

# SOFTWARE ENGINEERING

C03001

## CHAPTER 3 — REQUIREMENTS ENGINEERING

Truong Tuan Anh

# TOPICS COVERED

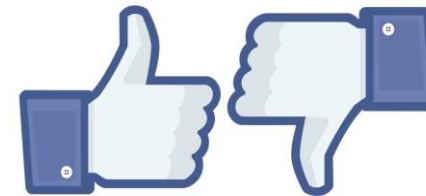
- ✓ Course's project – A smart printing service
- ✓ Functional and non-functional requirements
- ✓ Requirements engineering processes
- ✓ Requirements elicitation
- ✓ Requirements specification
- ✓ Requirements validation
- ✓ Requirements change

# REQUIREMENTS

# EFFECTS OF INADEQUATE REQUIREMENTS DEVELOPMENT – AIRBUS

- ✓ **Requirement:** *Reverse thrust may only be used when the airplane is landed.*
- ✓ **Translation:** *Reverse thrust may only be used while the wheels are rotating.*
- ✓ **Implementation:** *Reverse thrust may only be used while the wheels are rotating fast enough.*

- **Situation:** Rainstorm – aquaplaning
- **Result:** Crash due to overshooting the runway!
- **Problem:** Erroneous in the requirement phase



# The Ariane 5 accident – 1

- Single root cause failure!
- **The “bug”:** attitude deviation stored as 2-byte integer (max value 65,535) instead of 4-byte (max value 4,294,967,295)
- SW module was reused from Ariane 4
- Insufficient V&V of detailed requirements: larger attitude deviation tolerated in Ariane 5 than in Ariane 4
- Ariane 5 production cost 10 years and \$7 billion; luckily, no victims because it was unmanned.

65,535 = 00000000 00000000 11111111 11111111

65,536 = 00000000 00000001 00000000 00000000



# OTHER SOFTWARE RELATED ACCIDENTS & INCIDENTS

## ■ Accidents & incidents

- Alarm flooding, **power distribution** failure, BP Grangemouth Scotland, 29th May - 10th June 2000
- failure in a data bus + faulty logic in the software => engine power loss, **Airbus A340-642**, 2005
- failed accelerometer + software bug => Faulty airspeed metering, **Boeing 777-200**, 2005
- Software bug => shutdown of radio communication between **ATC and aircraft**, 2004. disrupted 800 flights
- Safety-related software flaws => recall of 200,000 **pacemakers** in 1990-2000
- **radiotherapy machines** attacked by computer **viruses**, 2005
- Buggy software incorporate computer + connection between corporate and control systems networks => shutdown of the **nuclear power plant**, USA 2008
- Many software failures are actually “bugs” in the **detailed requirement specifications**, i.e., poor understanding of the very detailed requirements



Korean Air 747 in Guam, 200 deaths (1997):  
incorrect configuration of  
“minimum altitude” warning  
system





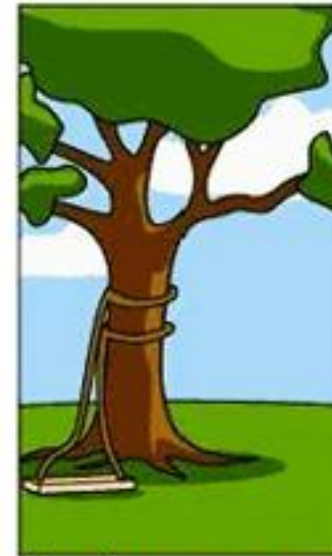
How the customer explained it



How the project leader understood it



How the engineer designed it



How the programmer wrote it



How the sales executive described it



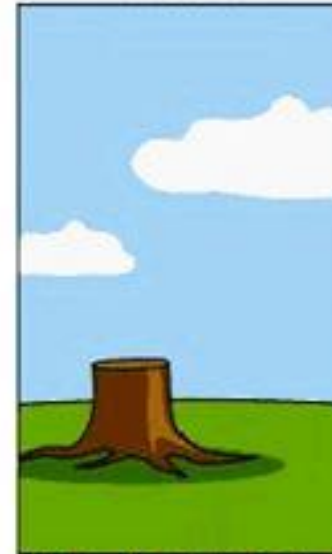
How the project was documented



What operations installed



How the customer was billed



How the helpdesk supported it



What the customer really needed

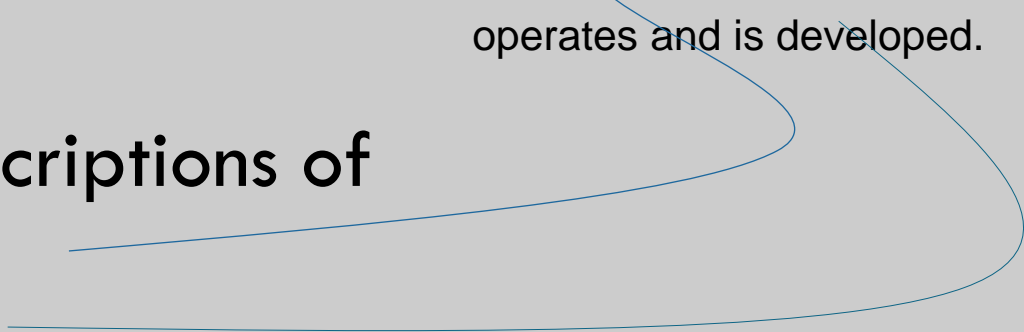
# REQUIREMENTS ENGINEERING

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



# WHAT IS A REQUIREMENT?

Requirement engineering = establishing the **services** that the customer requires from a system and the **constraints** under which it operates and is developed.



- ✓ Requirement = the descriptions of
  - the system services
  - and constraints
- ✓ It may range
  - from a high-level abstract statement
  - to a detailed mathematical functional specification.
- ✓ May serve a dual function
  - The basis for a bid for a contract - must be open to interpretation;
  - The basis for the contract itself - must be in detail;

# TYPES OF REQUIREMENT

## ✓ Requirements definition

- A statement in natural language plus diagrams of the services the system provides and its operational constraints. Written for customers

## ✓ Requirements specification

- A structured document setting out detailed descriptions of the system services. Written as a contract between client and contractor

## ✓ Software specification

- A detailed software description which can serve as a basis for a design or implementation. Written for developers

