Requirements Analysis

Salvatore Filippone, PhD

School of Aerospace, Transport and Manufacturing
salvatore.filippone@cranfield.ac.uk

Outline

Introduction

Requirements analysis

The process of establishing the services that the customer requires from a system and the constraints under which it operates and is developed.

The requirements themselves are the descriptions of the system services and constraints that are generated during the requirements engineering process.

Introduction

What is a Requirement?

- It may range from a high-level abstract statement of a service or of a system constraint to a detailed mathematical functional specification
- This is inevitable as requirements may serve a dual function
  - May be the basis for a bid for a contract — therefore must be open to interpretation;
  - May be the basis for the contract itself — therefore must be defined in detail;
  - Both these statements may be called requirement

Requirements Abstraction

"If a company wishes to let a contract for a large software development project .." (Davis)

- It must define its needs in a sufficiently abstract way that a solution is not predefined;
- The requirements must be written so that several contractors can bid for the contract, offering (perhaps) different ways of meeting the client needs;
- Once a contract has been awarded, the contractor must write a *system definition* for the client in more detail so that the client can understand and validate what the software will do;
- Both of these documents may be called the *requirements document* for the system.

Types of Requirements

User requirements Statements in natural language plus diagrams of the services the system provides and its operational constraints. Written for customers.

System requirements A structured document setting out detailed descriptions of the system's function, services and operational constraints. Defines what should be implemented, may be part of a contract between client and contractor.

Types of Requirements

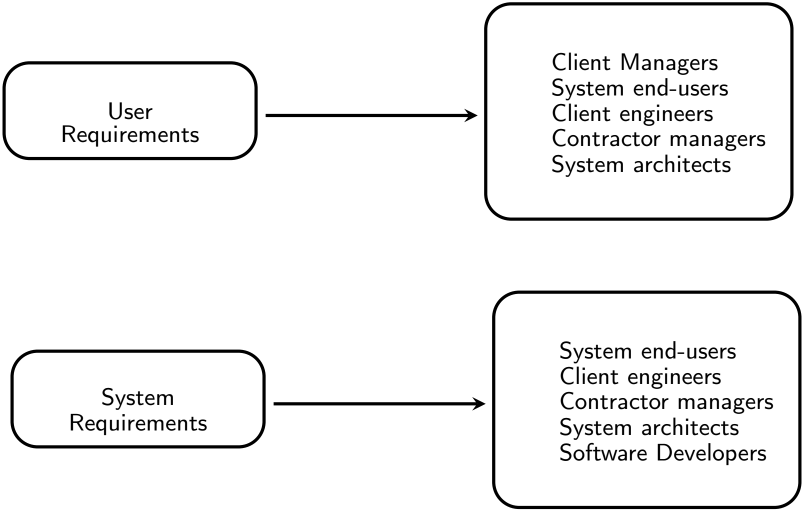
User Requirement Definition

1. The MHC-PMS shall generate monthly management reports on cost of drugs prescribed by each clinic during that month.

System requirements specification

1. On the last working day of each month, a summary of the drugs prescribed, their cost and the prescribing clinics shall be generated;
2. The system shall automatically generate the report for printing after 17:30 on the last working day of the month;
3. A report shall be created for each clinic and shall list the individual drug names, the total number of prescriptions, the number of doses prescribed and the total cost of the prescribed drugs;
4. If drugs are available in different dose units (e.g. 10mg, 20mg, etc) separate reports shall be created for each dose unit;
5. Access to all cost reports shall be restricted to authorised users specified in a management access control list.

Types of Requirements



Functional and Non-functional Requirements

Functional requirements: Statements of services the system should provide, how the system should react to particular inputs and how it should behave in specific situations. May state what the system should not do.

Non-functional requirements: Constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc. Often apply to the system as a whole rather than individual features or services.

Domain requirements: Constraints on the system from the domain of operation

Functional Requirements

- Describe functionality or system services.
- Depend on the type of software, expected users and the type of system where the software is used.
- Functional user requirements may be high-level statements of what the system should do.
- Functional system requirements should describe the system services in detail.

Example: Functional Requirements for the MHC-PMS

- A user shall be able to search the appointments lists for all clinics;
- The system shall generate each day, for each clinic, a list of patients who are expected to attend appointments that day;
- Each staff member using the system shall be uniquely identified by his or her 8-digit employee number.

Requirements Imprecision

Problems arise when requirements are not precisely stated; in fact, ambiguous requirements may be interpreted in different ways by developers and users.

Consider the term "search"

User interpretation: search for a patient name across all appointments in all clinics;

Developer interpretation: search for a patient name in an individual clinic. User chooses clinic then search.

Completeness and Consistency

In principle, requirements should be both complete and consistent:

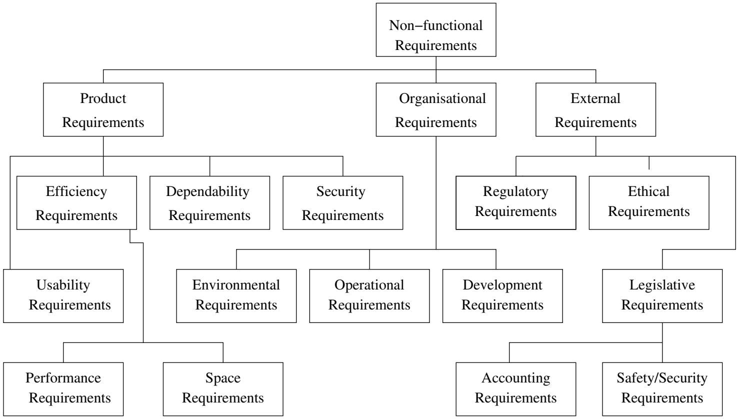
Complete:	They should include descriptions of all facilities required.
Consistent:	There should be no conflicts or contradictions in the descriptions of the system facilities.

In practice, it is essentially impossible to produce a requirements document that is both complete and consistent.

Non-functional Requirements

- These define system properties and constraints e.g. reliability, response time and storage requirements. Constraints include I/O device capability, system representations, etc.
- Process requirements may also be specified mandating a particular IDE, programming language or development method;
- Non-functional requirements may be more criticala than functional requirements. If these are not met, the system may be useless.

