estimates, resulting in the fact that beta requirements may end up above the margin line, as illustrated in Fig. 13.1 (b). Here the value is overestimated and the cost is underestimated so that a beta requirement is incorrectly judged to be an alfa requirement.

It should be noted that the value and cost of a requirement is not only depending on the requirement itself, but also on its relation to other requirements. As described in Chap. 5, requirements can have many different types of dependencies between pairs, or more generally among *n*-tuples of requirements, and the value and cost of one requirement may change depending on if other requirements are selected or not [7]. In addition, the value and cost of a requirement may also change over time, so that, e.g., an unanticipated delay in the implementation of a requirement may render another cost-value ratio than was expected at the point in time when the selection decision first was made.

In addition, the concept of "value" can be a complex combination of many different types of contributing values, e.g. value for a certain market segment, value for the internal architecture to enable future feature development, value for strengthening company image, value for entering new markets, etc. An example of how to visualize and balance several value estimates in a distributed marketing organisation is given in [28]. Examples of optimisation and trade-off analysis for release planning can be found in [9] and [31]. The alfa/beta model has been used as a basis for a survey among product managers [29], where it was found that a majority of the respondents that were able to consistently estimate process model pa-

Engineering and Managing Software Requirements
A. Aurum; C. Wohlin (Eds.)

Due to an error that occurred during the data conversion process, the figure on page 295 is not printed properly. Please find attached the correct version of the figure.

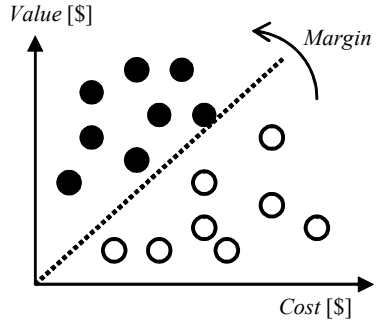

Figure 13.1 (a) Cost-Value Diagram with alfa requiremts (filled) and beta requirements (empty).

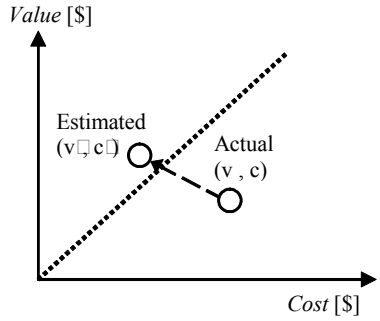

Figure 13.1 (b) Estimated values are differing from actual values causing wrong selection decision.

rameters revealed that most of their implemented product requirements were incorrectly selected. This result indicates that the potential of process improvement in MDRE within the surveyed companies is great.

In a case study in MDRE process improvement using a method called PARSEQ (Post-release Analysis of Requirements SElection Quality) [17], it was shown that retrospective investigation of selection quality, including a root case analysis of decisions that were suspected to be wrong based on a re-estimation of cost and value, revealed many interesting process improvement proposals.

13.3.2 Process Capacity

In empirical studies of the MDRE process it has been found that there is a risk that the process gets in a state of congestion [27, 15], as a consequence of allowing more requirements to enter the MDRE process than can be handled with the available resources. This, in turn, results in throughput problems and eventually a negative impact on both time-to-market and product quality. The MDRE process capacity and the risk of overloading have been further studied using both analytical modelling with queuing theory [29] and discrete event process simulation [10, 1, 30]. These studies show that if the process gets overloaded, the throughput is severely hampered and the mean-time-to-market increases rapidly.

In [30] the alfa/beta quality model was used as a basis for measurement in process simulation experiments, and the results showed that an important means of reducing the risk of overloading is the introduction of a screening activity. During screening a quick assessment of each requirements value and cost is made before further effort is spent on analysing that requirement. This results in a rough judgement whether the requirement should be rejected upfront or if it should be allowed to enter subsequent stages of refinement. (See further the requirements state model in Sect. 13.4). Of course, there is a higher risk of making a wrong rejection decision based on a quick and rough analysis, but the benefit of not pushing too many requirements into the further stages of the process and thus avoiding overloading may be greater than the loss of a few golden grains, as taking on more work than the available process capacity allows for may damage the whole development and result in an unreasonably long mean-time-to-market [30].

