Global Centre for ICT in Parliament Working Paper No. 1

January 2008

Requirements Management – Defining the Project Scope and Developing Use Cases

Avinash Bikha

Abstract

Often when an ICT system implementation project starts, project teams immediately get into the details of requirements even before they have established the project scope supported by a proper understanding of the real needs for the system and the stakeholder community or the constraints under which it is to be built. Starting to define requirements without this kind of foundation often causes problems during the project. Some projects are completed before the team realizes that the system produced does not meet any of the stakeholders' real needs and expectations. To avoid these kinds of problems, it is useful that project teams consider and try to adopt simple, but valuable guidelines to establish a good understanding of the stakeholder community, demonstrate an understanding of the problem to be solved by the ICT intervention, capture the real needs of the stakeholders and the system features required to fulfill these needs and ensure that the views of the stakeholder community are actively and appropriately represented and communicated throughout the project

INTRODUCTION

e-Parliaments

As Parliaments from around the world increasingly employ new technologies (ICT), they find themselves exposed to the multitude of challenges that come with them at the different levels in- and outside the parliamentary context.

Similar to other sectors, the introduction of the *"buzzword"* ICT in the Parliament creates a range of expectations. To note some examples:

- Citizens expect Parliaments that have deployed ICT to be more *open, transparent* and providing more channels through which they can *participate* and *interact*.
- MP's with access to new technologies are expected to fulfill their roles of representation, legislator, lobbyist (for constituent causes) and ombudsman practically *anywhere* and *anytime*.
- Parliaments as producers of (paper intensive) legislative documentation are called to improve their overall *efficiency* in terms of administering parliamentary sessions and reducing paper communications.

The list of expectations is off course longer, as ICT is able in many ways to change the way how Parliaments could work, for better or worse. With the rise of expectations and the multitude of solutions that ICT could provide, it becomes ever more important for ICT policymakers and decision makers to have a structured approach for introducing ICT solutions in Parliament that make sense and are beneficial.

It is very easy and popular to name ICT as the solution for everything; doing so creates high expectations which eventually might not be met. Choosing ICT as the right solution that responds to real problems and needs is a more complex exercise. The leadership in Parliament, ICT policymakers and administrators are able to apply structured approaches and be better equipped with the necessary tools to create a vision, define ICT strategies and manage the subsequent process of organizational change and the transformation of information delivery in and around Parliament, in a way that expectations are met.

A structured approach

Every introduction of ICT in the parliamentary work domain, be it the very basic purchase and installation of computers and printers, or a lengthy 12 months project to implement an enterprise-wide legislative information system, should be done as part of a higher rationale:
a strategic vision.

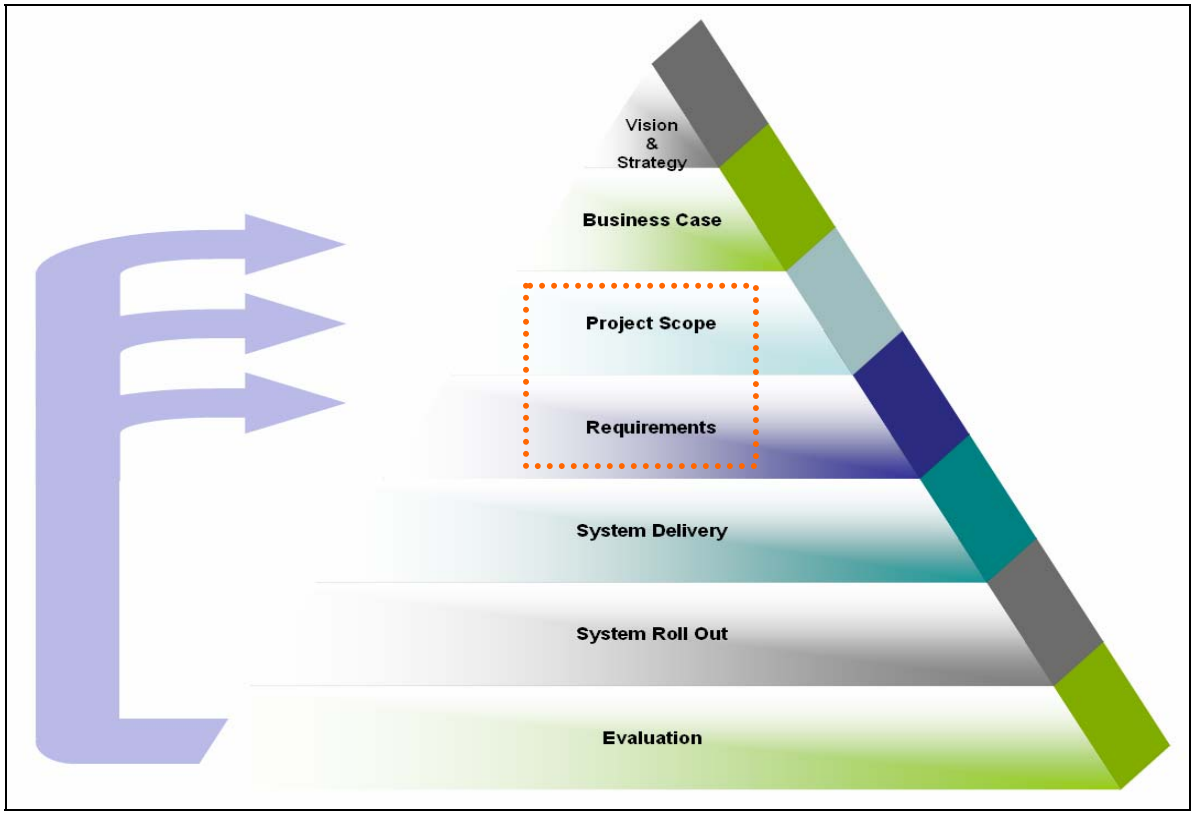
ICT comes at a price and every resource spent should be justified i.e. that ICT responds to real problems and needs. Approaching ICT strategically subsequently leads to making validated choices and the right formulation of ICT projects through which the right systems will be implemented. Maintaining the linkages between that higher rationale and implementing the right systems is a challenging and complex exercise, which in most cases spans years.

The Global Centre for ICT in Parliaments, through its ongoing research & analysis, and technical assistance activities, offers guidance to Parliaments by providing them with information and recommendations on approaches, tools and methodologies which can support Parliaments in this

process. From the Global Centre's point of view the road to ICT in Parliament encompasses the following levels: *vision and leadership* and *processes and organizational challenges*.

1) **Vision and leadership:** The adoption of innovation and technologies in Parliament implies an important institutional transformation and complex decisions ranging from human resources development to greater transparency and accountability, and must therefore emanate from a *political decision*. Realizing the future as envisaged by policy-makers needs also well defined *strategic plans*, based on a building block approach and modular implementation that allows for greater *control* and flexibility of the *implementation process*.
2) **Processes and organizational challenges**: The introduction of ICT in Parliament touches *people* and *processes* throughout the *organization* and therefore it requires an *integral multi-stakeholder approach*, which can yield more commitment and support throughout the implementation process.

The handbook in front of you is part of a structured approach that facilitates the introduction and integration of ICT in Parliament (figure 1.1 below) that maintains the linkages between the rationale for ICT (vision and strategy) and the subsequent stages of business case development, project scope definition, requirements development and the system delivery, roll out and evaluation.



From top to bottom let us look at the chronology and consider every stage:

1. After the *vision* for ICT in Parliament is clear and the political leadership has given its *support* for its realization, the *ICT strategic plan* is defined which outlines specific *objectives* that should be realized.

2. *Business cases* are then developed to describe the rationale, the political and economical justification for a chosen intervention (programmes or projects) which will lead to the achievement of one of more objectives mentioned in the strategic plan. The business case specifies the resources, time and effort and the area of parliamentary work where the intervention is needed.

3. Once the business case is approved the project start-up phase begins. Apart from the formalities of establishing the framework for project steering and control (for example through a committee or a project board) also the *project scope* needs to be determined. The business case mentions the outcomes of the project and the *project scope* details the *stakeholders* involved, the *problem domain* and the *solution:* a *description of the system* to be implemented and the *high-level requirements*.

4. Once the system and the high-level requirements have been defined the *detailed requirements* are elicited from stakeholders (e.g. future users) and captured through *requirements documentation methods*.

5. The captured set of detailed requirements becomes input for the *system delivery stage*, during which the system is built and tested against the requirements. Once delivered it is *rolled-out* into the organization. This implies the training of users and putting user support organization in place.

6. After a certain period of use the system can be evaluated in terms of its performance, usability and other metrics. Depending on the evaluation results the business case could be altered, follow-up projects could be defined, or simply the requirements are modified making technical changes and updates necessary.

The focus of this handbook is on the stages three and four: ***Defining the Project Scope*** and ***Documenting and Managing Requirements*** (see the dotted box in figure 1.1.)

The benefit of this handbook

Often when an ICT system implementation project starts, project teams immediately get into the details of requirements even before they have established the project scope supported by a proper understanding of the real needs for the system and the stakeholder community or the constraints under which it is to be built.

Starting to define requirements without this kind of foundation often causes problems during the project. Some projects are completed before the team realizes that the system produced does not meet any of the stakeholders' real needs and expectations. Other project teams find it impossible to produce a stable set of requirements or even to agree on one at all, due to a lack of understanding and agreement on the initial scope.

To avoid these kinds of problems, it is useful that project teams consider and try to adopt the following simple, but valuable guidelines:

- Establish a good understanding of the stakeholder community
- Demonstrate an understanding of the problem to be solved by the ICT intervention
- Capture the real needs of the stakeholders and the system features required to fulfill these needs
- Ensure that the views of the stakeholder community are actively and appropriately represented and communicated throughout the project

In this handbook for ICT administrations in Parliament we shall look at approaches and working methods that will guide project teams in these activities, and the positive effect that these have on the process of requirements development, management, and subsequent implementation of systems. This handbook describes how to develop your requirements and align them with the interests of the project stakeholders. It also explains a valuable methodology for gathering and documenting requirements: use case modeling.

Disclaimer: The approaches as described in this handbook should be tailored to the environment of each individual project. The methodologies are suitable for projects of various sizes; following this handbook will not guarantee success but it will be helpful, especially for those who are new to formal developing and managing requirements.

Audience

"Part I: Defining the Project Scope" is primarily written for ICT project managers who are responsible for implementing information systems in Parliament. This handbook will offer them useful guidelines on how to gather stakeholder support and involvement during the start-up phase of the system implementation and define a project scope, shared by all.

In chapters 1 and 2 we look at from where to start with the initial requirements development activities and who should be involved. In chapter 3 we look at the concept of the requirements pyramid which, through its structure can guide the process of developing requirements, resulting in an inclusive project scope.

In chapter 4 we look at how to record the results of the requirements development process in a scope document. In chapter 5 we look the application of the *requirements pyramid* and *scope document* through the use of case examples.

"Part II: Developing Use Cases" is aimed at business analysts / information analysts who under the supervision of the ICT project manager are responsible for further specifying and documenting requirements.

Chapter 6 is specifically dedicated to technical guidelines for writing and managing use cases, which are a very effective way to capture requirements.

In addition to ICT project managers and business analysts / information analysts this handbook is certainly of interest to senior ICT administrators in Parliament who are responsible for ICT programmes/projects, and who aspire to raise their operation to a higher level in terms of having a structured systems implementation approach with a higher degree of control.

PART 1: DEFINING THE PROJECT SCOPE

CHAPTER 1: DEPARTING FROM THE BUSINESS CASE

An ICT systems implementation project is usually born from a set of problems which have been previously identified (problem domain). This handbook assumes that the project team dedicated to developing requirements departs from a **business-case document** which contains:

- the problem domain
- the affected stakeholders
- the envisioned solution

The business-case document describes the need for an ICT intervention, through a programme or project, during which an ICT system is to be implemented. The business case also states the cost and resources it will take to realize the system, and who will endorse the project.

The business-case is commissioned and owned by high level decision makers and sponsors: people with political and resourceful backing in- and outside the organization. For example in Parliament the (Deputy) Speaker, MP's, Head of ICT, quaestors (budget supervisors) could all be involved in creating the business case of a Parliamentary Legislative Information System. Often the same people will participate in the project steering committee or project board which oversees the project.

The business-case being the departure point for scope definition and requirements development is of much value to the project team, since it gives a first impression of the need for the system and necessary background information to understand the problem domain, the affected stakeholders and the envisaged solution.

Sometimes a business case document will not exist at the start of the project; or it is very badly documented. In that case the project team should resort to all other documents leading up to the initiation of the project: documents that describe the problem, stakeholders and solution. This information could found in letters, memos, emails, reports and other correspondence.

See below the example case of the e-PS project.

The Business Case for the e-PS project

Upon recommendations from the "Modernization of the Parliament" (MoP) committee, the Deputy Speaker of the (……) Parliament commissions the development of an online petitioning system: e-Petition System (e-PS)

e-PS Project & Board

The development of e-PS will be realized through the e-PS Project, which will be governed by a Project Board presided by the Deputy Speaker. The other members of the board include:

- the Head of ICT in Parliament
- the project manager responsible for the delivery of e-PS
- representatives from the treasury department
- representatives from the stakeholder community:
    - three MPs who promote transparency, accountability and citizen participation

- one Committee chairman who is keen on receiving more citizen feedback
- An expert on Citizen Participation and e-Democracy, who represents the citizen's view

The MoP Committee

During its research the MoP has: created an overview of the current level of ICT / automation in the Parliament; gathered information from two parliaments that have online petitioning systems based on Open Source technology; made an overview of the stakeholders and the main processes in Parliament where e-PS will have an impact.

The recommendations by the MoP covers the:

1. **Problem analysis:**
   - Citizens have very few ways to communicate with Parliament other than in person (by traveling from remote areas to the capital).
   - MP's comment that there is minimum citizen feedback and opinion in the current legislative process
2. **Stakeholders:** Citizens, MPs and their staff and the Parliament
3. **New technologies & Opportunities:** Currently Open Source technology represents a viable alternative to propriety software developed by third parties. The main reason is that the cost of adoption and maintenance has considerably gone down as in recent years Open Source technology has become very accessible through the different online repositories and available documentation (wiki's and downloadable manuals)
4. **Envisaged solution:** An online petitioning system based on Open Source technology that will form an extension of the current parliamentary portal, enabling citizens to engage in interaction with MPs and the Parliament and express their opinion.

CHAPTER 2: INTRODUCING STAKEHOLDERS AND USERS

Before starting with defining the project scope and developing the initial requirements it is important to understand the stakeholder community and how to involve them in the project. This is an important prerequisite for:

- Establishing a proper understanding of the problem domain and those who are affected.
- Preparing for the requirements gathering workshops. If workshops are meant to gather the system requirements, then it is good to know who to invite and understand what portion of the daily parliamentary work processes they represent.
- Identifying the sources for system's requirements: requirements come from many sources, including future users and domain experts. Understanding these sources of requirements will allow the project team to decide how best to obtain information from them.

For the project to deliver an ICT system that will find wide acceptance across the stakeholder community there should be a clear understanding of the stakeholders and their particular needs. It is also important that the people asked to become involved in the project understand their role and the responsibilities that they are expected to fulfill.

In this section we will look at:

- What are stakeholders and why are they important?
- The different types of stakeholders
- The role of stakeholders in the project
- How to best identify and involve the stakeholders in the project

Who Are Stakeholders?

In the context of ICT in Parliament a stakeholder is: *"any individual who is materially affected by the introduction of information and communication technology in Parliament"*.

Using this definition, the first stakeholders who come to mind are ***future users of the system*** and the ***developers and administrators of the system***. In the parliamentary context there are specific stakeholders; for example, the decision to introduce a new legislative information system in Parliament could affect:

- Clerks, MPs and their staff
- The ICT department in Parliament
- The (Deputy) Speaker of Parliament and other key decision makers
- Political parties, committees and leaders
- The treasury department
- Third party software developers
- tax paying citizens

Depending on the problem domain, one can see who is directly / indirectly affected by the original problem. Figure 2.1 shows up the relationship between the stakeholders, the problem and

its solution. The stakeholder community consists of those people that experience the problem and/or are materially affected by the outcome of the solution.

