Market-Driven Requirements Engineering Process Model, version 1.0

Andrigo Gomes – andrigo.gomes@gmail.com
Andreas Pettersson – anpf02@gmail.com
Tony Gorschek – tony.gorschek@bth.se

School of Engineering
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden

MDREPM

Market-Driven Requirements Engineering Process Model (MDREPM)

The area of requirements engineering is largely acknowledged for its importance in successful software development. This stems from the fact that requirements are at the core of many activities, including project management, software development, and testing. Being at the core of different functional areas, requirements can be seen as the lowest common denominator between them, thus working as a means of communication.

Research in classical requirements engineering is not short of studies on techniques and processes that can help software organizations to improve their businesses by introducing good practices in requirements handling. By classical requirements engineering it is meant the type that occurs between software organizations that develop a customized software system for a specific customer, which is also known as bespoke requirements engineering (bespoke RE).

In contrast to that, there exists the market-driven approach to requirements engineering (market-driven requirements engineering or MDRE). This is the case applicable to software organizations that develop software to a market, which can be a combination of a number of known customers or, on another extreme, a mass market where customers cannot be clearly pinpointed.

In MDRE, several challenges arise as compared to the classical bespoke RE. Software organizations in a market-driven situation are faced with competition, the need for strategic planning, and the need for a market orientation to their businesses where strong marketing know-how is necessary to identify market opportunities. However, it should not be forgotten that these issues should all be reconciled with technical product development. Translating business goals into product features that add appealing selling points to software products becomes a challenge as the number of functional areas, and therefore perspectives in a market-driven situation augments (e.g. marketing, business, management, development). This challenge is about getting all perspectives on the same page, in ways that they can all work together to deliver maximum business benefit to the software development organization, whereas also providing customers with high value for the money they spend on products.

Classical requirements engineering does not contemplate techniques and processes for covering market-driven needs. Recently, though, research in requirements engineering started to pay more attention to that, and through collaboration with industry, techniques have been created to handle the specific needs of market-driven software organizations.

However, there is a gap between industry practice and research. It is not uncommon that industry practitioners are unaware of latest research findings. In addition, the mindset that academia is sometimes far from real world problems is unfortunately still common.

In a step to minimize that gap, this work presents the Market-driven Requirements Engineering Process Model (MDREPM). The main objective of this model is to spread the latest research findings in market-driven requirements engineering to industry practitioners. In addition, this model also has the purpose of showing that research in software engineering in general, and especially market-driven

requirements engineering, is more and more concerned with solving real world problems that industry has been facing.

The MDREPM is a collection of good practices in market-driven requirements engineering. It is a result of extensive literature research in the field, which counts on empirical studies in contribution with industry. Therefore the good practices suggested in the model are intended to provide software organizations with useful ideas that they can implement to improve their requirements process.

More than a collection of good practices, the MDREPM is also an assessment model. It intends to provide software organizations with a quick way to assess their current practices in requirements engineering. This is done through a questionnaire that evaluates whether the good practices in the model are fulfilled by the organizations or not, and by graphical representation of the assessment results.

Finally, the MDREPM also intends to provide software organizations with a step by step process improvement path towards a better requirements engineering process. This is done by organizing the good practices in the model in different levels, and also indicating dependencies between practices that should be considered when deciding to implement them.

In order to get an understanding of what motivated the creation of the MDREPM, the following section describes the problem it addresses. Later sections describe its organization and how it can help software organizations to improve their MDRE process.

The Different Perspectives in Software Product Development

Software product development is characterized by the presence of different perspectives, mostly due to internal and external stakeholders present in the development organization. It is a complex environment where interests from different parties play a role in decision making, and where a good interaction between them becomes crucial for maximum business benefit.

Figure 1 below summarizes some of the different perspectives that can be typically found in many software product development organizations. It also shows some of the needs they have in order to perform in a market-driven environment.

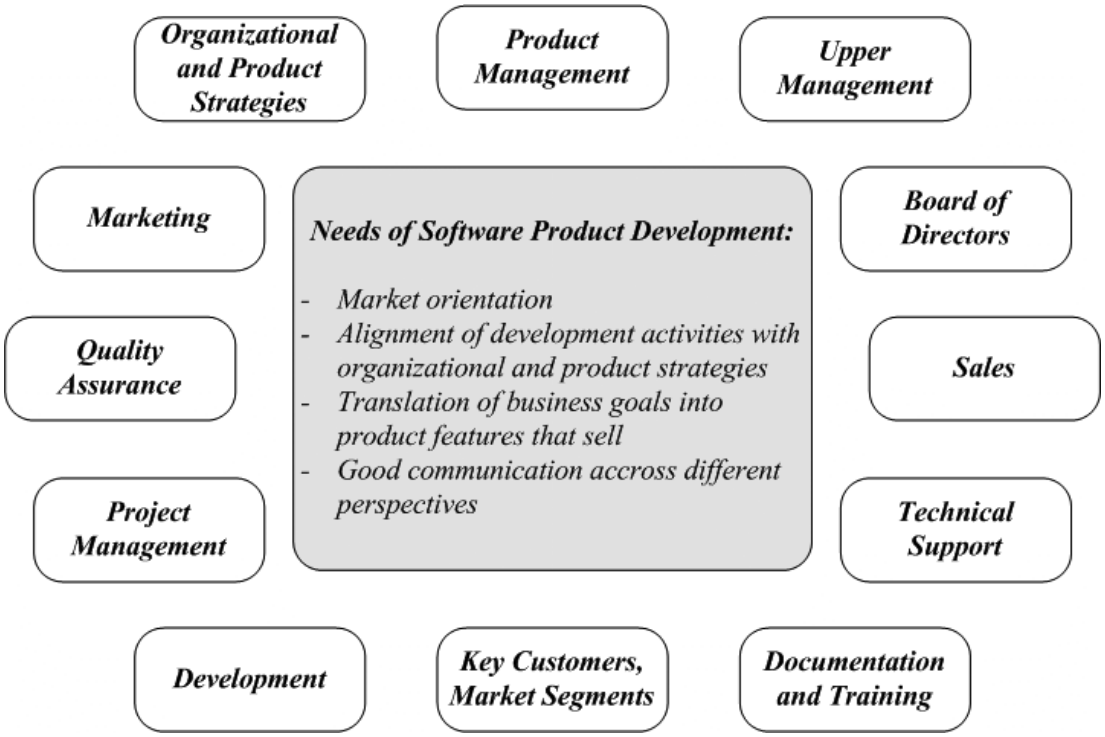

Figure 1 - Different perspectives in software product development

In a market-driven context, a strategic mindset in the organization can help by positioning it in the market in ways that put it in a good situation with respect to potential competitors. This can take place by defining organizational strategies, which are then refined through product strategies. The strategies set the context upon which the rest of the activities throughout the organization can be executed. Provided that the strategies are correct, aligning other activities with them should then help the organization achieve its goals.

From a strategy perspective towards actual product development, many stakeholders get involved. For example, board of directors, upper management, product management and marketing may be involved in strategy development. Once strategies are defined, the concern shifts to finding out which software products to develop, and which features to include in each product. This is the point where activities like market and competitor analysis can come to help. At this point, it is important to know what market needs are, what competitors have to offer, and what can be done to overcome those offers. For these activities, interested stakeholders can be, for example, marketing, sales, product management, and key customers.

Continuing with this analysis, when products and product features have been defined, the next step is to start their actual development. At this point, technical know-how is of great relevance. However, the management side is of no less importance. Product development usually takes place in projects; therefore project management is also a central area of concern. In addition to that, verifying that the product has met its quality goals before it is released is also important for ensuring it is going to deliver the expected value to customers. Areas of concern in this step relate to development, project management, quality assurance, documentation and training, and technical support.

The above description opens room for some questions:

- How to define organizational and product strategies?

- How to use those strategies to guide software product development, thus transforming business goals in actual products?
- How to identify what markets an organization can explore? How to identify the needs in those markets? Or how to create needs in those markets?
- How to balance the interests of so many different stakeholders involved in the process, in ways to bring them together towards a common goal?
- Having identified what product features to implement, how to distribute them in different product releases?
- How to balance the clash between limited resources, product features to implement, and time to market?

These questions reveal a bit of the complexity of software product development. Organizations in this business are faced with challenges related to communication across different stakeholders, as well as to how to translate market needs into actual features that can bring appealing selling points to their products.

It comes as no surprise that software product requirements lie at the core of the solution to solve the issues presented above. Requirements are the least common denominator across the different stakeholders presented in Figure 1. They can be derived from product strategies, they can be elicited from market and competitor analyses, or they can simply be invented within the organization itself. As the process of refining high-level product strategies into actual product features takes place, requirements are everywhere as a common language. They bring together areas like development, testing, project management, and marketing.

Requirements are also the means for preventing organizations to expend their resources in the development of product features that are not relevant from a sales perspective. That is, provided that the right requirements are elicited and chosen for implementation, development can focus on them to deliver maximum product value. Therefore, having a proper process to handle requirements can help organizations to focus their activities on developing products that have higher chances of success.

Even so industry practitioners often acknowledge the importance of requirements engineering, it is not always easy to get started with it due to its complexity. This, along with the challenges organizations face in a market-driven context, has motivated the development of the MDREPM. In addition, another driver was the realization that research findings in market-driven requirements engineering are scattered throughout many different sources. The idea was to concentrate them in one place, in the form of good practices that can be readily used by industry practitioners.

The following section describes how the MDREPM is organized and how it can help organizations to overcome the challenges of market-driven software development.

MDREPM - An Overview

Figure 1 above depicted the different perspectives that exist in software product development. In addition, the discussion in previous section referred to the challenges faced by market-driven organizations, and how requirements engineering can help overcoming them.

Based on the idea that a good process to handle requirements is at the core of successful software product development, the creation of the MDREPM took place

by considering the applicability of requirements practices to a market-driven situation.

This was done by first identifying what the main process areas are of interest in market-driven software development, and where requirements engineering could come to help. The analysis of the characteristics of market-driven software development led to the identification of the following process areas:

Organizational Support

Software product development has a strong market focus, which demands organizations to have an outward look towards markets, competitors, as well as opportunities and threats that may arise from those. On the other hand, an inward look towards the organization itself is also important, for example to foster innovative thinking. Ideas of new products or new product features that are created within the organization can potentially become a success if released in the market place.

In addition, roles and responsibilities to conduct activities related to marketing, product management, and requirements engineering are also relevant to ensure that ideas for product features get translated in actual software products by development personnel.

These organizational aspects are needed to support the execution of a requirements process. Therefore, the process area Organizational Support has been identified for the MDREPM. It contains several practices that software organizations can perform in order to give a strong market and strategic orientation to their businesses. In addition, this process area is also concerned with practices needed to setup the foundation for the development of a requirements process.

Release Planning

Software product development is characterized by usually short time-to-market, which often clashes with resource availability needed to implement desired product features. In addition, the process of deciding which features get implemented and when can be difficult to perform given the sometimes conflicting interests of different stakeholders.

This is where the process area of Release Planning of the MDREPM comes to help. This process area is dedicated to provide organizations with practices on how to prioritize requirements, which factors to consider when selecting them, and how to improve requirements selection process.

Requirements Management

Market-driven organizations are usually faced with high requirements influx, which can come from many different sources. These requirements are not only of interest for developers who will implement them, but also for many other roles. For example, a product manager will be interested in a high level product feature when deciding whether it should be selected for implementation in a release. On the other hand, developers will need more details about the functionalities that compose such feature in order to develop them properly. Therefore, the way requirements are specified will define their audience.

Requirements management takes care of that: the procedures to specify requirements so that they are understandable by different audiences. In addition, it is also concerned with controlling changes to requirements, controlling their versions, and providing proper tool support for managing requirements attributes, their lifecycle, as well as the relationships between them.

Requirements Elicitation

In a market-driven situation, potential customers can be as few as a dozen of known key customers, or as many as in a mass market. Therefore, finding out which requirements should be implemented in a certain product is challenging.

The Requirements Elicitation process area in the MDREPM contains good practices that can be used to identify what requirements sources can be considered, and techniques for eliciting requirements from them.

Requirements Analysis

As requirements are elicited, they can be specified in varied levels of detail and quality. Some requirements may be too poorly described that they can't be let go further without fixing problems in them. Other requirements may look good at a first glance, but may hide ambiguity that can cause misinterpretations later.

Requirements influx may also be high; in which case the amount of requirements received is more than what can be analyzed.

Requirements Analysis is the process area of the MDREPM that is concerned with the issues above. It contains good practices to aid organizations in assuring the quality of their requirements, and to also help managing a high requirements influx.

MDREPM as the Bridge between Different Perspectives

Previously, an introduction to the different perspectives in software product development was presented. In that discussion, communication across the different perspectives was commented to be important in order to have a positive outcome, having them all working together towards common goals.

The MDREPM was designed in order to foster the above. That is, the MDREPM contains suggestions of good practices that can be implemented by software organizations that can bring the different perspectives together through the use of an established requirements process.

Figure 2 below shows how the MDREPM intends to bridge the distances between all different perspectives. The way this is achieved is explained after the figure.

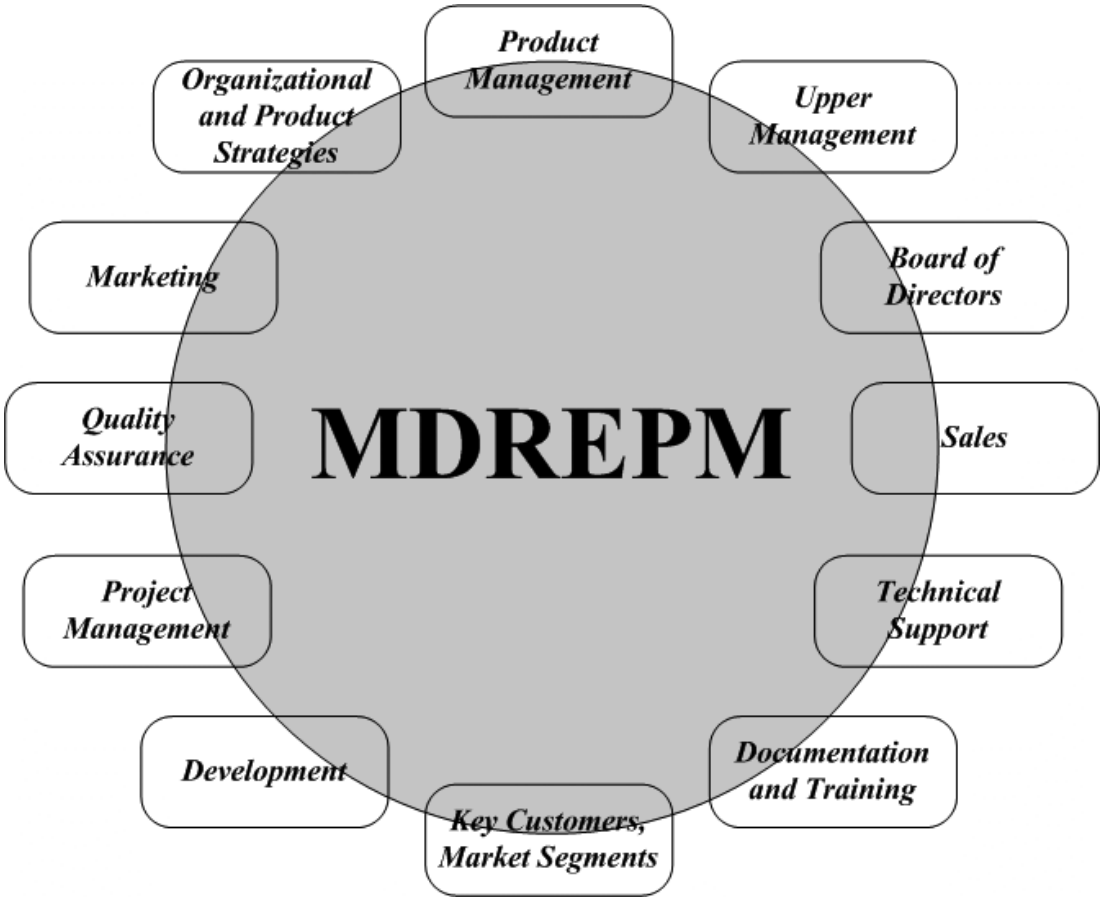

Figure 2 – MDREPM as the bridge between different perspectives in software product development

The process areas in the MDREPM are cross-functional in nature when looked at together.

Organizational Support intends to bring upper management, board of directors, product management, and marketing together by spreading a strategic mindset throughout the organization. This process area focuses on strategic planning as a way to align activities performed in other areas of the organization under a common framework set out by organizational and product strategies.

Requirements Management lays out the foundation upon which the requirements process will be executed. This process area does that by suggesting practices on how to specify requirements and managing them. It provides the structure needed for issuing requirements, accessing them for purposes such as analysis, prioritization and selection for implementation in different releases. This in turn is of interest for development, project management, and testing.

Release Planning, in turn, helps bridging the distance between perspectives by defining which features will be implemented at what point in time, and who will get them. This is of interest for key customers, product and project management, marketing, development and testing.

Requirements Elicitation has a crucial role of finding out what requirements can be considered for implementation. It therefore sets out the stage where all subsequent requirements and development activities will take place.

Finally, *Requirements Analysis* helps by handling requirements overload and by assuring requirements are of good quality. In this way it helps ensuring the work carried out, and also expected by many stakeholders can be done by relying on requirements as the common denominator among them.

Model Architecture

The architecture of the MDREPM consists of the following elements:

- Process areas: a process area defines areas of a market-driven requirements process. It is a holder of sub-process areas, as well as good practices, all of which are related to the main goal of the process area.
- Sub-process area: this is a more detailed grouping of practices that are related to each other, but that are ultimately also related to parent process area.
- Good practice: a good practice is a description of a procedure that organizations can consider to implement in order to improve their requirements process. The concept of a "good practice" per se is relative. They are called as such in the MDREPM because they have been identified from literature research. However, whether a practice is good or not to be implemented in a particular case is up to the organization to decide.
- Maturity level: a maturity level has the purpose of grouping practices according to the recommended order in which they should be implemented. Practices from different process areas may be grouped in the same level. Practices in higher levels depend on the implementation of practices in lower levels.