Another means of speeding up MDRE is to support the manual and labour intensive analysis of natural language requirements descriptions by means of linguistic techniques [22, 21], which is further described in Chap. 10.

13.4 MDRE Data Management

This section provides a general description of two typical ingredients in MDRE data management, the requirements state model used for progress tracking of requirements refinement and the requirements repository where relevant attributes of candidate requirements are stored. The description here is based on previous

studies of state models and repositories [6, 27] and our observation of industrial practice, but generalized and simplified in order to provide a broad and not too specific view of MDRE data management. One should therefore keep in mind that this perspective is quite different from tailor-made software, where the wishes and satisfaction of the customer are leading the requirements elicitation and capturing process. This implies that key principles are not the same in the processes and data management of MRDE.

13.4.1 Requirements State Model

At the conception of a requirement it is very uncertain whether it will finally get realized into a product release. Available resources and lead time until the planned date of the product release into the market limit the realization of any wish into the software product. Market-driven software implies that the vision and scope of the product are well established, thereby setting means to discern whether a requirement fits the standard or is to be rejected as it is too customer specific.

In keeping stock of the large volumes of requirements through the stages of the development a requirements state model is indispensable (see Fig. 13.2). We call this state model the requirements salmon ladder referring to the uncertainty of a salmon to get back upstream to the breeding currents.

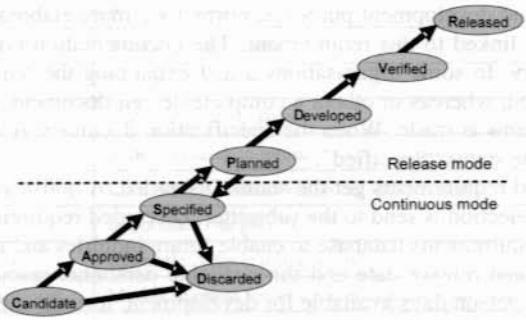

Fig. 13.2 Requirements state model, or requirements salmon ladder

Requirements are received at any time, but the development of a product is made in releases that are produced at discrete points in time. We therefore distinguish two modes: *continuous mode* and *release mode*. In the continuous mode, requirements are received and registered by the product manager from all kinds of submitters internal or external to the company, such as customers, sales representatives, or development teams.

The development of product releases is initiated at designated times according to the roadmap planning (see Sect. 13.6), and the requirements management activities are in release mode. During release development the product manager is in touch with other roles in the development team: project manager, software engi-

neers, testers, technical authors, translators, etc. In release mode the content of the next release, also called the release scope, is then frozen in order to manage the release development project properly. Changes to the scope are then decided through a scope change procedure.

In order to monitor the progress of the work on the requirements the following statuses of the requirements salmon ladder are usually distinguished.

Candidate: Each requirement received gets the status of "Candidate". It is preferred that the description of the requirement follow the wording of the submitter as precisely as possible in order to keep commitment from the submitting party to the requirement. (For an overview of the requirements sources and elicitation, see Sect. 13.5.)

Approved: At regular time intervals the requirements with status Candidate are being reviewed for a possible inclusion into the future product releases. Accepted requirements get the status "Approved". This judgement process is a very difficult and responsible task. First, a long term vision of the product is required, which is usually expressed in product roadmap documents (see Sect. 13.6). Then a thorough functional and technical understanding of the product is required to determine the meaning and consequences of the often very detailed requirements of the existing customer base. Finally, the product managers should be able to cope with the political and strategic issues brought in by possible new contracts, important customers, and insisting sales people.

Specified: As the original description of the requirements is likely not very suitable for planning and development purposes, normally a more elaborate specification is created and linked to this requirement. The documentation type of the specifications may vary. In some organisations a text explaining the requirement in more depth is created, whereas in others a complete design document with Use Cases and Class diagrams is made. When the specification document is available the requirement gets the status "Specified".