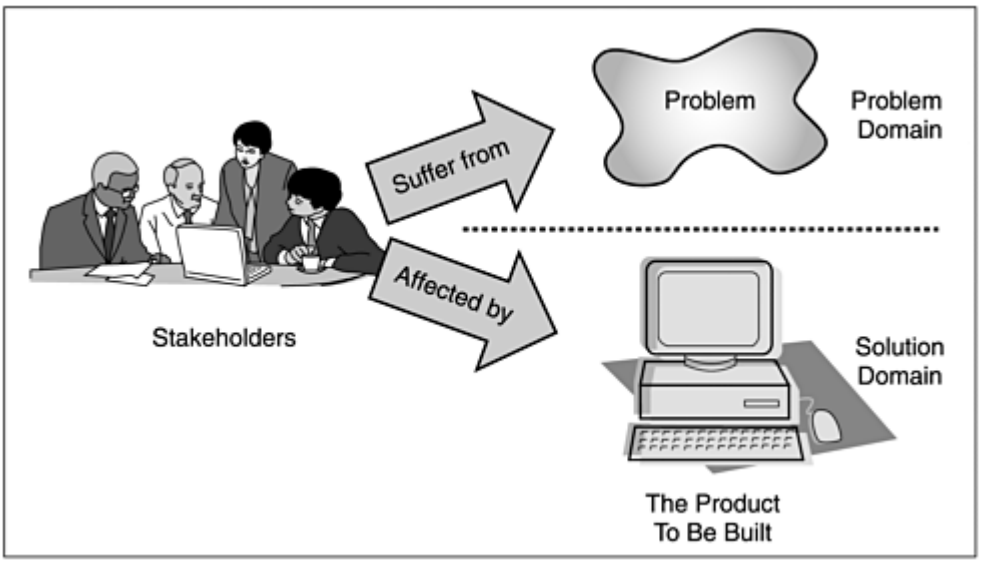


Figure 2.1

Identifying Stakeholder Types

The first step to understand the stakeholder community is to identify the **types** of stakeholders affected by the system. By **"stakeholder type"** is meant the classification of a set of stakeholders sharing the same characteristics and relationships with the system and/or the project that produces the system. Stakeholders typically fall into the following categories (analogue to the legislative information system example):

- **Users:** The most obvious types of stakeholder are the actual users of the system.
- **Sponsors:** The ICT managers, financiers, department heads, steering committee members all the way up to the level of the Speaker of Parliament. These stakeholders are only indirect users of the system, or are interested in the outcome of the ICT intervention, which would influence higher goals such *"strengthening Parliament"*, *"transparency, accountability and more openness towards citizens", "citizen participation"*.
- **Developers:** Project managers, system maintainers, testers, technical support staff, designers, programmers, technical writers, production staff, and any other types of developers involved in the production and support of the system.
- **Authorities:** Experts in a particular aspect of the problem or solution domain. These include legislative authorities, standards organizations, organizational governance departments, external and internal regulatory bodies, experts of legislative processes, and technology experts.

The actual list of stakeholder types for a project will be more concrete than this; it will identify specific user types, departments and organizational units. The key thing is to ensure that all those affected by the arrival of the system are considered. When identifying the stakeholder types, focus on understanding how they are affected by the project and the system it will put in place.

Stakeholders in the e-PS project

The business case for the e-PS Project identifies the following problem domain:

1. Citizens have very few ways to communicate with Parliament other than in person (by traveling from remote areas to the capital).
2. MP's comment that there is minimum citizen feedback and opinion in the current legislative process and committee work.

Each of these problem domains have their own directly affected stakeholders:

- Citizens: are at this moment not satisfied with the minimum level of interaction and being able to express their opinion directly to those who represent them
- MP's staff: are at this moment faced with the challenge on providing Committees and MP's with accurate and representative citizen opinion. Using a multitude of sources such opinion polls etc it is difficult to get direct information.
- MP's: who think it is in their interest to be connected to their constituents
- The ICT department who needs to find and allocate resources to develop, manage and support the new system

Lesser directly affected stakeholders of the problem domains are the key decision makers of the project such as the Deputy Speaker of the Parliament, the head of ICT in Parliament, the project manager and the treasury supervisor.

Though very important for the project and its mandate, these latter mentioned stakeholders will not provide the project team with requirements.

Stakeholder Representatives and Stakeholder Roles

Once stakeholders have been identified, the next step is to define their roles on the project to enable a fruitful representation of all stakeholder views. Appropriate people can then be recruited to fulfill these roles. The objective is to recruit a set of stakeholder representatives delegated and committed to the project, who can regularly attend project meetings, participate in the workshops and really add value to discussions.

A **Stakeholder Representative** is a member of the stakeholder community who is directly involved in determining the requirements scope of the project. A stakeholder representative represents one or more stakeholders from the same type. For example an MP stakeholder representative could sit in on behalf of other MPs; or a Clerk who has been delegated to the project represents the Clerk user community and their respective needs and opinions. In the case of a system which will be used to communicate with citizens, citizen representatives (citizen volunteers) should somehow be recruited and involved.

Before you can recruit an appropriate set of stakeholder representatives, you should first define how these representatives will participate in your project, i.e. how many hours of the working-week are they expected to dedicate to the project and what their exact role is.

The Role of Stakeholders: Primary Source of Requirements

Stakeholders and stakeholder representatives "own the problem" and are affected, directly or indirectly, by the proposed solution. For this reason they are also the primary source of requirements. The problem domain itself being something intangible, it is the role of the stakeholder representatives to bridge the gap between the problem domain and the specification of the proposed solution. Their participation in the discussions how to address the problem domain using ICT solutions gives life to stakeholder expectations, needs and requirements. The resulting requirements documentation in itself is *a formal articulation of the stakeholders' view*. Well documented and updated requirements must at all times reflect the viewpoint and expectations of the stakeholders.

Figure 2.2 sums up the relationships between the stakeholders, the system, and the requirements documentation. The requirements act as the representation of the stakeholders' goals.

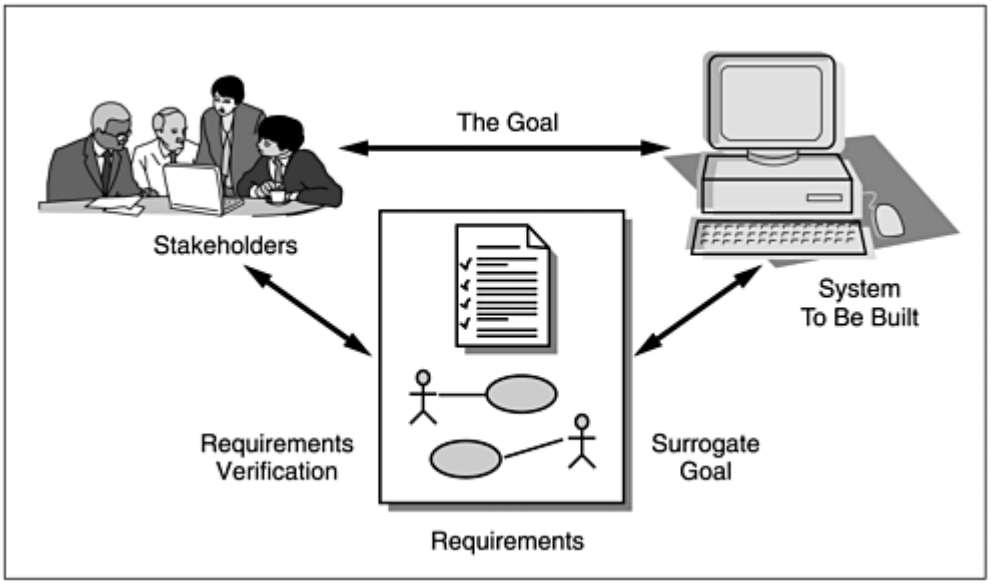


Figure 2.2

It is important to consider which stakeholder types will be the sources of the requirements when defining stakeholder roles and appointing stakeholder representatives. All stakeholder types should be represented, but it is important to focus attention on those which will actually provide the bulk of the requirements information. The treasury department and funding parties are a type of stakeholder and should be represented in the project, but they will not provide many or any requirements for the system during requirements gathering workshops. Their interest lies more at the outcome level and higher, long term goal of the project.

Stakeholder representatives who will act as the primary source of requirements information should be directly involved in the project and have a clear understanding of the role that they are expected to play in the requirements gathering process. Many projects run into trouble because the stakeholder representatives do not know why they are involved; they are not actively engaged in the project and do not contribute during a requirements gathering workshops.

Quality of stakeholders' involvement

Stakeholder representatives' indifference may manifest itself by their unavailability when trying to obtain needs, discuss features and requirements; not making time to review and sign off on requirements documentation; not committing themselves for the full lifetime of the project; or just plain forgetting why they are involved in the first place. The quality of the final result is often a direct result of the quality of the participation of the stakeholders. It is therefore critical that the project manager or the requirements development team-lead signals when these "symptoms" manifest themselves during requirements gathering workshops and related activities.

To overcome *stakeholder representative's indifference* it is advisable to clearly define the stakeholder roles and ensure that stakeholder representatives understand their roles and their responsibilities in representing different stakeholder communities. The role of stakeholder representative includes but is not limited to the following:

- Representing the views and needs of the section of the wider stakeholder community they represent
- Actively participating and contributing to requirements gathering workshops and reviewing requirements documentation
- Actively championing the project to the rest of the stakeholders' community
- Doing independent research, adding value to the discussions and being critical of proposed requirements
- Participating in the assessment and verification of the quality of system delivered

There are many ways of involving the stakeholder representatives in the project. If a legislative information system is developed for Parliament-internal use, it is advisable to invite people to participate who are familiar with parliamentary work domain and processes such as legislative drafting, enacting, amending and repealing processes.

If the envisaged system also aims to improve the communication of legislative information towards the public, experts who understand citizen's information/communication needs could be invited to join in the requirements development process. Also the project can employ tools such as surveys to get a better picture of the citizen expectations, needs and priorities.

Again, how the stakeholders' view is represented in the project is less important; that the stakeholders view is represented is more important. Remember that stakeholders own the problem and are the source of the project's requirements and if the system is not a success with stakeholders on board, then it is not a success.

Managing conflicts

Keeping stakeholders on board can be challenging. It is very likely that different stakeholders will come up with different and sometimes conflicting requirements. The requirements gathering process can get stagnated as the project wants to keep all stakeholders satisfied and therefore on board. This calls for tactful leadership coupled with transparency. Conflicting requirements can be resolved by giving priorities to the requirements and communicating and providing transparency into the prioritizing process. If done properly the prioritizing of requirements will be consistent

with the higher goals of the project (as described in the business case documentation). In chapter 3 the MoSCoW method of prioritizing is discussed.

Users: A Very Important Class of Stakeholder

As one of the most important types of stakeholder, future system **users** are essential to most ICT projects. The amount of user involvement required is variable: one user may be a full-time user representative permanently assigned to the project, another may be a member of a usability test panel, and yet another may simply submit ideas and feedback by questionnaire.

When defining the stakeholder roles, you should take into consideration the amount of user involvement necessary to support the project, the style of user involvement most suited to the project, the availability of the users to the project and the level of commitment to the project. In most projects it is impossible to involve all users of the system. What is essential is that the selection of stakeholder representatives includes relevant user representatives and that for each type of user there is clearly defined representation.

Examples of relevant users of Parliamentary information systems that could be representative are: staff of the MP's office, MP's themselves, citizens using the system online, Hansard writers, Bill drafters, Proofreaders, system administrators etc. What distinguishes these users is that they represent every possible area of interaction with the system, for example from writing /creating an online petition (Citizen-to-Parliament systems), to drafting bills to retrieving records from the system to prepare the Hansard (parliamentary legislative information systems).

The system users' community must clearly understand by whom and how they are represented in the project, and delegated user representatives must understand their responsibilities toward the system users' community they represent**.**

Techniques to Involve the Stakeholder Representatives

Various techniques can be used to involve the stakeholder representatives in the project. They include (but are not limited to) the following:

- Requirements Gathering Workshops: can be a very useful way to gather requirements, build coherence in the project team environment, and develop an understanding of the system. They should be well planned with an agenda that is sent to participants beforehand along with any background material.
- Interviews: Interviews are among the most useful techniques for involving stakeholders in a project.
- Questionnaires: Questionnaires are a very useful technique, particularly when a large number of stakeholder representatives are involved (citizens).
- Focus Groups: A focus group allows you to sample sets of stakeholder representatives to get their perspective on what the system must do. Focus groups tend to be used to gather specific feedback on specific topics.
- Advisory Boards (or Committees): An advisory board is a kind of standing focus group. It provides a way to gather stakeholder perspectives without the overhead of establishing a focus

group. The disadvantage compared to a focus group is that the composition of the advisory board cannot be varied according to topic.
- Reviews: Reviews are formal or informal meetings organized with the specific intent to review something, whether a requirement document or a prototype of the system.
- Role Playing: This is a facilitation technique that is typically used in combination with workshops to obtain specific information or feedback.

The choice of technique is very closely related to the definitions of the stakeholder roles and the availability of actual individuals to take on the responsibilities defined by the roles. There is no point in deciding that a project will have full-time user representatives attending weekly requirements gathering workshops if there are no experienced users in a position to take on this level of commitment.

This is why a continuous three steps evaluation of: **identify stakeholder → determine stakeholder role → involve stakeholder** should be applied iteratively by the project manager and the level of stakeholder representative involvement constantly monitored.

Paying attention to the stakeholder community and continuously involving them in the project in appropriate ways significantly increases the chances of project success. The technique most closely associated with the development of requirements and the subsequent creation of use cases is the *requirements gathering workshop*.

Workshops are also useful to investigate many other aspects of a project, for example to brainstorm on the characteristics of citizens and their information needs.

CHAPTER 3: THE REQUIREMENTS PYRAMID

As discussed in the previous chapter, prior to building a system there is the intensive phase of sitting together with stakeholders, gathering their input in terms of expectations and needs on the introduction of new technologies in their work domain. While gathering all this bulk input it is important to maintain a structured overview of everything and possibly to distinguish between different levels of stakeholder input and categorize them accordingly.

The Requirements Pyramid is one such tool which can be used by the project team for structuring the bulk of gathered requirements input. It has the advantage that requirements information can be distinguished and categorized into Needs, Features, and Requirements.

Requirements Pyramid: Needs

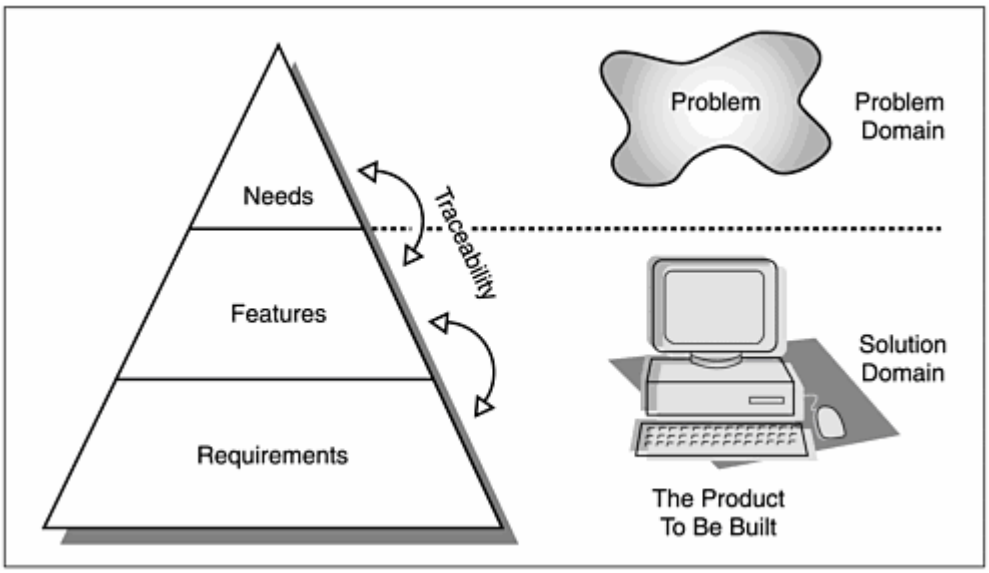


Figure 3.1

When beginning with the requirements gathering workshops, the project team will try to analyze and understand what the (stakeholder) needs are (figure 3.1). Having a good understanding of the needs is the first level of the requirements pyramid.