Figure 3 below provides an example of the aforementioned structure by showing a snapshot of the MDREPM.

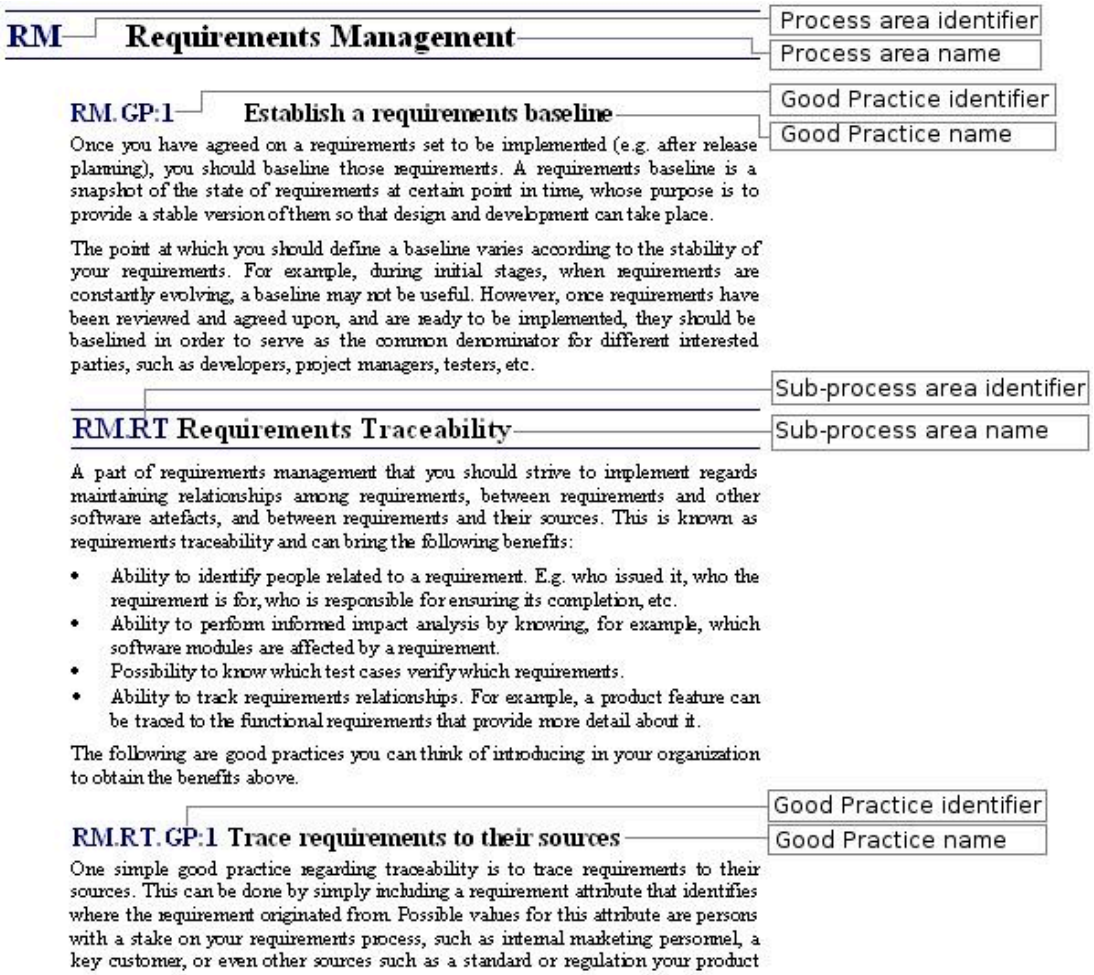

Figure 3 – MDREPM model structure

The figure shows some elements of the model. Each process area and sub-process area has a unique identifier that is composed by the initials of their names. In addition, good practices also have an identifier, which is composed by the initials of the process area (and optionally sub-process area), plus the suffix ".GP:x", where x is a sequential number within the process area.

Different Views: Practices by Process Area and by Level

The MDREPM can be consulted in two different ways. One way has been shown in Figure 3. This shows good practices according to their grouping in process areas and sub-process areas.

An alternative view shows practices according to the level of process maturity they belong. The MDREPM organizes practices in 5 different levels. The goals and main contents of each level are described below.

Level 1

The goal of Level 1 is to lay out the ground for the creation of a market-driven requirements engineering process. This is done by introducing practices from Organizational Support, Requirements Management and Requirements Elicitation.

By implementing practices of Level 1, organizations are expected to acquire a strategic and market orientation to their businesses. In addition, they also commit to implement a requirements process by introducing tool support for requirements management and assigning roles and responsibilities to ensure requirements-related activities get executed. Some basic elicitation techniques are also introduced in this level.

Level 2

Level 2 builds upon Level 1 by provoking the organization to attain a deeper awareness about its own strengths and weaknesses. In addition, it intends to make it get a better understanding of where it stands with regards to internal and external stakeholders and competitors. Development of product strategies is also recommended as part of practices in Organizational Support.

Level 2 also introduces for the first time practices in Release Planning and Requirements Analysis. Moreover, Requirements Management builds upon practices of Level 1 by introducing version and change control for requirements and basic traceability. Requirements Elicitation introduces more advanced techniques for eliciting requirements.

Level 3

The main goal of Level 3 is to take the strategic orientation acquired with the definition of organizational and product strategies in previous levels a step further. This is done by provoking the organization to better understand what customers and/or markets judge as important for their products.

Strategies are also used to guide release planning activities. These activities are also improved by introducing practices for requirements prioritization.

In addition, requirements management introduces practices for making requirements understandable for different audiences and usable for different purposes (like prioritization, design, development and testing).

Level 4

Level 4's main contributions are the introduction of advanced release planning practices. These are related to advanced prioritization of requirements which consider dependencies between them and different stakeholders' importance. Organizational Support contributes by introducing a practice for product portfolio management.

Level 5

In the top level of the MDREPM, advanced practices in release planning and elicitation are introduced. These consist of taking a pro-active approach to improve decisions on requirements selection by making post-mortem analysis of previous decisions in release planning. Elicitation activities are now guided by product strategies, and organizational support is improved by motivating the organization to foster the development of a creative environment.

Criteria for Practice Distribution in Levels

The main criteria used to distribute practices in different levels are the dependencies between them. Practices in higher levels depend on practices in lower levels, and sometimes on practices on the same level as well.

The goal of the levels is to provide a step-by-step process improvement path towards a robust market-driven requirements engineering process. However, it is acknowledged that not all practices will be useful for all organizations. Similarly, it is acknowledged that some organizations may already implement practices that are in higher levels of the MDREPM without implementing practices in lower levels.

Nonetheless, the goal of the model is to provide organizations with a broad view of what they can consider in order to strengthen their ability to handle requirements for their software products. Therefore, the placement of practices in levels in the MDREPM does not purpose to enforce a certain implementation order, but rather *recommending* a particular order based on identified dependencies between practices.

The Assessment Process

The assessment process using the MDREPM consists of a questionnaire and result presentation in graphical form. The assessment is performed through an interview with interested organizations, in which questions are asked to know whether they fulfill practices stated in the MDREPM.

More questions follow up the assessment questionnaire to gather feedback on the usefulness of the model for their case, as well to gather criticism about the model. The assessment questionnaire and the rules pertaining to the assessment are described in more detail in Appendix II.

Can the MDREPM help your organization? How?

The MDREPM was developed by having software product organizations in mind. Its goal was to gather a collection of good practices in market-driven requirements engineering in a single place. Moreover, the goal was also to structure such practices in a recommended order of implementation, by placing them in different levels and indicating, for each practice, which pre-requisite practices the organization should strive to implement first.

However, given the large variety of businesses in software product development, it is very difficult, if not impossible, to come up with a model that can contemplate the needs of all. The MDREPM, therefore, does not attempt to be a massive model that would try to fit every possible case. Rather, it has been conceived having in mind the most common needs that software product organizations face when handling requirements for their products. For example, the needs for market orientation, strategic thinking, and processes to elicit, analyze, deliver, and manage software requirements.

Therefore, the aim of the MDREPM is to help organizations to realize the scope of the activities involved when tackling market-driven software product development. By suggesting a collection of requirements practices divided according to process areas and levels, the model aims to provide guidance to organizations so that they can improve their way of handling requirements.

The model helps organizations by first providing a way to make an assessment of its current practices in MDRE against the practices in the model. Once assessment results are collected, they can be visualized in a graphical representation, which can then be used to point out which practices have not yet been implemented.

It is expected that the practices in the MDREPM will help organizations to be aware of areas they may be lacking expertise, and also point out which ones they are in most need for improvement. Revealing problem areas during the assessment is the first step towards process improvement, which can be tackled by considering practices that have not yet been implemented.

The next sections show the two views of the MDREPM. First, the view of practices by process area is shown. Next, practices are shown in their classification by level.

MDREPM – Practices by Level

This section shows the classification of practices of the MDREPM according to levels. Practices are grouped by process area and sub-process area within each level. Depending on the goals of the level, and on the dependencies among practices, some (sub) process area may sometimes not have any practices at all in certain levels.

Details about each practice can be found from next section onwards, where they are presented according to the process areas to which they belong.

Level 1
OS Organizational Support
OS.GP:1 Define roles and responsibilities
OS.GP:2 Maintain the requirements database
OS.GP:3 Assign a person to manage and own the requirements process
OS.GP:4 Train people in requirements engineering
OS.GP:8 Create a company-wide glossary of terms
OS.S Strategic
OS.S.GP:1 Define organizational strategies and introduce a strategic mindset
OS.M Marketing
OS.M.GP:1 Establish a marketing organization
OS.M.GP:2 Establish a product management organization
RP Release Planning
RP.P Prioritization
RM Requirements Management
RM.GP:1 Introduce tool support for requirements management
RM.CM Configuration Management
RM.RT Requirements Traceability
RM.RT.GP:1 Trace requirements to other software artifacts
RM.RS Requirements Specification
RM.RS.GP:1 Specify requirements attributes
RE Requirements Elicitation
RE.GP:1 Distinguish between end-user and customer
RE.GP:2 Identify requirements sources
RE.GP:3 Train personnel in eliciting requirements
RE.T Techniques practices
RE.T.GP:1 Elicit requirements through use cases
RA Requirements Analysis

Level 2
OS Organizational Support
OS.GP:5 Create a cross functional team to analyze and specify requirements
OS.GP:6 Establish a teamwork mindset
OS.S Strategic
OS.S.GP:2 Define product strategies
OS.S.GP:3 Perform periodical strategic planning meetings
OS.M Marketing
OS.M.GP:3 Know your own strengths and weaknesses
OS.M.GP:4 Identify and analyze competitors
OS.M.GP:5 Identify stakeholders and map their influence
RP Release Planning
RP.GP:1 Understand the challenges of release planning and define which of them to address
RP.GP:2 Use computer support for release planning activities
RP.P Prioritization
RP.P.GP:1 Perform systematic requirements prioritization
RM Requirements Management
RM.CM Configuration Management
RM.CM.GP:1 Have a change control process in place
RM.CM.GP:2 Define a cross-functional change control board
RM.CM.GP:3 Control versions of your requirements
RM.CM.GP:4 Make use of a requirements baseline
RM.RT Requirements Traceability
RM.RS Requirements Specification
RM.RS.GP:2 Track status of requirements
RM.RS.GP:3 Provide an initial priority and effort estimate to newly specified requirements
RE Requirements Elicitation
RE.GP:4 Use up-to-date market analysis information for elicitation
RE.T Techniques practices
RE.T.GP:2 Elicit requirements through user groups
RE.T.GP:3 Elicit requirements from product reviews
RA Requirements Analysis
RA.GP:1 Provide well informed effort estimates

Level 3
OS Organizational Support
OS.S Strategic
OS.S.GP:4 Spread strategic thinking throughout middle-management
OS.S.GP:5 Define a roadmapping process
OS.S.GP:6 Let requirements affect product strategies when applicable
OS.M Marketing
OS.M.GP:6 Identify critical success factors for specific markets and/or key customers
OS.M.GP:7 Identify the basis of the competitive advantage
RP Release Planning
RP.GP:3 Involve different perspectives in release planning
RP.GP:4 Use product strategies and product roadmap to guide release planning
RP.GP:5 Plan more than one release at a time
RP.P Prioritization
RP.P.GP:2 Prioritize requirements based on their abstraction levels
RP.P.GP:3 Prioritize requirements based on cost, value and risk
RM Requirements Management
RM.CM Configuration Management
RM.RT Requirements Traceability
RM.RT.GP:2 Trace relationships between requirements
RM.RS Requirements Specification
RM.RS.GP:4 Specify requirements in multiple abstraction levels
RM.RS.GP:5 Workup requirements
RM.RS.GP:6 Record reason for rejecting requirements
RE Requirements Elicitation
RE.GP:5 Create elicitation channels for requirements sources
RE.GP:6 Consider strengths and weaknesses of competitor's products
RE.T Techniques practices
RE.T.GP:4 Elicit requirements through personas
RA Requirements Analysis
RA.GP:2 Perform requirements triage
RA.GP:3 Analyze requirements using multiple views
RA.GP:4 Increase the quality of the requirements through inspections and reviews
RA.GP:5 Create requirements-based test cases during analysis
RA.GP:6 Perform requirements risk analysis
RA.GP:7 Track requirements dependencies

Level 4
OS Organizational Support
OS.S Strategic
OS.M Marketing
OS.M.GP:8 Manage your product portfolio
RP Release Planning
RP.GP:6 Consider product evolution during release planning
RP.P Prioritization
RP.P.GP:4 Consider requirements dependencies during prioritization
RP.P.GP:5 Consider multiple stakeholders during requirements prioritization
RP.P.GP:6 Let stakeholders' importance be a factor during prioritization
RP.P.GP:7 Re-plan releases upon changes
RP.P.GP:8 Make use of must and wish lists
RM Requirements Management
RM.CM Configuration Management
RM.RT Requirements Traceability
RM.RT.GP:3 Trace requirements to their sources
RM.RS Requirements Specification
RE Requirements Elicitation
RE.T Techniques practices
RE.T.GP:5 Elicit requirements through gap analysis
RE.T.GP:6 Elicit requirements through customer value analysis
RA Requirements Analysis
RA.GP:8 Perform impact analysis for requirements
RA.GP:9 Consider non-functional requirements during analysis

Level 5
OS Organizational Support
OS.GP:7 Create an environment that fosters innovative thinking ***OS.S Strategic*** ***OS.M Marketing***
RP Release Planning
RP.GP:7 Perform release planning post-mortem analysis RP.GP:8 Even out customer value between different releases ***RP.P Prioritization***
RM Requirements Management
RM.CM Configuration Management ***RM.RT Requirements Traceability*** ***RM.RS Requirements Specification***
RE Requirements Elicitation
RE.GP:7 Align elicitation activities with product strategies RE.GP:8 Let the importance of market segments guide elicitation efforts ***RE.T Techniques practices***
RA Requirements Analysis
RA.GP:10 Make use of personas for analysis

MDREPM – Practices by Process Area

This section presents the practices of the MDREPM according to the process areas.

The order in which the practices are shown is ascending by level. That is, when reading the practices in sequence in each process area, they are presented from the lowest to the highest level they belong in. This provides the reader with a reading sequence that resembles the order of implementation of the practices recommended by the MDREPM.

OS Organizational Support

OS.GP:1 Define roles and responsibilities

Having clear roles and responsibilities is a good start for a requirements process. This provides ownership and accountability, which helps enforcing the execution of important activities.

In a market-driven requirements engineering process, roles and responsibilities can be extensive depending on the size of the organization and the type of projects it executes. The following questions are listed to help you think about the case with your organization.

- Who issues requirements?
- Who specifies requirements?
- Who identifies what features the product should have to be competitive?
- Who decides on what product features should be implemented?
- Who decides on the release schedule/strategy of the product?
- Who is in charge of handling changes to existing requirements?
- Who analyses requirements for implementation feasibility? Are those the same who estimate the implementation effort?
- Who verifies whether requirements have been fulfilled?
- Who owns the requirements process in the organization?
- Who assures that the output of activities like market and competitor analysis are actually used by technical management?

The list above details some of the responsibilities of a requirements process and questions who is in charge of performing them all. You can consider these questions as a way to find out whether your organization is giving the right attention to the activities or not.

If you judge activities like the above important for your case, but your organization does not have assigned roles to perform them, you may run the risk of having them not properly executed.

OS.GP:2 Maintain the requirements database

The requirements database, be it a database per se (e.g. of a requirements management tool) or a spreadsheet, needs to be continuously and frequently managed. Maintaining the database involves activities like:

- Screen the database for newly issued requirements, and making an initial analysis of them
- Oversee requirements progress through their lifecycle
- Allocating requirements to a release
- Annotating requirements with priorities, effort estimates, and other attributes, and keeping them up to date
- Keeping traceability among requirements and between requirements and other artifacts (e.g. test cases, software design modules)

The list above is not exhaustive, but gives an idea of the tasks needed for maintaining the database. This could be done by e.g. a product manager, a requirements analyst, or project manger, depending on the organization. Regardless of who maintains the database, the point to stress is that maintaining the

requirements database shall be done constantly and often, preferably on a fulltime basis. If this is not done, it is likely that the following may occur:

- The database will deteriorate with the time.
- If the requirements influx is high, the database will soon be flooded with requirements that cannot be all analyzed, which can lead to important requirements being overlooked and forgotten in the requirements mass.
- Requirements statuses will lose their value if they are not updated constantly
- Outdated priorities will lead to implementation of wrong requirements
- Outdated effort estimates for requirements will cause schedule delays and cost overruns
- The likelihood of turning back to an ad-hoc requirements management becomes high.

Recommended pre-requisite practices:

- *RM.GP:1 Introduce tool support for requirements management*

OS.GP:3 Assign a person to manage and own the requirements process

A requirements process is seen here as a group of activities to handle requirements, which is performed by a group of people in the organization, with well-defined expected outcomes, in order to:

- Discover what requirements to implement
- Plan requirements for implementation in different releases
- Manage changes to requirements and control their versions
- Know which procedures to use to verify requirements
- Define where and how to specify requirements
- Track status of requirements

The MDREPM contains several practices to aid you in defining a requirements process for your organization. However, the point of this practice in specific is that you should assign a person to manage the requirements process. This person will be the one in charge of assuring that all activities will be executed properly, and that necessary tools, training, and templates are in place to support the process. In addition, this person can also be in charge of making adjustments to the process, so that it will adapt as needs arise, in a constant improvement fashion.