Discarded: Rejected requirements get the status Discarded. A notification with the motivation of the rejection is send to the submitter. Discarded requirements are not deleted from the requirements database to enable future inquiries and analyses.

Planned: The planned release date and the available personnel resources determine the number of person days available for development, testing, and product completion. The product release planning can accommodate a maximum number of requirements based on the effort estimates and a prioritization. All requirements selected get the status "Planned", and are input for the design and coding processes. As the estimates are usually too optimistic, some of the planned requirements have an indication of lower priority and may be candidate to be taken out of the release plan in case of shortage of time to complete the release.

Developed: Development entails technical design, coding, unit tests, and production of collateral materials, such as brochures, marketing campaign, and training material. When all these activities have been successfully completed, the requirement gets the status "Developed". Note, that de-scoping, i.e. taking a requirement out of the release plan, can happen anytime, even when development is substantially under way. In this case the code has to be brought back to a state

where the requirement was included. De-scoping usually happens if time runs out, or due to changing priorities.

Verified: Several tests are likely to be necessary in order to ensure an adequate level of quality before a developed requirement is released. Typical types of test are: functional unit tests for the small units performed by a tester not part of the development team; integration test focusing on dependencies between modules; system test for the complete software system; acceptance test for the complete product (software and collateral); and a final test of the installation files.

Released: When all activities for the product release have been completed the requirement finally gets the status of "Released", and the submitter is given a notification. Also released requirements are kept in the requirements repository for further analysis.

Most commercial requirement management tools allow the addition and definition of own statuses. The correspondence of status transfers with activities in the development, such as linkage to design and test documentation, can usually not be enforced by the tools, but require manual operation.

Table 13.3 Outline of a typical MDRE repository

Attribute	*Value*	*Assigned in State*
State	C / A / S / Di / P / De / V / R	-
ID	Unique identity	Candidate
Submitter	Who issued it?	Candidate
Company	Submitter's company	Candidate
Domain	Functional domain	Candidate
Label	Good descriptive name	Candidate
Description	Short textual description	Candidate
Contract	Link to sales contract enforcing requirement	Candidate
Priority	Importance category (1,2,3)	Approved
Motivation	Rationale: Why is it important?	Approved
Line of Business	Market segment for which requirement is important	Approved
Specification	Links to Use Case, Textual Specification	Specified
Decomposition	Parent-of / Child-of – links to other req's	Specified
Estimation	Effort estimation in hours	Specified
Schedule	Release for which it is planned for	Planned
Design	Links to design documents	Developed
Test	Links to test documents	Verified
Release version	Official release name	Released

13.4.2 Requirements Repository

In order to register the requirements properly many development teams use some kind of requirements repository. For smaller development efforts a simple spreadsheet may be sufficient. Larger-scale development is unlikely to be successfully executed without a requirements management tool due to the volume of requirements. Monolithic requirements specification documents are also considered problematic, as the document structure hinders the concurrent elaboration of different requirements by distinct teams. Individual registration of the requirements in an MDRE repository is indispensable. We present in Table 13.3 an outline of a typical MDRE repository in relation to the salmon ladder.

Aside from these generic attributes there are more attribute categories that are needed for specific markets. Country data is required for products that are sold internationally. Various countries have legal or financial rules that are required by law. Products sold on different technical platforms, such as operating systems, databases or multi-modal user interfaces, usually require specific requirements to cater with the particularities of these platforms. Some platforms may provide facilities that can be incorporated, whereas for other platforms these have to be completely developed.

Products with different product lines or being sold to different markets (line of business) require specific attributes related to the addressed functional domains. This is the case for products being sold in markets where safety is an important issue, such as the health care industry and in the avionics industry.

Tracing and tracking of requirements into the designs, code, and test reports is mainly an administrative task requiring proper support tools. As long as the tools employed in the requirements management and development lack proper means for interoperation, the tracing and tracking is condemned to be a labor-intensive error-prone manual task. Given the fact that developers often work at one requirement at a time, the tracing of changes made in the various work products would automatically provide insight into the requirements tracing process.