Besides workshops individual interviews with experienced users can be very helpful to understand their needs, going step-by-step through the daily work processes and identifying the bottlenecks.

To understand the needs the project team needs to understand the daily work processes. In the parliamentary context it simply it needs to know:

- What are the daily tasks? How is legislation drafted? What are the different steps? Who is doing what? What is the process of amending legislation? What are the input documents and output documents? What are the different steps? When do you proceed from one step to the other? And what is the role of the MP in this process? And the role of the Clerk? When does a task start and end? Etc. etc.

Creating Process maps

Gathering this type of basic information on the daily work processes should give the project team input to create **process maps** of the current situation. Process Maps are descriptions of the main processes such as *creating a petition, drafting legislation*, *amending legislation* etc. Process maps describe how work is currently carried out in the office without taking into account the system which will be built by the project. Process maps can be accompanied by flow charts. Each process depending on its complexity could be broken down into sub processes.

Process maps, once drawn up, are circulated, reviewed and discussed during the next workshops. The project team should get confirmation from the participants that this is indeed how work is currently being carried out. Once this base-line of work processes is achieved and agreed upon by everyone, the project team can then progress to identifying bottlenecks in the processes.

Bottlenecks = Needs

In the subsequent iteration of workshops the project team and participants can depart from the same baseline and understanding of work processes, and go through each and every process to have participants provide as much input as possible on what is currently not working well in the current situation, what are the bottlenecks and what could be ways to improve the current processes by using the envisaged solution or system. In other words what are the needs?

As members of the user community will express their opinion about the existing working process and give their view on how the processes could be improved with the envisaged solution, the discussions while trying to be participative and inclusive as possible to all parties, should remain aligned with business-case. It is therefore the responsibility of the project team lead or workshop moderator to keep the workshops on course. Time will be limited, discussions need to progress, for each process map *needs* need to be gathered.

In the extreme case of stakeholders not agreeing at all with the envisaged solution and too much deviation from the business-case this should be reported back to the project board; but that is an exercise out of scope for this handbook

The project team after gathering all this information and analyzing it, can make an revised overview of the processes, bottlenecks and **needs**.

Requirements Pyramid - Features

The next step for the project team is to introduce solutions that respond to the needs to the stakeholders. Again the most probable setting for this will be the workshop. Prior to the workshops the relevant documentation can be circulated: process maps, identified bottlenecks, needs and a preliminary very high level description of the system and its **features** (figure 1.2). Knowing and understanding the stakeholder needs and applying its in-depth knowledge of the envisaged solution and the enabling technology, the project team is able to create such high level description of the system to be built, in terms of its main characteristics.

When defining system features each must off course correspond to existing needs expressed by the stakeholder community, otherwise features do not make sense. Features when tabled for discussion during the workshop are either accepted or rejected, or revised and accepted. When accepted the list of features is well documented and circulated for review.

Project Scope

The list of accepted features to be implemented during the project is also called the *scope of the project*. In case the project is planned to have for example two system releases, then the features delivered in each system release are referred to as the *scope of a release*. Maintaining a clear overview of the project or release scope is very important since by communicating the scope to stakeholders, the project team can manage expectations i.e. the stakeholders know and understand and realize what system is arriving in their workplace and what it can (or cannot) do.

Requirements Pyramid - Requirements

Once the scope is determined, the next step is to describe each feature in terms of its requirements (figure 1.2). The difference between features and requirements is that features are the more general, high-level characteristics of a system; features are more one or two sentence descriptions of the product. Requirements are more detailed descriptions, outlining the conditions and limits of each of those characteristics.

Features, for example, would be mentioned by a (Deputy) Speaker of Parliament when he/she would proudly describe their parliamentary legislative information system or online petitioning system to a visiting Speaker of another Parliament. In our example of the e-PS project the envisaged system has the following features:

- e-PS allows citizens to have their petition live on the Internet, rather than just on paper. This way, a petition and supporting information can be made available to a potentially much wider audience, giving the opportunity to gather more names to support the petition.
- A petition may gather signatures in both forms - one can have a paper version and an online version, although repeat signatures will be removed.
- Each e-Petition also has its own discussion forum, where visitors and signatories can discuss the petition and surrounding issues online. There is also space for supporting information, so that one can add any background necessary and put ones petition in context.
- A petition can be addressed to a Committee or an MP (or MP office)
- Submitted petitions give a clear overview of the issue at hand and the number of votes
- The system guarantees privacy of personal information

The requirements would then describe what tasks the system would have to carry out in support of the *creation, assignment and submission of petitions*. The requirements would also describe who (user interacting with the system) would carry out this tasks, what are the pre- and post-conditions.

What vs. How

As a rule of thumb, requirements describe **what** function or task the system is required to do; requirements do not describe how the system executes its functions or task. The distinction between *what* and *how* is made in order to prevent writing the requirements too narrow and by doing so not allowing developers to choose how to build the system.

In chapter 6 we shall discuss an effective method of documenting requirements called use cases.

Managing the Requirements Pyramid: Traceability

As the project team works through each level of the requirements pyramid- by conducting iterative workshops to gather needs, propose features and discuss requirements – it will produce documentation. At the minimum at the end of each level there will be a needs, features and requirements document. Progressing through the requirements pyramid, features are born out of needs, and requirements are born out of features. Likewise there should also be linkages between the documents from each level. If the requirements pyramid is to be maintained it is important to manage your documentation and the linkages between them.

Needs, features and requirements, once distinguished and categorized will surely not remain static. In the course of the project stakeholder needs may change as external developments may take place. Needs may become more or less urgent; additional needs may emerge, or existing ones may become obsolete (!) Likewise the implementation of requirements during the built phase of the system may face technical limitations, i.e. requirements cannot be realized from a technical point of view. These dynamics taking place inside the requirements pyramid at each level could have an impact on the next or previous level. Therefore having an overview of the requirements pyramid and managing your documentation is a must.

For example if after the scope of the project is determined, and a new need is identified -by a stakeholder with lots of political backing etc - then the existing features need to be re-analyzed. If the existing features can serve this new need, the impact on the requirements pyramid is minimal. If, however, the existing features do not serve this new need, then new features have to be created to address this need. Ad-hoc workshops need to be organized; a complete iteration needs to be completed. Together with new features, new functional requirements need to be created. In addition the duplication or overlap of features and requirement must be prevented.

So the impact of one change at the top can be quite significant at the bottom of the pyramid. Sometimes, depending on the weight of these changes, the impact can extend into a change of the scope and/or planning of the project. This ad-hoc re-analyzing of needs, features and requirements throughout the different levels of the requirements pyramid is helped by traceability (figure 1.2).

In a well constructed and maintained pyramid the needs, features and requirements are linked i.e. a requirement must be linked to at least one feature, and a feature must be linked to at least one need. If there are requirements or features "loosely hovering" in the pyramid they will cause confusion and trouble.

And this is what traceability is all about: checking the linkages between needs, features and requirements, and determining whether there are non-connected items in the pyramid. It is a method for cleaning up the pyramid.

Requirements Management Tools

Project teams can use a variety of tools (software) that can assist them in managing their requirements. Simple tools such as a spreadsheet can be used to create an overview view of needs, features, requirements and the names of documents.

In addition there are requirements management tools both Open Source (free) and propriety which can support requirements management processes. In annex 5 more information is provided on these tools.

CHAPTER 4: THE SCOPE DOCUMENT- PUTTING IT ALL TOGETHER

The project scope document[1] (hereafter referred to as scope document) is a useful tool to capture summaries of all the requirements information that we have been discussing in chapters 2 and 3 of this handbook.

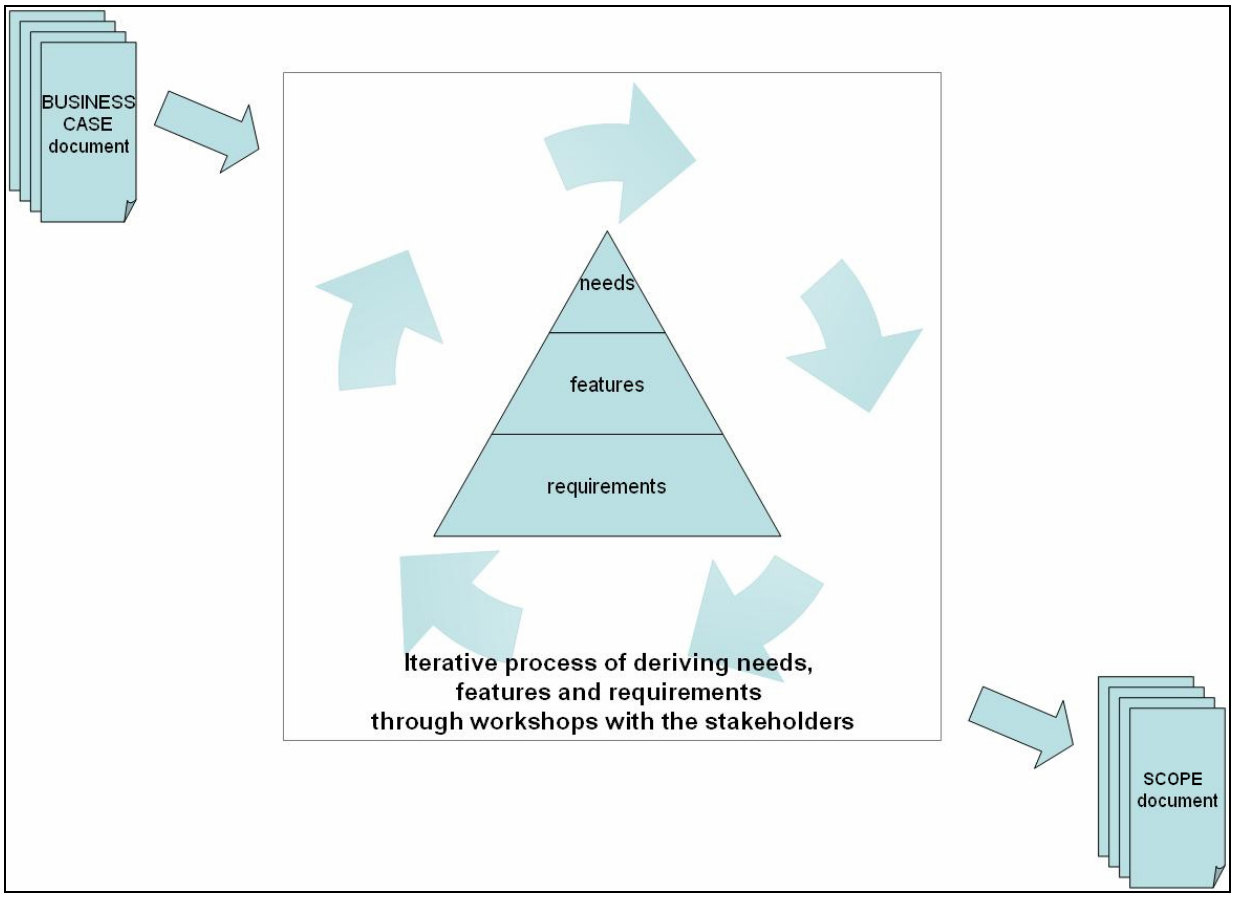


Figure 4.1

The scope document can be seen as the *grand result* of the whole requirements development cycle. It can be produced as a result of the process of deriving needs, features and requirements from the

[1] *The Project Scope document content-wise is equal to the "Vision Document" which is a deliverable of the* ***Open Unified Process*** *(OpenUP). The word "Vision" is not chosen for naming this document in this handbook to avoid confusion with the "Vision and Strategy" stage of the structured approach for integrating ICT in Parliament (figure 1.1).*

OpenUP is a part of the Eclipse Process Framework (EPF), which is an open source process framework developed by the Eclipse Foundation. It provides best practices from a variety of software development thought leaders and the broader software development community that cover a diverse set of perspectives and development needs.

stakeholders, through iterative workshops. It is not just a report of the whole cycle; it is also a document that reflects the contents of the requirements pyramid.

As with all requirements documentation, the scope document's primary purpose is communication. A scope document is written to give the reader an overall understanding of the system to be developed by providing a self-contained overview of the system and the motivation behind building it. To this end, it often contains extracts and summaries of other related artifacts, such as the business-case, needs, features and requirements documents. The purpose of the scope document is also to capture the focus, stakeholder needs, goals and objectives, target audience, user environments and target platforms.

It communicates the fundamental *"why's and what's"* related to the project, and it is a measure against which all future decisions should be validated. The scope document is the primary means of communication among the project's decision makers, IT management, stakeholder representatives and project team. It should be readable for all of the project's stakeholders, including non technical parties such as the (Deputy) Speaker of Parliament, funding authorities, donors and technical parties such as the head of IT, business analysts, use case writers, developers etc.

A scope document provides:

- A high-level (sometimes contractual) basis for the more detailed requirements
- Input to the project-approval process (and therefore it is intimately related to the business-case)
- A vehicle for eliciting initial stakeholders (users) feedback
- A means to establish the scope and priority of the product features

The points listed above imply that the scope document marks an important milestone in the progress of the project.

It is a document that gets "all parties working from the same book." When the scope is reviewed, agreed upon and signed off by the relevant stakeholders, in theory is means that all parties are in agreement to proceed with the next phases of the project such as requirements specification (through use cases), functional design and application development.

Because a scope document is used and reviewed by a wide variety of involved stakeholders, the level of detail must be general enough for everyone to understand. However it must also contain enough detail to provide the team with the information it needs to create a use case model and supplementary specification.

Contents

A typical scope document contains the following sections:

- **Positioning *(or project rationale)*:** This section summarizes the justification for the product and the problem or opportunity that the product is intended to address. Typically, the following areas should be addressed:

    - **The Business Opportunity:** A summary of the opportunity for supporting or improving the work of parliament that is being met by the product
    - **The Problem Statement:** A solution-neutral summary of the problem being solved focusing on the impact of the problem and the benefits required of any successful solution
    - **Product endorsement:** A summary of the forces that drive the product decisions i.e. who are the key decision makers and backers of the implementation of the product.
    - **User Environment:** The user environment where the product will be or could be applied.

- **Stakeholders and Users:** This section describes the stakeholders in, and users of, the product. The stakeholder roles and stakeholder types are defined in the project's scope document—the actual stakeholder representatives are identified as part of the project plan just like any other resources involved in the project.
- **Key Stakeholder and User Needs:** This section describes the key needs that the stakeholders and users perceive the product should address. It does not describe their specific requests or their specific requirements, because these are captured in a separate stakeholder requests artifact. Instead, it provides the background and justification for why the requirements are needed.
- **Product Overview:** This section provides a high-level view of the capabilities, assumptions, dependencies (including interfaces to other applications and system configurations), and alternatives to the development of the product.
- **Features:** This section lists the features of the product. Features are the high-level capabilities (services or qualities) of the system that are necessary to deliver benefits to the users and satisfy the stakeholder and user needs. This is the most important, and consequently usually the longest, section of the scope document.
- **Other Product Requirements:** This section lists any other high-level requirements that cannot readily be captured as product features. These include any constraints placed on the development of the product and any requirements the planned product places on its operating environment.