Types of Non-functional Requirements



Non-functional Requirements

Implementation

- Non-functional requirements may affect the overall architecture of a system rather than the individual components.
$\Rightarrow$ For example, to ensure that performance requirements are met, you may have to organize the system to minimize communications between components.
- A single non-functional requirement, such as a security requirement, may generate a number of related functional requirements that define system services that are required.
$\Rightarrow$ It may also generate requirements that restrict existinng requirements

Non-functional Requirements Classification

Product requirements: Requirements which specify that the delivered product must behave in a particular way e.g. execution speed, reliability, etc.

Organisational requirements: Requirements which are a consequence of organisational policies and procedures e.g. process standards used, implementation requirements, etc.

External requirements: Requirements which arise from factors which are external to the system and its development process e.g. interoperability requirements, legislative requirements, etc.

Example

Non-functional Requirements in the MHC-PMS

Product requirement The MHC-PMS shall be available to all clinics during normal working hours (Mon–Fri, 08:30–17:30). Downtime within normal working hours shall not exceed five seconds in any one day;

Organizational requirement: Users of the MHC-PMS system shall authenticate themselves using their health authority identity card;

External requirement: The system shall implement patient privacy provisions as set out in HStan-03-2006-priv.

Goals and Requirements

Non-functional requirements may be very difficult to state precisely and imprecise requirements may be difficult to verify;

Goal: A general intention of the user such as ease of use.

Verifiable non-functional requirement: A statement using some measure that can be objectively tested.

Goals are helpful to developers as they convey the intentions of the system users.

Usability Requirements

Goal

The system should be easy to use by medical staff and should be organized in such a way that user errors are minimized.

- Medical staff shall be able to use all the system functions after four hours of training;
- After training, the average number of errors made by experienced users shall not exceed two per hour of system use.

Testable, non-functional requirements

Metrics

Property	**Measure**
Speed	Processed transactions/second User/event response time Screen refresh time
Size	Mbytes; Number of ROM chips
Ease of use	Training time; Number of help frames

Reliability	Mean time to failure;
	Probability of unavailability;
	Rate of failure occurrence;
	Availability;

Robustness	Time to restart after failure; Percentage of events causing failure; Probability of data corruption on failure;
Portability	Percentage of target dependent statements; Number of target systems

Domain Requirements

The system's operational domain imposes addidional requirements

example

A train control system has to take into account the braking characteristics in different weather conditions.

- Domain requirements may be new functional requirements, constraints on existing requirements or define specific computations.
- If domain requirements are not satisfied, the system may be unworkable!!

Domain Requirements Issues

Understandability:

- Requirements are expressed in the language of the application domain;
- This is often not understood by software engineers developing the system.

Implicitness

- Domain specialists understand the area so well that they do not think of making the domain requirements explicit.

Key Points

- Requirements for a software system set out what the system should do and define constraints on its operation and implementation;
- Functional requirements are statements of the services that the system must provide, or descriptions of how some compuations must be carried out;
- Non-functional requirements often constrain the system being developed and the development process used;
- They often relate to the emergent properties of the system and therefore apply to the system as a whole.

The Software Requirements Document

The Software Requirements Document

The software requirements document is the official statement of what is required of the system developers.

- It should include both a definition of user requirements and a specification of the system requirements;
- It is NOT a design document. As far as possible, it should set WHAT the system should do rather than HOW it should do it.

Agile Methods and Requirements

- Many agile methods argue that producing a requirements document is a waste of time as requirements change so quickly;
- The document is therefore always out of date;
- Methods such as XP use incremental requirements engineering and express requirements as “user stories”;
- This is practical for business systems but problematic for systems that require a lot of pre-delivery analysis (e.g. critical systems) or systems developed by multiple teams.

Users of a Requirements Document

Customers	Specify the requirements and read them to check that they meet their needs, Customers specify changes to the requirements.
Managers	Use the requirements document to plan a bid for the system and to plan the system development process.
System Engineers	Use the requirements to understand what system is to be developed.
System Test Engineers	Use the requirements to develop validation tests for the system.
System maintenance Engineers	Use the requirements to understand the system and the relationships between its parts.

Requirements Document Variability

- Information in requirements document depends on type of system and the approach to development used;
- Systems developed incrementally will, typically, have less detail in the requirements document;
- Requirements document standards have been designed, e.g. IEEE standard. These are mostly applicable to the requirements for large systems engineering projects.

The Structure of a Requirements Document

Chapter	Description
Preface	This should define the expected readership of the document and describe its version history, including a rationale for the creation of a new version and a summary of changes made in each version.
Introduction	This should describe the need for the system. It should briefly describe the system's functions and explain how it will work with other systems. It should also describe how the system fits into the overall business or strategic objectives of the organization commissioning the software.
Glossary	This should define the technical terms used in the document. You should not make assumptions about the experience or expertise of the reader.
User requirements definition	Here, you describe the services provided for the user. The non-functional system requirements should also be described in this section. The description may use natural language, diagrams, or other notations that are understandable to customers. Product and process standards that must be followed should be specified.
System architecture	This chapter should present a high-level overview of the anticipated system architecture, showing the distribution of functions across system modules. Architectural components that are reused should be highlighted.

The Structure of a Requirements Document

Chapter	Description
System requirements specification	This should describe the functional and nonfunctional requirents in more detail. Interfaces to other systems may be defined.
System models	This might include graphical system models showing the relationships between the system components and the system and its environment. Examples of possible models are are object models, data-flow models, or semantic data models.
System evolution	This should describe the fundamental assumptions on which the system is based and any anticipated changes due to hardware evolution, changing user needs and so on. This section is useful for system designers as it may help them avoid design decision that would constrain likely future changes to the system.
Appendices	These should provide detailed, specific information that is related to the application being developed; for example, hardware and database descriptions. Hardware requirements define the minimal and optimal configurations for the system. Database requirements define the logical organization of the data used relationships between data.
Index	Several indexes to the document may be included. As well as a normal alphabetical index, there may be an index of diagrams, an index of functions and so on

Requirements Specification

- The process of writing down the user and system requirements in a requirements document;
- User requirements have to be understandable by end-users and customers who do not have a technical background;
- System requirements are more detailed requirements and may include more technical information;
- The requirements may be part of a contract for the system development
  $\Rightarrow$ It is therefore important that these are as complete as possible!