13.5 Market Analysis and Requirements Elicitation

Sources for requirements are numerous. When a new product is started, existing literature on the subject matter may provide insight in the domain. An efficient way to collect requirements in a structured manner is through the collaboration with key customers. In return for early knowledge transfer the key customer assist in requirements specification and in on-site testing. Care has to be taken that the focus of the product remains the full width of the market, and not deteriorate into a narrowing view of those key customer.

For larger enterprise applications markets, such as Enterprise Resource Planning (ERP) or Customer Relationship Management (CRM), analyst companies (e.g. Gartner, Forrester) provide functional and technical overviews of the underlying domains. A side effect of the analyst reports is the unification of the termi-

nology in a domain. The positioning of the current product release on the complete domain overview is a good source for additional requirements.

Recently, facilitated workshops were proposed as a means for effective and efficient elicitation of requirements. In this setting a group of domain experts is brought together in an intensive work setting to specify the requirements managed by a facilitator. Schalken et al. [32] reported an investigation into the advantages of facilitated workshops compared to traditional one-on-one interviews. The comparison was in terms of required effort, in terms of calendar time required, and in terms of the quality of the requirements. About 50 projects in both categories in a large financial company in the Netherlands were analyzed. It turned out that requirements' gathering with facilitated workshops is less effective for small projects, but for large projects it is more effective. Surprisingly, the customers were less satisfied with the quality of the resulting requirements. Time and group pressure of the facilitated workshop might be reasons for this.

Customer involvement in requirements specification is to be performed in a careful manner. Expectations have to be managed as the development of the requirements may be spread over various releases and years. Some companies have organized *Customer Working Groups (CWG)*. A CWG is a team of customer representatives together with product managers, which develops a specification document for a whole new functional area. The customer representatives are experts in the domain, who can also judge the priorities of the must-have and the nice-to-have requirements very well. Establishing a CWG in an area also sets expectations regarding the future availability in releases. Strategic roadmap changes that exclude the CWG theme from the roadmap may set pressure on the vendor-customer relationship.

13.6 Roadmapping and Release Planning

A roadmap is a document that provides a layout of the product releases to come over a time frame of three to five years. Customers want to be sure that the future of the software product on which they depend is in line with their future plans. Especially in markets where the costs and consequences of a vendor change are large, the customer wants to have a stake in the roadmap decision-making.

Roadmaps are available in several segments of society to support decision makers in the route to innovation [3]. Based on a variety of roadmaps reported in the literature, Schaller [32] has established a taxonomy that classifies roadmaps according to their location in an applications-objectives space. This taxonomy scheme classifies the roadmaps broadly into the following four categories:

- Science and Technology Roadmaps
- Industry Technology Roadmaps
- Corporate or Product-Technology Roadmaps
- Product or Portfolio Management Roadmaps

The Product-Technology Roadmaps is the type of roadmap of the software industry according to the taxonomy. Software development is a technology development and a roadmap is made for each of the products. A technology roadmap is the document that is generated by the roadmapping process. It identifies the critical system requirement themes, the product and process performance targets and the technology alternatives and milestones for meeting these targets [8]. The roadmap helps identify precise objectives and helps focus the required resources on meeting those objectives. Roadmapping has several potential uses and resulting benefits at both the individual corporate and industry levels. According to Garcia [8] the three major uses of roadmapping are:

- Development of a consensus about a set of needs and the technologies required to satisfy those needs
- Provision of a mechanism to help experts forecast technology developments in target areas
- A framework to plan and coordinate developments either within an organization or in an entire industry