In many cases, a lot more work is put into uncovering the business opportunity and understanding the product endorsement related to the proposed product than is reflected in the scope document. One could argue that this work is already captured in-depth in business-case documentation. Summaries of the business case can be included in the scope document to ensure that it is reflected in the ongoing evolution of the products specification.

Project scope vs. requirements pyramid documentation

In relation to the requirements documents it is recommended that the scope document be treated primarily as a summary report and that the actual stakeholder types, user types, stakeholder roles, needs, features, and other product requirements reside in separate documents (figure 4.2).

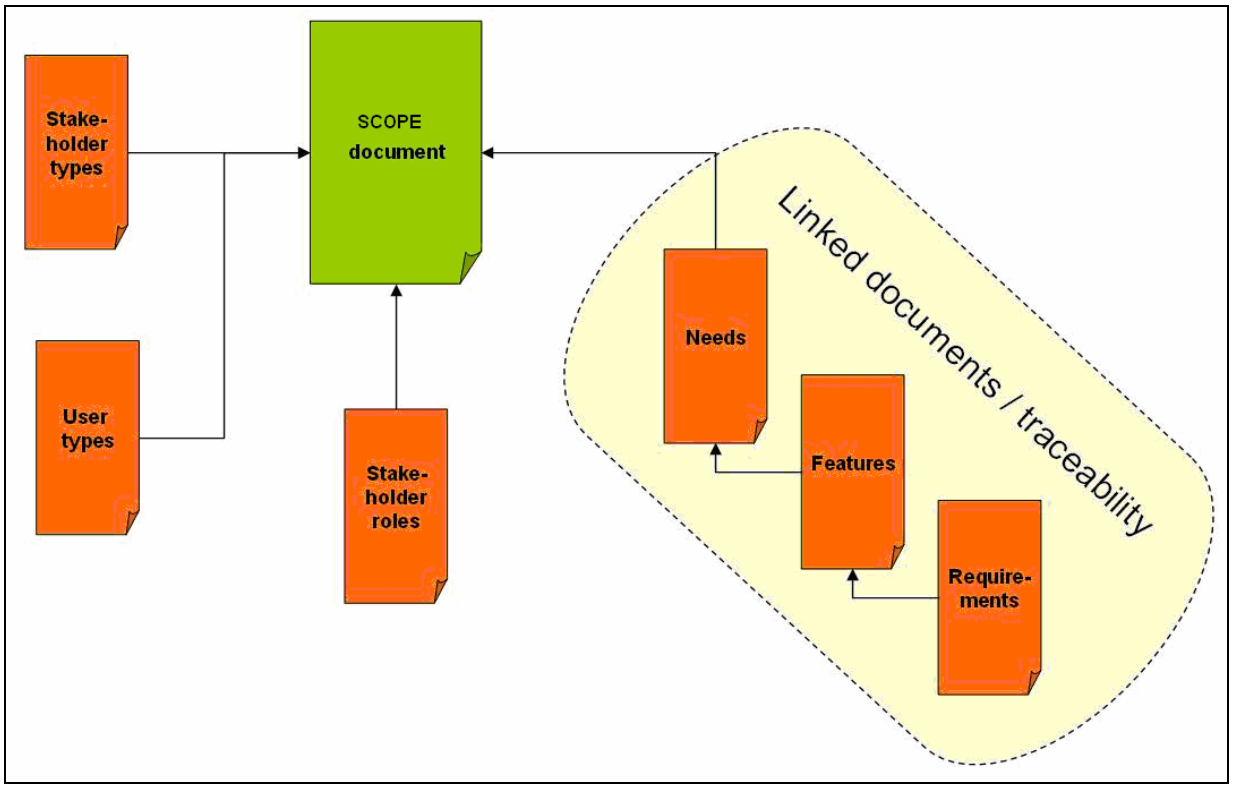


Figure 4.2

In figure 4.2 the linked documents – on the right - are those documents mirroring the requirements pyramid in between which traceability exists. The scope document should also provide insight into how the documents are linked i.e. traceable.

When presenting the list of features in the scope document it is recommended that they be presented in two sections "In-Scope features" and "Deferred features". As the scope document is mainly a communication document between stakeholders, it should be used by the project team to manage the expectations of what is to come.

Annex 1 contains a comprehensive scope document template that contains a full definition of the structure and contents of a typical scope document.

What is the benefit of all this work?

You are probably thinking that this all sounds like an awful lot of work, and you probably want to get started right away with the actual requirements specifications, use case modeling without producing all this additional overhead documentation. Projects are typically in one of the following four states when the requirements specifications (use case modeling) activities are scheduled to commence:

1. A formal scope document has been produced and communicated to all stakeholders.
2. The information has already been captured but not consolidated into a single scope document; the working documents have been circulated for review by the stakeholders.

3. There seems to be a shared project scope, but the working documents are still being consolidated and will be sent for review; there is no consolidated single scope document
4. There is no project scope.

If your project is in one of the first two states, and the information is available to all the stakeholder representatives, then you are in a position to proceed at full speed with the construction and completion of the use case model. If your project is in one of the last two states, then you should be very careful not to spend too much effort on the detailed requirements specifications, use case modeling activities. This does not mean that use case modeling cannot be started—it simply means that any modeling you do must be undertaken in conjunction with other activities aimed at establishing a documented project scope for the product. In fact, in many cases, undertaking some initial use case modeling can act as a driver and facilitation device for the construction of the project scope itself.

The recommendation would be to always produce a scope document for every project and to relate the information it contains to the use case model to provide focus, context, and direction to the use case modeling activities. Formally relating the two sets of information also provides excellent validation of their contents and quality. If there is sufficient domain knowledge and agreement between the stakeholder representatives, then producing and reviewing the scope document can be done very quickly. If there is little agreement, then there is no point in undertaking detailed use case modeling. The resulting specifications would be ultimately worthless as they would not be a reflection of the product's true requirements.

CHAPTER 5: APPLICATION OF THE REQUIREMENTS PYRAMID AND SCOPE DOCUMENT

In this chapter we will look at how following the requirements pyramid structure can guide us to complete the main sections of the scope document. While reading this chapter it is advisable to refer to annex 1 which contains a template of the scope document, outlining these main sections.

The application of the requirements pyramid and the scope document are demonstrated in the context of the e-PS project.

e-PS Project Team: first steps

During the first project board meeting further agreement is reached on how the board will operate and how it will oversee the e-PS project. Also the first deliverables of the e-PS project manager are discussed. These include:

1. Plan of action & budget planning
2. e-PS Project Team Formation
3. Scope & Release Planning

The latter two deliverables need further elaboration since they are strongly linked to the requirements pyramid:

e-PS project team

The e-PS implementation team shall consist of:

- 1 project manager: will lead the initial requirements gathering workshops together with the stakeholders of project and is responsible for producing the "Scope and release planning"
- 2 business analysts: will assist (prepare, organize, co-lead, report) the project manager during the requirements gathering workshops. The business analysts are responsible for documenting the "needs, features and requirements" of e-PS. They are responsible for the management of linkages between these documents according to the requirements pyramid.
- 1 systems architect: will carry out an assessment of the existing ICT infrastructure in Parliament and will provide an outline of technical constraints and opportunities. The systems architect will on/off be involved during the requirements gathering workshops, but will mostly sit in at the internal project team meetings, to technically guide the development of the features of the system.

Scope and Release Planning

After the first cycle requirements gathering workshops, the project manager, with input from his team, will make a proposed scope of the system, and divide this scope in releases. A release is a part of the complete system, containing a number of features and which can operate independently. The project manager will propose this scope and release planning to the project board for approval.

After the stakeholder representatives have been recruited and have joined the project, the goal is to arrive to a shared project scope of the system. To be effective this project scope must provide a

shared understanding of the problem that the system will solve, and should unify the various stakeholder perspectives. This will also help the project team to achieve a truly collaborative and cooperative working environment.

In the next sections we will look at how to:

A. Identify the underlying problem to be solved
B. Capture the stakeholders' needs
C. Describe features
D. Describe nonfunctional features
E. Provide a product position statement

A. Problem analysis

In Chapter 1 we discussed the importance of departing from the business case, as it describes the existing problem domain. Let us continue to look closer at the criteria of a problem statement and how it should be defined.

A problem can be defined as *"the difference between things perceived and things desired (by stakeholders)"*. This definition implies that there is more than one way to solve a problem. One can produce a solution; alternatively one can change the stakeholder's perception of what they experience now or change their perception of what they desire.

If you want to satisfy the stakeholder's real needs, you must first understand what problem they are trying to solve. When handing over a final system to stakeholders, you want to avoid hearing: *"Yes it meets the requirements listed on paper but it does not solve my problems."*

The best way to capture the problem is to construct a problem statement. This is a solution-neutral summary of the stakeholders' shared understanding of the problem to be solved. It includes an assessment of the problem's impact and the benefits to be gained by its solution. It can be captured using the scope document extract shown in Table 5.1.

The problem of	[describe the problem]
Affects	[the stakeholders affected by the problem]
The impact of which is	[what is the impact of the problem?]
A successful solution would	[list some key benefits of a successful solution]

Table 5.1 Problem statement template

The usefulness of the problem statement lies in its ability, as illustrated by Figure 5.1, to represent the tip of the requirements pyramid while summarizing the problem to be solved. Understanding the problem is the first step in understanding the requirements.

The stakeholders often describe the problem in terms of their own needs, but each need should reflect an aspect of the same underlying problem. All projects embarking on requirements analysis and subsequent use case modeling should take time to produce a problem statement.

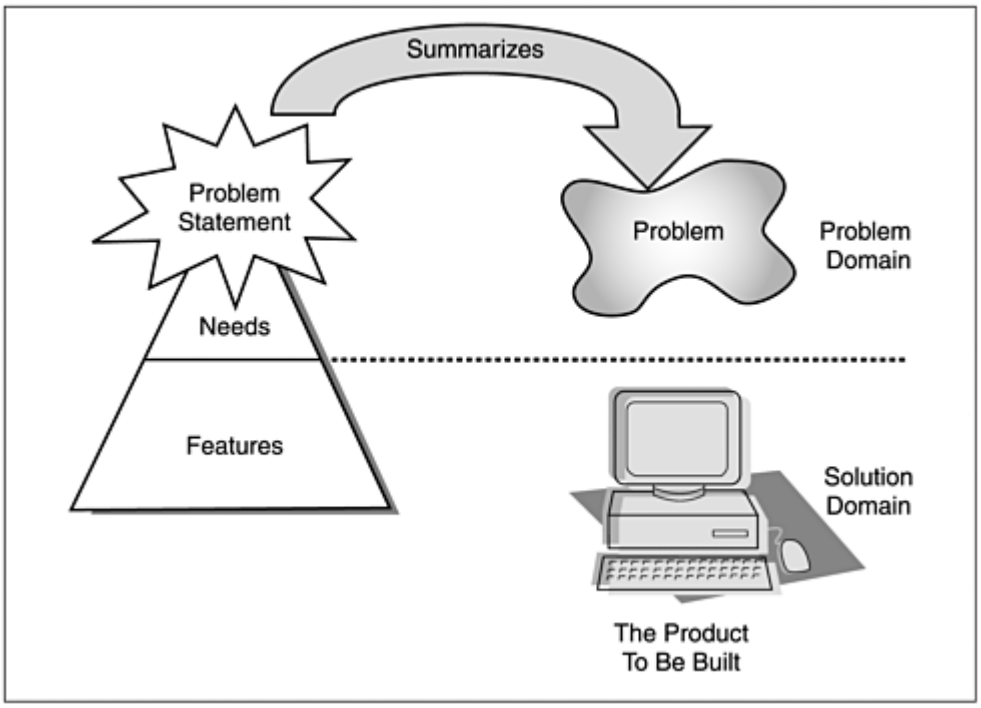


Figure 5.1

Often, the stakeholders have different perspectives of the same the problem, and it is very important that they reach agreement on a shared problem at some shared level of abstraction. If they cannot agree on a problem statement, then they are unlikely to agree on the scope or suitability of any proposed solution. Sometimes, achieving a shared definition of the problem can be very difficult, yet it is essential to understand why stakeholders want to do something new.

There are many ways to build up this understanding. During the requirements gathering workshops valuable tools can be used to perform some root-cause analysis such as: *fishbone diagrams* and then apply the *Pareto principle* to help in leveling the root causes (see annex 4 for templates). In addition the exercise of gathering stakeholders together in a workshop and trying to have them produce a shared problem statement also serves as a practical way for the project team to get a first-hand impression of the dynamics among stakeholders.

A (two-fold) problem statement for the e-PS project is given below in table 5.2:

The Problem Statement for e-PS	
The problem of	Lack of citizen participation in the legislative creation or amendment process
Affects	- Committees, MPs and their staff, in that they are not directly informed or do not get a confirmation of the issues most important to citizens - Citizens as they feel left out of the legislative creation or

	amendment process and gradually loose interest
The impact of which is	That on the one hand: - Committees, MPs and their staff are too occupied carrying out time consuming analyses of issues of importance to citizens and do not get a confirmation of their conclusions - And on the other citizens gradually loose interest in what is happening in parliament in general, and more specific what their representative is doing for them
A successful solution would	- Provide a way for citizens to communicate to their representatives (MP's) what they think are the most important issues and how legislation to address these issues could be created of amended. - Provide MP's and their staff with more reliable and direct information on what the most important issues are. In addition MP's would be able to receive valuable input for the creation and amendment of legislation.

B. Understand key stakeholders and their needs

Effectively solving any complex problem involves satisfying the needs of a diverse group of stakeholders. Typically, stakeholders will have different perspectives on the problem and different needs that must be addressed by the solution. These can be acknowledged and tracked by explicitly capturing and documenting the needs of each stakeholder type. We can define a stakeholder need as *"a reflection of the business, personal or operational problem (or opportunity) that must be addressed to justify consideration, purchase, or use of a new system".*

Capturing stakeholder needs allows us to understand how and to what extent the different aspects of the problem affect different types of stakeholders. This complements, and provides a deeper understanding of the shared problem statement. They will provide an insight into the root causes of the overall shared problem and define a set of solution-independent requirement statements that if met, will solve the underlying business[2] problem.

The description of each stakeholder need should include the reasons behind the need and clearly indicate why it is important to the affected stakeholders. The needs should be written in a solution-independent fashion and address the root causes of the problem only. It is important at this stage to not think about the solution: the product to be built or the technology involved.

In addition for each identified stakeholder need it is useful to understand:

- The priority of this specific need i.e the relative importance the stakeholders and users place on satisfying each need.

[2] *The word "business" in this context refers to the "parliamentary business" or "daily parliamentary work processes", as opposed to "business" in the commercial sense.*

- Which stakeholders perceive the need
- How this aspect of the problem is currently addressed. State the current situation. By specifying the current state you will better be able to understand the impact of the use cases you will write.
- What stakeholders would like to see i.e. a specification of the desired situation.

Below (table 5.3) is an example of some of the needs of the e-PS system (note that the needs are numbered for requirements management purposes)

e-PS project example		
Stakeholders	**Need Nr**	**Description**
Citizens	Need01	**Citizens' interactivity and participation:** Provide tools to allow citizens to participate in the legislative and oversight process by implementing functionalities like: - Petitition creation, Petition signature collection; - Web annotation; any documents (e.g. bill, committee reports, etc.); - Blog-like discussion; Feedback on any section/page of the website; - Poll and Surveys; etc.
Parliamentary staff, MP staff, MP, Committees	Need02	**Processing of petitions:** provide tools to allow Parliamentary front office staff, MP' staff to: - Process received petitions (remove duplicates and previously submitted petitions, reject spam-petitions) - Respond to petitions - Use petitions as input into the legislative creation/amendment process.
MP staff, MP, Committees	Need03	**Respond to petitions:** provide tools to allow MP' staff, MPs, Committee members to: - Respond to petitions - Use petitions as input into the legislative creation/amendment process.