Writing a System Requirements Specification

Notation	Description
Natural language	The requirements are written using numbered sentences in natural language. Each sentence should express one requirement.
Structured natural language	The requirements are written in natural language on a standard form or template. Each field provides information about an aspect of the requirements.
Design description languages	This approach uses a language like a programming language, but with more abstract features to specify the requirements by defining an operational model of the system. This approach is now rarely used although it can be useful for interface specifications.
Graphical notations	Graphical models, supplemented by text annotations, are used to define the functional requirements for the system; UML use case and sequence diagrams are commonly used examples.
Mathematical specifications	These notations are based on mathematical concepts such as finite-state machines or sets. Although these specifications can greatly reduce ambiguity in a requirements document, most customers don't understand them, hence they cannot check that they represent what they want and are reluctant to accept them as a system contract

Requirements and Design

- In principle, requirements should state what the system should do and the design should describe how it does this.
- In practice, requirements and design are inseparable
    - A system architecture may be designed to structure the design requirements;
    - The system may inter-operate with other systems that generate design requirements;
    - The use of a specific architecture to satisfy non-functional requirements may be a domain requirement;
    - This may be the consequence of a regulatory requirement.

Natural Language Specification

- Requirements are written as natural language sentences supplemented by diagrams and tables.
- Used for writing requirements because it is expressive, intuitive and universal. This means that the requirements can be (easily) understood by users and customers.

Guidelines for Writing Requirements

- Invent a standard format and use it for all requirements;
- Use language in a consistent way. Use shall for mandatory requirements, should for desirable requirements;
- Use text highlighting to identify key parts of the requirement;
- Avoid the use of computer jargon;
- Include an explanation (rationale) of why a requirement is necessary;

Problems with Natural Language

- Lack of clarity
  $\Rightarrow$ Precision is difficult without making the document difficult to read.
- Requirements confusion
  $\Rightarrow$ Functional and non-functional requirements tend to be mixed-up.
- Requirements amalgamation
  $\Rightarrow$ Several different requirements may be expressed together.

Example

Requirements for the Insulin Pump Software System

- The system shall measure the blood sugar and deliver insuline, if required, every 10 minutes. (Changes in blood sugar are relatively slow so more frequent measurement is unnecessary; less frequent measurement could lead to unnecessarily high sugar levels).
- The system shall run a self-test routine every minute with the conditions to be tested and the associated actions (A self-test routine can discover hardware and software problems and alert the user to the fact the normal operation may be impossible)

Structured Specifications

- An approach to writing requirements where the freedom of the requirements writer is limited and requirements are written in a standard way;
- This works well for some types of requirements e.g. requirements for embedded control system, but is sometimes too rigid for writing business system requirements.

Form-based Specifications

- Definition of the function or entity;
- Description of inputs and where they come from;
- Description of outputs and where they go to;
- Details about the information needed for the computation and other entities used;
- Description of the action to be taken;
- Pre and post conditions (if appropriate);
- The side effects (if any) of the function.

Structured Specification for an Insulin Pump

Insulin Pump/Control Software/SRS/

Function	Compute insulin dose: safe sugar level.
Description	Computes the dose of insulin to be delivered when the current measured sugar level is in the safe zone between 3 and 7 units.
Inputs	Current sugar reading (r_2); the previous two readings (r_1 and r_0).
Source	Current sugar reading from sensor. Other readings from memory.
Outputs	CompDose — the dose of insulin to be delivered.
Destination	Main control loop.

Structured Specification for an Insulin Pump

Action	CompDose is zero if the sugar level is stable or falling or if the level is increasing but the rate of increase is decreasing. If the level is increasing and the rate of increase is increasing, then CompDose is computed by dividing the difference between the current sugar level and the previous level by 4 and rounding the result. If the result is rounded to zero then CompDose is set to the minimum dose that can be delivered.
Requirements	Two previous readings so that the rate of change of sugar level can be computed;
Pre-condition	The insulin reservoir contains at least the maximum allowed single dose of insulin;
Post-condition	r_0 is replaced by r_1 then r_1 is replaced by r_2.
Side effects	None.

Tabular Specification

- Used to supplement natural language.
- Particularly useful when you have to define a number of possible alternative courses of action.
- For example, the insulin pump systems bases its computations on the rate of change of blood sugar level and the tabular specification explains how to calculate the insulin requirement for different scenarios.

Tabular Specification of Computation

Condition	Action
Sugar level falling $(r_2 < r_1)$	$CompDose = 0$
Sugar level stable $(r_2 = r_1)$	$CompDose = 0$
Sugar level increasing and rate of increase decreasing $((r_2 - r_1) < (r_1 - r_0))$	$CompDose = 0$
Sugar level increasing and rate of increase stable or increasing $((r_2 - r_1) \geq (r_1 - r_0))$	$CompDose = \mathrm{round}((r_2 - r_1)/4)$. If rounded result =0, $CompDose = MinimumDose$

Requirement Engineering Processes

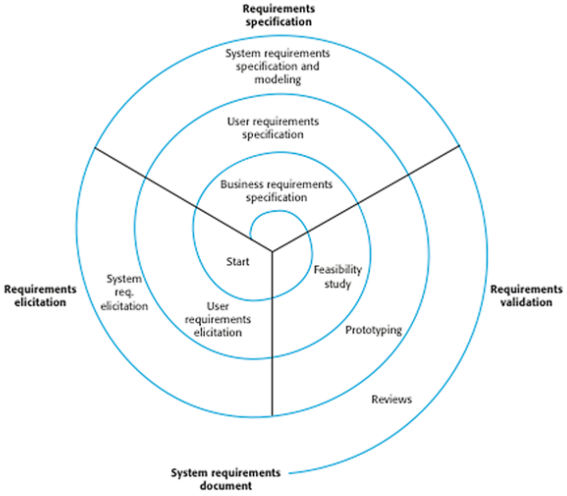
Requirement Engineering Processes

The processes used for RE vary widely depending on the application domain, the people involved and the organisation developing the requirements.
However, there are a number of generic activities common to al processes:

- Feasibility analysis
- Requirement elicitation;
- Requirement analysis;
- Requirement validation;
- Requirement management.