# MENTCARE SYSTEM



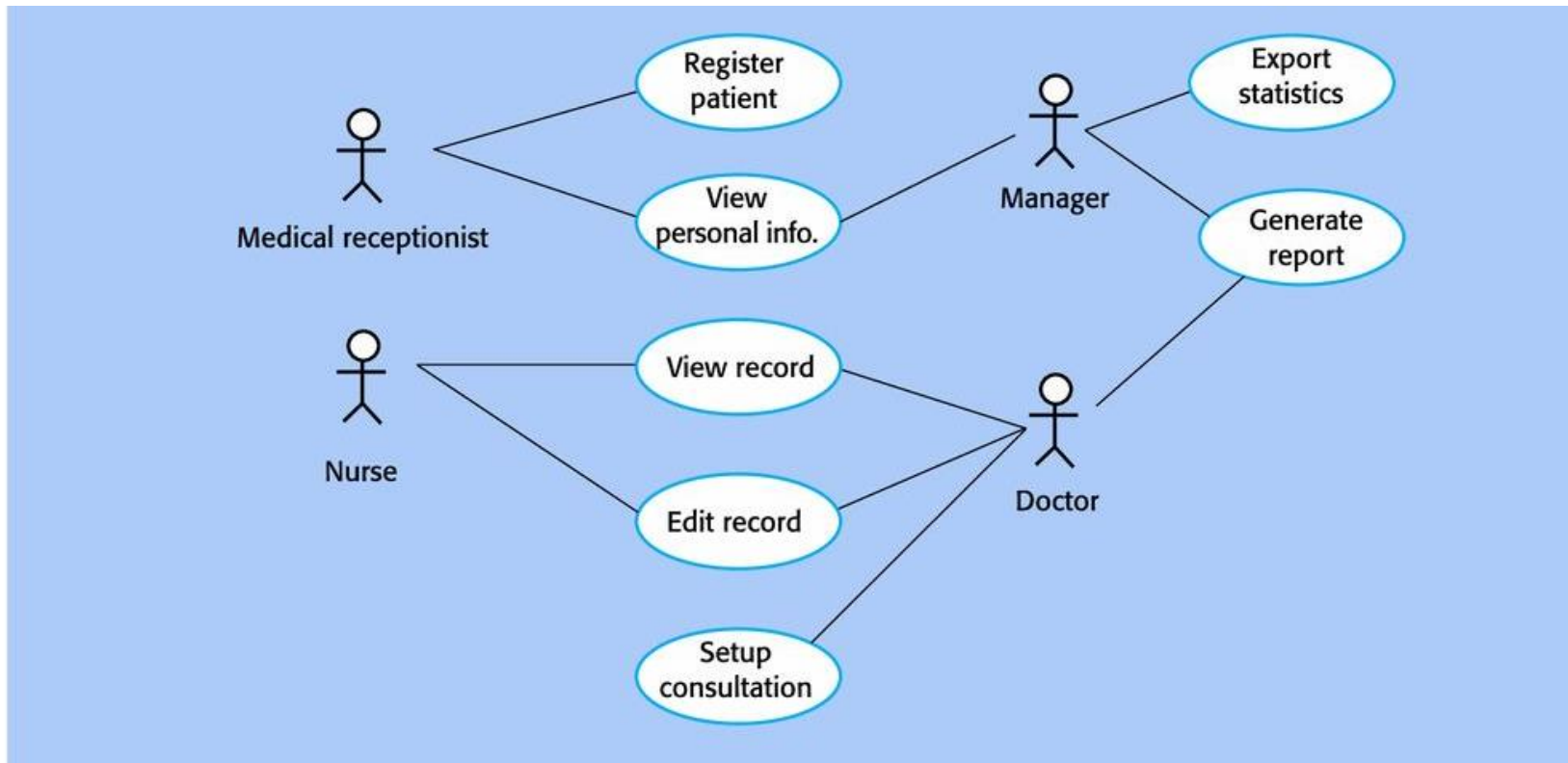
- ✓ Mentcare (Medical practice management system): manages the day-to-day operations of a clinic, such as appointment scheduling, billing and other administrative tasks.

<div> <div>📦</div> <div>All A B C D E F G H I J K L M N O P Q R S T U V W X Y Z # ...</div> <div>cat desc</div> </div>																
	S	Cat #	Catalog Description	Item #	Description	Qty / Purchased	Unit	Qty	Unit	Released / Due	Par Max	Current Stock	Par Min	Par Critical		
2		202407387	AMLODIPINE BESYLATE 5MG (NORVASC) TAB U/D; 100/BOX	51079045120	AMLODIPINE BESYLATE 5 MG TAB	2.0000 0	100 TA	200.0000	TA	0 2		27 / 27				0
3		202407388	AMLODIPINE BESYLATE 10MG (NORVASC) TAB U/D; 100/BOX	00054010220	AMLODIPINE BESYLATE 10 MG TAB	1.0000 0	100 TA	100.0000	TA	0 1		274 / 275				0
4		200805780	AMOXICILLIN CAP 500MG U/D; 100/BOX	00093310993	AMOXICILLIN 500 MG CAPSULE	2.0000 0	100 CA	200.0000	CA	0 2		186 / 188				0
5		202801738	ASPIRIN EC TAB 81MG U/D 25X30; 750/BOX	63739027201	ASPIRIN EC 81 MG TABLET	1.0000 0	750 TE	750.0000	TE	0 1		172 / 173				0
8		201202229P	BENZTROPINE MESYLATE TAB 1MG	00904105661	BENZTROPINE MES 1MG TABLET	1.0000	100 TA	100.0000	TA	0 1		104 / 104				0

# MENTCARE SYSTEM



- ✓ Mentcare (Medical practice management system): manages the day-to-day operations of a clinic, such as appointment scheduling, billing and other administrative tasks.