Prioritizing stakeholder needs

During the requirements gathering workshops the project team will try to understand how to prioritize the different stakeholder needs. Naturally every stakeholder representative will claim that their needs are the most important. One way to achieve a justified prioritization of the stakeholder needs is to trace back to the original problem domain described in the business case.

Also gathering input from senior level stakeholders is valuable in this process. Cumulative voting techniques can also be used to identify the needs that must be solved as opposed to those that the

stakeholders wish to see being addressed. One method for ranking the stakeholders needs by priority in a transparent and easy to understand manner is by applying the so-called "MoSCoW rules". **M**o**SC**o**W** is derived from the first letters of the following prioritizing criteria:

- Must have (Mo)
- Should have (S)
- Could have (Co)
- Will not have (W)

Prioritization at this stage is important to determine the scope of the project and to plan the use case writing and application development stages.

C. Describing Product Features and Supplementary Requirements

It is very useful to create a high-level description of the system provided by listing the set of product features. Features are the high-level capabilities (services, characteristics or qualities) of the system that are necessary to deliver benefits to the users and that help to fulfill the stakeholder and user needs, and eventually solve or address their problems. The list of features provides a summary of the advertised benefits of the product to be built.

Figure 5.2 illustrates the relationship among the needs, the features, and the system to be built.

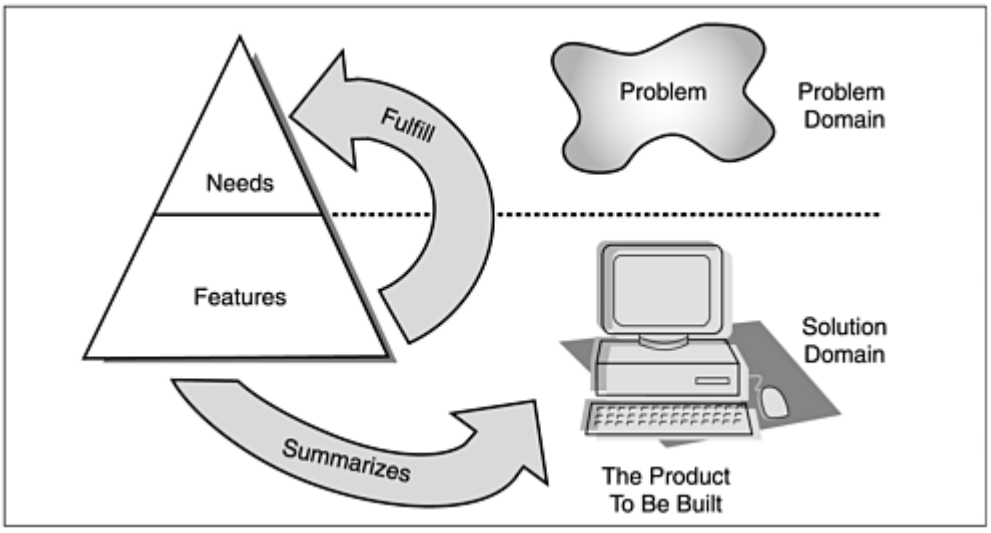


Figure 5.2

Features can be both functional[3] and nonfunctional. Many features describe externally accessible services that typically require a series of inputs from users (or interfacing systems) to achieve the desired result.

For example, a feature of a legislation-tracking system might be its ability to provide "status reports on legislative acts" indicating that the acts have the current status: drafted, discussed, passed or amended. This is typically an example of an externally visible quality of the system.

[3] When the word "features" is mentioned in this handbook, it is meant "functional features".

Another feature of the legislation-tracking system could be the "timely and accurate status reports". This is slightly more abstract and less visible externally.

Features are used to summarize the capabilities and qualities of the system that will be built; they must be accessible to all the members of the project team, including all the stakeholders. The level of detail must be general enough for everyone to understand them. And yet enough detail must be available to provide the use case writers with enough information to shape, validate, and manage the subsequent use cases and the supplementary specifications (more on use cases in chapter 6)

Features represent areas of the functionality of the system that are important to the users and other stakeholders of the system. Features do not provide a complete definition of the system. They represent the advertised benefits, the "hot aspects" of the system rather than a summary of its entire functionality. For this they are complemented by use cases which describe functional requirements.

The immediate and informal nature of features makes them a very powerful tool when working with the stakeholders in defining what they want from a system's releases. When asked, stakeholders will be able to quickly come up with a list of the top ten features they would like to see added to the system; in contrast, they will often struggle to identify any new use cases that are required. Features provide the fundamental basis for product definition and scope management.

Once the project team has an overview of all the features the next step is to look at when these features will be implemented. The implementation of an ICT solution, a set of system features, can be done in packages called a "release". A release is a fully working part/module of an ICT system that offers a number of features. The "scope of a release", by definition, is the list of features in the system which is released.

Documenting Features

When documenting features it is useful to:

- Include a description of functionality and any relevant usability issues that must be addressed.
- Avoid design. Keep feature descriptions at a general level. Focus on required capabilities and why (not how) they should be implemented.
- Assign each feature a unique identifier for easy reference and tracking (FEAT1, FEAT2)

The features of the system may be categorized and presented in many ways. For discussion with the stakeholders and verification, it is best to present the features by functional area and type. For scope management and publication purposes, it is best to group the features by target release, sorted in order of priority so that it is easy to distinguish between those that are in-scope and those that have been deferred. Again, as with the needs, we recommend the use of the MoSCoW rules to prioritize the feature set.

Table 5.5 (below) lists some of the features of e-PS with their priority rating. The column on the right shows the linkage to the needs, enabling traceability from features to needs.

e-PS project example: Features			
M	FEAT01	Parliament will have the possibility to allow registered members of the public to submit petitions and allow other members of the public to sign the petitions. It be possible to track the status of petitions on the portal.	NEED01
M	FEAT02	Petitions submitted will not be accessible before the designated staff of the parliament is satisfied that it complies with the specific regulations and requirements that each parliament will set.	
S	FEAT03	Parliament will have the possibility to have an online poll on any page of the site to solicit the opinion of the citizens and/or request them to fill in a survey if the information requested is more complex than a simple "yes/no" poll.	
M	FEAT04	Citizens can make comments and express their opinion on public forums.	
M	FEAT05	Parliamentary front-office staff can compare new petitions with previously submitted petitions by carrying out key-words match checks	NEED02
M	FEAT06	Parliamentary front-office staff can reject petitions and place the submitters (principal petitioner) on a SPAM filter list	
M	FEAT07	MP's (or staff) can formulate a response to a petition and post it online	NEED03
C	FEAT08	MP's (or staff) can communicate directly with the principal petitioner using e-mail	

D. Nonfunctional features: Constraints and Operational variables

Not all needs can be translated into functional features. Constraints placed on the development of the product, or needs regarding the future operating environments are not product features and should be documented separately from the features as nonfunctional features.

Constraints

No matter how technology independent the requirements-gathering and the software development processes are, some constraints are inevitable. Constraints are not related to fulfilling the stakeholders' needs; rather they are restrictions imposed on the project by external forces. For example the leadership of the Parliament may decide to only run Open Source software in Parliament. This certainly offers opportunity but at the same time limitations/constraints are imposed on the IT architecture of the Parliament. Although such constraints arise from the stakeholder community, they are not directly related to the problem to be solved. Figure 5.3 illustrates how the stakeholders' imposed constraints impact the project and system to be built.

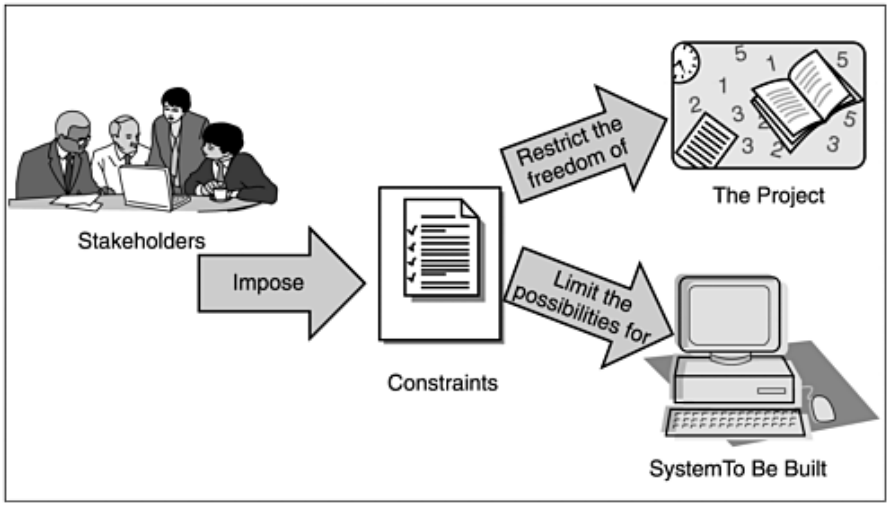


Figure 5.3

Many different kinds of constraint may be imposed on a project. These could include:

- **Business and Economic:** Cost and pricing, availability and licensing issues
- **Environmental:** External standards and regulations that are imposed on the development project
- **Technical:** The technologies that the project is forced to adopt or the processes that the project has to follow (such as a requirement that the system should be web-based)
- **System:** Compatibility with existing systems and operating environments (the system should be running on Open Source Linux servers instead of Windows OS)
- **Schedule and Resources:** Dates the project has been committed to or limitations on the resources that the project must use (e.g. the project must be completed before the end of the fiscal year etc)

Stakeholders may impose constraints for a variety of reasons:

- **Political:** Constraints may be placed on the project by the relationships among the stakeholders rather than the technical forces shaping the project.
- **Organizational Policies:** Organizational policies may be in place that constrain the way that the product can be developed. A parliament may have made a policy decision to move toward specific techniques, methodologies, standards, or languages.
- **Strategic Directions:** Strategic directions may be in place that constrains the project to use specific technologies and suppliers (e.g. open source software).
- **Organizational Culture:** The culture of the organization may itself constrain the project by limiting the way that the project must address the problem. (There is a limit to the amount of change that people can cope with at any one time, and this could prevent a project from adopting its preferred technologies and methods.)

E. Overview of the product

A list of features (functional and nonfunctional) is not sufficient to provide a complete overview of the system. One would also need to document the benefits, assumptions, dependencies (including interfaces to other systems), and alternatives to the development of the product.

Product Position Statement

Every system is built for at least one good reason. Like any good organization, the system requires a good "mission statement" or "statute". You should be able to state in clear terms what the system essentially does and why. This description need not to be long—in fact, the briefer the better; a short description that conveys the real value of the system. Most projects that find themselves in difficulties do so because, at least in part, no one really knows what is being built and for what reasons. The product position statement is an important vehicle for communicating a brief definition of the system to all stakeholders.

The scope document template extract shown below in Table 5.6 can be used to express the product positioning statement, a key element of the project scope. This format reminds people about all of the things that must be considered when establishing a project scope for the system. A description of the system is important because it gives everyone a common understanding of what the system does and for which reasons. Anyone associated with the project should be able to briefly describe what the system does in simple terms. Being able to do so creates a foundation for common understanding and further communication that pays dividends as the project progresses.

For	(target user)
Who	(statement of the need or opportunity)
The	(system name) is a (system category)
That	(statement of key benefit)
Unlike	(primary competitive alternative)
Our product	(statement of primary differentiation)

Let us consider product position statement for the e-PS project.

When asking the question "what does e-PS do", one might tend to start giving details by describing how the user logs-in and how bills are drafted and assigned to Clerks. These are important details, but they do not belong in the basic description of a product position statement.

When writing the product description, try to describe the system as you would to someone who is unfamiliar with it. Mention what problem it solves and what value it will deliver. Try to imagine speaking to one of the donors who plans to give funds to the project: What is the system going to do for MPs and Clerks? And what is the benefit to Parliament?

An example of the product position statement for e-PS is shown below (table 3.7).

For	Citzens and MP's/Committees
Who	Would like to participate in the legislative creation/amendment process respectively who would like to receive citizen feedback on the legislative process
The	e-PS is an online petitioning system for use between citizens and Parliament
That	Through web access will enable citizens to voice their opinion and provide feedback on the legislative process, and enable MP's and committees to communicate their position on certain issues to the public as well as include

	opinions and citizen input in the legislative process
Unlike	The first generation online petitioning systems that were build solely around gathering signatures on certain issues…
e-PS	In addition offers a platform where citizens can discuss these issues through online forums and discussion boards, which are also open to MP's.

This description is not fancy or over-complicated; it is simple, brief and it conveys the essence of what the system does and why.

It states what problem the system principally solves, who it principally serves, and what value it provides. If you cannot describe the system in very simple terms, you probably do not have a very clear idea of what the system will do. Note that this description does not try to capture even a fraction of the requirements, as it should not. Is it important to describe that one can print the legal documents with the system? Not at this point.

What about security? Not in the brief description. What about other kinds of drafting and editing actions that might be handled by the system? No need to describe them all here.

We merely want to capture the essence of what the system does so that everyone will be clear about it.

Completing the Product Overview

To provide a complete overview of the product, you may also need to summarize other aspects of the product not directly captured by the features. Typically, it is worth documenting:

- **Summary of Capabilities:** Summarize the capabilities that the system offers to its users. Presenting a brief overview of the use case model will summarize the functionality offered by the system.
- **User Benefits:** Summarize the benefits that the product offers to its users and which features provide the benefit. This may just be a matrix relating the stakeholder needs to the features.
- **Assumptions and Dependencies:** List any assumptions that have been made that if changed will alter the project scope established for the system. Also list any dependencies the product has on other products or the target environment.
- **Alternatives and Competition:** List any alternatives that the stakeholders perceive as available, including a description of their strengths and weaknesses, to allow comparison with the solution being proposed.

It is important in the scope document to provide the stakeholders with these additional perspectives on the product, so they demonstrate that the product is not being considered in isolation from its target business and operational environments.

It is not about writing a "scope document"

In this chapter we looked at what to write in the main sections of the scope document. Though this chapter was more focused on compiling a scope document according to a template (annex 1), writing the scope document is perhaps the easiest part.

Far more important and challenging are the discussions with all relevant stakeholders that will follow after the circulation of the scope document.

The aim of the scope document is to:

1. inform all parties about what kind of system will be implemented and why
2. and to facilitate the process of getting all parties to commit on working together from a agreed project scope

Following the review rounds and discussions the scope document will inevitably be revised until all parties are satisfied. After the final scope is signed-off by all parties the project can mark this important milestone.

PART 2: DEVELOPING USE CASES

CHAPTER 6: INTRODUCTION

In the first part of this handbook we focused on how to create a working environment with stakeholders, which in combination with requirements facilitation tools will produce the stakeholders' needs, system features and requirements. We also we looked at the role of the scope document which can be used to facilitate the process of establishing a shared project scope.

The requirements pyramid discussed in chapter 3 displayed the following three levels: Needs, Features and Requirements. In the previous chapters we discussed how needs can be elicited from the stakeholders and how subsequently features can be defined. We also discussed requirements in relation to needs and features i.e. what are the differences between needs vs. features vs. requirements. In this chapter we shall focus on use case[4] writing (or use case modeling) which is a very effective method of elaborating and documenting your requirements, according to a certain structure, and with the help of templates (annex 2). As use cases are introduced in this chapter it is important to note that besides use cases there are many other effective ways to capture requirements. The strength of use cases though is their ability to communicate details of the system to all stakeholders. Please see some positives and negatives of use cases in table 6.1

Postives
1. Employing Use case works well to include and involve non-technical people on the project, such as future users, early adopters of the system, project managers, project sponsors who are interested in knowing what the system, or parts of the system will do. 2. Experience with Use cases shows that usually 15 minutes of explanation is sufficient to allow non-technical stakeholders to start expressing requirements. Visualization of the user-system interaction with Use case diagrams keeps these stakeholders focused, involved and compartmentalized. 3. Breaking up the whole system into modular blocks of functionality, contained in use cases, allows for better project and scenario planning (i.e. what can users expect in the following months, what will change on the work floor and who will be involved). Keeping an overview of use cases with spreadsheets or requirements management tools supports the planning process.
Negatives
1. Employing use cases should not be seen as the "silver bullet". Gathering input for use cases, writing use cases and reviewing them takes time and is subject to error. 2. Use cases give a rough picture of the system to be implemented. It describes what the system-user interaction should contain in terms of information input and output. Description of how this interaction is realized is described in subsequent documentation: functional design. So when working with use cases still decisions must be made on a level of assumption and abstraction.