In practice, RE is an iterative activity in which these processes are interleaved.

A Spiral View


Requirements specification
System requirements specification and modeling
User requirements specification
Business requirements specification
Start
Feasibility study
Requirements elicitation
System req. elicitation
User requirements elicitation
Prototyping
Requirements validation
Reviews
System requirements document

Project System Request

We must first identify the project and its aims:

- Many projects fail or do not get approval because they are poorly specified;
- You must answer some fundamental questions regarding the reasons for proposing the project;
- This applies to both technical projects and business/enterprise oriented projects

The Project System Request provides a structured appr describing the proposal

- Generally raised by the Project Sponsor;
- Format will change according to organisation;
- First step in the requirements analysis process.

Project System Request

Some questions you should provide answers to:

Need: Why does the project need to be proposed in the first place ?
$\Rightarrow$ Increased sales? Improved market share?

Requirements: What extra capabilities does the project bring to the organisation ?
$\Rightarrow$ Improved application performance, better user support?

Value: What financial value will the project deliver?
$\Rightarrow$ Your unit of currency may vary from money to conference papers to journal papers. However, you should have a clear metric by which to understand the benefit of the proposal!

Special Issues: What constraints need to be factored into the request?
$\Rightarrow$ Deadlines, security, platforms

Project System Request – example

The following is a system request for an imaginary company that sells CDs and DVDs through a chain of music shops

- The system request proposes selling products through the Internet as well as the physical stores;
- It shows how each of the sections in the previous slides might be implemented
- Note that some initial estimates have been made regarding figures. This should be best guesses at this point but need to be validated and realistic

Project System Request - example

Requirement Process

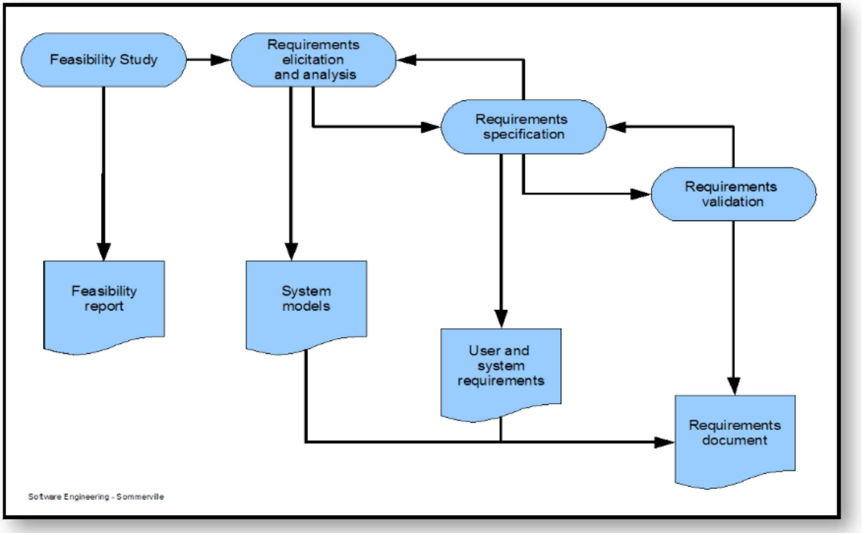
There are four main phases to the requirements analysis

- Feasibility study
- Requirements elicitation and analysis
- Requirements specification
- Requirements validation

In a traditional development process, these phases are completed sequentially:

- Also, they are completed only once
- In an iterative process, requirements may be revisited regularly;
- Changes to requirements are a source of delay and difficulties in projects;
- This is why Agile process focus on dealing with change in requirements!

Process Steps


Feasibility Study
Requirements elicitation and analysis
Requirements specification
Requirements validation
Feasibility report
System models
User and system requirements
Requirements document
Software Engineering - Sommerville

First Phase: Feasibility

Once we have decided we would like to implement the system, we must be sure that it is possible to do so

- We should understand the opportunities and limitations of the proposal;
- We should understand the specific risks associated with the project.

Feasibility studies usually occur at the start of the project

- However, we will often revisit feasibility as we progress through the project;
- If an unforeseen issue arises, it might be necessary to cancel the oject before more resources are assigned to it;
- In an iterative approach, we might perform a risk analysis at each cycle to assess feasibility.

First Phase: Feasibility

Typical questions we need to answer in this phase:

- Can the requirements be met with the current technologies?
- Are the expected costs likely to be acceptable?
- Can the project be realised within an acceptable time frame?

If the feasibility study is positive, a more detailed study

Feasibility : Three Key Areas

1. Technical feasibility — can we build it?
   - Application area
   - Technology
   - Scalability — large scale projects and infrastructure
2. Economic feasibility — can we afford to build it?
   - Cost-Benefit Analysis
   - Development costs
   - Operating costs
   - Intangibles
3. Organisational feasibility — does it match our organisation's goals/resources?
   - Is there a user base?
   - Are there leaders within the organisation?
   - Is this something the business should be doing?

Feasibility — Costs and Benefits

Development Costs

- Development Team Salaries
- Consultant Fees
- Development Training
- Hardware & Software
- Vendor Installation
- Office Space and Equipment
- Data Conversion Costs

Operational Costs

- Software Upgrades
- Software Licensing Fees
- Hardware Repairs
- Hardware Upgrades
- Operational Team Salaries
- Communications Charges
- User Training

Tangible Benefits

- Increased Sales
- Reduction in Staff Numbers
- Reductions in Inventory
- Reductions in IT Costs
- Better Supplier Prices

(In|Less) tangible Benefits

- Increased Market Share
- Increased Brand Recognition
- Higher Quality Metrics
- Improved Customer Service
- Better Supplier Relationships

Feasibility

Important economic metrics

- Return on Investment: Revenues generated from the investment
  (Total Benefits - Total Costs) / Total Costs
- Break Even Point: The point at which the return is eequal to or greater than the investment
  (Yearly Net Cash Flow - Cumulative Net Cash Flow)/Yearly Net Cash Flow

Note that these figures by themselves do not take into account variations in the value of cash, such as inflation, interest rates or alternative forms of investment

- Present Value:
  Amount / $(1 + \text{interest rate})^n$ (n=number of years)
- Net Present Value:
  PV Benefits - PV Costs