Having a person to manage the requirements process may mean the difference between a working process and a “shelfware” process (which is forgotten, outdated, and not performed). Besides, given the fact that the requirements process is the point where different functional areas intersect in a software development organization, it is important that the process is well-managed so that it does its job of bringing them together, aligning them towards fulfilling the organization’s business goals.

OS.GP:4 Train people in requirements engineering

Requirements engineering is a difficult endeavor, to the same extent it is crucial for organizations. Therefore, it needs to have people with experience in performing tasks like requirements elicitation, analysis, management, and release planning.

Once a requirements process has been established, training of personnel is essential for assuring activities work as planned. Therefore, preferably perform training sessions with all staff with a stake in the requirements process to explain their responsibilities and their rights.

Educating developers, testers, project managers, marketing personnel, and so on, can be done through examples. Provide examples of specified requirements, pointing out characteristics such as attribute meanings and why they exist, how to specify a requirement in order for it to be unambiguous and testable, and so forth. If you adopted a requirements management tool, you can provide tool training for maximizing the benefits the tool can bring to your organization as well.

In addition, you can make personnel involved with requirements engineering, especially developers, aware that their work is done ultimately for achieving business goals, and not just for overcoming technical challenges. This is important to establish a common understanding of where the focus of the work should be put on.

Finally, you can utilize training sessions to spread a teamwork mindset throughout all staff with a stake in the requirements process, in which cooperation, respect and understanding for each party's interests become part of their daily work on requirements.

Related practices:

- *OS.GP:1 Define roles and responsibilities*
- *OS.GP:6 Establish a teamwork mindset*

OS.GP:5 Create a cross functional team to analyze and specify requirements

The tasks of requirements analysis and specification should be done with the help of personnel from different backgrounds, in what here is suggested as a cross-functional team. The purpose of this practice is to assure that requirements are well understood by all internal stakeholders. Typical stakeholders within a product company are marketing personnel, developers, software architects, testers, and project and product managers.

Requirements should work as the common language among all these stakeholders. By that it is meant that they should convey what a product goal is to decision makers (e.g. product managers), what features a product release will have (which is of interest to marketing), what functional requirements are needed to implement those features (which is of interest to developers and project managers), and how to verify that those features were implemented correctly (which is of interest for testers).

The participation of marketing personnel or a product manager in specifying product goals and product features assures that the latter two are meaningful from a market/business perspective. The participation of developers and/or software architects as R&D representatives assures that the breakdown of product goals and features generates functional requirements that can be used for impact analysis, effort estimation and software design. Finally, the participation of testers ensures that functional requirements will be specified in a testable way, i.e. by providing measurable evidence of their fulfillment, so that testers can perform acceptance tests.

Related practices:

- *RM.RS.GP:5 Workup requirements*
- *RM.RS.GP:4 Specify requirements in multiple abstraction levels*
- *OS.GP:6 Establish a teamwork mindset*

OS.GP:6 Establish a teamwork mindset

Marketing and Technical Development organizations sometimes have different views of what is important regarding product requirements. For example, marketing

sees requirements from a selling perspective, whereas development sees them from a technical perspective.

This may lead to a clash regarding technical feasibility and importance of requirements for marketing. For example, development can deny the implementation of some requirements that marketing wants, claiming technical infeasibility (e.g. the product architecture cannot support them).

This could lead to a situation where both parties lose. By being too resistant to marketing wishes and making things in its own way, development can implement a product in time, but which is a failure in terms of successful sales due to missing features proposed by marketing. By pressuring development with unrealistic demands, marketing will see its product being delivered late in the market window. Worse, this will also lead to cost overruns.

In order to fix problems like the above, all parties involved shall be aware that they are together in a joint effort of putting the best product out in the market. For that to happen, a teamwork mindset shall exist, and energies shall focus on understanding each party's side, analyzing different possibilities, making compromises, and not on fuelling adversarial relationships.

This practice could be implemented by performing workshops to develop an understanding of each team's tasks, challenges, and problems faced. Cross-functional team building activities can also help strengthening relationships, in ways to create a feeling of one bigger team striving towards common goals.

The output of activities like above could be documented as, for example, product roadmaps and product strategies. These should then reflect both marketing and development perspectives, and work as a guide to requirements engineering activities.

OS.GP:7 Create an environment that fosters innovative thinking

Many market-driven organizations reach success due to technological innovations. Assembling a group of employees and telling them to invent new features will probably not succeed. Instead your organization can create environment that fosters innovative thinking. However, it is not enough to only create an innovative environment, the innovations need to be picked up and conformed into solid business ideas as well. This usually requires a great deal of interaction between developers and managers. Different ways to aid this are exemplified below:

- Incentive programs for employees, or to whole teams that strive for innovation. This is a way to reward them for innovative thinking, which can include financial compensation or other benefits.
- Encouraging managers to maintain close technical contact with their employees. This in addition to innovative requirements can help both parties gain a deeper respect towards each other.
- Establish projects that developers work on for the fun of it and with limited control. Sometimes these prototype projects can turn out to be great business ideas later on. One example of this is Google Labs.
- Provide a working environment that fosters interaction between people. For example, replacing individual cubicles by rooms where team members can collaborate.
- Employees can become more innovative with managers who have a more open and more informal management style.
- Enable employees to voluntarily choose the assignments or projects they want to be assigned to.

OS.GP:8 Create a company-wide glossary of terms

It is often the case that different people have different views on specific terms within the same organization. Take as an example the term “requirement priority". What does it mean: urgency to deliver or requirement importance? Or is it a combination of the two? As another example, consider the term “release date”. Does that mean the date of delivery of the release to customers? Or is it to the testing department?

These are examples that illustrate the need for a common terminology across different functional areas. Having such definitions available and easily accessible will help prevent misunderstandings during discussions such as in release planning meetings.

OS.S Strategy

OS.S.GP:1 Define organizational strategies and introduce a strategic mindset

A market-driven organization has several responsibilities such as paying attention to competitors, to markets, and to opportunities and threats generated by them. It should therefore think strategically in order to position itself in ways of becoming proactive rather than reactive in the market. A strategic planning mindset is important to make the organization aware of the context it finds itself in, and how it should proceed to gain a long term position in the market place where it has an advantage over its competitors. These advantages should not only come from technical competence, but also from the ability to deliver products that add value to customers.

This practice has the purpose of introducing a strategic mindset in your organization. At the initial state the intent is not to create a strict formal process but rather lay the foundation for strategic planning through introduction of a strategic mindset. This can done by defining where your organization is *now* (e.g. looking at the past as well as present state), *where* it should be heading (e.g. target market segment), *how* it should proceed to get there (e.g. managing changes in the organization). These definitions can be on an abstract level, but still they should be well thought through since they will set the direction for the organization as well as for several other activities that align themselves with it. At the early stages of strategic planning, the formalization of the activities is not a must; however as your organization maturity grows there is a need to formalize them. Therefore, you should preferably formalize the strategic activities and its outcome from the beginning, in order to save time in the long run.

The initial definitions of the *now*, *where*, and *how* provide the bases for the definition of your *organizational strategies*. In other words, they are the bases for defining the overall purpose and scope of your organization as well as how value will be added to the different products in your organization.

The approach described so far takes a top down approach. However, as the maturity grows a mixture between a top down and bottom up approach can be adopted. For example, this can be done by exploring your core competences and capabilities (stretching existing ones or adding new ones).

Related practices:

- *OS.S.GP:2 Define product strategies*

OS.S.GP:2 Define product strategies

The definition of product strategies should follow up from the definition of organizational strategies. The latter provide the context in which product strategies are defined. That is, once you know where your organization is *now, where* it should head, and *how* it will do that, you can then go into more detail and define strategies for the products the organization will use to realize its organizational strategies.

Defining product strategies is closely related to product portfolio management, market and competitor analyses, to knowing your organization's and its competitors' strengths and weaknesses, and the opportunities and threats posed by external (environmental) and internal factors. For example, you may want to introduce a new product to your portfolio to capture a new market segment that can now be explored because your organization masters a new technology. This takes into consideration your current product portfolio, market and competitor analyses, as well as the analysis of your strengths (new technology), and opportunities that you see with the use of such technology.

The definition of product strategies should be done in ways to support organizational strategies. For example, if organizational strategies are directed toward increasing market share, product strategies may be defined in a way to foster the development of new product features that can help gaining that share.

Product strategies should also define *how* they will be achieved. This consists of choosing which customer targets to focus on (e.g. existing customers, or new customers); the competitive targets (e.g. first competitor, second competitor); and the differential advantage of the product. The above can be discovered by activities such as market segmentation, customer analysis, and competitor prioritization. The output of this should then be compared to the organizational strategies to define the product strategies.

Finally, one important aspect of product strategy definition is to choose *when* things will happen. A common way to define this is through a *product roadmap*. A product roadmap draws a timeline of when things will happen and combines different aspects of product strategies, such as commercial and technological. The MDREPM contains a practice named *Define a roadmapping process* which discusses roadmaps further and their relation to product strategies.

The benefits of defining product strategies are many. Not only can it provide a long-term view for the products of your organization, which fosters a proactive rather than a reactive approach, but it also guides you through requirements selection. When the amount of requirements to be handled is large, requirements can be compared against product strategies in order to decide whether they should be taken for further analysis or whether they can be discarded upfront, in a technique known as *requirements triage*. This comparison can take different perspectives, when considering for example competitive targets and target customers. E.g. if strategies state that A is a target competitor that has feature X in its product, that feature may need to be considered in your product too, and thus requirements related to it should be accepted. On a more concrete example, if your product strategies state you are to increase your share in the domestic market, you could then discard a requirement related to multiple language support, since that would be interesting in case the strategies were related to international expansion.

Related practices:

- *OS.S.GP:6OS.M.GP:6 Identify critical success factors for specific markets and/or key customers*
- *OS.S.GP:6OS.M.GP:7 Identify the basis of the competitive advantage*

- *OS.S.GP:6 Let requirements affect product strategies when applicable*
- *OS.S.GP:6OS.M.GP:8 Manage your product portfolio*
- *RA.GP:2 Perform requirements triage*
- *RPRP.GP:4 Use product strategies and product roadmap to guide release planning*

OS.S.GP:3 Perform periodical strategic planning meetings

Strategic planning is the activity in which long term goals are set for the company and its products. It considers current and forecasted market situation, and positions the company and its products in relation to them. The positioning consists of defining or updating organizational and product strategies.

Strategic planning meetings are a cross-functional activity and thus should include personnel from marketing, product management, and executive management. It is recommended that these meetings be performed periodically in order to keep the company and its products aware of market situation and in a competitive position. The periodicity may vary depending on the volatility of the markets in which the company is involved, but typically it can be semestral.

The output of strategic planning is the definition of organizational, business and product strategies, which form a top-down view of how to ultimately achieve organizational goals through product sales and services.

The benefit of conducting such meetings is to prevent the company to expend its R&D resources in the implementation of product features that do not have a market appeal; rather, the focus can then be on the implementation of features that are demanded by market segments and/or key customers, or that can create such demand (e.g. through technology innovation).

An organization should strive to know where it is going and where to position its products in order to succeed. Without explicit strategic planning, product success, and ultimately organizational success, can be put at risk by subjecting the company to difficulties in generating revenues through selling products that are not what markets want.

Recommended pre-requisite practices:

- *OS.S.GP:1 Define organizational strategies and introduce a strategic mindset*

OS.S.GP:4 Spread strategic thinking throughout middle-management

Middle-management realizes the strategic directions set by the organization through operational strategies. In order to broaden the understanding of the organizational strategies and how these cascade down, middle-management can participate in periodically strategic workshops.

Managers from several different departments can participate in basic strategic formulating activities e.g. identifying their own department's strengths and weaknesses and then doing the same for the organization as a whole. The benefit of this is threefold. Firstly, a broader understanding of the organization and product strategies is developed, which creates a "this is how I fit in the big picture" view. Secondly, the strengths and weaknesses identified can be used as input when specifying the organizational strategies. Thirdly, mid-management responsibilities become more aligned with the organizational direction.

Recommended pre-requisite practices:

- *OS.S.GP:1 Define organizational strategies and introduce a strategic mindset*

Related practices:

- *OS.S.GP:3 Perform periodical strategic planning meetings*

OS.S.GP:5 Define a roadmapping process

Linking requirements engineering practices with a business view in order to create long-term planning for a product can be done through roadmaps. This way requirements engineers will take more business oriented decision when e.g. eliciting and analyzing requirements or when performing release planning. In other words product roadmaps can be used to keep focus on the long-term intentions, which are often defined on a high level (organizational strategies), as well as keeping the focus on the right issues that reflect these intentions (product strategies). This leads to that the short term solutions, which might seem correct at the time, are removed in favor for the long-term ones.

The aspects of market, product, and technology can be included in a roadmap and are often represented in a time based chart showing when in the lifecycle their activities occur. This way different viewpoints are conceived in the product lifecycle e.g. from forecast stage to business decisions.

Since roadmaps have a long term perspective and builds upon, among others, marketing information it is important to frequently update and review these. However, this is dependent on how often the market segments that your organization products target change their viewpoints.

Except from product roadmaps your organization can define technological and risk roadmaps or a combination of the three. Technological roadmaps containing evolution pace and technological forecast and risk roadmaps containing the potential problems or other factors affecting your products. These two roadmaps can be included in the product roadmap and are this way more frequently observed by its users' creating a reminder of extra risk awareness during certain phases of the product life cycle.

Below the benefits of defining roadmaps summarized:

- Roadmapping fosters communication and collaboration among different functional areas, at the same time as providing a holistic view of problems, opportunities and new ideas. A common language for the roadmaps is adopted.
- They provide a framework upon which the organization can work towards common goals. For example, marketing and sales can start preparing their activities, at the same time as development can start analyzing and implementing requirements to realize what is in the roadmap.
- Resource allocation can be aided by knowing what is in the roadmap at what point in time. Similarly, skill development programs can also benefit from that.
- Helps organizations to keep focused on right issues, by establishing a long-term view for products and technologies
- Helps customers in that they know what they can expect from a product evolution perspective

Recommended pre-requisite practices:

- *OS.S.GP:1 Define organizational strategies and introduce a strategic mindset*
- *OS.S.GP:2 Define product strategies*

OS.S.GP:6 Let requirements affect product strategies when applicable

During requirements triage, requirements are compared against product strategies in order to know whether they should be accepted for further processing or not. However, do not be unaware to new opportunities that may arise with a requirement that is not currently aligned with product strategies.

It can be beneficial to also let requirements influence product strategies when applicable. As requirements are elicited from various sources, or even invented within the organization, some new opportunities may be identified (e.g. a new technology opens room for innovative features that can capture new markets). If these requirements are not currently aligned with product strategies, do consider changing the strategies in order to reflect the newly identified opportunities. However, be cautious when doing so, since this may have significant implications in the company's business.

Before deciding about changing the strategies, make sure the new requirements can actually lead the product to success in the market place. You could do that by, for example, using market analysis results, or even conducting new analyses to assess whether it is a good idea to allow for changing the strategies.

Recommended pre-requisite practices:

- *OS.S.GP:1 Define organizational strategies and introduce a strategic mindset*
- *OS.S.GP:2 Define product strategies*

OS.M Marketing

OS.M.GP:1 Establish a marketing organization

This practice intends to establish the marketing view in your organization by assigning marketing personnel to responsibilities. A marketing organization is important for successful product development, as it is one of the main sources of requirements that reflect what market and key customer needs are. When choosing to fulfill these needs, your organization and its stakeholders should be aware of the costs and benefits this will incur, and consciously decide which ones to tackle so that benefits outweigh implementation costs.

A marketing organization is seen here as one person, or a team, whose responsibilities include, but are not limited to the following examples:

- Analyze market segments
- Market and competitor analysis
- Eliciting and negotiating requirements from key customers
- Planning product launches and marketing campaigns
- Identifying and analyzing Critical Success Factors (CSF)
- Identify bases of competitive advantage

The level of fulfillment for some of these responsibilities have a direct impact on the requirements process in your organization and are therefore reused in several of the practices in the MDREPM. The importance of the responsibilities should be assessed and possibly customized by your organization. Moreover, for some they might seem limiting and therefore you can as well assess what other responsibilities you find as important for your organization to succeed with establishing a marketing organization. Information gathered should in combination with organizational

strategies be used when defining product strategies e.g. adjustments depending on chosen market segment size, competitors, and critical success factors.

OS.M.GP:2 Establish a product management organization

A product management organization is seen here as a person, or a team, in full charge of the lifecycle of a software product, with responsibilities like the following:

- Manage product lifecycle, from strategic planning through to delivery
- Develop product strategies and document them in a product roadmap, keeping it up to date throughout the product lifecycle
- Research market needs (possibly supported by Marketing personnel)
- Write business requirements. These are seen here as product goals, which are high level, as opposed to lower level functional requirements that will be specified later based on these goals
- Bring development, marketing, sales and finances together, balancing their interests, in order to develop a product and take it to the market.
- Perform requirements triage, i.e., the constant screening of the requirements database, checking whether candidate requirements can be discarded or should be accepted for further analysis.
- Perform release planning, selecting requirements to be implemented in different product releases
- Define a release strategy

Of special interest from requirements engineering point of view is the specification of business requirements (which in turn should be aligned with product strategies), as well as requirements triage and the planning of requirements that pass triage in different releases.

Requirements triage is a technique for fast acceptance or dismissal of requirements to avoid overload, and to avoid putting effort on requirements that are irrelevant. Not all requirements will be accepted due to reasons such as non-alignment with product strategies. Hence, product management plays an important role in constantly verifying the requirements database for candidate requirements and performing triage.

Related practices:

- *RA.GP:2 Perform requirements triage*

OS.M.GP:3 Know your own strengths and weaknesses

Knowing your own strengths and weaknesses, both in the perspective of the business environment as well as your strategic capabilities, is a practice that can help you succeed in the market. It enables you to avoid the threats to your organization. Furthermore it creates an understanding of how to capitalize on your opportunities. A SWOT Analysis (Strengths, Weaknesses, Opportunities and Threats) is an example of a tool that can be used by your organization. Not only because it is lightweight and a simple, but also because it can widen the understanding of strategic planning and marketing.