# USER AND SYSTEM REQUIREMENTS EXAMPLE

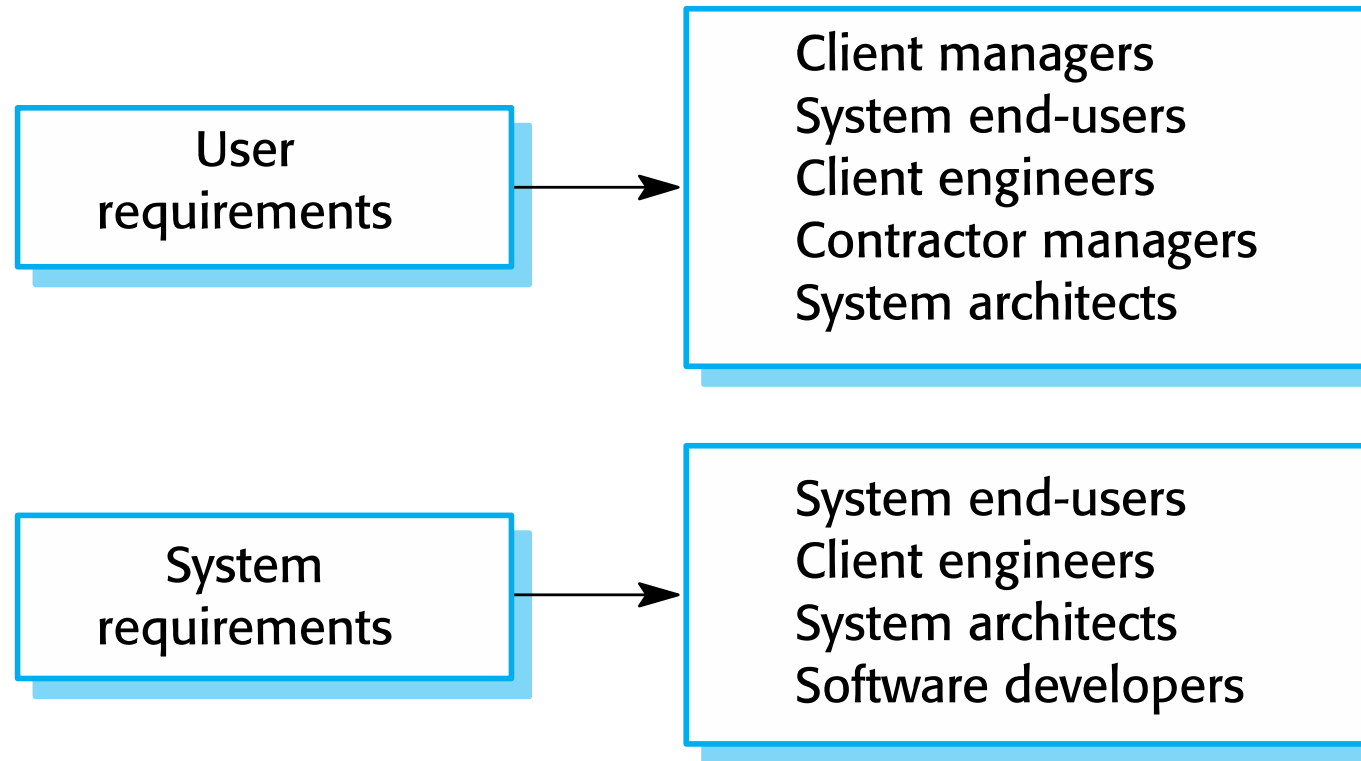
## User requirements definition

- 1.** The Mentcare system shall generate monthly management reports showing the cost of drugs prescribed by each clinic during that month.

## System requirements specification

- 1.1** On the last working day of each month, a summary of the drugs prescribed, their cost and the prescribing clinics shall be generated.
- 1.2** The system shall generate the report for printing after 17.30 on the last working day of the month.
- 1.3** 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.
- 1.4** If drugs are available in different dose units (e.g. 10mg, 20mg, etc) separate reports shall be created for each dose unit.
- 1.5** Access to drug cost reports shall be restricted to authorized users as listed on a management access control list.

# READERS OF DIFFERENT TYPES OF REQUIREMENTS SPECIFICATION



# SYSTEM STAKEHOLDERS

- ✓ Any person or organization who is affected by the system in some way and so who has a legitimate interest
- ✓ Stakeholder types
  - End users
  - System managers
  - System owners
  - External stakeholders



# STAKEHOLDERS IN THE MENTCARE SYSTEM

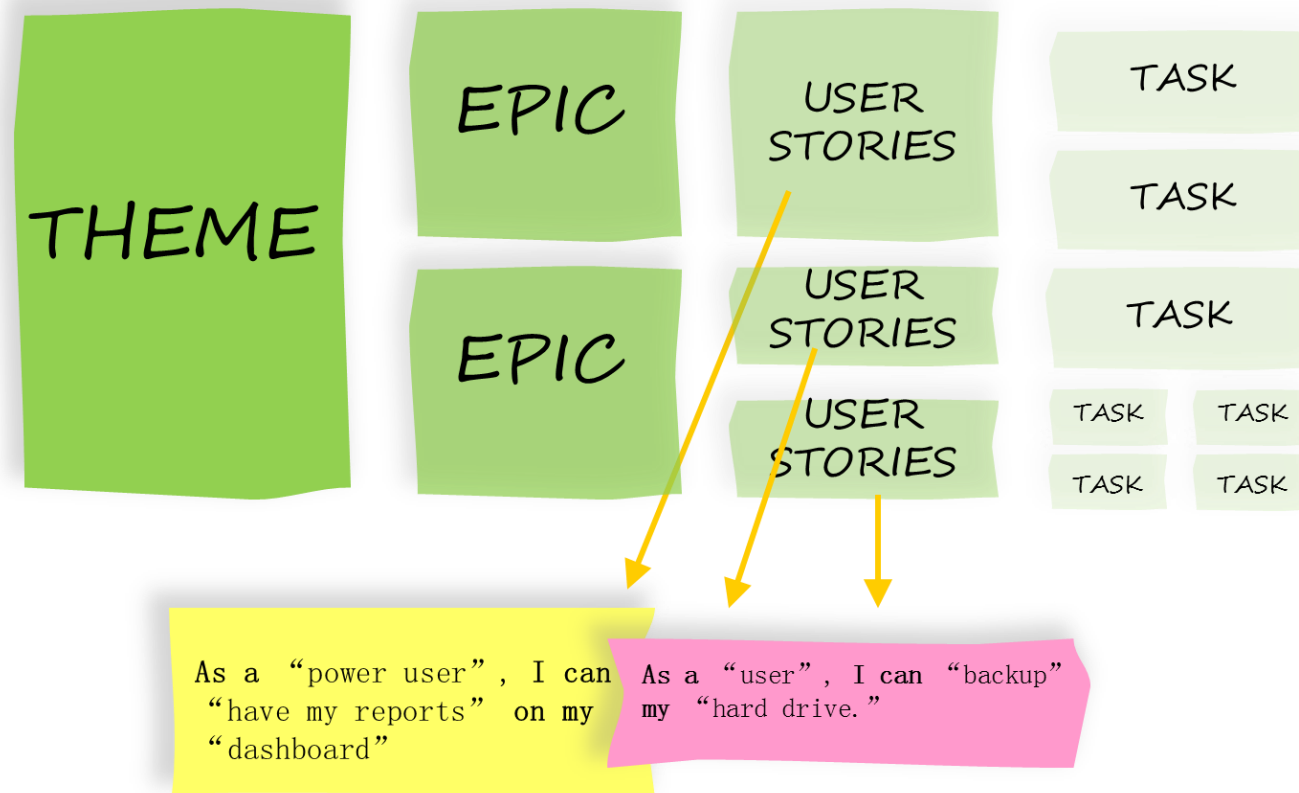
Stakeholder	Why? - Role
Patients	whose information is recorded in the system
Doctors	responsible for assessing and treating patients
Nurses	coordinate the consultations with doctors and administer some treatments
Medical receptionists	manage patients' appointments
IT staff	responsible for installing and maintaining the system
Medical ethics manager	ensure that the system meets current ethical guidelines for patient care
Health care managers	obtain management information from the system
Medical records staff	responsible for ensuring that system information can be maintained and preserved, and that record keeping procedures have been properly implemented.