[4] *Use cases are deliverables of the **Open Unified Process** (OpenUP). OpenUP is a part of the Eclipse Process Framework (EPF), which is an open source process framework developed by the Eclipse Foundation. It provides best practices from a variety of software development thought leaders and the broader software development community that cover a diverse set of perspectives and development needs.*

What are Use Cases?

Use cases vs. requirements pyramid

In figure 6.1 we see that the requirements layer of the pyramid has been replaced with the use cases and supplementary specifications layer.

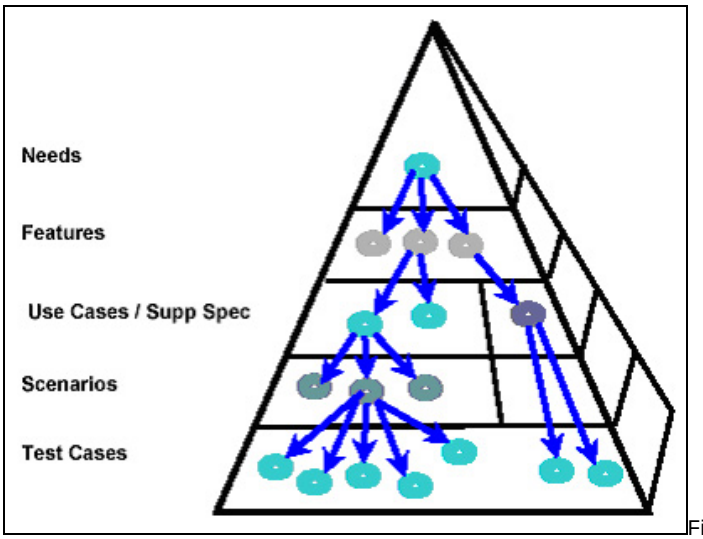


Figure 6.1

This is exactly how use cases must be seen; they are a notation of requirements. What we also see is that use cases only exist because of a parent-feature. No use case exists without a parent-feature, and no feature should exist without a parent-need.

Use cases modeling is the process of defining detailed descriptions of the higher level features. Use cases are written according to a specific format and describe scenarios between the actor (user) and the system. They describe the interaction in terms of input and output given and received by the actor.

Figure 6.2 shows a use case diagram, which is a graphical notation of use cases. On the left and right, the actors are shown in this case the Clerk and the MP. And in the center the system is depicted. The ovals are the use cases, in this case Submit a bill and Draft a bill.

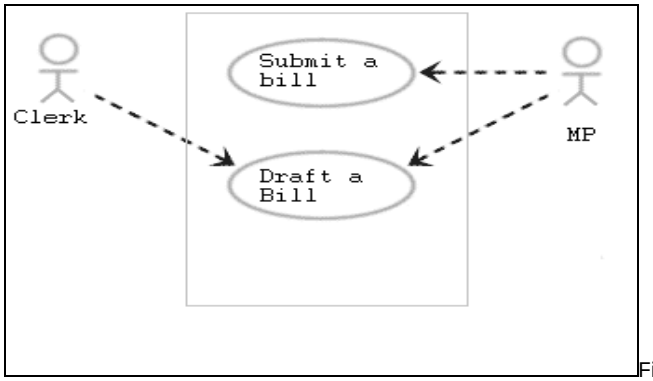


Figure 6.2

What are use cases - what are they not?

A use case describes what a system must do during the interaction with the actor. A use case describes the sequence of actions, the corresponding input into the system by the actor, and the subsequent output by the system. If a use case describes the interaction between the system and more than one actor (as in figure 5.1 *UC-Draft a Bill*, actors *Clerk* and *MP*) then the use case must explicitly distinguish between the different interactions for each actor.

A use case serves as input into the functional and technical design of the system, which is the work domain of application developers. A use case does specify what information is entered into the system. So, for example, the use case *Draft a Bill* must note that the *MP enters the date, the title, the bill number,* etc. It does not, however, mention where on the screen this data is entered; this is up to the designer initially to decide. A use case should not describe how the system looks like. It does not mention how many buttons there are on the screen or where the buttons are located.

A use case does describe the actions and functions which the system should be able to perform. A proper use cases would say: "the user saves the record" or the "the user deletes the record". A proper use case never says: "the user saves the record by clicking on the save button located at the bottom of the screen" or the "the user deletes the record by selecting the record and clicking on the delete button". Those are details to be determined by the developer together with the business analyst.

For whom are use cases written?

Use cases are mainly written for the following audiences in the project:

1. **Stakeholder representatives (user community):** use cases ideally should be read by stakeholders from the user community, and by experts of the parliamentary work-domain to verify that the scenarios described in the use case are in line with the agreed working processes in Parliament, e.g., the way a bill is drafted within the system must be in line with the adopted bill drafting-process and the use case *Draft a Bill* must reflect this accurately.
2. **Application Developers:** During the requirements gathering workshops application developers are seldom directly involved, except on those occasions when technical constraints of the system are discussed, and the development team lead is asked for his/her opinion. But apart from that, developers are not involved in the requirements gathering workshops. When developers are given the go-ahead to commence with the technical design of the system, they will need input. That input is provided by use cases. Use cases describe what the system must do, and given this input the technical design is created: "screenshots of a dummy application", technical parameters etc and subsequently a "prototype" is built (an initial, not fully working version of the application for demo purposes to stakeholders).
3. **Test team:** Parallel to the development team, the test team starts to prepare for the functional and system test phase by writing test-scripts. These are simple instructions on paper on how to test the system and verify that it was built as specified in the use cases. As soon as the development team releases an alpha version of the system, the test team can commence with the execution of tests.
4. **Training team:** As soon as the test team has produced test-scripts, the team of system trainers can re-use the test-scripts as input, together with use cases, to get a complete

understanding of how the system must be operated: how to carry out certain functions, which screens to navigate to and from, which buttons to click etc. Having all of this information at their disposal will help; the trainers write the user manual.

Figure 6.3 below shows the life-cycle of use cases and the respective audiences/readers. From this figure the significance and the impact of use cases on the delivery by other teams can be seen (bold arrow). Formally the process of use case writing starts after the scope document is signed off. After the stakeholders' and developers' sign-off the use cases are considered final. They then formally become input to the application development process, and are used to for test preparation purposes i.e. they support the test team in creating their first test documentation. Finally use cases are valuable input for writers of the user manual.

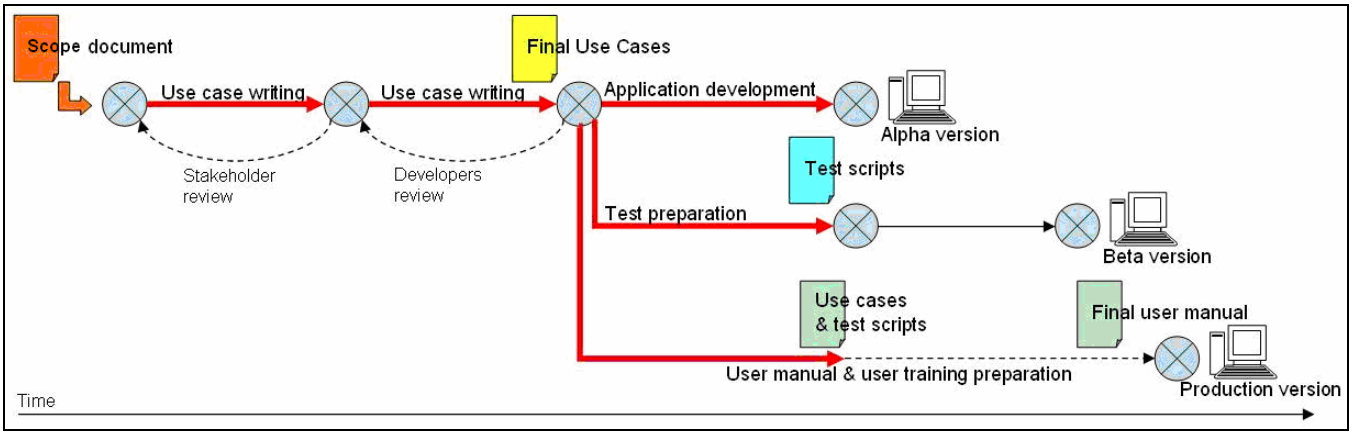


Figure 6.3

Who writes and reviews use cases?

In most projects the task of writing use cases is done by the business analysts or information analysts as they are able to bridge the gap between technology and the parliamentary work domain. Stakeholder representatives can be involved in the review of use cases. To facilitate an effective contribution, it helps to assign stakeholders with specific parliamentary work process knowledge to review those use cases linked to their area of expertise. Most probably these stakeholders were also involved in the requirements gathering process. It is important that they validate how their needs and requirements have been interpreted and that the use cases do describe the system with all preferred features and functions.

The structure of use cases

To begin with each use case has a reference written in the notation: UC<number><name>.

The name of a use case should reflect the name of the task to be carried out on the system. Some examples:

- *UC001-Login*
- *UC03-Draft a New Bill*
- *UC07-Edit an Existing Bill*

- *UC11- Publish a Bill Online*
- *UC14- Submit a Proposed Amendment*

Use case numbering is important for your requirements management. A use case may not exist by itself; every use case must be traceable to a feature (FEAT). In addition to its name and reference number a use case contains the following paragraphs:

1. Brief description
2. Flow of events
    - Basic flow
    - Alternative flow 1
    - Alternative flow 2
3. Special requirements
4. Preconditions
5. Post-conditions
6. Extension points
7. Context diagram
8. Activity diagram

Let us look in more detail at these paragraphs.

1. Brief description: Briefly conveys the role and purpose of the use case. A single paragraph will suffice for this description

2. Flow of events - Basic flow: The basic flow contains the most likely or frequent sequence of actions, the steps that happen when everything goes correctly. The use case starts when the actor does something. An actor always initiates use cases. The use case describes what the actor does and what the system does in response. It is phrased in the form of a dialog between the actor and the system.

3. Flow of events - Alternative flow 1: More complex alternatives are described in a separate section, referred to in the Basic Flow subsection of Flow of Events section. Think of the Alternative Flow subsections like alternative behavior— each alternative flow represents alternative behavior usually due to exceptions that occur in the main flow. They may be as long as necessary to describe the events associated with the alternative behavior. When an alternative flow ends, the events of the main flow of events are resumed unless otherwise stated.

4. Flow of events - Alternative flow 2: There may be, and most likely will be, a number of alternative flows in a use case. Keep each alternative flow separate to improve clarity. Using alternative flows improves the readability of the use case, as well as preventing use cases from being decomposed into hierarchies of use cases. Keep in mind that use cases are just textual descriptions, and their main purpose is to document the behavior of a system in a clear, concise, and understandable way.

5. Special requirements: A special requirement is typically a nonfunctional requirement that is specific to a use case, but is not easily or naturally stated in the text of the use case's event flow. Examples of special requirements include legal and regulatory requirements, application standards,

and quality attributes of the system to be built including usability, reliability, performance or supportability requirements. Additionally, other requirements—such as operating systems and environments, compatibility requirements, and design constraints—should be captured in this section.

6. Preconditions: A precondition of a use case is the state of the system that must be present prior to a use case being performed. For example most use cases (except for the use case "Login") will have as a precondition: "the user is logged onto the system". Another example is when a use case involves doing something with an existing record or file. The use case "Publish a Bill Online" will have as precondition: "a bill has been entered into the system". Without a "bill in the system" the use case would not be executable.

7. Post-conditions: A post-condition of a use case is a list of possible states the system can be in immediately after a use case has finished. For example in the case of "Publish a Bill Online" a post-condition could be "the system informs the user that the bill has been successfully published online" or "the system indicates that the bill is now online". But a post-condition could also be "the system indicates that the bills were not published". Every possible end state that could result from the flow of events must be included in the list of post-conditions.

8. Extension points: An extension point is where one use case connects to another use case. For example after a user logs into a system, he/she could then carry out other tasks. For example "check email" or "draft a new bill". In this case the use case "login" has an extension to the use cases "check email" and "draft a new bill".

Supplementary requirements specifications

Use cases can capture a great part of the requirements, in particular every requirement related to the interaction between the actor and the system, and the corresponding input and output. There will also be requirements which are not related to the actor-system interaction but are more general.

For example requirements such as "the system must be user-friendly and fast" cannot be captured in a use case. These are called supplementary specifications of a system. Supplementary Specifications capture the system requirements that are not readily captured in the use cases of the use case model.

Such requirements include:

- Legal and regulatory requirements, including application standards.
- Quality attributes of the system to be built, including usability, reliability, performance, and supportability requirements.
- Other requirements such as operating systems and environments, compatibility requirements, and design constraints.

These are verifiable / testable requirements. For example usability tests have the aim of knowing how easy-to-use a system is. Likewise performance and reliability can be tested: Does it take one

second to show the results of a query or does it take 10 seconds? Is the system stable or are there unexpected interruptions?

Annex 5 contains a template for capturing supplementary specifications.

Use Case Examples: Creating Petitions with e-PS

In this paragraph we will look at how use cases are written for the features of e-PS listed in the scope document. In table 5.5 below the most relevant features of the e-PS are listed:

FEAT01	Parliament will have the possibility to allow registered members of the public to submit petitions and allow other members of the public to sign the petitions. It is possible to track the status of petitions on the portal.
FEAT02	Petitions submitted will not be accessible before the designated staff of the parliament is satisfied that it complies with the specific regulations and requirements that each parliament will set.
FEAT05	Parliamentary front-office staff can compare new petitions with previously submitted petitions by carrying out key-words match checks
FEAT06	Parliamentary front-office staff can reject petitions and place the submitters (principal petitioner) on a SPAM filter list
FEAT07	MP's (or staff) can formulate a response to a petition and post it online

The proposed petitioning process or life-cycle with e-PS which has been discussed with stakeholders is as follows:

Step 1: Citizen creates petition: the citizen will be asked to give personal and contact information. The citizen will also be asked to write the petition text. This information will be used to give citizen petition a unique URL (website address) that the citizen can use to publicize their petition. The citizen will be able to specify the length of the petition up to 12 months.

Step 2: Submit citizen petition: the citizen selects an MP or Committee and once the petition is submitted, the citizen will receive an email asking citizen to click a link to confirm. The proposed petition will then be delivered to the chosen office of MP or Committee.

Step 3: Petition approval: MP staff or Parliament front office staff will check the proposed petition to make sure that it meets the basic requirements set out in our acceptable use policy and the civil service code. If for any a petition cannot be accepted, the citizen will be written to explaining the reasons why. Citizen will be able to edit and resubmit citizen petition if citizen wish. Once citizen petition is approved, the MP staff of Parliament front office sends an email to the citizen to confirm a date for it to appear on the website, usually within three working days. If the amended petition cannot be approved, staff will write again to the citizen explaining the reason(s). Any petitions that are rejected or not resubmitted will be published on this website, along with the reason(s) why it was rejected.

Any content that is offensive or illegal or clearly SPAM will be left out. If the petition is classified as SPAM the submitter could be put on a SPAM filter list.

Step 4: Petition live: Once the petition is live, the citizen will be able to publicize the URL citizen, and anyone will be able to come to the website and sign it by providing their personal and contact

information. After checking for duplication or invalid signatures the signer receives a confirmation email.

Step 5: Petition close: When a petition closes, usually provided there are 200 signatures or more, MP's or committee members will ensure citizen get a response to the issue that has been raised. The principal petitioner and everyone who has signed the petition will be given the details of the response.