The initial SWOT Analysis can be conducted by the senior management of your organization as well as marketing personnel and it should focus on just the business environment and strategic capabilities. However as the maturity of performing the analysis settles in, the scope of usage can be widened by inclusion of mid-management view of the organizations strategic capabilities. By doing so the initial top down strategic planning is enriched with a bottom up view.

The outcome of the SWOT Analysis should aid your decisions when choosing between different strategic plans by knowing to which extent your organization is able to fulfill them.

Unless the information gathered during a SWOT Analysis is compared against the same type of analysis conducted on your competitors, the value of it becomes rather limited. To combine this practice with several of the other practices in this process area are beneficial.

Related practices:

- *OS.M.GP:4 Identify and analyze competitors*
- *OS.M.GP:6 Identify critical success factors for specific markets and/or key customers*

OS.M.GP:4 Identify and analyze competitors

Your competitor's strategic intentions when marketing their products should preferably be identified and analyzed extensively both from an organizational and marketing point of view. However, the case might be that your organization is innovative and consequently be the only one in the present market. Nevertheless, it will probably be the case that you will face competitors sooner or later when the wave of technical innovations fades away. Therefore, answering the questions listed below may help you to identify and analyze your competitors.

- Who are your main competitors?
- Is there any competitor that is ahead in the technological aspect and that is planning to release a product within 6 months, or 1 year?
- If they are doing something that seems to be working, how can your organization imitate that?
- How do you preserve your strengths in a way that competitors can not imitate your organization?
- If the competitors deliver a product first to the market how can you block that attack? What features can be included in the next release to differentiate you so your competitors can not lock in the customers?
- What are your competitors' weaknesses? How can that knowledge be used without attacking them directly (Because if you do then they will learn about their weaknesses and will change thereafter.)?
- Is there even a place in the current market segment for your product?
- How are the competitors marketing their products?
- What level of resources do your competitors have? E.g. number of employees, financial situation.
- What is the customer's perspective of the competitors?
- How can you become unpredictable and not predictable to your competitors?
- Knowing the above, is it possible to create misleading strategic intentions to the eyes of your competitors?

When your competitors have been identified it is important to analyze their strengths, weaknesses, opportunities, and threats, from the perspective of your organization. By doing so, several of the hidden features that contribute to their potential success can be revealed.

However, to create a full understanding of the strategies that can be undertaken to e.g. minimize your competitors' strengths as well as enhance their threat; you must first know your own strengths, weaknesses, opportunities and threats. Check related practices below for knowing how to do that.

Recommended pre-requisite practices:

- *OS.M.GP:1 Establish a marketing organization*
- *OS.M.GP:3 Know your own strengths and weaknesses*

Related practices:

- *OS.M.GP:6 Identify critical success factors for specific markets and/or key customers*
- *RPRP.GP:8RP.P.GP:1 Perform systematic requirements prioritization*
- *RE.GP:5 Create elicitation channels for requirements sources*

OS.M.GP:5 Identify stakeholders and map their influence

When software engineers in market-driven organizations discuss stakeholders they often refer to e.g. marketing personnel, developers, software architects, testers, and project and product managers. However the stakeholders might be different when discussing with senior management, marketing and sales department e.g. financial institutions, customers, shareholders and unions. What software engineers refer to as stakeholders (except from the inclusion of customer) should all be considered as internal stakeholders while others can all be seen as external. Therefore, you should take as a good practice to identify your stakeholders as well as defining them from your perspective to minimize the risk of ambiguity.

One way is to categorize your external stakeholders according to:

- Market Environment: e.g. competitors, shareholders, key customers, partners, distributors, subcontractors, competitors, contractual commitments with customers
- Social / Political Environment: e.g. policy makers, government agencies
- Technological environment: e.g. standard agencies (ISO, 3GPP)

And some examples of internal stakeholders can follow organizational roles such as:

- Owners of the organization
- Marketing personnel
- Product manager
- Project Manager
- Software quality manager

Once you have identified stakeholders, you can proceed by mapping them according to their influence. This can be an important factor when guiding decisions about which requirements sources to consider, which requirements get higher priority, or which get selected for a release. One way to map stakeholder influence is to assign relative weights to them, which should represent their level of interest and power on the decisions you need to take regarding requirements.

Not only is the outcome a more clearer mapping of your organizations stakeholders, but this information is useful for other process areas in the MDREPM as well. For example, release planning and requirements elicitation, since these all base their discussions on your organizations strategies which are affected by the stakeholders in return. As the maturity of your organization grows, beware that the number of roles and responsibilities will also grow, and keeping definitions of the stakeholders up to date is important to get the most out of this practice.

Related practices:

- *RPRP.GP:8RP.P.GP:5 Consider multiple stakeholders during requirements prioritization*

- *RPRP.GP:8RP.P.GP:6 Let stakeholders' importance be a factor during prioritization*
- *RE.GP:8 Let the importance of market segments guide elicitation efforts*

OS.M.GP:6 Identify critical success factors for specific markets and/or key customers

Understanding what factors customers of different market segments value and to what extent, is important information. You should try to excel and outperform the competitors in identifying what is known here as critical success factors (CSF).

Some of the factors known as thresholds are not considered as CSF since they are always expected to be delivered by any organization in the same branch e.g. a mobile phone is expected to be able to receive phone calls. On the other hand a CSF for an organization producing mobile phones for a market segment, mainly consisting of young customers, can for example be that the phone offers an Mp3 player.

Remember that the strengths of your organization should only be considered as such if they target the CSF of the intended market segment, e.g. if your organizations outperforms all competitors in after sales activities but this is not critical for the customers, then it can not be consider as a strength either.

Related practices:

- *RPRP.GP:8RP.P.GP:1 Perform systematic requirements prioritization*

OS.M.GP:7 Identify the basis of the competitive advantage

Competitive advantage is not only achieved by identifying what the market segment wants and needs, but additionally the requirements preferably have to be delivered in a way that exceeds the competitors.

Customers from diverse market segments may have different views on what "value-for-money" is. You should therefore take as a practice to measure which requirements increase the perceived product/service benefits against what the price for the product is. These measurements can then be used as input when positioning the product in a specific market segment. When performing release planning these perspectives, documented in the product strategy, must be a factor in the equation. For example: Is the price of the product the main concern for our customers or are the functionality, service, packaging more important (CSF)? But more importantly what requirements will offer more benefits than what the competitors can offer and at what price?

Related practices:

- *OS.M.GP:4 Identify and analyze competitors*
- *OS.M.GP:3 Know your own strengths and weaknesses*
- *OS.M.GP:6 Identify critical success factors for specific markets and/or key customers*

OS.M.GP:8 Manage your product portfolio

Market-driven organizations usually either have a variety of products or are in the phase of adding or removing products in their portfolio. These activities should not be performed on gut feeling. Rather your organization should take as practice to

adopt a methodology for the management of the product portfolio such as Boston Consulting Groups matrix, or the Market attractiveness/product strength matrix.

The choice of which products to keep in your portfolio should be guided by:

- The balance: whether to keep a product in a market segment that is disappearing, in order to enforce the ability of your company to have a wide range of products.
- The attractiveness: how attractive are the products for different market segments?
- The fit: are the products aligned, or in other words, do they make sense together as part of the same portfolio?

For example: a product that is losing its appeal in the market or draining resources and time might be considered to be removed from the portfolio, freeing resources to focus on new ideas. On the other hand, your decision should also consider the negative consequences of removing the product from the portfolio. For example, you may decide to remove the product because sales figures are not explicitly favorable. However, just the presence of the product in the market may be a good way to keep the name of your organization known, and thus helping it to sell other products.

The rationale behind different approaches to, e.g. either removing or adding products to your portfolio should be consulted with the methodology used.

RP Release Planning

RP.GP:1 Understand the challenges of release planning and define which of them to address

Release planning is an activity full of constraints related to budget, human resources, and time. Examples of factors that may affect release planning include:

- Cost of implementing requirements
- Requirements dependencies
- Value of requirements (as perceived by customers, or internal stakeholders)
- Architectural importance of requirements
- Availability of resources
- Time to market
- Product Roadmap and product strategies
- Risks of implementing requirements
- Specific customers/market segments
- Market window
- Market size
- Product evolution

Not all factors above may be relevant to your organization. Therefore, you should decide which of them are applicable to your case, and perform your release planning activities, such as requirements prioritization and requirements selection, keeping in mind the factors you have chosen.

The MDREPM contains several practices to address the challenges above. Once you have got an understanding of the challenges of release planning and the factors that are relevant to your organization, strive to implement these practices to improve your release planning process.

Related practices:

- *OS.S.GP:2 Define product strategies*
- *OS.M.GP:6 Identify critical success factors for specific markets and/or key customers*
- *RA.GP:7 Track requirements dependencies*
- *RA.GP:5 Create requirements-based test cases during analysis*
- *RP.GP:6 Consider product evolution during release planning*

RP.GP:2 Use computer support for release planning activities

Release planning is a complex activity given the numerous factors that influence it. These factors can be roughly categorized into budget, time, technology, and resource constraints.

An example of the difficulty of performing release planning can be seen in the following example:

Different key customers, each with different importance, suggest new requirements for the next release. At the same time, internal marketing department identifies new features that need to be implemented for strengthening the product in the market, as bug reports reported by support personnel also need to be considered for the next release. The time set for the release date is short, and the amount of resources

available is not enough to cover all requests. Moreover, some of the requirements identified depend on each other, which makes it more difficult to group them as part of one release whereas keeping the value added to the product in that release in a good level.

As the example above shows, constraints coming from all sides converge into release planning. Such complexity introduced by so many factors makes it hard, if not impossible, to handle release planning manually.

Therefore, consider using computer support to perform this activity. Computer aid in release planning can be achieved by introducing tools such as spreadsheets for computing requirement priorities, or, in a more advanced way, by full fledged release planning tools, which can consider numerous factors at the same time, and provide alternative release plan suggestions. For example, Release Planner and its associated EVOLVE methodology for release planning is a suggestion of such a full fledged tool.

The use of computer support for release planning can make it more objective, as it can weigh different factors and, depending on the tool used, come up with different plan suggestions. This can be achieved by combining it with human judgment on effort estimations, requirements' value, and priorities as seen by different stakeholders/key customers.

RP.GP:3 Involve different perspectives in release planning

Release planning is the activity that selects requirements that marketing wants, that development can implement, and whose implementation can be funded by finances. Therefore, these three perspectives: marketing, development and finances, are all interested in release planning, and therefore should preferably take part in the decisions taken in it. Failing to have one of them may lead to the following situations.

- If marketing is absent, development and finances can remove expensive requirements and only implement cheaper ones. However, this may leave valuable requirements out.
- If development is absent, marketing and finances can agree on a set of requirements that is impossible to implement within the requested time.
- If finances are absent, development can agree on developing what marketing wants, but at an unacceptable cost (e.g. due to extra resources that should be added).

Therefore, it is advisable that representatives from marketing, finances and development take part in release planning in order to ensure that compromises can be made to develop reasonably strong products, at an acceptable cost, and within a realistic schedule. Their inputs can be given to activities such as requirements prioritization.

Related practices:

- *RP.GP:8RP.P.GP:5 Consider multiple stakeholders during requirements prioritization*

RP.GP:4 Use product strategies and product roadmap to guide release planning

Product strategies can be used to guide release planning efforts. They are especially important when deciding the priority of requirements. For example, if product strategies state that the product is going to be internationalized, requirements related

to multiple-language support may gain high importance as compared to others that are not immediately related to the strategies.

Similarly, using a product roadmap is important to determine the contents of a release that fulfils previous product planning. For example, if the roadmap states that the upcoming release should reinforce the presence of the product in a specific market; requirements which were discovered from that market would then gain priority over different ones.

Related practices:

- *OS.S.GP:2 Define product strategies*
- *OS.S.GP:5 Define a roadmapping process*

RP.GP:5 Plan more than one release at a time

If only one release is planned at a time, binary decisions in requirements selection are taken in the sense that a requirement either is included in the release or not. If it is not, the requirement goes back to the pool of available requirements, but with no expected date for its delivery since no other release has yet been planned. This, in turn, can dissatisfy customers who are expecting certain requirements to be delivered, as all they can be told is that the requirements will be delivered "in the future".

In order to be more forthcoming towards customers, strive to plan at least one release ahead. Therefore, when a requirement is not selected for the coming release 'n', it can be postponed and included as part of release 'n+1'. Alternatively, yet one more release can be considered: the current release project, i.e., the one undergoing development at the moment as next release 'n' is being planned. This release, which can be called 'n-1', can accommodate highly urgent requirements that have appeared after development efforts started, and that have to be implemented during the current release project. In this specific situation, some requirements may be dropped from 'n-1' and allocated in either 'n' or 'n+1'.

Related practices:

- *OS.S.GP:5 Define a roadmapping process*

RP.GP:6 Consider product evolution during release planning

In some organizations, the development of completely new products does not happen so often. In most of the cases, organizations are improving their existing products over and over through several releases. Therefore, product evolution becomes an important concern when performing release planning.

Keeping that in mind when deciding which features to implement, you should consider basically two things: (i) the current state of the product and how it can support the addition of new features; (ii) how new features will impact the existing product.

The current state of the product can be characterized by attributes such as reliability, maintainability, and modifiability. Metrics such as post-release number of defects per module can give an indication of reliability. If you decide to add new features in a non-reliable module of your product, you may run the risk of delivering an instable product. In a similar example, if you decide to add features to a module that has low maintainability and modifiability, you should expect the implementation time to be longer, and the risk of introducing new defects into the product could be higher.

Finally, you should also consider the impact new features will have in the product from an evolvability perspective. Will the new features be accommodated nicely within the product architecture? Or will they break design constraints, contributing to a complex architecture? If the latter is the case, do you think it is worth penalizing system evolvability in exchange of a feature that can sell today, but that can cause problems to the addition of other features in the long run?

Weigh the benefits you may get and the risks you will run, and make decisions on feature selection also from a product evolvability perspective.

RP.GP:7 Perform release planning post-mortem analysis

This practice intends to create an environment where measurements are gathered for later analysis in order to help to improve release planning activities. This is done by post-analyzing requirements selection quality by comparison with the measurements.

Gathering e.g. customer value, market penetration, profit, and revenue your organization can gauge the impact that different product releases had when they were released. You can then use this information to rate the quality of the decisions you took when planning previous releases, and to see what you have got right and wrong during that time. A method that can help with this is called PARSEQ (Post-Release Analysis of Requirements Selection Quality).

By knowing your mistakes as well as your right decisions of previous releases, you can then build upon them to guide your decisions for future releases.

RP.GP:8 Even out customer value between different releases

One of the main challenges of release planning is to select an optimal set of requirements. When doing so, some key customers whose requirements were not fulfilled may be disappointed with a certain release.

In order to prevent long term customer dissatisfaction because of that, you can strive to even out the requirements value over the releases to satisfy as many customers as possible, provided that this is financially feasible and aligned with your organizational strategies.

When deciding which customers to satisfy first, prior financial and marketing data can be used, e.g. profit for last release, revenue in the previous year, number of potential customers in market segment, or size of total market segment.

By keeping track of the discarded requirements for a release your organization can notify the affected key customers (if possible) and let them know that their requirements were not included in the first release, and that they are considered for the second or third one. However customer groups with high interest for a release and that have the power to affect your organization should be considered before the ones with low interest and power.

Related practices:

- *RM.RS.GP:6 Record reason for rejecting requirements*
- *RP.GP:5Plan more than one release at a time*
- *RP.GP:7 Perform release planning post-mortem analysis*
- *OS.M.GP:5 Identify stakeholders and map their influence*

RP.P Prioritization

RP.P.GP:1 Perform systematic requirements prioritization

A common fact when planning a product release is that there is a mismatch between the amount of existing requirements and the time and resources available to implement them. Therefore, requirements should be prioritized in order to rank them in a way that an optimal subset can be selected for implementation in a release project.

Making the prioritization process systematic involves defining which criteria to choose for prioritizing requirements by, that is, *you should define what a requirement priority consists of.* This may involve, for example, a combination of any of the following: requirements cost and value, risk, urgency, strategic importance, architectural importance, architectural impact, and critical success factors for customers. Some of these criteria may also be considered from the perspective of different market segments, key customers or other stakeholders. Systematic prioritization also involves the definition of which prioritization technique to use. Many techniques exist, such as Cumulative Voting, Top Ten Requirements, and 100 dollar method to name a few. Moreover, you should define who the stakeholders that will participate in the process are. These may include, e.g. marketing personnel, developers, project and product managers, and key customers.

Systematic prioritization is at the core of release planning. Requirements selection for different releases is done on the basis of the priorities set for those requirements. Therefore, making this activity explicit and systematic is essential. The purpose of doing this is exactly to prevent it to happen implicitly and by gut feeling, which poses great risks to product success. The latter form always happens regardless of you planning for it or not. For example, some requirements will be prioritized on behalf (or detriment) of others when time for implementation becomes short, and some need to be dropped to give room for others.

Not making requirements prioritization a systematic process can cause you problems. By postponing prioritization decisions rather than tackling them from the outset, you run the risk of implementing wrong requirements, e.g. the ones with least value and higher cost. As a result your product may not meet market/customer needs, therefore putting the success of your business at risk.

Related practices:

- *RP.P.GP:2 Prioritize requirements based on their abstraction levels*
- *RP.P.GP:3 Prioritize requirements based on cost, value and risk*
- *RP.P.GP:5 Consider multiple stakeholders during requirements prioritization*
- *OS.M.GP:6 Identify critical success factors for specific markets and/or key customers*

RP.P.GP:2 Prioritize requirements based on their abstraction levels

Once requirements have been specified considering different levels of abstraction, it becomes easier to prioritize them, since you can for example focus on feature level requirements, which are less numerous than their function level counterparts.