# AGILE METHODS AND REQUIREMENTS

- ✓ Many Agile methods argue that producing detailed system requirements is a waste of time as requirements change so quickly.
- ✓ The requirements document is therefore always out of date.
- ✓ Agile methods usually use incremental requirements engineering and may express requirements as ‘user stories’
- ✓ This is practical for business systems but problematic for systems that require pre-delivery analysis (e.g. critical systems) or systems developed by several teams.

# AGILE METHODS AND REQUIREMENTS – USER STORY

## USER STORIES



[www.agile-scrum.be](http://www.agile-scrum.be)

# AGILE METHODS AND REQUIREMENTS — USER STORY

- ✓ As a medical record staff, I wish to view current medication from primary care general practice
- ✓ As a medical receptionist, I want to search appointments by patients' first name, their telephone number, their date of birth or their prescribed medicine
- ✓ As a healthcare manager, I want to see the monthly report including number of prescribed medicines ordered by their categories, by weeks, and by cost



# FUNCTIONAL AND NON-FUNCTIONAL 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 the system should behave in particular 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.
  - Functional user requirements may be high-level statements of what the system should do.
  - Functional system requirements should describe the system services in detail.



# MENTCARE SYSTEM: FUNCTIONAL REQUIREMENTS

- ✓ 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.

# NON-FUNCTIONAL REQUIREMENTS

- ✓ Define system properties and constraints
  - Properties: reliability, response time and storage requirements.
  - Constraints: I/O device capability, system representations, etc.
  
- ✓ Non-functional requirements may be more critical than functional requirements.
  - If these are not met, the system may be useless.

# NON-FUNCTIONAL REQUIREMENTS IMPLEMENTATION

- ✓ Non-functional requirements may affect the overall architecture of a system
  - rather than the individual components
  - cross-cutting concern
  
- ✓ A single non-functional requirement
  - may generate a number of related functional requirements
  - and may also generate requirements that restrict existing requirements.

# NON-FUNCTIONAL CLASSIFICATIONS

- ✓ **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.

# EXAMPLES OF NONFUNCTIONAL REQUIREMENTS IN THE MENTCARE SYSTEM

- ✓ **Product requirement**
  - The Mentcare system shall be available to all clinics during normal working hours (Mon–Fri, 0830–17.30). Downtime within normal working hours shall not exceed five seconds in any one day.
- ✓ **Organizational requirement**
  - Users of the Mentcare 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.

## EXERCISE: 15 MINS (+ 10 MINS BREAK)

- ✓ Watch the clip: <https://www.youtube.com/watch?v=J9p47AO0Xco>
- ✓ Imagine that each employee will install a new Messenger App as a part of the UWC 2.0 system.

### Task:

- Identify the stakeholders of the app
- Specify 03 functional requirements, at least one functional requirement extracted from the clip. Write them as an user story (guideline: <https://www.mountaingoatsoftware.com/agile/user-stories>)
- Specify 02 non-functional requirements you can think of! Make sure they are testable!
- Document your answers:

[https://docs.google.com/spreadsheets/d/1JAKNwIClopl4gjk\\_0cP9fJq-54TS1zc004gb3Q5mahl/edit?usp=sharing](https://docs.google.com/spreadsheets/d/1JAKNwIClopl4gjk_0cP9fJq-54TS1zc004gb3Q5mahl/edit?usp=sharing)

# WHAT IS A GOOD REQUIREMENT?

- Precise/ Unambiguous
- Complete
- Consistent
- Correct
- ✓ Testable
- ✓ Ranked for importance and/or stability



# REQUIREMENTS IMPRECISION

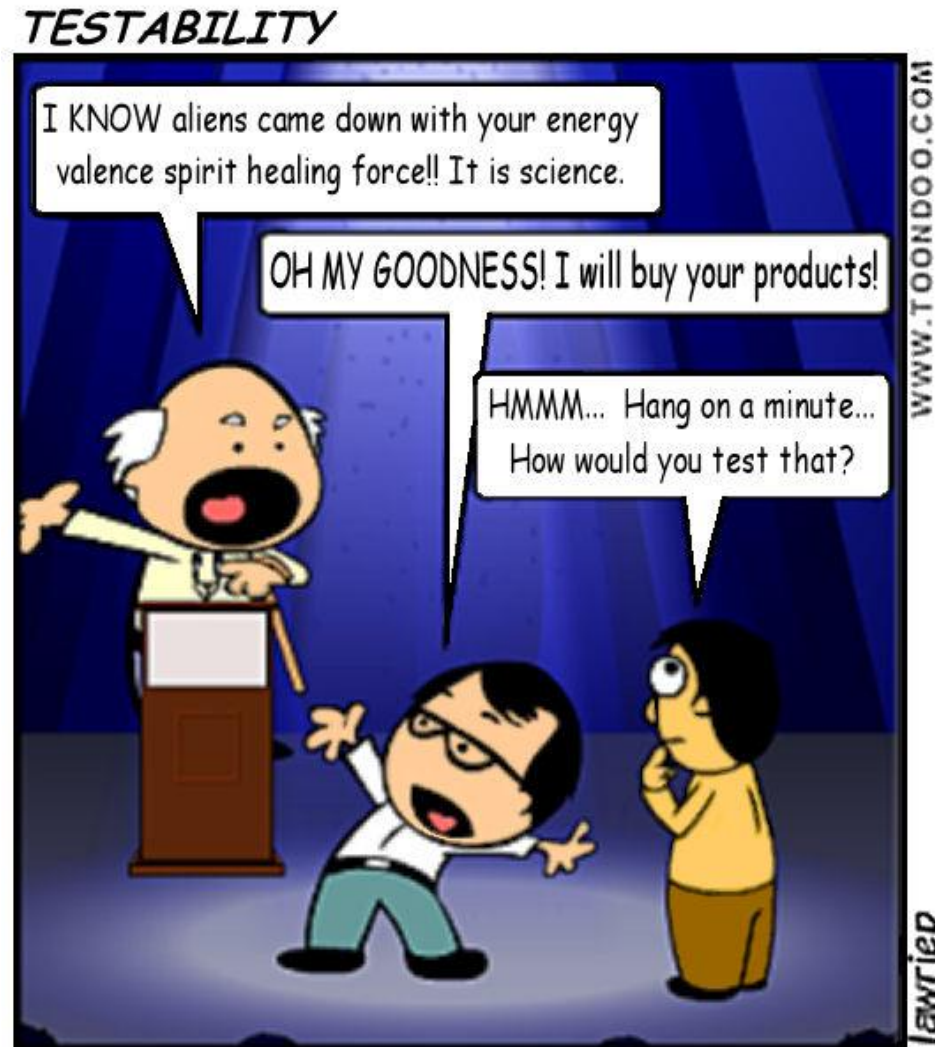
- ✓ A software requirement is unambiguous, if and only if, every requirement stated therein has only one interpretation
- ✓ Problems arise when requirements are not precisely stated.
  - Ambiguous requirements may be interpreted in different ways by developers and users.
- ✓ For the term 'search' in requirement 1
  - User intention – 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.