Feasibility

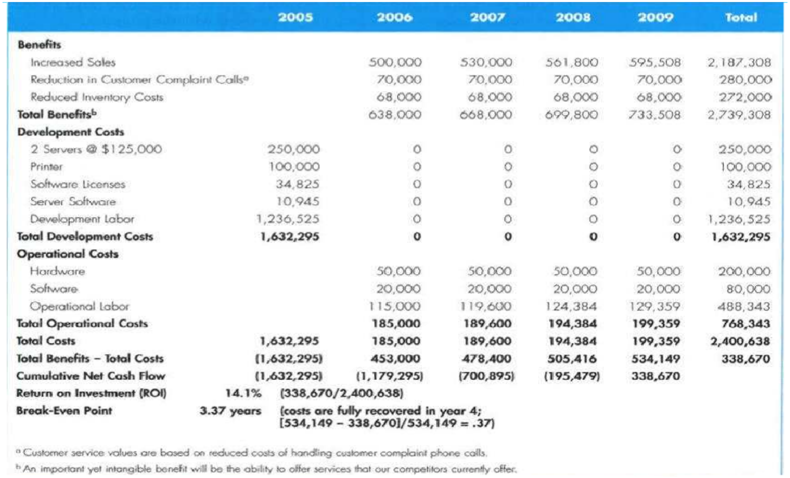
Now, for an example of a cost-benefit analysis over 5 years period

- For a given outlay, we wish to see what the returns will be by the end of the period;
- At the same time, we need to understand whether the project will be economically viable;
- If it is not viable, no matter how interesting the project is, it should not proceed;
- This approach ensures confidence in the project and shows that you understand itseconomic implications ;

We can see the total costs and the total benefits:

- Knowing something about these figures allows us to say something about the return;
- Being able to represent this information in a form that is understood by management will make it more likely to get your project funded

Cost Benefit Analysis — Cash Flow



	2005	2006	2007	2008	2009	Total
Benefits						
Increased Sales		500,000	530,000	561,800	595,508	2,187,308
Reduction in Customer Complaint Calls[a]		70,000	70,000	70,000	70,000	280,000
Reduced Inventory Costs		68,000	68,000	68,000	68,000	272,000
Total Benefits[b]		638,000	668,000	699,800	733,508	2,739,308
Development Costs						
2 Servers @ $125,000	250,000	0	0	0	0	250,000
Printer	100,000	0	0	0	0	100,000
Software Licenses	34,825	0	0	0	0	34,825
Server Software	10,945	0	0	0	0	10,945
Development Labor	1,236,525	0	0	0	0	1,236,525
Total Development Costs	**1,632,295**	**0**	**0**	**0**	**0**	**1,632,295**
Operational Costs						
Hardware		50,000	50,000	50,000	50,000	200,000
Software		20,000	20,000	20,000	20,000	80,000
Operational Labor		115,000	119,600	124,384	129,359	488,343
Total Operational Costs		**185,000**	**189,600**	**194,384**	**199,359**	**768,343**
Total Costs	**1,632,295**	**185,000**	**189,600**	**194,384**	**199,359**	**2,400,638**
Total Benefits – Total Costs	**(1,632,295)**	**453,000**	**478,400**	**505,416**	**534,149**	**338,670**
Cumulative Net Cash Flow	**(1,632,295)**	**(1,179,295)**	**(700,895)**	**(195,479)**	**338,670**	
Return on Investment (ROI)	**14.1%**	**(338,670/2,400,638)**				
Break-Even Point	**3.37 years**	**(costs are fully recovered in year 4; [534,149 – 338,670]/534,149 = .37)**				

[a] Customer service values are based on reduced costs of handling customer complaint phone calls.

[b] An important yet intangible benefit will be the ability to offer services that our competitors currently offer.

Dennis, Systems Analysis and Design, 3rd ed., 2006

Feasibility

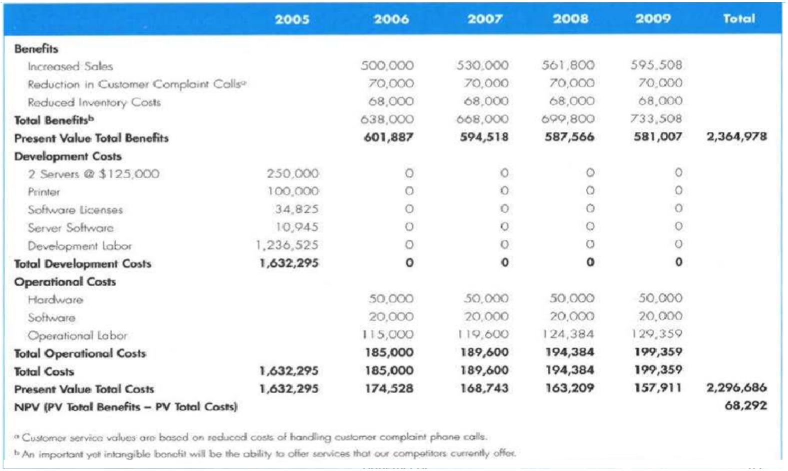
The same cost benefit analysis can be performed using Present Value figures:

- This is useful for comparison with the more basic cash flow analysis;
- Note that the PV adjusted total benefits are appreciably smaller than the previous version whilst the PV costs are only slightly smaller

The basic idea is that a unit of currency will be worth less in the future than it is today:

- We could take the unit of currency and invest it to make a return in the future;
- However, by committing the funds today we are forgoing this opportunity, therefore the value of the future fund will be less than it is today;
- This devaluation needs to be taken into account when calculating the potential return on investment!