Considering that feature level requirements are abstractions of their function level counterparts, they can be prioritized among themselves on behalf of their related functional requirements. For example, a feature level requirement A that has associated function level requirements a1, a2, and a3 can be compared against

feature level requirement B, which in turn has function level requirements b1, b2, and b3. In this example, only two requirements need to be compared against each other, rather than 6 if function level ones would have been considered.

Once feature level requirements have been prioritized, you can then subject their related function level ones to prioritization as well. This could be done for the purpose of, for example, choose which ones to implement first.

In order to define which requirement gets a higher or lower priority, you can consider different aspects by which prioritize them. And different aspects can be considered for feature and function level prioritization. For example, at a feature level, aspects such as strategic importance and customer value can guide prioritization. This would help defining which requirements are more important from a business perspective. On the other hand, at function level, risk could be used as a prioritization aspect to decide which requirements should be implemented first. For example, implementing the riskiest functional requirements early in the project could be helpful to guide architectural design. An architecture that is created without considering requirements risks could prove to be faulty, hence the benefits of addressing requirements-related risks early to avoiding that.

See related practice RP.P.GP:1 for details about how to define what a requirement priority consists of.

Recommended pre-requisite practices:

- *RM.RS.GP:4 Specify requirements in multiple abstraction levels*
- *RP.P.GP:1 Perform systematic requirements prioritization*

RP.P.GP:3 Prioritize requirements based on cost, value and risk

Some common criteria by which to prioritize requirements are the cost of implementing them, the value they can add to the product (and ultimately to its customers), as well as the risk of implementing the requirements (e.g. of not getting them right in the first try).

Several prioritization techniques can be used to rank requirements using these criteria. For example, AHP (Analytical Hierarchy Process), which uses pair-wise comparisons; 100 dollar test, by which 100 units are distributed to requirements according to how they are rated as compared to others. Alternatively, rating requirements in a scale from 1 to 10 regarding their cost, value and risk can also be used.

These techniques are effective up to a limited number of requirements, such as a few tens of them. Therefore you should select a small subset of your requirements that need to go through prioritization. Leave out of this group all requirements that are a must to the product (e.g. requirements stemming from regulations, essential product features). In addition, you can also consider feature level requirements only for prioritization, since they are less in number (see *RP.P.GP:2 Prioritize requirements based on their abstraction levels* for details).

By prioritizing according to these criteria, you will be able to distinguish requirement sets that can add the biggest value to the product, at the lowest cost and risk, from requirements that only add cost (and possibly risk), whereas not adding so much value.

Related practices:

- *RP.P.GP:2 Prioritize requirements based on their abstraction levels*

RP.P.GP:4 Consider requirements dependencies during prioritization

One practice related to requirements specification is to track requirements dependencies. Once you have implemented that practice in your organization, you can then consider dependencies when performing release planning.

Even if no formal technique for prioritizing requirements that consider dependencies is used, being aware of them is important to decide the final priority of a requirement. For example, a requirement A may initially be seen as high priority given its market/customer value. However, a dependency analysis on this requirement may reveal that it requires a number of low priority ones to be implemented first. When those are considered as a part of the equation, they may keep the development team busy throughout the time available for the release project. Consequently, the resulting release value will not be as high as too many low value requirements were implicitly chosen. This may be acceptable in some cases, but if not, a different combination of requirements should be agreed upon in order to maximize the release value.

As looking at dependencies when deciding the priority of every requirement can be very difficult to implement, a suggestion is to perform an initial prioritization without considering dependencies, and only make use of them once the initial prioritization list has been obtained. Dependencies can then be considered to evaluate whether the list of requirements is feasible to implement.

Recommended pre-requisite practices:

- *RA.GP:7 Track requirements dependencies*

RP.P.GP:5 Consider multiple stakeholders during requirements prioritization

Stakeholders participating in release planning activities e.g. marketing, finance and development should also actively participate in the actual prioritization of the requirements. Each of the stakeholders follows a predefined role description that truthfully must be represented during the prioritization session.

Moreover, several of the participant roles are a combination of information sources that also affect what they consider as important factors e.g. marketing represents the wishes of several market segments maybe in combination of a few key customers while finance can have their view of the factors from the perspective they get through shipment and sales records. The different views must be represented from each stakeholder perspective if a mutual agreement should be reached on how to prioritize the requirements for one or a sequence of releases.

Related practices:

- *RP.GP:3 Involve different perspectives in release planning*

RP.P.GP:6 Let stakeholders' importance be a factor during prioritization

When including multiple stakeholders in requirements prioritization, their different interests can affect the outcome (e.g. when different stakeholders assign very different prioritizes to the same requirement). This can have a great impact on the release contents if only a few viewpoints are considered. In order to minimize the risk of improper release contents due to that, your organization can assign weights to stakeholders to define their importance. These are not absolute but rather of

changing nature e.g. dependent on market segments size, or time to market. It is important that the weights reflect what has been defined in the product strategies (regarding e.g. key customers or market segments to address).

The assignment of weights is in this context mainly targeting the internal stakeholders. However it shall not be forgotten that individuals can be considered both as external and internal stakeholders in some situations e.g. individuals from marketing department might be under high pressure by a larger key customer and consequently undermine requirements of the other customers. Therefore the assignment of weights can have a greater impact than intended if the stakeholders are not considering all the factors in the prioritization equation. Additionally by mapping the power of the external stakeholder, several problematic areas such as the above can be minimized.

Recommended pre-requisite practices:

- *OS.M.GP:5 Identify stakeholders and map their influence*
- *RP.P.GP:5 Consider multiple stakeholders during requirements prioritization*

RP.P.GP:7 Re-plan releases upon changes

As a release project advances in its lifecycle, the priority of requirements may change. Not only requirements being implemented in the project may undergo priority changes, but requirements external to the project may also do so. In addition, change requests to existing requirements may also appear.

In this case, re-prioritization of requirements, as well as release re-planning can be performed in order to reflect the new reality. If this is not done, the development organization runs the risk of spending efforts in the development of a product that is no longer aligned with customer or market needs, and projects may be delayed due to ignoring such changes without updating the plan.

Release re-planning may be necessary after re-prioritization of requirements because some requirements may need to be dropped from a release, and others may need to be included. If more than one release is planned at a time, all releases need to be considered, and reallocation of requirements among them shall be done as necessary.

Re-prioritization here can be as simple as exchanging a newly discovered high priority requirement with a low priority one that was allocated to the release project. Or it can involve a more comprehensive effort such as the one that allocated the requirements to the release project.

Example*:*

> *If requirement A is currently planned for release 'n', and its priority increases, it may be allocated to release 'n-1' instead, i.e., the release that is currently undergoing development. As a result, requirement B that was planned for release 'n-1', but that had not yet been implemented, may be dropped from release 'n-1' and reallocated to release 'n'. Alternatively, it may be completely dissociated from any release, and go back to the pool of available requirements.*

Related practices:

- *RP.GP:5 Plan more than one release at a time*
- *RP.P.GP:1 Perform systematic requirements prioritization*

RP.P.GP:8 Make use of must and wish lists

When doing requirements selection for a release, consider allocating them to must and wish lists. This practice intends to assure that requirements that must be implemented take priority over requirements that should also be in the release, but whose implementation may still be postponed.

One way to use must and wish lists is exemplified below, from a market-driven requirements engineering process called REPEAT.

In order to create these lists, requirements should be annotated with priority and effort estimation. A prioritization technique as suggested in related practices could be used to find out the highest priority requirements. You should then allocate them to a must list, in ways that they fill up to 70% of the effort. Requirements that do not make it to the must list according to these criteria should then be put in a wish list. The wish list can take up to 60% of the effort that can be put in the release.

Therefore, must plus wish lists account for 130% of the total effort. This means you have a 30% buffer after you implement the most critical requirements, which can be used to implement up to half of the requirements in the wish list.

This practice allows you to concentrate your efforts in the highest priority requirements, whereas providing a contingency buffer in case a delay happens during their implementation. If no delays happen, you still have 30% of non-used effort that can then be allocated to the implementation of requirements in the wish list.

Related practices:

- *RP.P.GP:1 Perform systematic requirements prioritization*
- *RP.P.GP:3 Prioritize requirements based on cost, value and risk*
- *RP.P.GP:2 Prioritize requirements based on their abstraction levels*

RM Requirements Management

RM.GP:1 Introduce tool support for requirements management

Your requirements process can become better implemented and sustained if you have proper tools to support it. Tool support is important to enable easily issuing requirements, specifying them further with the use of attributes, tracking their statuses, versions, and traceability links.

The widespread document-based approach to requirements specification poses several drawbacks to managing requirements, such as:

- It is hard to store requirements attributes,
- There is no easy way for the many requirements sources to issue requirements (especially when personnel are geographically dispersed),
- Tracking requirements statuses and traceability links between them is cumbersome,
- It is difficult to handle requirements during release planning (e.g. how to assign a requirement to the next release, or how to postpone it to a future release)
- It is not possible to baseline a sub-set of requirements within a document; rather, the whole document needs to be baselined.

Using a database-based approach to requirements specification helps solving the problems above. This approach can be as advanced as introducing a commercial requirements management tool, or as simple as using a spreadsheet.

The former comes along with benefits like versioning of individual requirements, query facilities to search requirements, access control, and change management. The latter approach of using spreadsheets is much simpler and cheaper, yet it can still yield benefits over a document-based approach. For example, it is more practical to specify requirements attributes and statuses, associating requirements to releases, and sorting and filtering requirements.

Regardless of the type of tool you use, bear in mind that a tool does not replace a process. Having a tool that offers many facilities to managing your requirements will be of no use if your organization does not know how to specify them and if your personnel do not exploit all the facilities of the adopted tool. More importantly, tool support will also be of no use if the requirements you elicit are not relevant from a product or business perspective (i.e. a tool won't bring you benefits if the requirements managed through it are wrong).

Therefore, see the adoption of a tool for requirements management as essential to support your requirements process, but do not forget all the other aspects of the process, such as practices on requirements elicitation, analysis, release planning, and the organizational aspects needed to support them.

Recommended pre-requisite practices:

- *OS.GP:3 Assign a person to manage and own the requirements process*

Related practices:

- *OS.GP:4 Train people in requirements engineering*

RM.CM Configuration Management

RM.CM.GP:1 Have a change control process in place

Once you have created a requirements baseline, you should have a change control process in place in order to control changes to them. Controlling changes to requirements is necessary for the following reasons:

- Change requests can occur often and simply accepting all of them may end up ruining initial project plans.
- Not all change requests have the same urgency and importance; therefore you should be able to differentiate critical from less important ones.
- Change requests should be evaluated from a business perspective to assess whether they are relevant from a product, market or strategic perspective. Not doing this may lead to increased development costs without the expected financial return when developers accept change requests that are meaningless from those perspectives.
- Change requests may not be aligned with the release objectives and therefore need to be rejected. On the other hand release objectives may change along the way and thus such change requests may need to be accepted. Therefore, analyzing change requests for their relevance from a product perspective is important before accepting them.
- Accepted change requests must be communicated to all parties involved (e.g. developers, testers, and documentation and marketing personnel)
- A change control process should not only handle changes to existing requirements that are undergoing implementation within a release project. It should also handle, for example, reports of defects that need to be fixed, and suggestions of new requirements.

When defining your own change control process, you should think of aspects as shown in the list below.

- Roles: who are the parties involved in the process? For example, who approves a change request? Who performs impact analysis? Who implements and verifies changes? The process should also define which roles integrate the change control board (entity in charge of approving/rejecting change requests)
- Change request statuses: which statuses does a change request go through? For example, Issued, Approved, Implemented, Verified, and Rejected.
- Analysis procedure: which factors to consider when analyzing a change request? For example, cost, impact on architecture, alignment with business requirements, resources available.
- Verification procedure: which techniques to use, to verify a change request? For example, peer reviews, requirements based testing, and inspections.
- Tool support: which tool to use to track change requests?
- Status reporting: which types of reports shall be generated and when? For example, how many change requests have been accepted, rejected, approved, implemented or verified within a certain period of time?
- Feedback to issuer: issuers of change requests may be revealing important improvements to the product. Therefore your change control process should include channels that enable feedback to issuers on the status of their change requests, so that they will keep on giving you feedback on the product.

When defining your change control process, beware of the level of formality you need for your organization. Making the process too bureaucratic may lead to people taking shortcuts and falling back to an ad-hoc process. In addition, be sure to make it easy to issue change requests (e.g. through an online form available to interested

parties), so that you can funnel changes through one unique channel where they can later be analyzed.

Recommended pre-requisite practices:

- *RM.RS.GP:2 Track status of requirements*
- *RM.CM.GP:4 Make use of a requirements baseline*
- *RM.CM.GP:2 Define a cross-functional change control board*

Related practices:

- *RM.GP:1 Introduce tool support for requirements management*
- *RM.RS.GP:6 Record reason for rejecting requirements*

RM.CM.GP:2 Define a cross-functional change control board

A change control board (CCB) is a group of people defined by your change control process who is in charge of deciding which new features, which changes to existing requirements, or which defect reports should be accepted for further processing or rejected.

It is important that you include different perspectives in the CCB by bringing together roles from different functional areas of your organization. For example, you should consider having representatives from development, marketing, and management. These roles can provide their view on technical feasibility and effort estimates, strategic and market importance of change requests.

Recommended pre-requisite practices:

- *RM.CM.GP:4 Make use of a requirements baseline*
- *RM.CM.GP:1 Have a change control process in place*

RM.CM.GP:3 Control versions of your requirements

You should control versions of your requirements for the following reasons:

- It will prevent that people waste time working on obsolete versions
- It will offer a way to keep a history of changes to requirements along with documented reasons for them, who made them, and when.
- The version control for requirements is at the heart of baselining and change control procedures (see related practices for these)

You can control versions of your requirements in different ways depending on the way you specify them and the tools you have available.

If you use a document-approach to requirements specification, you should keep track of different versions of the documents. This can be done manually or by using a version control system like CVS or Subversion. If you use a more advanced requirements management tool, you may have the advantage of controlling versions of individual requirements as well.

Recommended pre-requisite practices:

- *RM.GP:1 Introduce tool support for requirements management*

Related practices:

- *RM.CM.GP:1 Have a change control process in place*
- *RM.CM.GP:4 Make use of a requirements baseline*

RM.CM.GP:4 Make use of a requirements baseline

Once you have agreed on a requirements set to be implemented (e.g. after release planning), you should baseline those requirements. A requirements baseline is a snapshot of the state of requirements at certain point in time, whose purpose is to provide a stable version of them so that design and development can take place.

The point at which you should define a baseline varies according to the stability of your requirements. For example, during initial stages, when requirements are constantly evolving, a baseline may not be useful. However, once requirements have been reviewed and agreed upon, and are ready to be implemented, they should be baselined in order to serve as the common denominator for different interested parties, such as developers, project managers, and testers.

Recommended pre-requisite practices:

- *RM.CM.GP:3 Control versions of your requirements*

Related practices:

- *RM.GP:1 Introduce tool support for requirements management*

RM.RT Requirements Traceability

A part of requirements management that you should strive to implement regards maintaining relationships among requirements, between requirements and other software artifacts, and between requirements and their sources. This is known as requirements traceability and can bring the following benefits:

- Ability to identify people related to a requirement. E.g. who issued it, who the requirement is for, and who is responsible for ensuring its completion?
- Ability to perform informed impact analysis by knowing, for example, which software modules are affected by a requirement.
- Possibility to know which test cases verify which requirements.
- Ability to track requirements relationships. For example, a product feature can be traced to the functional requirements that provide more detail about it.

The following are practices you can think of introducing in your organization to obtain the benefits above.

RM.RT.GP:1 Trace requirements to other software artifacts

One practice you can consider implementing regarding traceability is to trace requirements to other software artifacts. For example, requirements can be traced to use cases, to software modules or components, to test cases, and to project plans.

Depending on the type of artifact you trace requirements to, you may have different benefits. For example, when a change request is handled, you can identify which requirement is involved and from this requirement you will know which modules are affected. This can help you perform impact analysis. Once changes are performed, you can identify which test cases are affected and that may also need to be changed to verify the requirement.

This practice is expensive to implement in the sense it demands discipline from many different roles involved, like software architects, developers, testers, project managers, and so on, depending on which type of traceability is kept.

However, if your organization is in a business that involves safety or life critical systems, the expense of implementing traceability to other software artifacts is worth the benefits it can bring.

Recommended pre-requisite practices:

- *RM.GP:1 Introduce tool support for requirements management*

Related practices:

- *RM.CM.GP:1 Have a change control process in place*

RM.RT.GP:2 Trace relationships between requirements

When specifying requirements, you will probably come across with relationships between them. These relationships are seen here as those between requirements that are interconnected by the different level of abstraction in which they are specified.

For example, a product feature has many associated functional requirements that provide more details on how it should be implemented. These functional requirements are then related to the feature, and vice-versa. You may also relate features with product level requirements (which are in a higher level, and are more like business requirements for the product).

If you specify requirements in multiple abstraction levels, you should implement this practice in order to keep track of the relationships between them.

In the MDREPM, requirements dependencies are also seen as relationships between requirements, but are handled in another practice. See related practices for details.

Recommended pre-requisite practices:

- *RM.RS.GP:1 Specify requirements attributes*
- *RM.RS.GP:4 Specify requirements in multiple abstraction levels*

Related practices:

- *RM.GP:1 Introduce tool support for requirements management*
- *RA.GP:7 Track requirements dependencies*

RM.RT.GP:3 Trace requirements to their sources

One simple practice regarding traceability is to trace requirements to their sources. This can be done by simply including a requirement attribute that identifies where the requirement originated from. Possible values for this attribute are persons with a stake on your requirements process, such as internal marketing personnel, a key customer, or even other sources such as a standard or regulation your product must adhere to.

This practice can be helpful in the sense of, for example, keeping track of the original author of the requirement. The author can then be consulted for clarification purposes. In addition, accountability is introduced, as the person who recorded the requirement can be easily identified.

Recommended pre-requisite practices:

- *RM.RS.GP:1 Specify requirements attributes*

Related practices:

- *RM.GP:1 Introduce tool support for requirements management*

RM.RS Requirements Specification

RM.RS.GP:1 Specify requirements attributes

A requirement usually is specified with a title and a description. However, those are only two out of other attributes that you may consider specifying for your requirements. The following list provides some examples of other attributes.