# REQUIREMENTS COMPLETENESS AND CONSISTENCY

- ✓ Requirements should be both complete and consistent.
- ✓ Complete
  - They should include descriptions of all facilities required.
  - All possible situations must be covered
- ✓ Consistent
  - There should be no conflicts or contradictions in the descriptions of the system facilities.

# TESTABLE REQUIREMENT

- ✓ If a method cannot be devised to determine whether the software meets a particular requirement, then that requirement is not testable.



# EXAMPLE OF NIGHTMARE REQUIREMENTS

- ✓ The system shall perform at the maximum rating at all times except that in emergencies it shall be capable of providing up to 125% rating unless the emergency condition continues for more than 15 minutes in which case the rating shall be reduced to 105% but in the event that only 95% can be achieved then the system shall activate a reduced rating exception and shall maintain the rating within 10% of the stated values for a minimum of 30 minutes.

More than one requirement in a paragraph  
Rambling: like a novel  
Vague terms: emergencies

# ANOTHER EXAMPLE OF NIGHTMARE REQUIREMENTS

- ✓ The system shall provide general word processing facilities which shall be easy to use by untrained staff and shall run on a thin Ethernet Local Area Network wired into the overhead ducting with integrated interface cards housed in each system together with additional memory if that should be necessary.

More than one requirement in a paragraph

Rambling: like a novel

Let-out clauses: if that should be necessary

Vague words: be easy to use, additional memory

# ISO/IEC 25010:2011 SOFTWARE QUALITY (NON-FUNCTIONAL) REQUIREMENTS

## Product quality model

### Functional suitability

Functional completeness  
Functional correctness  
Functional appropriateness

### Performance efficiency

Time behavior  
Resource use  
Capacity

### Compatibility

Coexistence  
Interoperability

### Usability

Appropriateness  
recognizability  
Learnability  
Operability  
User-error protection  
User-interface aesthetics  
Accessibility

### Reliability

Maturity  
Availability  
Fault tolerance  
Recoverability

### Security

Confidentiality  
Integrity  
Nonrepudiation  
Accountability  
Authenticity

### Maintainability

Modularity  
Reusability  
Analyzability  
Modifiability  
Testability

### Portability

Adaptability  
Installability  
Replaceability

# GOALS VS. REQUIREMENTS

## ✓ Goal

- A general intention of the user such as ease of use.

## ✓ Requirements are often

- Concrete
- Measureable
- Testable



# GOALS VS. REQUIREMENTS (CONT.)

## Goal:

The system should be easy to use by medical staff.




## Non-functional requirement:

Medical staff shall be able to use all the system functions after four hours of training.

# COURSE'S PROJECT – UWC 2.0

<https://docs.google.com/spreadsheets/d/1-FVIRZ62PFLCLW92koVbdvorSvfjknIPeIWPJGe20Dw/edit#gid=1655354121>



# SOFTWARE ENGINEERING

C03001

## CHAPTER 4 — REQUIREMENTS ENGINEERING PROCESS

Anh Nguyen-Duc  
Tho Quan Thanh

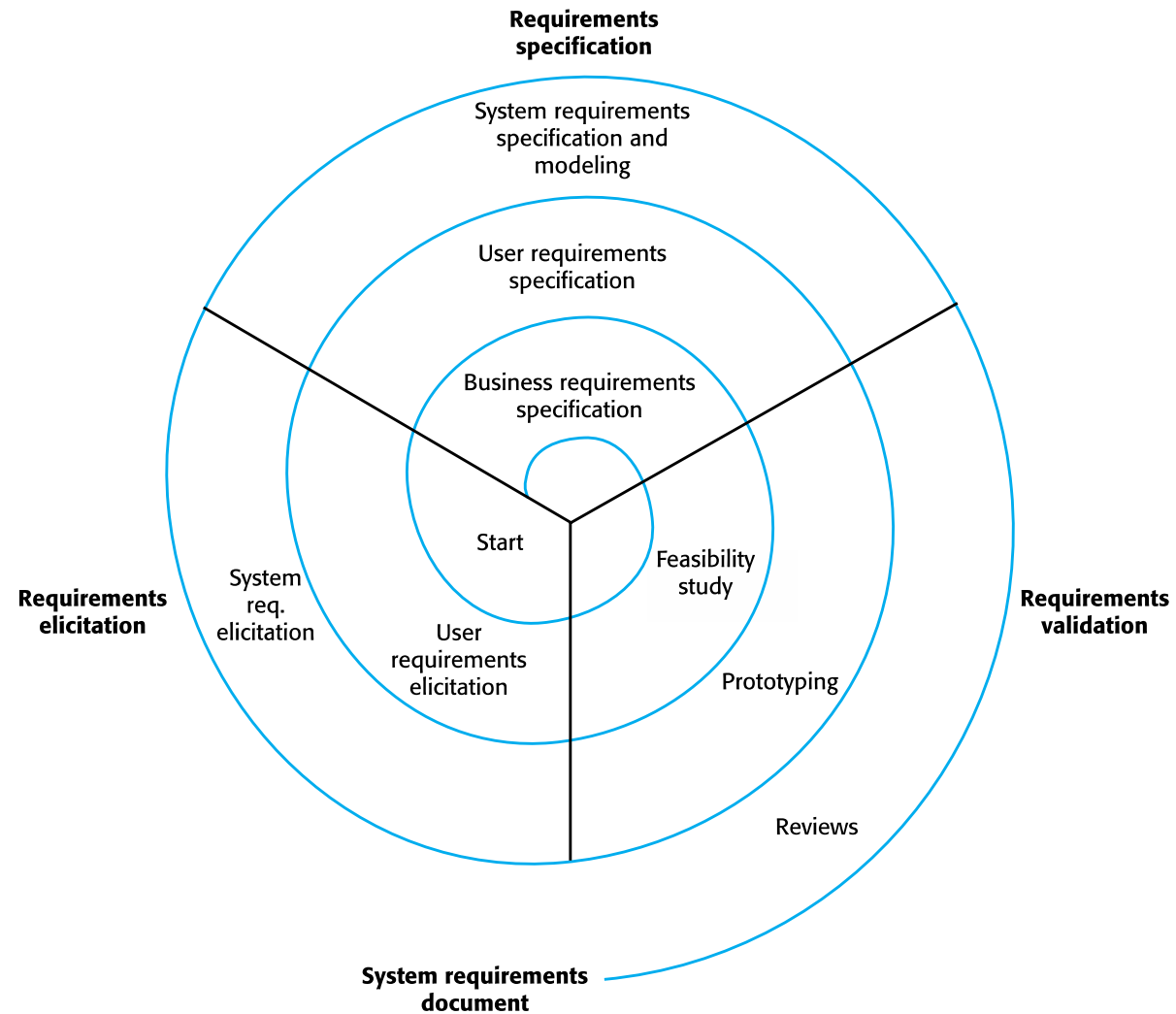
# REQUIREMENTS ENGINEERING PROCESSES

- ✓ Processes to “generate” all requirements
- ✓ Generic activities common to all processes
  - Requirements elicitation;
  - Requirements analysis;
  - Requirements validation;
  - Requirements management.
- ✓ In practice, RE is an iterative activity