Cost Benefit Analysis — Present Value



	2005	2006	2007	2008	2009	Total
Benefits						
Increased Sales		500,000	530,000	561,800	595,508	
Reduction in Customer Complaint Calls[a]		70,000	70,000	70,000	70,000	
Reduced Inventory Costs		68,000	68,000	68,000	68,000	
Total Benefits[b]		638,000	668,000	699,800	733,508	
Present Value Total Benefits		**601,887**	**594,518**	**587,566**	**581,007**	**2,364,978**
Development Costs						
2 Servers @ $125,000	250,000	0	0	0	0	
Printer	100,000	0	0	0	0	
Software Licenses	34,825	0	0	0	0	
Server Software	10,945	0	0	0	0	
Development Labor	1,236,525	0	0	0	0	
Total Development Costs	**1,632,295**	**0**	**0**	**0**	**0**	
Operational Costs						
Hardware		50,000	50,000	50,000	50,000	
Software		20,000	20,000	20,000	20,000	
Operational Labor		115,000	119,600	124,384	129,359	
Total Operational Costs		**185,000**	**189,600**	**194,384**	**199,359**	
Total Costs	**1,632,295**	**185,000**	**189,600**	**194,384**	**199,359**	
Present Value Total Costs	**1,632,295**	**174,528**	**168,743**	**163,209**	**157,911**	**2,296,686**
NPV (PV Total Benefits – PV Total Costs)						**68,292**

[a] Customer service values are based on reduced costs of handling customer complaint phone calls.

[b] An important yet intangible benefit will be the ability to offer services that our competitors currently offer.

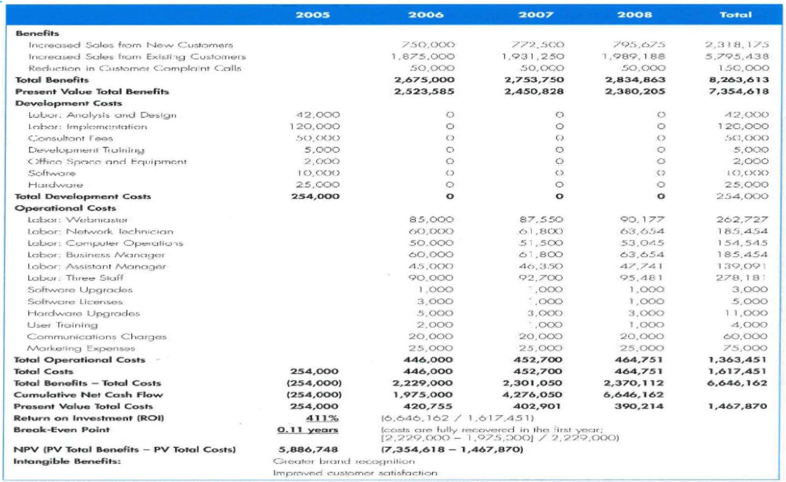
Dennis, Systems Analysis and Design, 3rd ed., 2006

Feasibility Analysis

We can bring all this information and analysis into a single report

- Provide a summary without too much detail
- Economic figures will necessarily be estimates but should be sufficient to make a decision;
- Brings all the relevant information together into one place;
- Different companies have different expectations from the report and different ways of presenting the information;

Cost Benefit Analysis — CD company



	2005	2006	2007	2008	Total
Benefits					
Increased Sales from New Customers		750,000	772,500	795,675	2,318,175
Increased Sales from Existing Customers		1,875,000	1,931,250	1,989,188	5,795,438
Reduction in Customer Complaint Calls		50,000	50,000	50,000	150,000
Total Benefits		**2,675,000**	**2,753,750**	**2,834,863**	**8,263,613**
Present Value Total Benefits		**2,523,585**	**2,450,828**	**2,380,205**	**7,354,618**
Development Costs					
Labor: Analysis and Design	42,000	0	0	0	42,000
Labor: Implementation	120,000	0	0	0	120,000
Consultant Fees	50,000	0	0	0	50,000
Development Training	5,000	0	0	0	5,000
Office Space and Equipment	2,000	0	0	0	2,000
Software	10,000	0	0	0	10,000
Hardware	25,000	0	0	0	25,000
Total Development Costs	**254,000**	**0**	**0**	**0**	254,000
Operational Costs					
Labor: Webmaster		85,000	87,550	90,177	262,727
Labor: Network Technician		60,000	61,800	63,654	185,454
Labor: Computer Operations		50,000	51,500	53,045	154,545
Labor: Business Manager		60,000	61,800	63,654	185,454
Labor: Assistant Manager		45,000	46,350	47,741	139,091
Labor: Three Staff		90,000	92,700	95,481	278,181
Software Upgrades		1,000	1,000	1,000	3,000
Software Licenses		3,000	1,000	1,000	5,000
Hardware Upgrades		5,000	3,000	3,000	11,000
User Training		2,000	1,000	1,000	4,000
Communications Charges		20,000	20,000	20,000	60,000
Marketing Expenses		25,000	25,000	25,000	75,000
Total Operational Costs		**446,000**	**452,700**	**464,751**	**1,363,451**
Total Costs	**254,000**	**446,000**	**452,700**	**464,751**	**1,617,451**
Total Benefits – Total Costs	**(254,000)**	**2,229,000**	**2,301,050**	**2,370,112**	**6,646,162**
Cumulative Net Cash Flow	**(254,000)**	**1,975,000**	**4,276,050**	**6,646,162**	
Present Value Total Costs	**254,000**	**420,755**	**402,901**	**390,214**	**1,467,870**
Return on Investment (ROI)	**411%**	(6,646,162 / 1,617,451)			
Break-Even Point	**0.11 years**	(costs are fully recovered in the first year; [2,229,000 – 1,975,000] / 2,229,000)			
NPV (PV Total Benefits – PV Total Costs)	**5,886,748**	**(7,354,618 – 1,467,870)**			
Intangible Benefits:	Greater brand recognition				
	Improved customer satisfaction				

Dennis, Systems Analysis and Design, 3rd ed., 2006

Requirements Elicitation and Analysis

- Sometimes called requirements elicitation or requirements discovery;
- Involves technical staff working with customer about the application domain, the services that the system should provide and the systems' operational constraings:
- May involve end-users, managers, engineers involved in maintenance, domain experts, trade unions, etc. These are called *stakeholders.*

Problems of Requirements Analysis

- Stakeholders don't know what they really want;
- Stakeholders express requirements in their own terms;
- Different stakeholders may have conflicting requirements;
- Organisational and political factors may influence the system requirements;
- The requirements change during the analysis process. New stakeholders may emerge and the business environment may change.