Within the context of the process described above the following use case diagram can be created.

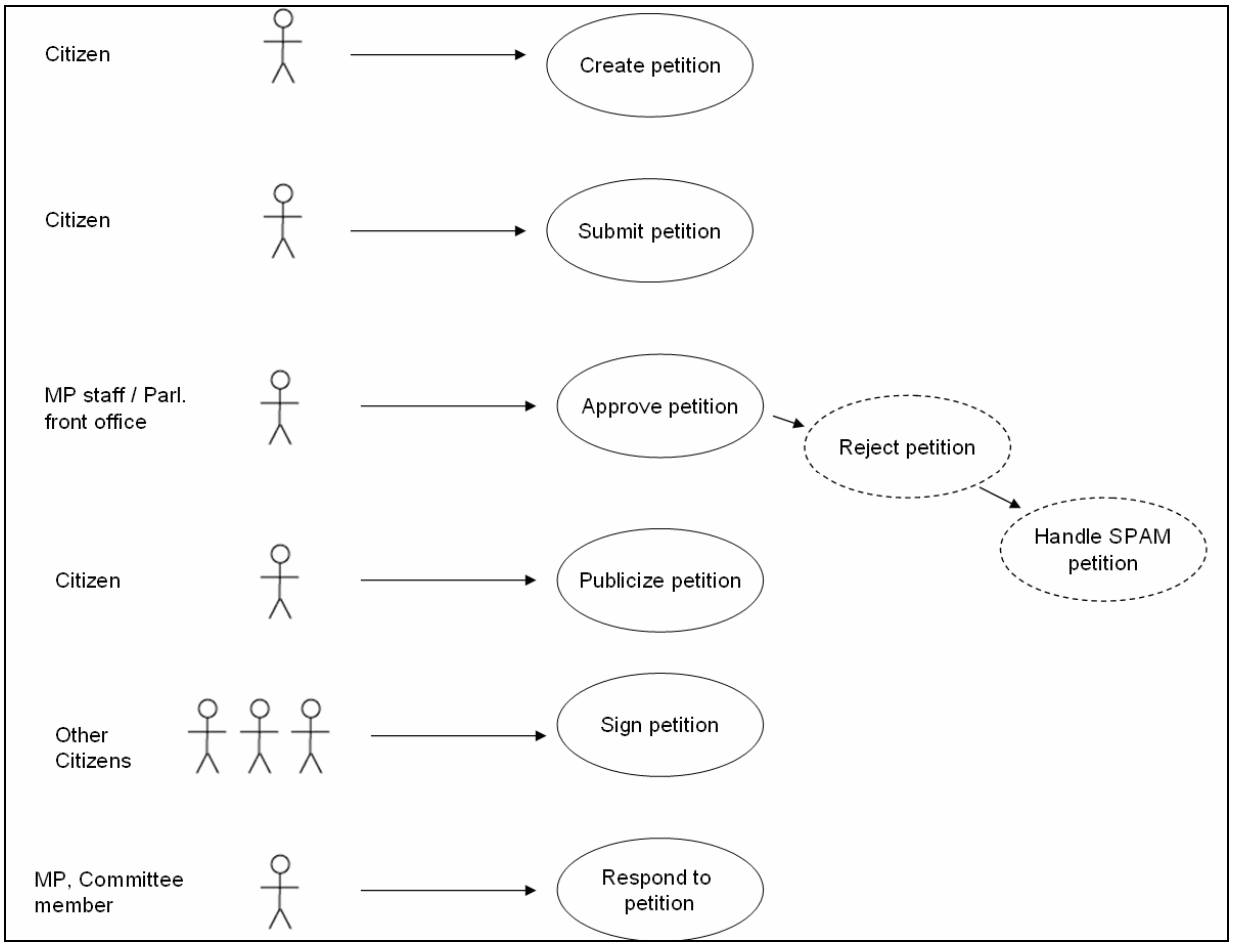


Figure 6.4

In figure 6.4 we can see that the petitioning process can be segmented into several independent actions:

- Create petition
- Submit petition
- Approve petition *(with extended actions reject petition and handle SPAM petitions)*
- Publicize petition
- Sign petition

- Respond to petition

These independent actions in petition process or lifecycle can be described by distinct use cases since they describe the specific interaction between actor and system. The use cases describe the requirements of what e-PS is supposed to offer to the citizens and MP/parliamentary staff, as well as what they are supposed to input into the system.

Thus use cases describe the *interface of e-PS* in terms of what information it offers, and the *interaction between the actor and the system*: i.e. when what type of information is entered into the system, and what information is given back by the system to the actor.

Let us now look in more detail how the following two use cases are written: *Create a Petition* and *Sign a Petition* are written. Keep in mind that use cases should be easily readable since they are intended for a wide audience varying from future users, application developers, senior project members and executive sponsors.

Use Case Name: Create a petition

Identifier: UC005

Description: this use case describes the creation and modification an online petition by the principal petitioner (PP)

Preconditions:

- The PP is registered and has a valid login account to enter e-PS
- The e-PS website is online

Flow of Events – Basic Flow:

- The PP logs into e-PS (see UC001.Login)
- The PP chooses the Create Petition function
- The PP enters personal details: name, organization (if applicable), address and email address)
- The PP enters the petition details: title, petition text, start and finish date of the petition; the PP also selects the addressee (MP or Committee)
- The PP saves the data

Alternative Flow A:

- While trying to save the petition the system indicates that this is not possible due to technical problems
- The Drafter is able to save the created document on the local workspace (desktop PC)

Post-conditions:

- The petition has been completely entered and stored in the petition database of e-PS
- The petition is partially entered and stored in the petition database of e-PS

- The petition is partially entered and not stored in the petition database of e-PS, but on a local workspace

Extension points:*

This use cases UC005.Create a Petition extends **from** UC001.Login and extends **to** the use cases UC006.Submit Petition and **to** UC020.Logout

**this simply means that UC005 can be performed after UC001, and after completion of UC005 the use cases UC006 and UC020 can be performed*

The example of UC005 above demonstrates how brief and easy-to-read use cases are. They are not complicated technical documents only to be understood by application developers or programmers; nor are they high-level business documents which often lack concreteness and details.

Use cases are the bridge between the main parties of the project i.e. those who have needs and requirements, and those who have to interpret those needs and requirements and produce a system accordingly.

Use cases serve the purpose that when read by application developers and programmers it gives them a framework of the minimum requirements the system needs to comply to. Having this framework in mind they can approach future users and other senior members of the project to further discuss and fill-in the functional details of the system. These discussions will produce functional designs of the system: for example the layout of the screens, how many and which buttons should be on it, and where they are ideally placed and what is the function of each function.

Vice versa, the purpose of use cases is for future users and senior level project members to maintain control over what will be delivered by the application development team 3-4 months down the road. It is the responsibility of that they (future users and senior level project members) review the use cases carefully and provide feedback and corrections.

Let us consider the other two use case examples for e-PS: *UC011.Sign a Petition*

Use Case Name: Sign a Petition

Identifier: UC011

Description: this use case describes how an ordinary citizen can sign a petition

Preconditions:

- A complete, approved petition is published online on e-PS
- The e-PS system is online

Flow of Events – Basic Flow:

- The Signer navigates to the e-PS website searches for a petition by listed categories and by entering free text keywords
- The Signer retrieves the petition in question and is able to view all its details: principal petitioner, title, text, period of the petition, how many signatures have been placed, the list of previous signers (if they have opted to be published on the petition)
- The Signer signs the petition by providing his/her personal information and contact details: name, address and email address
- e-PS confirms that the signature information has been submitted and that the signer will receive an e-mail confirming that the signing process is concluded.
- The Signer can navigate away from the petition in question.

Alternative Flow A:

- While trying to submit the signature information the system indicates that this is currently not possible due to technical problems

Post-conditions:

- The signer receives an email confirming whether the signing process is concluded

Extension points: N/A

In the latter example there are no extension points since for the act of signing no previous actions are needed, such as logging in or out.

Use Cases - Next Steps

Review

After use cases have been written by the business/information analysts it is time to have them reviewed:

- *Review – peer to peer:* it is always useful to let peers review your use cases. They could be from the same project of from other projects in Parliament. The focus with peer-to-peer review is to make sure that the use cases are written conform the guidelines i.e. that they are brief, concise, that they not cover functional design.
- *Review by stakeholders:* review by stakeholders is a test of the quality of requirements gathering during the workshops, of the note taking and documentation. However no use case is immediately perfect and subsequent additions, changes are inevitable. Besides questions that arise and discussions that follow are indicators of stakeholder interest in the requirements of the system.
- *Review by developers:* developers will be the main receivers of the signed off use cases as they serve as input for the functional design process (during which the number of buttons on screen are determined, why the buttons are left- and not right aligned etc). For them the use

cases must be sufficient as input documents to commence they design activities. All that is not clear in terms of the process, what information needs to be inputted and outputted should be clarified.

Managing the use case documentation: requirements baseline

As discussed in previous sections in the requirements pyramid it is necessary that the requirements (in this case manifested by use cases) be linked to features. As such it is also recommended to document the use cases.

By employing a simple spreadsheet, the use case numbers (UC<number>), or full use case names can be listed under the feature (FEAT<number>). It is also useful to provide for each use case some metadata such as what the status of review of the use case is, and to whom the use case is allocated.

Use cases are never really handed over to the system development teams. The development team receives a copy of the requirements baseline of a certain point in time. Requirements are owned by the requirements managers (business/information analysts assigned to that role.

As the project progresses the requirements documentation will become needed again. For example as the system and functional testing is prepared, the test team will need the updated version the requirements baseline that was issued to the development team.

As the training manual is being written and user training workshops are being prepared for the system roll out phase the updated requirements documentation needs to be available to the user-trainers. In the same way the requirements documentation becomes an important commodity for the future helpdesk and system administrators.

It does not stop at the current project: when the definition of follow-up projects is underway project managers and key stakeholders will want to see on paper what the current project is delivering.

The requirements documentation: project scope document and in more detail, the use cases shall give the answer.

SUMMARY

Understanding the stakeholder community is essential as the stakeholders are the primary source of requirements. The following are the key to understanding the stakeholder community:

- **Stakeholder Types:** Definitions of all of the different types of stakeholders affected by the project and the product(s) it produces.
- **User Types:** Definitions of characteristics and capabilities of the users of the system. The users are the people and things that will take on the roles defined by the actors in the use case model.

To establish a **shared project scope** for the project, the following are essential:

- **The Problem Statement:** A solution-neutral summary of the problem being solved, focusing on the impact of the problem and the benefits required of any successful solution.
- **Stakeholder Needs:** The true "business requirements" of the stakeholders presented in a solution-neutral manner. These are the aspects of the problem that affect the individual stakeholders.
- **Features, Constraints, and Other High-Level Product Requirements:** Collectively these provide a high-level definition of the system to be developed. They complement and provide a context for the use case model and enable effective scope management.
- **Product Overview:** A summary of the other aspects of the product not directly captured by the high-level requirements.

The scope document can be used to capture all of this information in a form that is accessible to all the stakeholders of the project. The project scope does not have to be complete before use case modeling activities start, but to a certain extend there should be general agreement over its main elements.

Undertaking some initial use case modeling can act as a driver and facilitation device for the construction of the project scope itself, but if the project scope is not established alongside the use case model, there is a strong possibility that it will not be a true reflection of the real requirements.

For **the use case modeling** activities to be successful, the stakeholders and users will need to be actively involved. The stakeholders and users directly involved in the project are known as stakeholder representatives.

To ensure that the stakeholder representatives understand their commitment to the project, it is worthwhile to clearly define the "stakeholder roles" that they will be adopting. The stakeholder roles serve as a contract between the stakeholder representatives and the project, reflecting the responsibilities and expectations of both sides.

ANNEX

1. Template: Scope document

<Project Name>

Project scope

Introduction

Positioning

Problem Statement

[Provide a statement summarizing the problem being solved by this project. The following format may be used:]

The problem of	[describe the problem]
Affects	[the stakeholders affected by the problem]
the impact of which is	[what is the impact of the problem?]
a successful solution would be	[list some key benefits of a successful solution]

Product Position Statement

[Provide an overall statement summarizing, at the highest level, the unique position the product intends to fill in the marketplace. The following format may be used:]

For	[target customer]
Who	[statement of the need or opportunity]
The (product name)	is a [product category]
That	[statement of key benefit; that is, the compelling reason to buy]
Unlike	[primary competitive alternative]
Our product	[statement of primary differentiation]

[A product position statement communicates the intent of the application and the importance of the project to all concerned personnel.]

Stakeholder Descriptions

Stakeholder Summary

Name	Description	Responsibilities
[Name the stakeholder type.]	[Briefly describe the stakeholder.]	[Summarize the stakeholder's key responsibilities with regard to the system being developed; that is, their interest as a stakeholder. For

			example, this stakeholder: - ensures that the system will be maintainable - ensures that there will be a market demand for the product's features - monitors the project's progress - approves funding and so forth]

User Environment

[Detail the working environment of the target user. Here are some suggestions:

- Number of people involved in completing the task? Is this changing?

- How long is a task cycle? Amount of time spent in each activity? Is this changing?

- Any unique environmental constraints: mobile, outdoors, in-flight, and so on?

- Which system platforms are in use today? Future platforms?

- What other applications are in use? Does your application need to integrate with them?

- This is where extracts from the Business Model could be included to outline the task and roles involved, and so on.]

Product Overview

Product Perspective

[This subsection of the Scope document puts the product in perspective to other related products and the user's environment. If the product is independent and totally self-contained, state it here. If the product is a component of a larger system, then this subsection needs to relate how these systems interact and needs to identify the relevant interfaces between the systems. One easy way to display the major components of the larger system, interconnections, and external interfaces is with a block diagram.]

Assumptions and Dependencies

[List each factor that affects the features stated in the Scope document. List assumptions that, if changed, will alter the Scope document. For example, an assumption may state that a specific operating system will be available for the hardware designated for the software product. If the operating system is not available, the Scope document will need to change.]

Needs and Features

[Avoid design. Keep feature descriptions at a general level. Focus on capabilities needed and why (not how) they should be implemented.]

Need	Priority	Features	Planned Release

Alternatives and Competition

[Identify alternatives the stakeholders perceive as available. These can include buying a particular commercial product, building a homegrown solution, or simply maintaining the status quo. List any known competitive choices that exist or may become available. Include the major strengths and weaknesses of each competitor as perceived by the stakeholders or end users.]

Other Product Requirements

[At a high level, list applicable standards, hardware, or platform requirements; performance requirements; and environmental requirements.

-Define the quality ranges for performance, robustness, fault tolerance, usability, and similar characteristics that are not captured in the Feature Set.

-Note any design constraints, external constraints, or other dependencies.

-Define any specific documentation requirements, including user manuals, online help, installation, labeling, and packaging requirements.

-Define the priority of these other product requirements. Include, if useful, attributes such as stability, benefit, effort, and risk.]

2. Template: Use Case Specification

Use case Specification: <Use case Name>

Brief Description

[The description briefly conveys the role and purpose of the use case. A single paragraph will suffice for this description.]

Flow of Events

Basic Flow

[This use case starts when the actor does something. An actor always initiates use cases. The use case describes what the actor does and what the system does in response. It is phrased in the form of a dialog between the actor and the system.

The use case describes what happens inside the system, but not how or why. If information is exchanged, be specific about what is passed back and forth. For example, it is not very illuminating to say that the actor enters customer information. It is better to say the actor enters the customer's name and address. A Glossary of Terms is often useful to keep the complexity of the use case manageable—you may want to define things like customer information there to keep the use case from drowning in details.

Simple alternatives may be presented within the text of the use case. If it only takes a few sentences to describe what happens when there is an alternative, do it directly within the Flow of Events section. If the alternative flow is more complex, use a separate section to describe it. For example, an Alternative Flow subsection explains how to describe more complex alternatives.