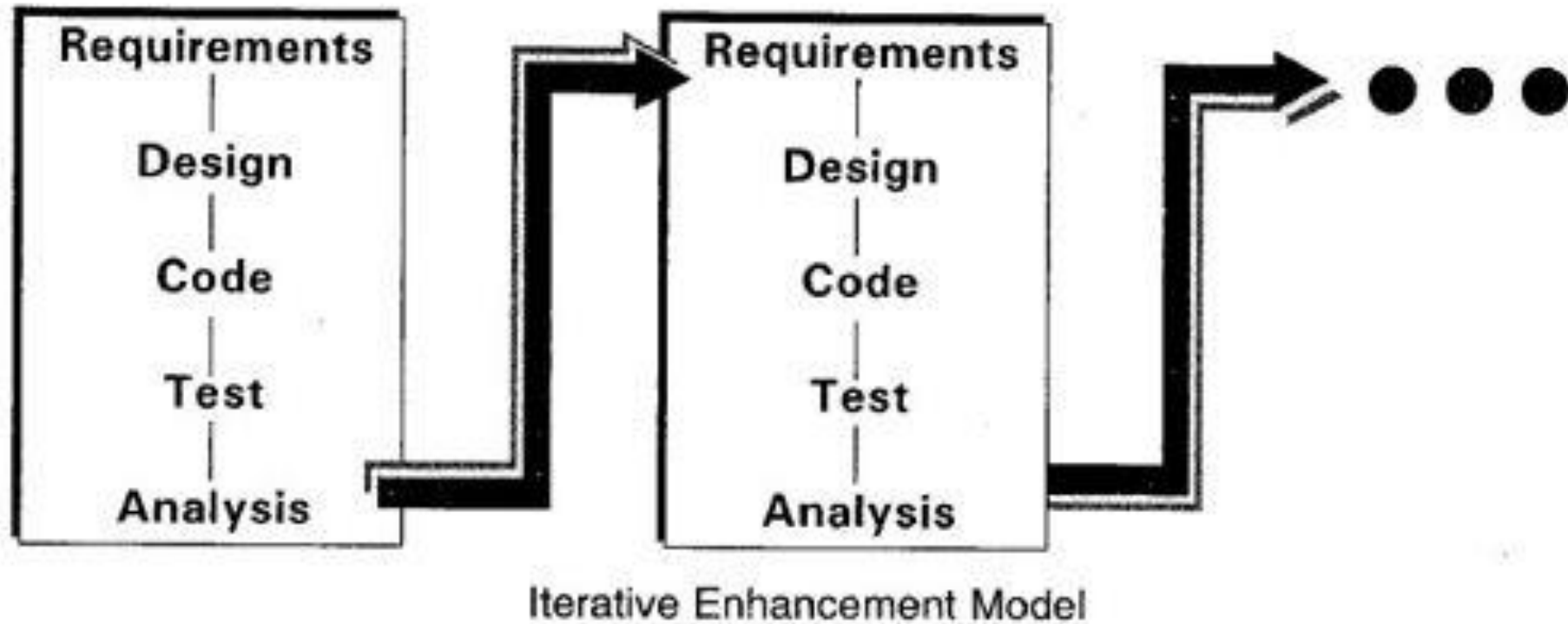


*"The client kept changing the requirements on a daily basis, so we decided to freeze them until the next release."*

# A SPIRAL VIEW OF THE REQUIREMENTS ENGINEERING PROCESS



# A AGILE VIEW OF THE REQUIREMENTS ENGINEERING PROCESS





# REQUIREMENT ELICITATION AND ANALYSIS

# REQUIREMENTS ELICITATION AND ANALYSIS

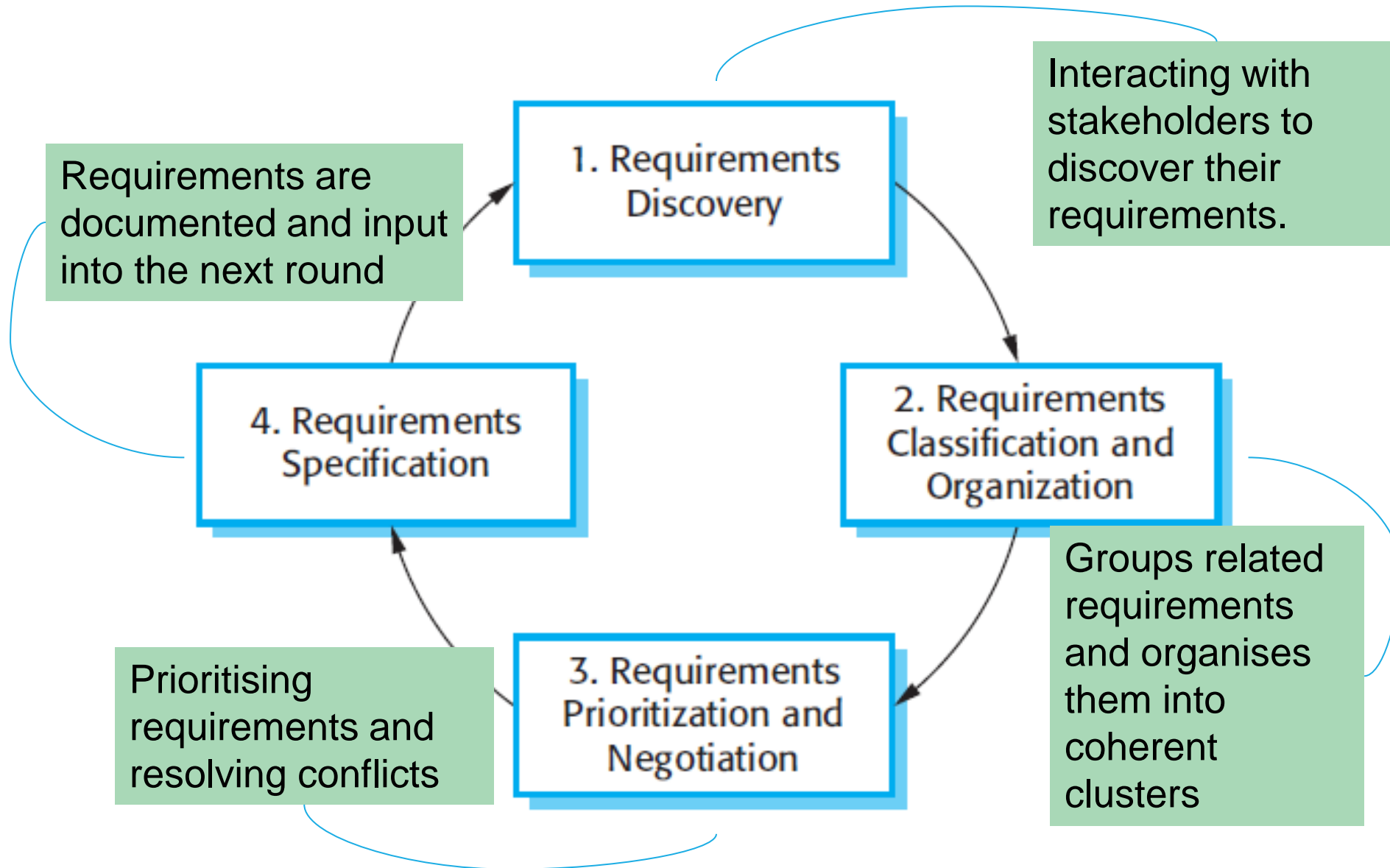
- ✓ ~ requirements elicitation or requirements discovery.
- ✓ Work with customers to find out:
  - the application domain, the services and the operational constraints (system performance, hardware constraints, etc.).
- ✓ May involve
  - end-users, managers, engineers involved in maintenance, domain experts, trade unions, etc. These are called stakeholders.