- Identifier: a unique id for the requirement
- Rationale: a reason why the requirement exists
- Release: the number of the release to which the requirement is allocated
- Status: the status of the requirement (see practice *RM.RS.GP:2 Track status of requirements*)
- Originator/Source: traces the source of the requirement (e.g. market research, key customer)
- Implementation priority: a number indicating how soon the requirement should be implemented
- Effort estimate: an estimation of the implementation effort (e.g. in man-hours)
- Impact: a quantitative measure of the impact of the requirement on the existing system. E.g. considering how many components it will affect.
- Dependencies: states other requirements which depend on a specific one.
- Responsible: defines the person in charge of implementing, or making sure the requirement is satisfied.

Although many of the attributes above may seem useful, be careful to select only the ones that really applicable to your case. In addition, make sure to define responsibilities to fill out values for the attributes. For example, a developer may assign values for the Impact and Effort attribute after performing impact analysis and effort estimate. Assigning such responsibilities ensures the attributes will be used.

Attributes give you the benefit of separating important pieces of information about a requirement from its description. Depending on the tool you use for requirements specification, you could be able to filter requirements by certain attributes. For example, you could query for all requirements associated with release 'X' that have status 'Y' and that are of priority higher than 'Z'.

Recommended pre-requisite practices:

- *RM.GP:1 Introduce tool support for requirements management*

RM.RS.GP:2 Track status of requirements

Requirements should have a well defined lifecycle, with statuses that allow for keeping track of which point in the lifecycle a requirement currently is. The following statuses are an example of what you may consider for your organization:

- Candidate – this status indicates a requirement has been proposed, but not yet gone through further analysis.
- Approved – requirements that are accepted for further analysis after requirements triage reach this status, and become eligible for inclusion in a future release.
- Specified – this status represents requirements that have been further analyzed and specified in more detail (with e.g. effort estimates, impact analysis)
- Planned – a planned requirement has been allocated to a release.
- Developed – status that represents requirements that have been implemented
- Verified – verified requirements have been tested and proven to be fully functional

- Released – a final status to represent requirements that have been released
- Rejected – in case a requirement has been rejected, it reaches this state. A documentation of the rejection reason should also be provided in this case.

The benefit of having a well defined requirements lifecycle is providing visibility of the state of the requirements database. For example, it is possible to see which candidate requirements there are, which have been accepted for further analysis, which have been rejected (and on what grounds), which are assigned to a certain release project, what state within the release project is the requirement in (e.g. implemented, verified, delivered), and so on.

Recommended pre-requisite practices:

- *RM.GP:1 Introduce tool support for requirements management*
- *RM.RS.GP:1 Specify requirements attributes*

Related practices:

- *RM.CM.GP:3 Control versions of your requirements*

RM.RS.GP:3 Provide an initial priority and effort estimate to newly specified requirements

Newly specified requirements shall be given an initial priority and effort estimate. Priority here is seen as the urgency of implementing the requirement, and gives an idea of how soon the requirement should undergo further analysis. The initial effort estimate is also important as it gives a rough indication of the cost of the requirement and the resource implications it will cause.

These two attributes shall be provided as soon as the requirement is specified because they will facilitate *requirements triage*, the activity of screening the requirements database in order to select requirements for further analysis, or simply reject non-pertinent ones.

Recommended pre-requisite practices:

- *RM.RS.GP:1 Specify requirements attributes*

Related practices:

- *RA.GP:2 Perform requirements triage*

RM.RS.GP:4 Specify requirements in multiple abstraction levels

A common fact that occurs when requirements are issued is that they can be in different levels of abstraction.

On the highest level, a requirement can be as abstract as a *product goal* (e.g. “The product shall enter the Scandinavian market”). In a slightly more refined abstraction level, product features represent ways to achieve product goals (e.g. “The user interface shall support all Scandinavian languages”). Continuing the refinement of requirements, product features can be broken down into functional requirements, which describe an essential part of some functionality needed to implement a feature (e.g. “The user shall be able to choose the language at log in time”.

Different levels of abstraction will be useful for different audiences. For example, developers may find product and feature level requirements too abstract to start design or implementation. However, those levels can be appropriate to marketing, sales and product management when deciding, e.g. whether the requirement should be implemented in a release, or how to plan a market campaign to advertise new product features.

Accepting the fact that no one abstraction level fits all audiences, you should then strive to specify requirements in multiple levels of abstraction, like in the above example. By doing this, the following benefits can be achieved.

- The multiple abstraction levels help bring marketing, development, sales, and product management together by conveying a general understanding of what requirements mean and imply to different stakeholders.
- Product and feature level requirements are helpful in requirements triage. Such requirements can be compared against product strategies to see whether they should be accepted for further analysis or simply be rejected.
- Feature level requirements facilitate prioritization, and ultimately release planning. Prioritization of features rather than functional requirements becomes more efficient due to their lower number (since a feature may be composed of several functional requirements). In addition, features represent a more holistic view than a specific functionality, which makes prioritization decisions more understandable and intuitive.

The idea of using multiple abstraction levels for specifying requirements is systematically covered in the Requirements Abstraction Model (RAM).

Recommended pre-requisite practices:

- *RM.GP:1 Introduce tool support for requirements management*
- *RM.RT.GP:2 Trace relationships between requirements*

Related practices:

- *RA.GP:2 Perform requirements triage*
- *RP.P.GP:2 Prioritize requirements based on their abstraction levels*

RM.RS.GP:5 Workup requirements

When a requirement is issued, it can come in different levels of detail. The requirement may be a rough idea of a new feature got through a phone call with a key customer, or it may be a low level, detail-rich description of some functionality.

In the latter case, it may not be always clear whether the functionality is part of a feature that is aligned with product strategies. Therefore, new, higher level requirements may need to be derived from the low level one, in order to abstract up the functionality it represents to a feature or product level. This "work up" process allows identifying whether a newly issued requirement is pertinent to the product or not by comparing it against product strategies, thus allowing for *requirements triage*.

On the other hand, if the issued requirement is on a product level – a high level description of a feature the product shall contain – it needs to be broken down in lower level, functional requirements. This needs to be done in order to make the requirement useful for development purposes. A product or feature level requirement may hold significant functionality behind it, which shall be specified in the form of functional requirements to ensure that development will implement features as they should, rather than based on developer's gut feeling and assumptions.

Recommended pre-requisite practices:

- *RM.RS.GP:4 Specify requirements in multiple abstraction levels*

Related practices:

- *RA.GP:2 Perform requirements triage*

RM.RS.GP:6 Record reason for rejecting requirements

When analyzing requirements, you will often come across with some you may want to reject. There may be several reasons why to take such a decision, such as:

- The requirement is out of scope (e.g. does not belong in the product)
- The requirement is within scope, but it is too expensive to implement
- The requirement does not bring significant value to the product
- The requirement has not been specified properly (e.g. missing information)

A requirement that is rejected today is probably done so after some time is spent on analyzing it, which can involve the participation of expensive resources such as a product or project manager. If this requirement or a similar one reappears in the future (which is not uncommon) you can search your requirements database for it. Having a documented rejection reason will save you analysis time to judge whether to reject the reappeared requirement or not. In addition, you will also be able to compare the current situation against the informed reason and decide whether the reason is still applicable or not and maybe "resurrect" the requirement.

Recommended pre-requisite practices:

- *RM.RS.GP:2 Track status of requirements*

Related practices:

- *RA.GP:2 Perform requirements triage*
- *RM.RS.GP:1 Specify requirements attributes*

RE Requirements Elicitation

RE.GP:1 Distinguish between end-user and customer

The customers of a specific product might not be the same as the end users e.g. a larger customer that invest in word processing software which then several hundred employees use, but not the customer himself. By separating the customer from the end users the elicitation techniques become more accurate. In addition, this information can be used as input when deciding how the product will be marketed and what requirements that are elicited for that purpose only.

If the suppliers of the word processing programs, mentioned above, do not distinguish between the user and the customer there is an apparent risk that several requirements are not elicited. For example, only focusing on the customer will lead to that requirements giving user-friendliness, functionality, spellchecker will be ignored. On the other hand solely focusing on the end-user might remove the option to share documents with others not using exactly the same version of the program or other security issues, that e.g. are ignored by open source versions of word processing programs.

RE.GP:2 Identify requirements sources

The goals with the elicited requirements are to, through analysis and release planning activities, generate the maximum perceived value for the market(s) they are aimed for. The markets might differ in what is considered value and therefore it is better to divide the market into segments. Each of the segments should then be screened for possible sources of requirements, in addition to the sources internal to your organization. Knowing where to focus your elicitation efforts will save time and resources since you will concentrate on the most relevant sources of requirements for your products.

The following list shows some examples of requirements sources you may want to consider when eliciting requirements.

- Influences: standards, trends, and regulations.
- External stakeholders: partners, distributors, competitors, subcontractors, and contractual commitments with customers.
- Internal stakeholders: owners of the organization, marketing personnel, product manager, project manager, software quality manager, and technical support.
- Internal sources: bug reports, technology opportunities – innovation, existing requirements (requirements reuse), existing functionality enhancements, history (previous releases), similar projects/products, product strategies, and product roadmaps.
- Techniques that aid elicitation: scenarios focus groups, market analysis, surveys, competitor analysis, and segment analysis.

Recommended pre-requisite practices:

- *OS.M.GP:1 Establish a marketing organization*
- *OS.M.GP:5 Identify stakeholders and map their influence*

RE.GP:3 Train personnel in eliciting requirements

Technical support in most organizations has frequent contact with users and customers that are in need of help with your products e.g. through regular phone support or when installing the products at the location of your customers. When giving support they hear what areas user have the most problems with, what is not appreciated, and most importantly what they would like to see in future releases.

Technical support staff can therefore be trained to identify what new requirements are and what are not. However, strong motivation of the support personnel is needed, so that they do not let these moments pass. When support personnel elicit requirements they can preferably quickly document them in an internal system. The draft versions of the requirements can then be refined during requirements analysis or requirements management activities.

The same approach can be used when performing certain test activities for a release since the test personnel can be seen as the "initial customers" e.g. when performing acceptance tests and functional testing. Improvement ideas that testers have can turn out to be relevant requirements. Therefore, testers can also be seen as potential requirements sources as they verify already implemented requirements during their activities.

Related practices:

- *RM.GP:1 Introduce tool support for requirements management*
- *RE.GP:5 Create elicitation channels for requirements sources*

RE.GP:4 Use up-to-date market analysis information for elicitation

The reasons behind market analysis are to find out if the new product's functions, prices, and appearances will be appreciated by the market segments before they are released. This does not only give your organization the option to postpone or cancel the product before the launch (e.g. the market segment have shifted direction, or already been filled by the competitors), but the information gathered during the market analysis can also be used as an elicitation source for future releases.

The current market segment demands have to be fitted with a product as well as a marketing strategy for it. However the market segment may shift direction rapidly and the information gathered therefore have an "expiration date" which should be reflected in the elicited requirements. For example requirement A is only of interest if put in release 'X' while requirements B, C, D can be released later due to low demand at the moment.

One way for a market driven organization to stay ahead of its competitors can be to take as a practice to periodically perform market analysis so that the requirements elicited are not outdated. By doing so your organization also identifies upcoming trends, e.g. user interface, new standards, or other opportunities through technological combinations such as mobile phones that support VOIP-services and payment services as if the phone was a bank card.

Recommended pre-requisite practices:

- *OS.M.GP:1 Establish a marketing organization*

Related practices:

- *RE.GP:8 Let the importance of market segments guide elicitation efforts*

RE.GP:5 Create elicitation channels for requirements sources

Enabling different channels for customers to put forward requests and feedback do not only give the customers a feeling that their product supplier listens to them, but is also a good source to use for requirements elicitation. The requirements can be elicited through different channels such as beta releases, FAQ, customer complaints, or simply by adding a suggestion page on your organization homepage.

If customers suggest a function that is later considered for a product release, it is recommended to follow up their request by thanking them for input on top of the decision whether to discard or when to add the suggested functionality to the product. This gives customers a feeling that the organization cares about them which increases the chances that they will continue to issue requirements or purchase future products from that organization.

The benefit of this practice, which is making it easy to issue requirements, can also be its drawback, which is requirements overload. It is therefore recommended to fulfill other practices prior to this one, such as *RA.GP:2 Perform requirements triage*, which can help handling the high requirements influx that may take place once you have created elicitation channels.

Recommended pre-requisite practices:

- *RM.GP:1 Introduce tool support for requirements management*
- *RM.RS.GP:4 Specify requirements in multiple abstraction levels*
- *RM.RS.GP:5 Workup requirements*
- *RA.GP:2 Perform requirements triage*

RE.GP:6 Consider strengths and weaknesses of competitor's products

One of the main sources of requirements for a market driven organization is its competitive rivals. In order to be competitive, your products should not only contain the appealing features your competitors' products have, but also fill gaps left by them. Therefore it is important that you conduct an analysis of the main strengths and weaknesses of competitor's products, and focus your efforts in finding requirements that can make your product superior to theirs. The information gathered during the analysis can be complemented through other channels as well e.g. analysis of competitor product manuals usually available on their homepages or available product white papers.

It is important to remember that the strengths and weaknesses of competitor's products should be defined from the customer's perspective. This can for example be done through market analysis, by including questions about what the intended market segment appreciates and not, in the products supplied to them by your competitors.

This practice can be seen as a complement to *OS.M.GP:4 Identify and analyze competitors*, from the process area organizational support in the MDREPM. That practice focuses on providing a view of competitors as an organization e.g. resources, technical expertise, marketing strategies. By implementing this practice here you can then gain a broader view of your competitors by knowing strong and weak points of their products, and therefore taking advantage of that on your behalf.

Recommended pre-requisite practices:

- *OS.M.GP:3 Know your own strengths and weaknesses*

- *OS.M.GP:4 Identify and analyze competitors*

Related practices:

- *OS.M.GP:6 Identify critical success factors for specific markets and/or key customers*
- *OS.M.GP:7 Identify the basis of the competitive advantage*

RE.GP:7 Align elicitation activities with product strategies

Elicitation activities should be guided by what is stated in the product strategies. For example if the focus of the product strategies is to introduce a new product in a certain market, efforts should be put on finding out what are the important features required by that market segment, or which can create a demand for the product in that market. That is, elicitation activities should focus on that market.

The elicitation activities can also be guided by the product roadmaps, which reflect on several releases. However, in some cases new requirements might be elicited and considered, instead of the ones specified in the roadmap. But by using the roadmap and the product strategies as a means to align the elicitation resources with, the time and money you spend on elicitation is assured to be aligned with overall goals for the product.

Recommended pre-requisite practices:

- *OS.S.GP:1 Define organizational strategies and introduce a strategic mindset*
- *OS.S.GP:2 Define product strategies*
- *OS.S.GP:5 Define a roadmapping process*

RE.GP:8 Let the importance of market segments guide elicitation efforts

This practice aims to aid market-driven organizations which have one or a few products that target a specific market, for which they need to elicit requirements from. The below suggested elicitation techniques for specific market segments can still be considered by all organizations.

Depending on the targeted market segment(s) the elicited requirements might differ. If the targeted market segment(s) demand products that have high quality and accept a higher price, then the resources spent on elicitation should reflect this as well.

For example:

Porsche's targeted customers are looking for features that a sports car provide e.g. big engine, leather seats, nice interior, design. On the other hand a very little segment of Porsche's customers would prefer to have a towing hook on their Porsche. Requirements from these different segments are elicited and documented. When analyzing the requirement it is concluded that the towing hook would agitate the main customer group and is therefore discarded. This way, time and resources were spent on something that might be considered obvious to exclude from the beginning.

This example might be a little bit out of scope of the software industry but it can be considered when eliciting software requirements as well. In other words the importance of the market segment and its stereotypic view should set restrictions on resources and time spent on eliciting requirements from that market. This will help handling a high requirement influx and thereby minimize the time spent on analyzing the elicited requirements.

When deciding which segments of market that guide the elicitation process in your organization the following techniques can help:

- Key customers: Which are the key customers? What made them choose you? What are they expecting from the next release? What do they not prefer in the long-term?
- Historical data: Do the targeted market segment tend to change their mind frequently regarding features of the product, e.g. mobile phones? Are the customers loyal to your brand?
- Sales department: In addition to the feeling of personnel from the sales department; what requirements contributed to the easiness to sell the last product? If it was hard to sell, what requirements do the sales personnel felt lacking or were not appreciated by the customers? What requirements do they feel that the customers would appreciate better? Information perceived during discussions in sales meetings, informal discussions with customers.
- Market analysis: Customers questionnaires.

Recommended pre-requisite practices:

- *OS.M.GP:1 Establish a marketing organization*

RE.T Techniques practices

RE.T.GP:1 Elicit requirements through use cases

One good way to find out about the requirements your product needs to fulfill is to think of the usage scenarios it will be subjected to. Use case is a technique that can help with that. It consists of identifying actors that interact with the system and the steps involved in this interaction (actors can be complemented with personas for deeper understanding, see *RE.T.GP:4 Elicit requirements through personas*).

As an example, consider a charging system for a mobile telecom operator. One use case could be the following. A user that belongs to a different telecom operator enters the current operator network, and makes a phone call. Analyzing this use case will reveal requirements related to roaming agreements, fees, charging scheme (e.g. per minute, flat rate).

A use case does not need to be created with the presence with the actual user of the system (use case actor). It could rather involve experts within your organization that know typical usage scenarios your product will and can be exposed to.

Beware, though, that use cases do not replace requirements. Rather, they are a means through which you can identify requirements your product needs to fulfill.

Recommended pre-requisite practices:

- *RE.GP:1 Distinguish between end-user and customer*

Related practices:

- *RE.T.GP:4 Elicit requirements through personas*

RE.T.GP:2 Elicit requirements through user groups

If representatives can be identified from different user groups, a technique you can use to elicit requirements is to gather them in meetings to discuss their wishes in relation to your products. User group meetings complement other elicitation techniques such as market analysis realized through surveys by offering a more personal and natural interaction with actual user group representatives.

These meetings have the advantage of bringing together people with different opinions about your products and hearing from them what they are happy with, what they want to see improved what new features they want to see implemented. The selection of which user representatives to invite can be done according to market segments to get more varied opinions since interests could be different.