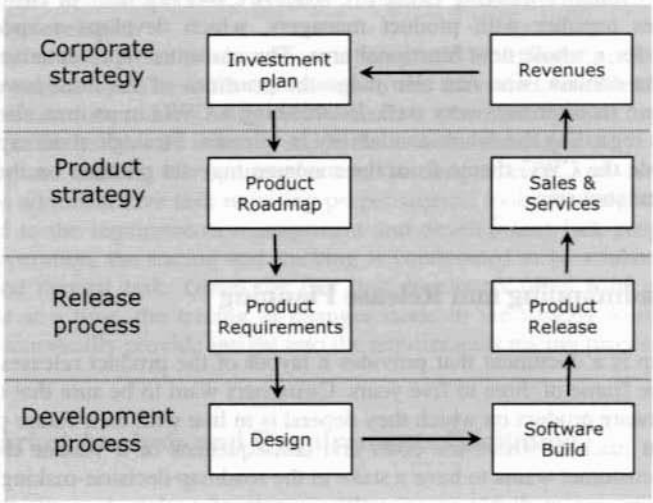

Fig. 13.3 Product roadmap in the investment cycle

The determination of the product roadmap in a MDRE context cannot be seen independent from the overall strategy of the company. As shown in Fig. 13.3 it serves best to distinguish a cyclic, four layer structure to stratify from strategy making to the development of the software product. First, on an annual basis the investment plan is devised based on revenues and forecast plans of the current product lines: an extension of the product line with a next release, a start of a new product line, and the termination of a product line. These plans also include the

investment levels in terms of money or headcount, and some strategic issues regarding the content of the products.

The investment plan is then input for the management of the product development unit to create or update the current product roadmaps. In several product companies the main manager responsible for the product roadmap is called Chief Technology Officer. The roadmaps are created taking the views of the units for sales and consulting services into account, as these units know best what the strengths and weaknesses of the current products are, and what kind of market trends and functionality is appreciated by current and prospective customers.

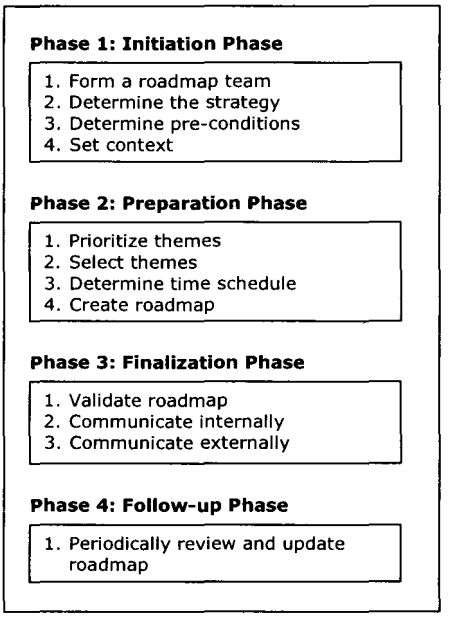

Fig. 13.4 Roadmap processes

Product managers are responsible for the release process at the next layer of operation. They elaborate the product roadmap into a set of product requirements for the various releases. Either they select the suitable requirements from the available candidate requirements in the requirements database (see Fig. 13.3), or they look for additional requirements (see Fig. 13.4) from various sources in the product domain. This step is especially needed when new product lines are initiated or when an existing product line is expanded with a new functional area. The set of product requirements is then input for the development process, which results into the kernel of the software product, the software build. The software build together with the auxiliary materials, such as user manuals, training material, marketing collateral, is then packaged as a new product release.

Example: Roadmapping at Baan
Recently, the roadmapping processes of Baan (now SSA Global) were evaluated and redesigned [3]. The process flow of the roadmap process, which resulted from this effort, is shown in Fig. 13.4 and explained subsequently.