A picture is sometimes worth a thousand words, though there is no substitute for clean, clear prose. If it improves clarity, feel free to paste graphical depictions of user interfaces, process flows or other figures into the use case. If a flow chart is useful to present a complex decision process, by all means use it! Similarly for state-dependent behavior, a state-transition diagram often clarifies the behavior of a system better than pages upon pages of text. Use the right presentation medium for your problem, but be wary of using terminology, notations or figures that your audience may not understand. Remember that your purpose is to clarify, not obscure.]

Alternative Flows

< First Alternative Flow >

[More complex alternatives are described in a separate section, referred to in the Basic Flow subsection of Flow of Events section. Think of the Alternative Flow subsections like

alternative behavior— each alternative flow represents alternative behavior usually due to exceptions that occur in the main flow. They may be as long as necessary to describe the events associated with the alternative behavior. When an alternative flow ends, the events of the main flow of events are resumed unless otherwise stated.]

< An Alternative Sub flow >

[Alternative flows may, in turn, be divided into subsections if it improves clarity.]

< Second Alternative Flow >

[There may be, and most likely will be, a number of alternative flows in a use case. Keep each alternative flow separate to improve clarity. Using alternative flows improves the readability of the use case, as well as preventing use cases from being decomposed into hierarchies of use cases. Keep in mind that use cases are just textual descriptions, and their main purpose is to document the behavior of a system in a clear, concise, and understandable way.]

Special Requirements

[A special requirement is typically a nonfunctional requirement that is specific to a use case, but is not easily or naturally specified in the text of the use case's event flow. Examples of special requirements include legal and regulatory requirements, application standards, and quality attributes of the system to be built including usability, reliability, performance or supportability requirements. Additionally, other requirements—such as operating systems and environments, compatibility requirements, and design constraints—should be captured in this section.]

< First Special Requirement >

Preconditions

[A precondition of a use case is the state of the system that must be present prior to a use case being performed.]

< Precondition One >

Post conditions

[A post condition of a use case is a list of possible states the system can be in immediately after a use case has finished.]

< Post condition One >

Extension Points [Extension points of the use case.] <Name of Extension Point>

3. Template: Supplementary Requirements Specifications

<Project Name>

Supporting Requirements

Introduction

System wide Functional Requirements

[Statement of system-wide functional requirements, not expressed as use cases. Examples include auditing, authentication, printing, reporting]

System Qualities

[Qualities represent the Usability, Reliability, Performance, Supportability (URPS+) classification of supporting requirements].

Usability

[Describe requirements for qualities such as ease of use, ease of learning, usability standards and localization].

Reliability

[Reliability includes the product and/or system's ability to keep running under stress and adverse conditions. Specify requirements for reliability acceptance levels, and how they will be measured and evaluated. Suggested topics are availability, frequency of severity of failures and recoverability]

Performance

[The performance characteristics of the system should be outlined in this section. Examples are response time, throughput, capacity and startup or shutdown times.]

Supportability

[This section indicates any requirements that will enhance the supportability or maintainability of the system being built, including adaptability and upgrading, compatibility, configurability, scalability and requirements regarding system installation, level of support and maintenance.]

System Interfaces

[Interface Requirements part of the supporting requirements. Define the interfaces that must be supported by the application. It should contain adequate specificity, protocols,

ports and logical addresses, and so forth, so that the software can be developed and verified against the interface requirements.]

User Interfaces

[Describe the user interfaces that are to be implemented by the software. The intention of this section is to state requirements relating to the interface. Interface design may overlap the requirements gathering process.]

Look & Feel

[A description of the spirit of the interface. Your client may have given you particular demands such as style, colors to be used, degree of interaction and so on. This section captures the requirements for the interface rather than the design for the interface.]

Layout and Navigation Requirements

[Requirements on major screen areas and how they should be grouped together]

Consistency

[Consistency in the user interface enables users to predict what will happen. This section states requirements on the use of mechanisms to be employed in the user interface. This applies both within the system, and with other systems and can be applied at different levels: navigation controls, screen areas sizes and shapes, placements for entering / presenting data, terminology]

User Personalization & Customization Requirements

[Requirements on content that should automatically displayed to users or available based on user attributes. Sometimes users allowed to customize the content displayed or to personalize displayed content]

Interfaces to External Systems or Devices

[Are there any external systems with which this system must interface? Are there any constraints on the nature of the interface between this system and any external system, such as the format of data passed between these systems, and any particular protocol used? Consider both provided and required interfaces.]

Software Interfaces

[This section describes software interfaces to other components of the software system. These may be purchased components, components reused from another application or components being developed for subsystems outside of the scope of this SRS, but with which this software application must interact.]

Hardware Interfaces

[This section defines any hardware interfaces that are to be supported by the software, including logical structure, physical addresses, expected behavior, and so on.]

Communications Interfaces

[Describe any communications interfaces to other systems or devices such as local area networks, remote serial devices, and so on.]

[Business rules are statements that define or constrain some aspect of the business. Business rules are often represented as production rules when they are meant to be directly executed by an ICT System: a production rule is an independent statement of programming logic that specifies the execution of one or more actions in the case that its conditions are satisfied. Production Rules define the operation semantic for the system in a technologic independent way. They constrain the behavior expressed in system use cases.

Organize this document on rule classes, a high level grouping of candidate or actual rules about one business concept with a specific kind of logic processing , example: Driver Risk Assessment Rules or Customer Validation Rules]

Business Rules

<Rule name and ID>

[The description defines the rule. It can be made in natural language typically following a decision table or a pattern like: if [condition-list] then [action-list], example:

"If a drafted bill document that has been entered is not yet signed off, it cannot be submitted for online publication"

System Constraints

[Constraints are part of supporting requirements. Describe any design; implementation or deployment constraints on the system being built that have been mandated and must be adhered to. Examples include software implementation languages, prescribed use of developmental tools, third-party components or class libraries, platform support, resource limits and requirements on the shape, size or weight of the resulting hardware housing the system.]

System Compliance

Licensing Requirements

[Define any licensing enforcement requirements or other usage restriction requirements that are to be exhibited by the software.]

Legal, Copyright, and Other Notices

[This section describes any necessary legal disclaimers, warranties, copyright notices, patent notice, word mark, trademark, or logo compliance issues for the software.]

Applicable Standards

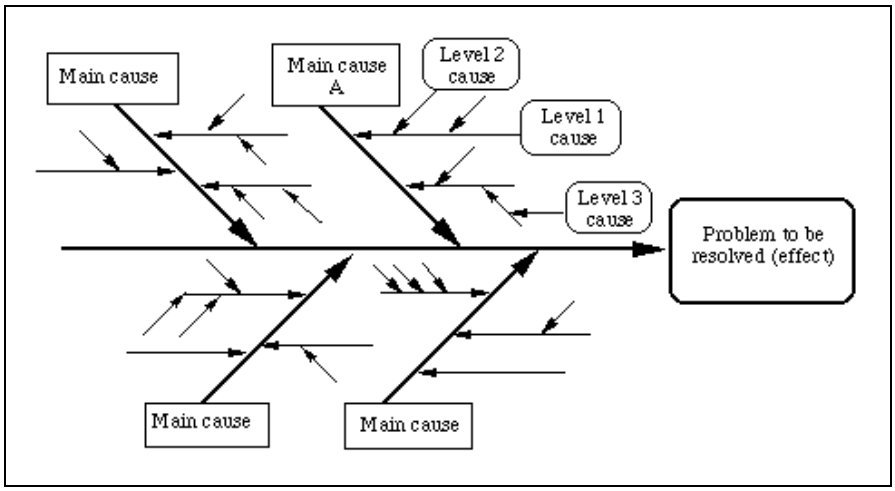
[This section describes by reference any applicable standards and the specific sections of any such standards that apply to the system being described. For example, this could include legal, quality and regulatory standards, industry standards for usability, interoperability, internationalization, operating system compliance, and so forth.]

System Documentation

[Describes the requirements, for on-line user documentation, help systems, help about notices, and so on. Set expectations for the documentation and to identify who will be responsible for creating it.

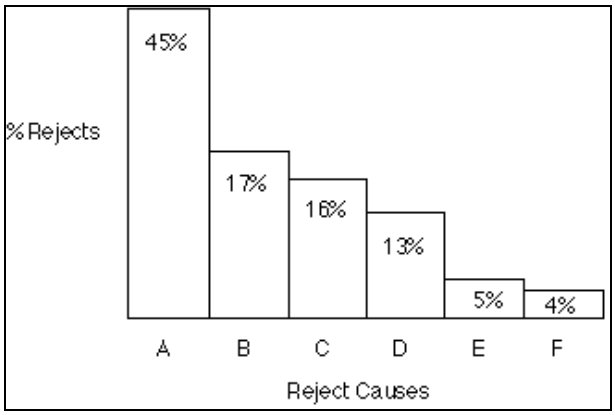
4. Problem Analysis Tools

Fishbone Diagram (Cause and Effect Diagram)



The cause-and-effect diagram is a method for analyzing process dispersion. The diagram's purpose is to relate causes and effects. Effect = problem to be resolved, opportunity to be grasped, result to be achieved. This is an excellent tool for capturing team brainstorming output and for filling in from the 'wide picture'. It helps to organize and relate factors, providing a sequential view. It deals with time direction but not quantity.

Pareto Principle



The Pareto principle suggests that most effects come from relatively few causes. In quantitative terms: 80% of the problems come from 20% of the causes (machines, raw materials, operators etc.); 80% of the wealth is owned by 20% of the people etc. Therefore effort aimed at the right 20% can solve 80% of the problems. Double (back to back) Pareto charts can be used to compare 'before and after' situations to decide where to apply initial effort for maximum effect.

5. Requirements Management Tools

Below is a list of requirements management software. Develop requirements from your project or enterprise needs and thoroughly investigate candidate software. Performance demonstrations are critical: additional software packages may be required to achieve full traceability to other work products. Tests and design are part of the software product and should be maintained along with the code and documentation.

Some RM tools cover the entire development environment, and they may dictate how you approach system development. The features and capabilities of any tool you are considering should be tested to ensure it works with your development approach.

Requirements Management Software

Accept 360	Active!Focus	AnalystPro
Caliber-RM	Catalyze	CORE
Cradle	DOORS & DOORSrequireIT	Enterprise Architect
GatherSpace	GMARC	IRqA
Leap SE	Lighthouse RM	Mac A&D and Win A&D
MKS Requirements	objectiF	Open Source RM
Optimal Trace	PACE	Profesy
Projectricity	Qualica QFD	Rally
RaQuest	Reconcile	Reqtify
Requirements Mgmt Database	Requirement Tracing System	Requisite Pro
RMTrak	RTM Workshop	SoftREQ
SpeedReq	Teamcenter	Tiger PRO (free)
TopTeam Analyst	Tracer (free)	TRUEreq (free)
WIBNI	XTie-Requirements Tracer	

Source: International Council on Systems Engineering (INCOSE) that publishes a comparison of features of many requirements management tools, which INCOSE updates periodically. The site also has a good discussion of the minimum capabilities of RM tools.

Website: http://www.incose.org/practice/techactivities/wg/rqmts/#

6. Glossary

Actor: The user(s) of the system. There can be several user types or categories. Actors are external entities (people or other systems) who interact with the system to achieve a desired goal.

Application Developer: A member of the project team who, by creating technical designs and writing software (programming), is responsible for delivering technical modules and/or components for the system.

Business Analyst (or Information Analyst): A member of a project team who is responsible for mapping the stakeholder needs, developing requirements, writing use cases and managing all requirements documentation. A business analyst does not necessarily need to have a technical background; he/she must have thorough knowledge and understanding of the "business at hand" i.e. daily work processes within a particular organization.

Functional Design: Functional designs (or functional design documents) describe how the interface of the system should look like such as the screen-layout, buttons and their functions. Functional designs usually created after use cases. Functional designs are written in the application development phase of the project.

ICT system implementation project life cycle: The phases of an ICT project. A typical ICT-SI project life cycle is divided into the following stages that produce the following deliverables:

Stage	Deliverables
Problem analysis/business case development	→ Business case document
Requirements development	→ Requirements documentation: Scope document, Use Cases
System development	→ Non-tested release 1.0 of the system (alpha version)
Testing (system and functional)	→ Tested release 1.0 of the system (beta version)
Users training	→ User manual and trained users
System roll-out	→ Operational system, users, system management

Open Unified Process (OpenUP): is a part of the Eclipse Process Framework (EPF), an Open Source process framework developed by the Eclipse Foundation. It provides best practices from a variety of software development thought leaders and the broader software development community that cover a diverse set of perspectives and development needs. OpenUP is based on *UML language* (see below). For more information: http://www.eclipse.org/

Process mapping: (or business process mapping) refers to activities involved in defining exactly what a business or organizational entity does, who is responsible, to what standard a process should be completed and how the success of a process can be determined. For increased clarity a process illustration (or process map) is produced. It is the first step in gaining control over an organization by knowing and understanding the basic processes

Requirements: A set of conditions that a system must comply with upon its completion. If a completed system complies with all the requirements, the system development phase of the project can be labeled as successful. Requirements are used as input during the testing phase to measure the accuracy, completeness, and quality of the system development.

Requirements development: this is second stage of the system implementation project life cycle. During this stage needs, features are gradually produced, by gathering requirements information from stakeholders and further analyzing and managing them.

- **Requirements gathering:** The process (supervised by the project team lead) of meeting with all project stakeholders in order to learn, understand and document their needs and expectations for the system. Requirements gathering can take place during "requirements gathering workshops" and in individual interviews

- **Requirements analysis and management:** After requirements have been gathered from the respective stakeholders, the project team analyzes what has been said by the stakeholders. It tries to organize the raw information by using a requirements pyramid to distinguish between needs, features and detailed requirements. Requirements management refers to the ongoing discipline of consistently documenting the requirements through the hierarchy of needs, features and detailed requirements, which enables "tracing the linkages" between these levels.

Unified Modeling Language (UML): Is widely recognized and used modeling language for specifying, visualizing, constructing, and documenting the artifacts of a system. It simplifies the complex process of software design, creating a blueprint for construction. UML was developed in 1994 by Dr. James Rumbaugh, Ivar Jacobson and Grady Booch.

Use Cases: A technique used in software and systems engineering to capture the functional requirements of a system. Use Cases originated as part of the UML, but are frequently used outside the UML context as an effective tool to capture system requirements. Use Cases are often co-authored by Business Analysts and system end-users.

7. Resources

References

G. Booch, J. Rumbaugh, and I. Jacobson, I. "The Unified Modeling Language User Guide", 1999, Addison-Wesley.

L. Constantine, "The Case for Essential Use Cases," Object Magazine, May 1997, SIGS Publications.

I. Jacobson, M. Christerson, P. Jonsson, G. Overgard "Object-Oriented software engineering: A use case driven approach" Addison-Wesley 1992

I. Jacobson, M. Griss, P. Jonsson "Software Reuse: Architecture, Process and Organization for Business Success" Addison-Wesley 1997

e-PS Case research done at:

- Africa i-Parliament Action Plan (www.bungeni.org)
- Scottish Parliament (www.scottish.parliament.uk/business/parliamentaryProcedure)
- 10 Downing Street – Petitions (http://petitions.pm.gov.uk/)

Images & graphs resources:

Figures 2.1 – 5.3: D. Leffingwell, D. Widrig "Managing Software Requirements: A Unified Approach" Addison Wesley Professional 1999

Figures 6.1: P. Zielczynski "Traceability from Use Cases to Test Cases", IBM -Developers Work, 2006

Annex 5. Requirements Management Tools, INCOSE (http://www.incose.org/practice/techactivities/wg/rqmts/#)

Annex 6. Problem Analysis Tools: Institute for Manufacturing, University of Cambridge (http://www.ifm.eng.cam.ac.uk/dstools/represent/tqm.html)

Templates resources:

Annex 1-3: Courtesy of Mr. Per.Kroll, Project Lead, Eclipse Process Framework (http://www.eclipse.org)