These meetings can be realized with the participation of a mixture of technical and business personnel from the side of your organization. This provides different perspectives on suggestions gathered from the group. For example, business benefits and technical feasibility of suggestions can be evaluated together.

In order to guide discussions, the presence of your product (for demonstration purposes) can be helpful to foster participation of the meeting participants. You may also consider the use competitor's products for the purpose of discovering what type of features they have that appeal to the users, and that therefore you should focus on as a way to overcome your competitors.

In order to get to a consensus about the main wishes identified, you may employ a prioritization technique, such as Top Ten requirements, or Cumulative Voting in order to rank the requirements you found out according to the participants' preferences.

Recommended pre-requisite practices:

- *RE.GP:2 Identify requirements sources*
- *RE.GP:1 Distinguish between end-user and customer*

RE.T.GP:3 Elicit requirements from product reviews

Customers considering purchasing a product, either online or from a regular store, usually search for information about the product on internet first. This search often targets web pages containing customer reviews about the product (e.g. auction sites, price comparison sites). In these pages, potential customers can find what others liked and disliked in the products. Furthermore, magazine reviews, forums, and blogs may also be used as source of feedback on products, either for your own products or your competitor's products.

This information is for free and it is not uncommon that hundreds of comments may have been provided for a single popular or unpopular product. It is therefore a valuable source of potential requirements your organization can explore in addition to the channels discussed in practice *RE.GP:5 Create elicitation channels for requirements sources.*

Related practices:

- *RE.GP:5 Create elicitation channels for requirements sources*

RE.T.GP:4 Elicit requirements through personas

When you are faced with a mass market for your products, you may find it difficult to identify a typical user profile to target. This is where the use of personas for eliciting requirements can come to help.

A persona is a description of a typical user of your product. This description is very detailed, and can include social activities with which the person is involved, his/her personal likes and dislikes, daily routine, hobbies, interests, social connections, technology literacy, technical frustrations, age, gender, life style. To make it more individual, a persona is usually given a name and a photograph, which help in memorizing them.

The information you will want to include in a typical persona can be derived from the analysis of a market segment. The persona is then created as a way to communicate the characteristics of the typical user profile throughout your organization. However, the correctness of the information behind a persona is important. If it does not reflect reality, the persona can create the illusion of a user that does not exist. This illusion can cascade through several requirements engineering activities, with the possibility to cause damages. For example, the identification of requirements that are later worked on and that are not really what the target user profile wanted.

Still, the advantages this technique brings are several:

- It fosters communication. Referring to "Daniel as a 16 year old Swedish boy who likes listening to music and going out with friends" rather than the "market segment of teenagers in Sweden" gives a much clearer picture of who the target user profile is. Within your organization, the design of, for example, a game, a mobile phone, or an MP3 player can then be done considering Daniel's preferences (which are more concrete) rather than the market segment preferences (which are more abstract).
- A persona, or a small set of them, can be a powerful tool as they are a more concrete representation of market segments which can be referred to when finding out which requirements the product should have, how its user interface should be designed, and so on.
- Personas save you the time of going out to the market and try to elicit requirements from many hundreds of users.
- They are relatively simple to develop, and can bring you benefits during many phases of your development process, from elicitation of requirements, through release planning, design, implementation, through to testing.

Recommended pre-requisite practices:

- *OS.M.GP:1 Establish a marketing organization*
- *RE.GP:1 Distinguish between end-user and customer*

Related practices:

- *RE.GP:4 Use up-to-date market analysis information for elicitation*
- *RA.GP:10 Make use of personas for analysis*

RE.T.GP:5 Elicit requirements through gap analysis

Gap analysis is done in order to verify if a gap exists between what a product delivers and what gives satisfaction to the customers. The level of customer satisfaction is compared with the features and characteristics that the product offers to differentiate the two types of gaps that can exist. If the product lacks features or characteristics, from the customer perspective, then the gap is said to be negative but if the products contains features and characteristics that is more than what the customer whished for then it is said to be a positive gap.

This analysis will not only provide a fairly accurate measurement of how well release planning activities have been performed but the negative identified gap is also an excellent source to elicit requirements from. On the other hand the positive gap can be used during requirements analysis as an indicator to which requirements that should not be included in future releases of the product. Moreover, Gap analysis can also be used during requirements prioritization. For example, if performance issues were identified in the gap analysis then requirements can be prioritized after which ones that improve performance the most.

This shows that by performing Gap analysis several requirements can be elicited in addition to the improvements areas that are identified.

Recommended pre-requisite practices:

- *OS.M.GP:1 Establish a marketing organization*
- *RE.GP:1 Distinguish between end-user and customer*

Related practices:

- *RE.GP:4 Use up-to-date market analysis information for elicitation*

RE.T.GP:6 Elicit requirements through customer value analysis

This practice has similar characteristics with *RE.T.GP:5 Elicit requirements through gap analysis*. However, while that practice analyzes the gap between the product and the customer satisfaction, the customer value analysis (CVA) can be used to verify the perceived customer value. This is done through comparison with the perceived value the competitor's products provide. The information can be gathered through surveys answered by your customers and your competitor's customers. The perceived value of several attributes can be compared e.g. technology, pricing, sales, support, features.

When performing CVA the conjoint analysis tool can be used. This tool compares features or attributes of a product that can have two or more levels. The customers are asked to provide the level of importance for the attributes from their perspective and these levels are then compared against one another e.g. what are preferred weight, price, and battery time for a mobile phone. For example, most people prefer 80g weight of a mobile phone while a few prefer 60g, and 100g. These different attributes are then compared with other attributes in order to conclude the optimum values e.g. weight of a mobile phone can be 80g, 120g, or 160g which is put in relation to its battery time 200h, 300h, or 400h as well as the price 200€, 300€ or 400€.

Not only will your organization be rated against your competitors (which are useful measurements for several practices in Organizational Support), but the information gathered in the survey can also be used to elicit new requirements or improve already elicited ones (both functional and non-functional). Moreover, by performing this analysis the disconnections between what the customer values and what the organization think the customer values are limited. This will in the long run through improved elicitation efforts increase the perceived quality and value for future releases, compared to what your competitors provide.

Recommended pre-requisite practices:

- *OS.M.GP:1 Establish a marketing organization*
- *RE.GP:1 Distinguish between end-user and customer*

Related practices:

- *RE.GP:6Consider strengths and weaknesses of competitor's products*
- *RE.GP:4 Use up-to-date market analysis information for elicitation*
- *OS.M.GP:4 Identify and analyze competitors*

RA Requirements Analysis

RA.GP:1 Provide well informed effort estimates

Providing accurate effort estimates is a practice your organization should strive to master. The reason is that it is at the core of project management and release planning. It will help you minimize disruptions in project plans and deliver the expected value for your product releases.

Estimations can be done using different approaches. In a bottom-up manner, a project is broken down in individual tasks and requirements, and each is estimated individually. In a top-down manner, expert opinion on high level description of features is used to derive the estimates. A variation of this approach is to include several experts and refine their estimates in several rounds, to improve their accuracy. In addition, comparisons to previous, similar projects can also help, in what is known as estimating by analogy.

Estimating effort can rely on historical data to strengthen their accuracy. For example, graphs comparing probability percentage of success against a timeline (such as months of implementation) can guide you to decide whether an estimation of, say, 6 months for a release project has good chances to succeed as compared to previous and similar attempts. This, along with documented team productivity and software size measures of previous projects, can help you to consciously estimate effort for a new release project.

As estimates have a degree of uncertainty when they are made, they should be updated as more is known about the work at hand. Therefore, you can start by providing a rough effort estimate at the time you specify a requirement, and update it as you proceed with the requirement analysis. This can be done and will be useful in even pre-project activities, such as requirements selection for implementation during release planning. You can then use the estimates for guiding your release project plan creation. As you proceed with implementation of estimated requirements, you should then revise your estimates as you know more about the effort to be expended in their implementation.

At the end of a project, you can then archive your project plan with provided estimates for use as historical data by future projects.

RA.GP:2 Perform requirements triage

Release planning is one of the cornerstones of market driven requirements engineering and a too high pressure on it can limit its benefits. However by performing requirements triage before release planning the pressure is reduced which limits the workload on the product managers in your organization as well.

Requirements triage is the activity of going through candidate requirements in order to choose which will be rejected and which will be allowed to go through further analysis, and eventually be selected for implementation. It allows for fast acceptance or dismissal of requirements to avoid overload, and to avoid putting effort on requirements that are irrelevant.

This is done by comparing the requirements with product strategies which are derived from organizational strategies. If the strategies are not defined properly they can become a bottleneck for requirements triage. They should therefore be defined in a detailed enough level so that requirements can be compared to them. The

requirements need to be worked up as well before used in triage activities, e.g. they can not be flattened to one level but should rather be defined in several abstraction levels (like suggested in *RM.RS.GP:5 Workup requirements* in requirement management).

In other words the triage activities are an essential part of a market-driven requirements engineering process in that it fights requirements overload – a problem that can arise when more requirements are issued than can be analyzed. The direct outcome of this initial screening is that effort and costs are reduced later on in the process. Therefore, requirements triage shall be carried out periodically for screening the requirements database for newly issued requirements and to judge whether they can go through further analysis or should be rejected.

Recommended pre-requisite practices:

- *RM.RS.GP:4 Specify requirements in multiple abstraction levels*
- *RM.RS.GP:5 Workup requirements*
- *RA.GP:1 Provide well informed effort estimates*

RA.GP:3 Analyze requirements using multiple views

The most typical way to specify requirements is using natural language. However, that poses limitations and risks. Limitations in the sense of making it difficult to express processes and interactions. And risks in the sense of introducing ambiguity, which in many cases are only perceived when a requirement has already been implemented.

Therefore, a good practice is to use multiple views to analyze requirements. The following list outlines some of the artifacts you can use for that purpose.

- UML models: use cases, activity, sequence, and class diagrams offer a good way to model interactions between users and the system, different parts of the system itself and with its exterior, as well as entities that are part of your system.
- Test cases: if a requirement provides a description of what the product should do, a test case for that requirement tries to prove the product won't be able to do what the requirement states. When you create a test case for a requirement, you are using an acceptance view that tries to verify whether the requirement has been implemented properly and that the product does what it is expected to do.
- Decision trees: in complicated processes where many decisions and several possibilities exist, natural language becomes cumbersome to provide clear descriptions. In that case, you may consider the use of decision trees. They are a graphical representation where nodes represent a question and arches between nodes are the decision taken, which leads to the next step in the process.

The main benefit of using multiple views for requirements analysis is that you may catch inconsistencies with the different representations early, which may save time and money by preventing them to be propagated to later stages of development. In addition, graphical representations are good in providing an overview which is not always easy to provide with natural language.

Related practices:

- *RE.T.GP:1 Elicit requirements through use cases*
- *RA.GP:5 Create requirements-based test cases during analysis*

RA.GP:4 Increase the quality of the requirements through inspections and reviews

Software requirements inspection can increase the quality of your requirements. When executing all activities included in an inspection on all requirements, it can be a time and resource consuming task. Inspections activities can therefore initially target just those requirements that are identified as risky (e.g. through risk analysis), or the ones assigned for implementation in a release project. This way the personnel participating will grow an understanding and respect for the outcome of inspections when seeing the increase of requirements quality.

The quality of the requirements can also be increased by performing structured reviews early in the requirements lifecycle e.g. through perspective based reading of the requirements. Except from instructing the reviewers it is also important to tell them of how important their contribution is increasing the quality of the requirements otherwise you might run the risk of getting poor results.

By using developers, testers, and key customers as reviewers time spent in other activities can be shortened down e.g. developers already discussed requirements before implementation starts which can enable better understanding of the requirement or test personnel can start creating acceptance tests as part of the review process.

One of the main benefits of using inspections and reviews is that you can catch errors and misunderstandings early in the development process. At this point, it is much cheaper to fix them than when the product is undergoing acceptance testing, for example.

Related practices:

- *RA.GP:6 Perform requirements risk analysis*
- *RA.GP:5 Create requirements-based test cases during analysis*

RA.GP:5 Create requirements-based test cases during analysis

Requirements-based test case creation can be helpful during analysis of requirements, whereas it also provides a way to perform acceptance tests.

The benefits of creating test cases based on requirements are the following:

- Involves testers in the requirements process.
- Test cases provide a different view of requirements. They allow for revealing inconsistencies between textual description of requirements and other analysis models related to them, such as sequence and activity diagrams, dialog maps.
- If you try to derive test cases from a requirement, and you can't come up with a tangible way to evaluate whether the requirement will be fulfilled, you just came across with a requirement that may be ambiguous, or not clear enough. This in turn prompts you to fix such a requirement before it is given away for implementation, thus saving you time and money you would spend in fixing errors after development.

In many organizations, testers work within projects, and it may not be always easy to allocate them to create requirements-based test cases which are created in pre-project activities. However, the benefit of creating such test cases still exists, and it is recommended that this practice be taken into consideration as part of testing activities. In this way, it would be easier to have access to testing resources to perform the practice.

Related practices:

- *RA.GP:3 Analyze requirements using multiple views*

RA.GP:6 Perform requirements risk analysis

Assessing risks and preparing action plans for handling their possible materialization are considered a good practice of project management. In a market-driven situation, where projects are initiated to implement requirements that are selected for a product release, risk management can be done based on requirements, since requirements are at the core of all development activities.

When analyzing requirements, consider evaluating how risky they are. Documenting requirements risks will benefit you in activities such as release planning. You may consider risk as one of the factors when prioritizing requirements. This will help you select a sub-set of them whose combined risk is acceptable given the circumstances for the release plan in question.

Risk factors you may consider can be many, as exemplified in the list below:

- Changing requirements: pay attention to requirements that are highly volatile and that can change often. These may be rated with relatively higher risk.
- Requirements that demands new and unproven technologies. This can lead to large development costs which mean high risk.
- Requirements that are not aligned with product strategies: if you accept requirements for implementation without concerns to whether they belong in the product or not, you risk wasting development effort in the implementation of features that may not add any value to the product.

Related practices:

- *RP.P.GP:3 Prioritize requirements based on cost, value and risk*

RA.GP:7 Track requirements dependencies

A common fact when analyzing requirements is that you will notice dependency relations between them. These may be of several forms, as detailed below:

- Require dependency: a requirement R1 may depend on R2, which means R2 needs to be implemented before R1. For example, implementing a video player will require the implementation of video codecs before.
- Cost dependency: implementing a requirement R1 may lead to an increase of cost in implementing another requirement R2. For example, a requirement that demands very short response times may increase the complexity of the system and thus the cost of implementing other requirements.
- Value dependency: it may also be the case that implementing a requirement R1 will increase (or decrease) the value of a requirement R2. As an example, consider an Mp3 player. Having an advanced graphical user interface for the player, which allows for creating playlists, sorting songs by name, artist, and genre, would be of little value if the player does not have high memory capacity to store several songs. Therefore, high memory capacity increases the value of the advanced graphical user interface for the player.

Tracking requirements dependencies as you analyze requirements can help you in later phases of your requirements process. For example, knowing dependencies is useful when doing release planning – requirements that depend on one another may be scheduled for implementation in the same release (or in a specific sequence of releases). Dependencies are also useful when prioritizing requirements. For example, a requirement that is rated as high priority initially may have its priority lowered after it is known that it depends on too many requirements (thus becoming too

expensive to implement). In addition, knowing dependencies can also help during impact analysis.

Although useful, keeping track of requirements dependencies can be labor-intensive job. In order to bring the benefits it can deliver, the tracking of dependencies should be done constantly to keep them up to date.

You should decide which types of dependencies are of interest for your organization, and to what extent you would like to keep track of them. A way to minimize dependency tracking is to use them in feature level requirements only, which are less numerous then function level ones. See practice RM.RS.GP:4 for details on that.

Related practices:

- *RP.P.GP:4 Consider requirements dependencies during prioritization*
- *RM.RS.GP:4 Specify requirements in multiple abstraction levels*

RA.GP:8 Perform impact analysis for requirements

Before performing an impact analysis it is recommended that the requirements have gone through triage activities to verify the rationale behind them and alignment with product strategies.

When analyzing requirements it is important to consider potential consequences to other requirements, as well as to the existing product implementation. For that, being aware of requirements dependencies can be very helpful (see related practice) There may be side effects that need to be considered, and if you decide to include a requirement, a rough estimation of the extra costs and time the impact will bring can be done and documented.

Underestimating the impact can have severe consequences to the rest of the development process, such as schedule delays and over pressured staff. Therefore, by consulting experts, e.g. developers, marketing, and testers, instead of only consulting documentation (e.g. requirements specification and design documentation) a better knowledge base is created not just for the person analyzing the requirements but for the experts as well (making them aware of the potential changes). In other words the requirements are analyzed by a cross functional team suggested in the process area Organizational Support in MDREPM.

Recommended pre-requisite practices:

- *RA.GP:7 Track requirements dependencies*
- *OS.GP:5 Create a cross functional team to analyze and specify requirements*
- *RM.RS.GP:4 Specify requirements in multiple abstraction levels*

RA.GP:9 Consider non-functional requirements during analysis

Functional and non-functional requirements should be separately specified after being elicited and then considered again when analyzing the requirements.

The non-functional requirements, also known as quality attributes, cannot usually be traced directly into ready code, design objects, or individual test cases while functional requirements usually can. This leads to that non-functional requirements can be neglected since they are not directly reflected in “visible” functionality.

It can therefore be taken as practice to analyze the results from e.g. market surveys, focus groups, key customer requests, in order to create a list of what the intended market segment seek in their products, e.g. portability, security, performance. Functional requirements can then be traced to related non-functional requirements

and be analyzed in order to understand how they contribute to parent non-functional requirement. The outcome of this analysis also enables traceability the other way as well, i.e. from non-functional requirements to implemented code through functional requirements.

This information can in addition to enabling a more fully understanding of the system be used for setting priorities for the functional requirements, by first prioritizing all of the parental nonfunctional requirements. For example if the market segment demand high security in your products, then functional requirements that are concerned with e.g. login procedures automatically get higher priority, due to the priority of the non-functional requirement.