# PROBLEMS OF REQUIREMENTS ELICITATION

- ✓ 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 change.

# THE REQUIREMENTS ELICITATION AND ANALYSIS PROCESS



# REQUIREMENTS DISCOVERY

- ✓ To gather information about the required and existing systems and distil the user and system requirements from this information.
- ✓ Main concerns:
  - Stakeholders
  - Discovery techniques/approaches/...

# DISCOVERY TECHNIQUE - INTERVIEWING



- ✓ Part of most RE processes.
- ✓ Types of interview
  - Closed vs Open => mixed?
- ✓ Be effective
  - 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.

# DISCOVERY TECHNIQUE - ETHNOGRAPHY

## ✓ Observational technique

- used to understand operational processes and requirements for these processes

## ✓ How

- 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.



# STORIES AND SCENARIOS


- ✓ Scenarios and user stories can be used.
- ✓ Stories and scenarios can be used for a particular purpose.
- ✓ Because they are based on real life, they can relate to them and respect to the story.

## TOBI DAY

PERSONA TEMPLATE

**AGE** 26  
**OCCUPATION** Record Store Manager  
**STATUS** Single  
**LOCATION** New York, NY  
**TIER** Enthusiast  
**ARCHETYPE** The Maestro

Ambitious Admired Focused



"If I had a way to share projects and collaborate in real time, that would make my workload so much easier to manage."

### MOTIVATIONS

Incentive	
Fear	
Achievement	
Growth	
Power	
Social	

### GOALS

- To grow a strong industry reputation
- To build an audio-pro portfolio
- To keep track of everything

### FRUSTRATIONS

- Slow download times
- Data crashes
- Poor communication

### BIO


Tobi has a day job at a record store, but on the side she does all kinds of production work for up-and-coming artists. She never hesitates to learn something new and she often acts as tech support for her friends and clients. She is usually working on a dozen projects at a time and is trying to establish herself in the industry, so she hates data crashes or anything that makes her look bad. Because she works alone and in her home, collaboration is everything.

### PERSONALITY

Extrovert	Introvert
Sensing	Intuition
Thinking	Feeling
Judging	Perceiving

### TECHNOLOGY

IT and Internet	
Software	
Mobile Apps	
Social Networks	





# REQUIREMENT ELICITATION TECHNIQUES

	Interviews	Workshops	Focus groups	Observations	Questionnaires	System interface analysis	User interface analysis	Document analysis
Mass-market software	x		x		x			
Internal corporate software	x	x	x	x		x		x
Replacing existing system	x	x		x		x	x	x
Enhancing existing system	x	x				x	x	x
New application	x	x				x		
Packaged software implementation	x	x		x		x		x
Embedded systems	x	x				x		x
Geographically distributed stakeholders	x	x			x			

**Suggested** elicitation techniques by project characteristics



# REQUIREMENTS SPECIFICATION



# REQUIREMENTS SPECIFICATION

- ✓ The process of writing down the user and system requirements in a requirements document.
  
- ✓ Notes:
  - 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

# WAYS OF WRITING A SYSTEM REQUIREMENTS SPECIFICATION

Notation	Description
Natural language	Sentences in natural language. Each sentence should express one requirement.
Structured natural language	Natural language statements on a standard form or template
Design description languages	Uses a language like a programming language, but with more abstract features
Graphical notations	Graphical models, supplemented by text annotations (best for functional requirements); UML use case and sequence diagrams are commonly used.
Mathematical specifications	Based on mathematical concepts such as finite-state machines or sets; Can reduce the ambiguity but hard to understand (and hard to check manually)

# NATURAL LANGUAGE SPECIFICATION

- ✓ Used for writing requirements because it is expressive, intuitive and universal.
  - The requirements can be understood by users and customers.
  
- ✓ Problems
  - Lack of clarity: Precision is difficult without making the document difficult to read.
  - Requirements confusion: Functional and non-functional requirements tend to be mixed-up.
  - Requirements amalgamation: Several different requirements may be expressed together.

# EXAMPLE REQUIREMENTS FOR THE INSULIN PUMP SOFTWARE SYSTEM

- ✓ Req 3.2. The system shall measure the blood sugar and deliver insulin, if required, every 10 minutes.
- ✓ Req 3.6. The system shall run a self-test routine every minute with the conditions to be tested and the associated actions defined in Table 1.