The roadmapping effort starts in Phase 1 with the formation of the roadmap team. Obviously, some senior employees with in-depth product knowledge and access to the key people are candidates for this role. The strategy and preconditions are usually laid out by corporate management, e.g. time line (three or five years), products in scope, range of investment, and release frequency. The team then formulates its own plan and context. In the next phase the themes for functional and technical extension to the products are identified and prioritized. Themes can be seen as high-level requirements, usually well known generic issues in the product domain. The themes are elaborate in a set of coherent requirements to be planned in one or subsequent releases. Typical themes are "Enabling for Workflow", "Porting to Linux platform", and "Extensions for a new market". Themes should be so well defined and attractive, that they are candidates for the functional extensions to be listed on the brochures that cover the release products

Schedules of roadmaps are often expressed in quarters of a year. A timeline shows the various product lines with the releases plotted. The release frequency is dependent on the size of the product. For Baan ERP the frequency was about 1.5 year as the market is not receptive for too many disruptive system upgrades. Bookkeeping software is usually upgraded once a year. Changed legislation requires that the financial processes are brought up-to-date. When the roadmap has been drafted, it requires to be validated by the various stakeholders groups: general management, large customers, sales and consultancy teams, and development teams. Comments and feedback is integrated, and the roadmap is handed over to the general management, who is the owner and communicator of the roadmap. The formal communication of the roadmap is often launched at some large event where many customers meet.

Finally, in the Follow-up phase the roadmap team is thanked for its efforts and dissolved. Some product managers remain responsible for the maintenance of the product roadmap documentation and the updating with new themes. After about three years a new roadmap team is formed and the cycle of phases is repeated.

13.7 Conclusion

When the requirements engineering process is enacted in a market-driven context the developing organization faces special challenges. Continuously arriving requirement candidates provide input to the decision-making that should result in a strategic roadmap and a prioritized release plan. A major challenge is to cope with the potentially enormous amount of information and to represent and organize it in an efficient way so that it can provide a good basis for efficient and effective decision-making, which in-turn provides the basis for a profitable software business.

This chapter offers input to the design of a competitive MDRE process through the following elements as explained previously:

- A process quality model for assessing the goodness of requirements selection
- A typical requirements state model to be used in progress tracking
- A typical requirements repository to be used in data management
- An example of an industrial release management process

The MDRE has to be adapted to its specific context. The maturity of the organization and its products, as well as the market and its customers, are critical parameters that have to be considered when formulating and establishing a well-balanced process. It is also important that there is a built in mechanism for learning and improving in order to stay ahead even as the competition gets smarter. In [2], the following four research topics were identified based on a systematic assessment of research contributions in relation to the Capability Maturity Model Integration [4]:

- Release planning: means to select requirements for the next release based on priority, development effort estimates, and expected revenues
- Experience evaluations of industrial requirements management processes: a study in MDRE efforts in a variety of companies
- Tracking and tracing: tools to track and trace the requirements over the various work products of the development process, such as designs, code, tests, and manuals;
- Measuring requirements management efficiency and effectiveness: development of measurements to provide means to assess the efficiency and effectiveness of the requirements processes

Other important areas providing challenges to RE researcher in the market-driven context are: accurate prioritization, efficient management of dependencies, and tool support for handling very large requirement repositories, as well as the general area of RE decision support (see further Chaps. 4, 5, 10 and 12 respectively). Both descriptive and prescriptive research is needed to provide both a deeper understanding of the nature of MDRE as well as to offer solutions to industrial problems in combination with scientific evidence on how to best apply them.

Acknowledgements

We would like to thank all researchers that have been involved in the many projects that have formed the basis for this chapter. Special thanks to Johan Natt och Dag and Lena Karlsson, both at Lund University, who have during their PhD studies actively participated in the advancement of the research frontier within market-driven requirements engineering. We would also like to thank Dr. Joachim Karlsson and Per Beremark for providing rewarding opportunities of industrial collaboration. Thanks also to the product managers of Baan (now SSA Global) who participated in the company-wide requirements management processes. Especially

thanks to Pierre Breuls, Mike Chouinard, Wim van Rijswijk and Shirley Bodegraven for their time and involvement.