Related practices:

- *RM.RT.GP:1 Trace requirements to other software artifacts*
- *RP.P.GP:1 Perform systematic requirements prioritization*
- *RE.GP:4 RE.GP:4Use up-to-date market analysis information for elicitation*

RA.GP:10 Make use of personas for analysis

User centric requirements elicited through marketing analysis, GAP analysis and CVA can be analyzed by comparing them against personas. This creates a common base for discussion between e.g. marketing personnel, product manager, and developers when analyzing requirements.

The persona can also be used in situations where there is a difference in belief regarding the rationale of the requirement (as a complement to the product strategies). E.g. would Daniel really appreciate this feature? How much would Daniel appreciate it?

Recommended pre-requisite practices:

- *RE.T.GP:4 Elicit requirements through personas*

Related practices:

- *OS.GP:6 Establish a teamwork mindset*
- *RA.GP:3 Analyze requirements using multiple views*

MDREPM – Main Reference Sources by Practice

The good practices in the MDREPM were created based on extensive literature research in market-driven requirements engineering. The following table contains the main reference sources used for the creation of good practices. It should be noted though that the sources do not always reflect explicitly the contents of good practices. Rather, they served as inspiration for them.

OS Organizational Support	Main Source(s)
OS.GP:1 Define roles and responsibilities	16, 28
OS.GP:2 Maintain the requirements database	20
OS.GP:3 Assign a person to manage and own the requirements process	47
OS.GP:4 Train people in requirements engineering	20
OS.GP:5 Create a cross functional team to analyze and specify requirements	20, 46
OS.GP:6 Establish a teamwork mindset	15
OS.GP:7 Create an environment that fosters innovative thinking	51
OS.GP:8 Create a company-wide glossary of terms	3
OS.S Strategic	
OS.S.GP:1 Define organizational strategies and introduce a strategic mindset	47
OS.S.GP:2 Define product strategies	55
OS.S.GP:3 Perform periodical strategic planning meetings	33
OS.S.GP:4 Spread strategic thinking throughout middle-management	33
OS.S.GP:5 Define a roadmapping process	39, 43, 44
OS.S.GP:6 Let requirements affect product strategies when applicable	50
OS.M Marketing	
OS.M.GP:1 Establish a marketing organization	47, 48, 49
OS.M.GP:2 Establish a product management organization	55
OS.M.GP:3 Know your own strengths and weaknesses	47
OS.M.GP:4 Identify and analyze competitors	47, 48
OS.M.GP:5 Identify stakeholders and map their influence	47, 48
OS.M.GP:6 Identify critical success factors for specific markets and/or key customers	47
OS.M.GP:7 Identify the basis of the competitive advantage	47, 48
OS.M.GP:8 Manage your product portfolio	47

RP Release Planning	
RP.GP:1 Understand the challenges of release planning and define which of them to address	7, 15
RP.GP:2 Use computer support for release planning activities	26
RP.GP:3 Involve different perspectives in release planning	15
RP.GP:4 Use product strategies and product roadmap to guide release planning	1, 36
RP.GP:5 Plan more than one release at a time	15, 16
RP.GP:6 Consider product evolution during release planning	25
RP.GP:7 Perform release planning post-mortem analysis	12, 56
RP.GP:8 Even out customer value between different releases	26
RP.P Prioritization	
RP.P.GP:1 Perform systematic requirements prioritization	3, 20, 52
RP.P.GP:2 Prioritize requirements based on their abstraction levels	17, 20
RP.P.GP:3 Prioritize requirements based on cost, value and risk	20, 30
RP.P.GP:4 Consider requirements dependencies during prioritization	31
RP.P.GP:5 Consider multiple stakeholders during requirements prioritization	21, 52
RP.P.GP:6 Let stakeholders' importance be a factor during prioritization	21, 47
RP.P.GP:7 Re-plan releases upon changes	15
RP.P.GP:8 Make use of must and wish lists	16
RM Requirements Management	
RM.GP:1 Introduce tool support for requirements management	20, 46
RM.CM Configuration Management	
RM.CM.GP:1 Have a change control process in place	20
RM.CM.GP:2 Define a cross-functional change control board	16, 46
RM.CM.GP:3 Control versions of your requirements	3, 20
RM.CM.GP:4 Make use of a requirements baseline	3, 20
RM.RT Requirements Traceability	
RM.RT.GP:1 Trace requirements to other software artifacts	3, 20
RM.RT.GP:2 Trace relationships between requirements	3, 20
RM.RT.GP:3 Trace requirements to their sources	3, 20
RM.RS Requirements Specification	
RM.RS.GP:1 Specify requirements attributes	10, 20
RM.RS.GP:2 Track status of requirements	2, 16, 20
RM.RS.GP:3 Provide an initial priority and effort estimate to newly specified requirements	16, 32
RM.RS.GP:4 Specify requirements in multiple abstraction levels	17, 20
RM.RS.GP:5 Workup requirements	17
RM.RS.GP:6 Record reason for rejecting requirements	3, 10

RE Requirements Elicitation	
RE.GP:1 Distinguish between end-user and customer	20
RE.GP:2 Identify requirements sources	3, 20, 23
RE.GP:3 Train personnel in eliciting requirements	20
RE.GP:4 Use up-to-date market analysis information for elicitation	47
RE.GP:5 Create elicitation channels for requirements sources	46
RE.GP:6 Consider strengths and weaknesses of competitor's products	47
RE.GP:7 Align elicitation activities with product strategies	47
RE.GP:8 Let the importance of market segments guide elicitation efforts	47
RE.T Techniques practices	
RE.T.GP:1 Elicit requirements through use cases	46
RE.T.GP:2 Elicit requirements through user groups	57
RE.T.GP:3 Elicit requirements from product reviews	-
RE.T.GP:4 Elicit requirements through personas	27
RE.T.GP:5 Elicit requirements through gap analysis	48
RE.T.GP:6 Elicit requirements through customer value analysis	53
RA Requirements Analysis	
RA.GP:1 Provide well informed effort estimates	46
RA.GP:2 Perform requirements triage	15
RA.GP:3 Analyze requirements using multiple views	3, 20, 46
RA.GP:4 Increase the quality of the requirements through inspections and reviews	20
RA.GP:5 Create requirements-based test cases during analysis	20, 46
RA.GP:6 Perform requirements risk analysis	20
RA.GP:7 Track requirements dependencies	31
RA.GP:8 Perform impact analysis for requirements	20, 58, 59
RA.GP:9 Consider non-functional requirements during analysis	20
RA.GP:10 Make use of personas for analysis	27

References

[1] Wohlin Claes, Aurum Aybuke (2005). Engineering and Managing Software Requirements. Springer.

[2] Regnell B, Brinkkemper S (2005). Market-Driven Requirements Engineering for Software Products. Engineering and Managing Software Requirements. Springer, New York NY.

[3] Kotonya G, Sommerville I (1998). Requirements Engineering – Process and Techniques. John Wiley, New York NY.

[4] Sawyer P, Sommerville I, Kotonya G (1999). Improving Market-Driven RE Processes. VTT SYMPOSIUM

[5] Gorschek T (2004) Software Process Assessment & Improvement in Industrial Requirements Engineering. Licentiate Series No. 2004/07, Blekinge Institute of Technology.

[6] Gorschek T (2006) Requirements Engineering Supporting Technical Product Management. Doctoral Dissertation Series No. 2006:01. Blekinge Institute of Technology.

[7] Carlshamre P (2002). Release Planning in Market-Driven Software Product Development: Provoking an Understanding. Requirements Eng 7:139–151

[8] Brinkkemper S (2004) Requirements Engineering Research the Industry Is and Is Not Waiting For. Requirements Engineering: Foundation for Software Quality (REFSQ'04)

[9] Xu L, Brinkkemper S. (2005) Concepts of Product Software: Paving the Road for Urgently Needed Research. Proc. of 1st International Workshop on Philosophical Foundations of Information Systems Engineering (PHISE'05). Lecture Notes in Computer Science, Springer Verlag.

[10] Robertson S, Robertson J (1999). Mastering the Requirements Process. Addison Wesley.

[11] Karlsson L et al. (2002) Challenges in Market-Driven Requirements Engineering - An Industrial Interview Study. REFSQ'2002 Workshop.

[12] Karlsson L et al. (2003) Market-Driven Requirements Engineering Processes for Software Products - A Report on Current Practices. Proceedings of International Workshop on COTS and Product Software.

[13] Keil, M., Carmel, E (1995) Customer-Developer Links in Software Development. Communications of the ACM. 38 (5), 1995.

[14] Davis A (2003) The Art of Requirements Triage. IEEE Computer 36 (3): 42-49.

[15] Davis A. (2005) Just Enough Requirements Management: Where Software Development Meets Marketing. Dorset House Publishing. New York, NY.

[16] Regnell, B., Beremark, P., Eklundh, O., A Market-driven Requirements Engineering Process - Results from an Industrial Process Improvement Programme. *Requirements* Engineering Journal, 3(2):121-129, Springer, 1998.

[17] Gorschek T, Wohlin C, Requirements Abstraction Model, Requirements Engineering, Volume 11, Issue 1, Mar 2006, pp. 79-101

[18] Hall, T Beecham, S Rainer, A (2002). Requirements problems in twelve software companies: an empirical analysis. Software, IEE Proceedings 149 (5): 153- 160

[19] Lubars, M, Potts C, Richter C (1993), A Review of the State of the Practice in Requirements Modeling, in: Proceedings of the IEEE International Symposium on Requirements Engineering, San Diego, CA: IEEE Computer Society, pp. 2-14.

[20] Wiegers K (2003) Software Requirements. Microsoft Press. Redmond, Washington.

[21] Regnell B, Höst M, Natt och Dag J, Beremark P, Hjelm T. An industrial case study on distributed prioritization in market-driven requirements engineering for packaged software. Requirements Eng 2000; 6(1):51–62

[22] Nikula, U, Sajaniemi, J, Kälviäinen, H (2000) A State-of-the-Practice Survey on Requirements Engineering in Small- and Medium-Sized Enterprises. TBRC Research Report 1, Telecom Business Research Center. Lappeenranta, Lappeenranta University of Technology.

[23] Sommerville I, Sawyer P (1997) Requirements engineering: a good practice guide. John Willey & Sons.

[24] Gorschek T, Tejle K (2002) A Method for Assessing Requirements Engineering Process Maturity in Software Projects. Blekinge Institute of Technology. Master Thesis Computer Science no. MSC-2002:2

[25] Saliu O, Ruhe G (2005) Supporting Software Release Planning Decisions for Evolving Systems. Proceedings of the 2005 29th Annual IEEE/NASA Software Engineering Workshop (SEW'05)

[26] Ruhe G (2005) Software Release Planning. Handbook of Software Engineering and Knowledge Engineering

[27] Aoyama M (2005) Persona-and-Scenario Based Requirements Engineering for Software Embedded in Digital Consumer Products. Proceedings of the 2005 13th IEEE International Conference on Requirements Engineering (RE'05)

[28] Carlshamre P, Regnell B (2000) Requirements Lifecycle Management and Release Planning in Market-Driven Requirements Engineering Processes. Proc. 11th International Workshop on Database and Expert Systems Applications.

[29] Gorschek T, Davis A. Requirements Engineering: In Search of the Dependent Variables. Submitted to Information and Software Technology

[30] Karlsson J, Ryan K (1997) A Cost-Value Approach for Prioritizing Requirements. IEEE Software 14(5): 67-74, Sep/Oct 1997

[31] Carlshamre, P.; Sandahl, K.; Lindvall, M.; Regnell, B.; Natt och Dag, J. (2001) An industrial survey of requirements interdependencies in software product release planning, Requirements Engineering, 2001. Proceedings. Fifth IEEE International Symposium on , vol., no.pp.84-91, 2001

[32] Rautiainen, K.; Lassenius, C.; Vahaniitty, J.; Pyhajarvi, M.; Vanhanen, J., "A tentative framework for managing software product development in small companies," *System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on* , vol., no.pp. 3409- 3417, 7-10 Jan. 2002

[33] Berry M, Strategic Planning in Small High-Tech Companies. Long Range Planning 31(3):455-466, 1998.

[34] Vahaniitty, J., Key Decisions in Strategic New Product Development for Small Software Product Businesses, *Euromicro Conference, 2003. Proceedings. 29th* , pp. 375- 383, 1-6 Sept. 2003

[35] Deek F, McHugh James A M, Eljabiri O M (2005) Strategic Software Engineering – An Interdisciplinary Approach. Auerbach Publications.

[36] Vähäniitty J (2005) A Tentative Framework for Connecting Long-Term Business and Product Planning with Iterative & Incremental Software Product Development.
Proceedings of the Seventh International Workshop on Economics-driven Software Engineering Research. St. Louis, Missouri, pp 1 - 4

[37] Hitt, M., D. Ireland, and R. Hoskisson, Strategic Management: Competitiveness and Globalization: Concepts, 2 ed., West Publishing Company, 1997.

[38] Rajala, R., Rossi, M., Tuunainen, V.K. A Framework for Analyzing Software Business Models. European Conference on Information Systems 2003.

[39] Lehtola L, Kauppinen M, Kujala S, "Linking the Business View to Requirements Engineering: Long-Term Product Planning by Roadmapping," *re*, pp. 439-446, 13th IEEE International Conference on Requirements Engineering (RE'05), 2005

[40] Ebert, C., "Understanding the Product Life Cycle: Four Key Requirements Engineering Techniques," *IEEE Software*, vol.23, no.3, pp. 19- 25, May-June 2006

[41] Kamsties, E., Hörmann, K., Schlich, M.: "Requirements Engineering in Small and Medium Enterprises: State-of-the-Practice, Problems, Solutions and Technology Transfer", Proc. Conference on European Industrial Requirements Engineering (CEIRE'98), Hammersmith, UK, 1998.

[42] Juristo, N.; Moreno, A.M.; Silva, A., "Is the European Industry Moving Toward Solving Requirements Engineering Problems?"*, IEEE Software* , vol.19, no.6pp. 70- 77, Nov/Dec 2002

[43] Phaal R., C. Farruk, and D. Probert, "Technology Roadmapping – A Planning Framework for Evolution and evolution", Technological Forecasting & Social Change, Vol. 71, No. 1-2, 2003, pp. 5-26

[44] Kappel T., "Perspectives on Roadmaps: How Organizations Talk about the Future", The Journal of Product Innovation Management, Vol. 18, 2001, pp. 39-50.

[45] Ruhe G et al, (2005) "The Art and the Science of Software Release Planning". *IEEE Software*, Vol. 22, No. 6, pp. 47-53, November/December

[46] Wiegers K (2006) More about Software Requirements – Thorny Issues and Practical Advice. Microsoft Press. Redmond, Washington.

[47] Johnson G., Scholes K., "Exploring Corporate Strategy", 7th edition, Prentice Hall, London

[48] Wallin L. (2003) "Industriell Marknadsföring på KTH-Syd", LOWEK, Bromma

[49] Kotler P. et al. (2002). Principles of marketing. Harlow, England ; New York, Prentice Hall.

[50] Mintzberg, H et al. Strategic Safari. The Free Press

[51] Humphrey S. (1997) Managing Technical People. Addison-Wesley, 326p.

[52] Berander P. (2004) “Prioritization of Stakeholder Needs in Software Engineering - Understanding and Evaluation”, Karlskrona. Blekinge Institute of Technology

[53] Harmon R.R and Laird G., (1997) “Linking marketing strategy to customer value: implications for technology marketers”, Innovation in Technology Management - The Key to Global Leadership, page 896-900

[54] Dahan, E.; Hauser, J.R., (2002) “The virtual customer”, Journal of Product Innovation Management, Elsevier, Volume 19, Number 5, page 332-353

[55] Lehmann D., Winer R. (2002) “Product Management”, 3rd edition, McGraw-Hill Irwin.

[56] Karlsson L. et al. (2003) “Post-Release Analysis of Requirements Selection Quality – An Industrial Case Study”

[57] Joseph A. et al. (1993) “Techniques for requirement elicitation”, IEEE Computer Society, pages 152-164.

[58] S.A. Bohner and R.S. (1996) “Software impact analysis” IEEE Computer Society Press

[59] M Lindvall, (1997) “An Empirical Study of Requirements-Driven Impact Analysis in Object-Oriented Systems Evolution”, Ph.D. thesis no 480, Linköping Studies in Sceince and Technology, Linköong, Sweden

[60] Simmons E. ”Requirements Triage: What Can We Learn from a "Medical" Approach?”, IEEE Software, page 86-89,

[61] Zahran, Sami. Software Process Improvement – Practical Guidelines for Business Success. *Pearson Education*, 1998

[62] Ahern Dennis M. et al, CMMI Distilled – A Practical Introduction to Integrated Process Improvement. Addison Wesley 2001

[63] CMMI Product Team. Capability Maturity Model Integration Version 1.1. Software Engineering Institute Carnegie Mellon. 2001

[64] Sawyer Pete, Sommerville Ian, Viller Stephen. Capturing the Benefits of Requirements Engineering. IEEE Software March/April 1999

[65] Kauppinen M. Aaltio T. Kujala S. Lessons Learned from Applying the Requirements Engineering Good Practice Guide for Process Improvement. Software Quality - ECSQ 2002 : 7th International Conference, Helsinki, Finland, June 9-13, 2002. Proceedings

[66] Sawyer P. "Maturing Requirements Engineering Process Maturity Models", in J. Maté and A. Silva (Eds), Requirements Engineering for Socio-technical Systems, Idea Group Inc. (invited contribution), in press 2004

[67] Awan R (2005) Requirements Engineering Process Maturity Model for Market Driven Projects – The REPM-M Model. Blekinge Institute of Technology. Master Thesis Software Engineering no. MSE-2005-17.

[68] Gorschek, T.; Garre, P.; Larsson, S.; Wohlin, C., "A Model for Technology Transfer in Practice," Software, IEEE , vol.23, no.6pp.88-95, Nov.-Dec. 2006

[69] Robson C, *Real World Research*, 2nd Edition, Blackwell Publishing, Malden, 2002

[70] J.W. Creswell, Research Design: Qualitative, Quantitative, and Mixed Methods Approaches, 2nd Edition, Sage Publications, Thousand Oaks, 2003

[71] Novarita R. J., Grube G. (1996) Benefits of Structured Requirements Methods for Market-Based Enterprises, Telelogic.