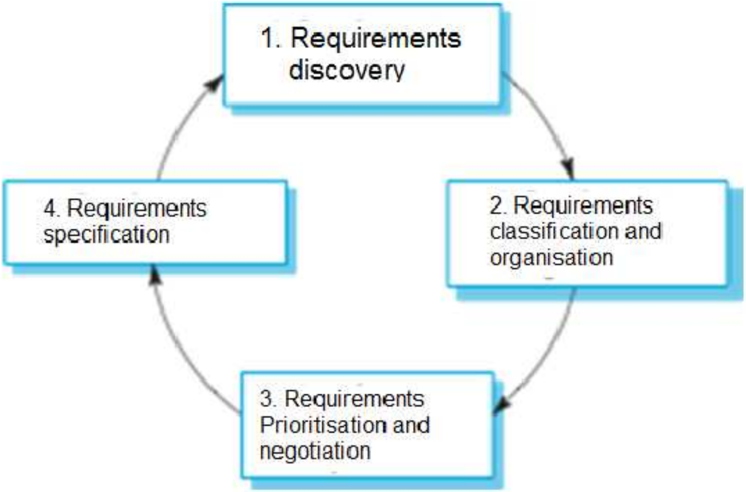
Requirement Elicitation and Analysis

Software engineers work with a range of system stakeholders to find out about the application domain, the services that the system should provide, the required system performance, hardware constraints, other systems, etc.

Stages include:

- Requirements discovery;
- Requirements classification and organization;
- Requirements prioritization and negotiation;
- Requirements specification.

Requirement Elicitation and Analysis


1. Requirements discovery
2. Requirements classification and organisation
3. Requirements Prioritisation and negotiation
4. Requirements specification

Process Activities

Requirements discovery: Interacting with stakeholders to discover their requirements. Domain requirements are also discovered at this stage;

Requirements classification and organisation: Groups related requirements and organises them into coherent clusters;

Prioritisation and negotiation: Prioritising requirements and resolving requirements conflicts;

Requirements specification: Requirements are documented and input into the next round of the spiral.

Key Points

- The software requirements document is an agreed statement of the system requirements. It should be organized so that both system customers and software developers can use it;
- The requirements engineering process is an iterative process including requirements elicitation, specification and validation;
- Requirements elicitation and analysis is an iterative process that can be represented as a spiral of activities — requirements discovery, requirements classification and organization, requirements negotiation and requirements documentation.

Requirements Discovery

Requirements Discovery

- The process of gathering information about the required and existing systems and distilling the user and system requirements from this information;
- Interaction is with system stakeholders from managers to external regulators;
- Systems normally have a (wide) range of stakeholders.

Stakeholders in the MHC-PMS

- Patients whose information is recorded in the system;
- Doctors who are responsible for assessing and treating patients;
- Nurses who coordinate the consultations with doctors and administer some treatments;
- Medical receptionists who manage patients' appointments;
- IT staff who are responsible for installing and maintaining the system.

Stakeholders in the MHC-PMS

- A medical ethics manager who must ensure that the system meets current ethical guidelines for patient care;
- Health care managers who obtain management information from the system;
- Medical records staff who are responsible for ensuring that system information can be maintained and preserved, and that record keeping procedures have been properly implemented.

Interviewing

Formal or informal interviews with stakeholders are part of most RE processes.

Types of interview:

- Closed interviews based on pre-determined list of questions;
- Open interviews where various issues are explored with stakeholders.

Effective interviewing:

- Be open-minded, avoid pre-conceived ideas about the requirements and are willing to listen to stakeholders;
- Prompt the interviewee to get discussions going using a springboard question, a requirements proposal, or by working together on a prototype system.

Interviews in Practice

Normally we have a mix of closed and open-ended interviewing.

- Interviews are good for getting an overall understanding of what stakeholders do and how they might interact with the system;
- Interviews are *NOT GOOD* for understanding domain requirements:
  - Requirements engineers cannot understand specific domain terminology;
  - Some domain knowledge is so familiar that people find it hard to articulate or think that it isn't worth articulating.

Scenarios

Scenarios are real-life examples of how a system can be used. They should include

- A description of the starting situation;
- A description of the normal flow of events;
- A description of what can go wrong;
- Information about other concurrent activities;
- A description of the state when the scenario finishes.

Scenario for Collecting Medical History in MHC-PMS

Initial assumption: The patient has seen a medical receptionist who has created a record in the system and collected the patient's personal information (name, address, age, etc.). A nurse is logged on to the system and is collecting medical history.
Normal: The nurse searches for the patient by family name. If there is more than one patient with the same surname, the given name and date of birth are used to identify the patient.
The nurse chooses the menu option to add medical history.
The nurse then follows a series of prompts from the system to enter information about consultations elsewhere on mental health problems (free text input), existing medical conditions (nurse selects conditions from menu), medication currently taken (selected from menu), allergies (free text), and home life (form).

Scenario for Collecting Medical History in MHC-PMS

What can go wrong:

- *The patient's record does not exist or cannot be found*, The nurse should create a new record and record personal information;
- *Patient conditions or medication are not entered in the menu* The nurse should choose the "other" option and enter free text describing the condition/medication;
- *Patient cannot/will not provide information on medical history* The nurse should enter free text recording the patient's inability/unwillingness to provide information.
- The system should print the standard exclusion form stating that the lack of information may mean that treatment will be limited or delayed. This should be signed and handed to the patient.

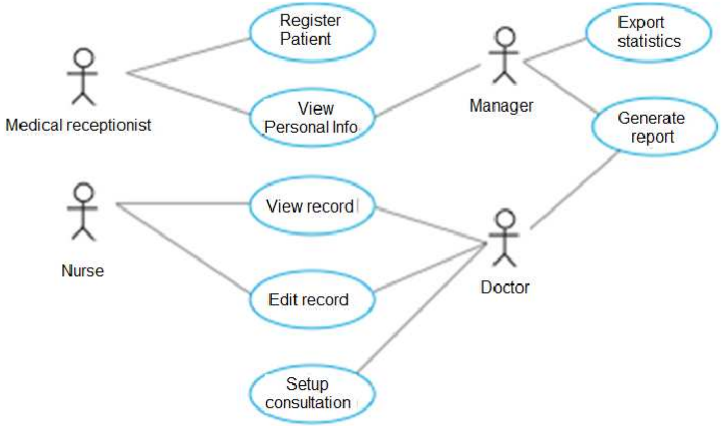
Scenario for Collecting Medical History in MHC-PMS

Other activities: Record may be consulted but not edited by other staff while information is being entered.
System state on completion: User is logged on. The patient record including medical history is entered in the database, a record is added to the system log showing the start and end time of the session and the nurse involved.