References

1. Booth R, Regnell B, Aurum A, Jeffery R, Natt och Dag J (2001) Market-driven requirements engineering challenges: An industrial case study of a process performance declination. In: Proceedings of 6th Australian Workshop on Requirements Engineering (AWRE'01), Sydney, Australia, pp.41–47
2. Brinkkemper S (2004) Requirements engineering research the industry is (and is Not) waiting for. In: Proceedings of the 10th Anniversary International Workshop on Requirements Engineering: Foundation of Software Quality, Regnell B, Kamsties E, Gervasi V (Eds.), Essener Informatik Berichte, 9:251-264, ISBN 3-922602-91-6
3. Bodegraven S, Brinkkemper S (2004) Product software roadmap determination process: Where marketing and technology come together. Technical report, ICS, Utrecht University
4. Chrissis MB, Konrad M, Shrum S (2003) CMMI: Guidelines for process integration and product improvement. Addison-Wessley, ISBN: 0-321-15496-7
5. Carlshamre P (2002) A usability perspective on requirements engineering – From methodology to product development. Dissertation No. 726, Linköping University, Sweden
6. Carlshamre P, Regnell B (2000) Requirements lifecycle management and release planning in market-driven requirements engineering processes. International Workshop on the Requirements Engineering Process: Innovative Techniques, Models, and Tools to support the RE Process (REP'00), September 6-8, Greenwich UK, pp.961–965
7. Carlshamre P, Sandahl K, Lindvall M, Regnell B. Natt och Dag J (2001) An industrial survey of requirements interdependencies in software product release planning. In: Proceedings of 5th IEEE International Symposium on Requirements Engineering (RE'01), August 27-31, Toronto, Canada, pp.84–92
8. Garcia M, Bray O (1998) Fundamentals of technology roadmapping. Sandia National Laboratories, Technical Report, http://www.sandia.gov/Roadmap/home.htm
9. Greer D, Ruhe G (2004) Software release planning: an evolutionary and iterative approach. Information & Software Technology 46(4): 243–253
10. Höst M, Regnell B, Natt och Dag J, Nedstam J, Nyberg C (2001) Exploring bottlenecks in market-driven requirements management processes with discrete event simulation. Journal of Systems and Software, 59(3): 323–332
11. Hermann K, Brinkkemper S, Bubenko JA Jr, Farbey B, Greenspan SJ, Heitmeyer CL, Leite JCS, Mead NR, Mylopoulos J, Siddiqi J (2002) Requirements engineering and technology transfer: Obstacles and incentives. Requirements Engineering, 7(3):113–123
12. Kamsties E, Hörmann K, Schlich M (1998) Requirements engineering in small and medium enterprises. Requirements Engineering, 3, pp.84–90
13. Karlsson J, Ryan K (1997) A cost-value approach for prioritizing requirements. IEEE Software, Sept/Oct pp.67–74
14. Karlsson J, Wohlin C, Regnell B (1998) An evaluation of methods for prioritizing software requirements. Information and Software Technology, 39(14-15): 939–947