Use Cases

- Use-cases are a scenario based technique in the UML which identify the actors in an interaction and which describe the interaction itself;
- A set of use cases should describe all possible interactions with the system;
- High-level graphical model supplemented by more detailed tabular description;
- Sequence diagrams may be used to add detail to use-cases by showing the sequence of event processing in the system.

Use Cases for the MHC-PMS



Ethnography

- A social scientist spends a considerable time observing and analysing how people actually work;
- People do not have to explain or articulate their work;
- Social and organisational factors of importance may be observed;
- Ethnographic studies have shown that work is usually richer and more complex than suggested by simple system models.

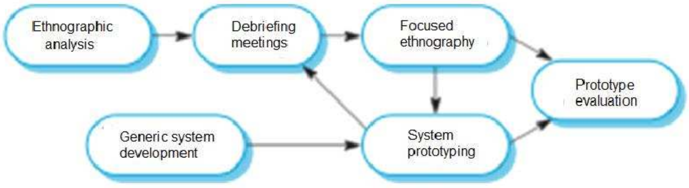
Scope of Ethnography

- Requirements that are derived from the way that people *actually* work rather than the way in which process definitions suggest they ought to work;
- Requirements that are derived from cooperation and awareness of other poeple's activities;
- Awareness of what other people are doing leads to changes in the ways in which we do things;
- Ethnography is effective for understanding existing processess but cannot identify new features that should be added to a system.

Focused Ethnography

- Developed in a project studying the air traffic control process;
- Combines ethnography with prototyping;
- Prototype development results in unanswered questions which focus the ethnographic analysis;
- The problem with ethnography is that it studies existing practices which may have some historical basis that is no longer relevant.

Ethnography for Requirement Analysis



Requirements Validation

- Concerned with demonstrating that the requirements define the system that the customer *really* wants;
- Requirements error costs are high so validation is very important;
- Fixing a requirements error after delivery may cost up to 100 times the cost of fixing an implementation error!

Requirements Checking

Validity	Does the system provide the functions which best support the customer's needs?
Consistency	Are there any requirements conflicts?
Completeness	Are all functions required by the customer included?
Realism	Can the requirements be implemented given the available budget and technology?
Verifiability	Can the requirements be checked?

Requirements Validation Techniques

Requirements reviews	Systematic manual analysis of the requirements.
Prototyping	Using an executable model of the system to check requirements;
Test-case generation	Developing tests for requirements to check testability.

Requirement Reviews

- Regular reviews should be held while the requirements definition is being formulated;
- Both client and contractor staff should be involved in reviews;
- Reviews may be formal (with completed documents) or informal.

Good communications between developers, customers and users can resolve problems at an early stage.

Review Checks

Verifiability	Is the requirement realistically testable?
Comprehensibility	Is the requirement properly understood?
Traceability	Is the origin of the requirement clearly stated?
Adaptability	Can the requirement be changed without a large impact on other requirements?

Requirement Management

Requirement management

is the process of managing changing requirements during the requirements engineering process and system development.

- New requirements emerge as a system is being developed and after it has gone into use;
- You need to keep track of individual requirements and maintain links between dependent requirements so that you can assess the impact of requirements changes. You need to establish a formal process for making change proposals and linking these to system requirements.

Changing Requirements

The business and technical environment of the system changes after installation.

New hardware may be introduced, it may be necessary to interface the system with other systems, business priorities may change (with consequent changes in the system support required), and new legislation and regulations may be introduced that the system must necessarily abide by.

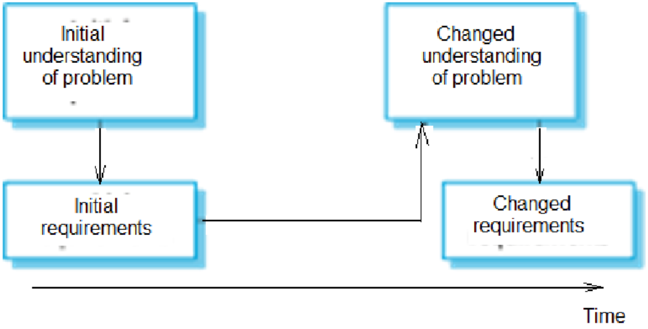
The people who pay for a system and the users are rarely the same people.

System customers impose requirements because of organizational and budgetary constraints. These may conflict with end-user requirements and, after delivery, new features may have to be added for user support if the system is to meet its goals.

Changing Requirements

- Large systems usually have a diverse user community with many users having different requirements and priorities that may be conflicting or contradictory;
- The final system requirements are inevitably a compromise between them and, with experience, it is often discovered that the balance of support given to different users has to be changed.

Requirements Evolution


Initial understanding of problem
Changed understanding of problem
Initial requirements
Changed requirements
Time

Requirement Management Planning

- Establishes the level of requirements management that is required;
- Requirements management decisions:

  Requirements identification Each requirement must be uniquely identified so that it can be cross-referenced with other requirements;

  A change management process The set of activities that assess the impact and cost of changes;

  Traceability policies These define the relationships between each requirement and between the requirements and the system design that should be recorded;

  Tool support Tools that may be used range from specialist requirements management systems to spreadsheet and simple database systems.

Requirements Change Management

Deciding if a requirements change should be accepted:

Problem analysis and change specification During this stage, the problem or the change proposed is analyzed to check that it is valid. This analysis is fed back to the change requestor who may respond with a more specific requirements change proposal, or decide to withdraw the request

Change analysis and costing The effect of the proposed change is assessed using traceability information and general knowledge of the system requirements. Once this analysis is completed, a decision is made whether or not to proceed with the requirements change.

Change implementation The requirements document and, where necessary, the system design and implementation, are modified. Ideally, the document should be organized so that changes can be easily implemented.

Key Points

- You can use a range of techniques for requirements elicitation including interviews, scenarios, use-cases and ethnography;
- Requirements validation is the process of checking the requirements for validity, consistency, completeness, realism and verifiability;
- Business, organizational and technical changes inevitably lead to changes to the requirements for a software system. Requirements management is the process of managing and controlling these changes.