15. Karlsson L, Dahlstedt ÅG, Natt och Dag J, Regnell B, Persson A (2002) Challenges in market-driven requirements engineering - An industrial interview study. In: Proceedings of 8th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'02), September 09-10th, Essen, Germany, pp.37–49
16. Karlsson L, Regnell B (2004) Aligning the requirements engineering process with the maturity of markets and products. In: Proceedings of 10th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'04), June 7-8, Riga, Latvia, pp.69–74
17. Karlsson L, Regnell B, Karlsson J, Olsson S (2003) Post-release analysis of requirements selection quality - An industrial case study. In: Proceedings of 9th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'03), June 16 -17, Klagenfurt/Velden, Austria, pp.47–56
18. Lauesen S, Vium JP (2004) Experiences from a tender process - The customer's dreams and the supplier's frustrations. In: Proceedings of 10th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'04), June 7-8, Riga, Latvia, pp.29–46
19. Lubars M, Potts C, Richter C (1993) A review of the state of the practice in requirements modeling. In: Proceedings of IEEE International Symposium on Requirements Engineering (RE93), Los Alamitos, USA. IEEE Computer Society Press, pp.2–14
20. Morisio M, Torchiano M (2002) Definition and classification of COTS: A proposal. In: Proceedings of 1st International Conference on COTS Based Software Systems (ICCBBS), Orlando, February 4-6, pp.165–175
21. Natt och Dag J, Gervasi V, Brinkkemper S, Regnell B (2004) Speeding up requirements management in a product software company: Linking customer wishes to product requirements through linguistic engineering. In: Proceedings of 12th IEEE International Conference on Requirements Engineering (RE'04), Kyoto, Japan, pp.283–295
22. Natt och Dag J, Regnell B, Carlshamre P, Andersson M, Karlsson J (2002) A feasibility study of automated natural language requirements analysis in market-driven development. Requirements Engineering, 7(1): 20–33
23. Natt och Dag J, Regnell B, Madsen OS, Aurum A (2001) An industrial case study of us-ability evaluation in market-driven packaged software development. In: Proceedings of 9th International Conference on Human-Computer Interaction (HCII'2001), August 5-10, New Orleans, USA, pp.425–429
24. Maiden NA, Ncube C (1998) Acquiring COTS software selection requirements. IEEE Software, March/April, pp.46–56
25. Novorita RJ, Grube G (1996) Benefits of structured requirements methods for market-based enterprises. In: Proceedings of 6th Annual International INCOSE Symposium. Seattle, USA, INCOSE
26. Potts C (1995) Invented requirements and imagined customers: Requirements engineering for off-the-shelf software. In: Proceedings of Second IEEE International Symposium on Requirements Engineering (RE'95), pp.128–130 Los Alamitos, USA
27. Regnell B, Beremark P, Eklundh O (1998) A market-driven requirements engineering process: Results from an industrial process improvement programme. Requirements Engineering, 3(2):121–129
28. Regnell B, Höst M, Natt och Dag J, Beremark P, Hjelm T (2001) An industrial case study on distributed prioritization in market-driven requirements engineering for packaged software. Requirements Engineering, 6(1):51–62

29. Regnell B, Karlsson L, Höst M (2003) An analytical model for requirements selection quality evaluation in product software development. In: Proceedings of 11th IEEE Inter-national Conference on Requirements Engineering, (RE'03), September 8-12, Monterey Bay, California USA, pp.254–263
30. Regnell B, Ljungquist B, Thelin T, Karlsson L (2004) Investigation of requirements selection quality in market-driven software process using an open source discrete event simulation framework. In: Proceedings of 5th International Workshop on Software Process Simulation and Modeling (ProSim 2004), May 24-25, Edinburgh, UK
31. Ruhe G, Eberlein A, Pfahl D, (2003) Trade-off analysis for requirements selection. Software Engineering and Knowledge Engineering, 13(4): 345–366
32. Schalken J, Brinkkemper S, van Vliet H (2004) Assessing the effects of facilitated workshops in requirements engineering. In: Proceedings of 8th IEEE International Conference on Empirical Assessment in Software Engineering (EASE2004), pp.135–144
33. Kostoff RN, Schaller RR, (2001) Science and technology roadmaps. IEEE Transactions on Engineering Management, 48(2): 132–143

Author Biography

Dr. Björn Regnell is associate professor in Software Engineering at the Department of Communication Systems, Lund University, Sweden, and senior member of the Software Engineering Research Group (SERG). Dr. Regnell is the project leader of the requirements engineering research at SERG. His research interests include empirical software engineering, requirements engineering, and market-driven software development. He has published one book and more than 40 refereed papers in these areas. He was program co-chair of the International Workshop on Requirements Engineering – Foundation for Software Quality (REFSQ) in 2002–2004, and he is a member of the program committee of the International Conference on Requirements Engineering (RE) since 2002.

Dr. Sjaak Brinkkemper is professor in Organization and Information at the Institute of Information and Computing Sciences of the Utrecht University, the Netherlands. Previously he was a consultant at the Vanenburg Group and a Chief Architect at Baan Research and Development, where he was responsible for overall software process improvement initiatives in Requirements Management, Architecture and Design. He has published five books and more than 90 papers on software product development and information systems methodology. He is a member of the Editorial Board of the Requirements Engineering Journal, Journal of Database Management, and Journal on Information Systems and e-Business Management.