# STRUCTURED SPECIFICATIONS

- ✓ Writing on a standard form or template:
  - Name
  - Inputs, outputs
  - The information needed for the computation
  - Action
  - Pre and post conditions (if appropriate)
  - The side effects (if any)
  
- ✓ This works well for some types of requirements e.g. requirements for embedded control system

Insulin Pump/Control Software/SRS/3.3.2	
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 (r2); the previous two readings (r0 and r1).
Source	Current sugar reading from sensor. Other readings from memory.
Outputs	CompDose—the dose in insulin to be delivered
Destination	Main control loop.
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	r0 is replaced by r1 then r1 is replaced by r2.
Side effects	None

# TABULAR SPECIFICATION

- ✓ Particularly useful when you have to define a number of possible alternative courses of action.
- ✓ Example:

Condition	Action
Sugar level falling ( $r2 < r1$ )	CompDose = 0
Sugar level stable ( $r2 = r1$ )	CompDose = 0
Sugar level increasing and rate of increase decreasing ( $(r2 - r1) < (r1 - r0)$ )	CompDose = 0
Sugar level increasing and rate of increase stable or increasing ( $(r2 - r1) \geq (r1 - r0)$ )	CompDose = round $((r2 - r1)/4)$ <b>If</b> rounded result = 0 <b>then</b> CompDose = MinimumDose

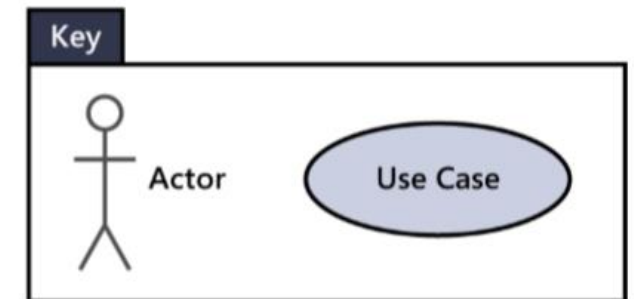
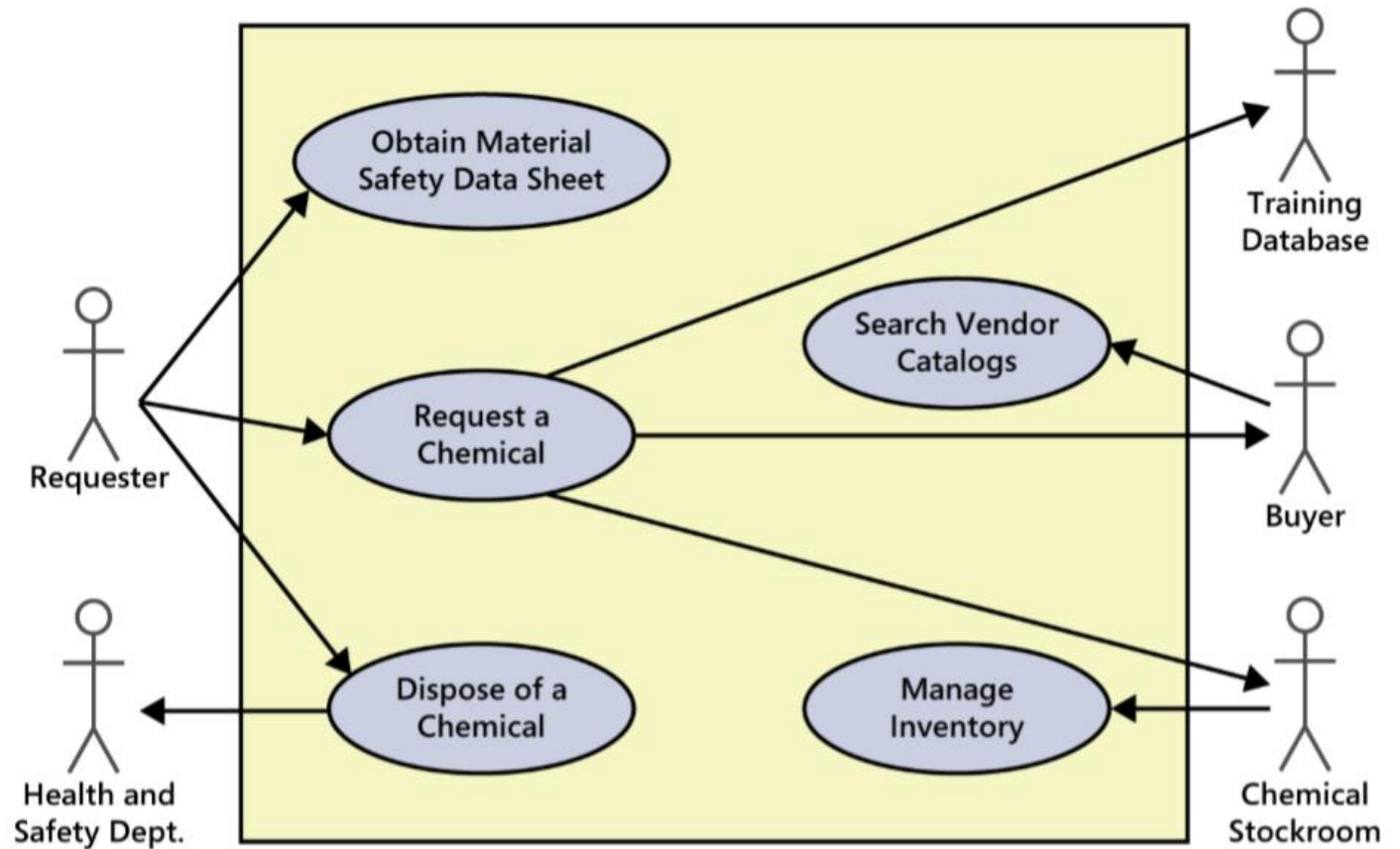
# USE CASES

- ✓ Use-cases are a kind of scenario
  - identify the actors in an interaction and which describe the interaction itself
  - Included in the UML
- ✓ A set of use cases should describe all possible interactions with the system.
- ✓ UML sequence diagrams may be used to add detail to use-cases
  - show the sequence of event processing in the system



# USE CASE DIAGRAM

- Actors
- Use cases
- Association
- System boundary boxes




**Partial use case diagram for the Chemical Tracking System (CTS)**

# USE CASE TEMPLATE AND EXAMPLE

ID and Name:	UC-4 Request a Chemical		
Created By:	Lori	Date Created:	8/22/13
Primary Actor:	Requester	Secondary Actors:	Buyer, Chemical Stockroom, Training Database
Description:	The Requester specifies the desired chemical to request by entering its name or chemical ID number or by importing its structure from a chemical drawing tool. The system either offers the Requester a container of the chemical from the chemical stockroom or lets the Requester order one from a vendor.		
Trigger:	Requester indicates that he wants to request a chemical.		
Preconditions:	PRE-1. User's identity has been authenticated. PRE-2. User is authorized to request chemicals. PRE-3. Chemical inventory database is online.		
Postconditions:	POST-1. Request is stored in the CTS. POST-2. Request was sent to the Chemical Stockroom or to a Buyer.		

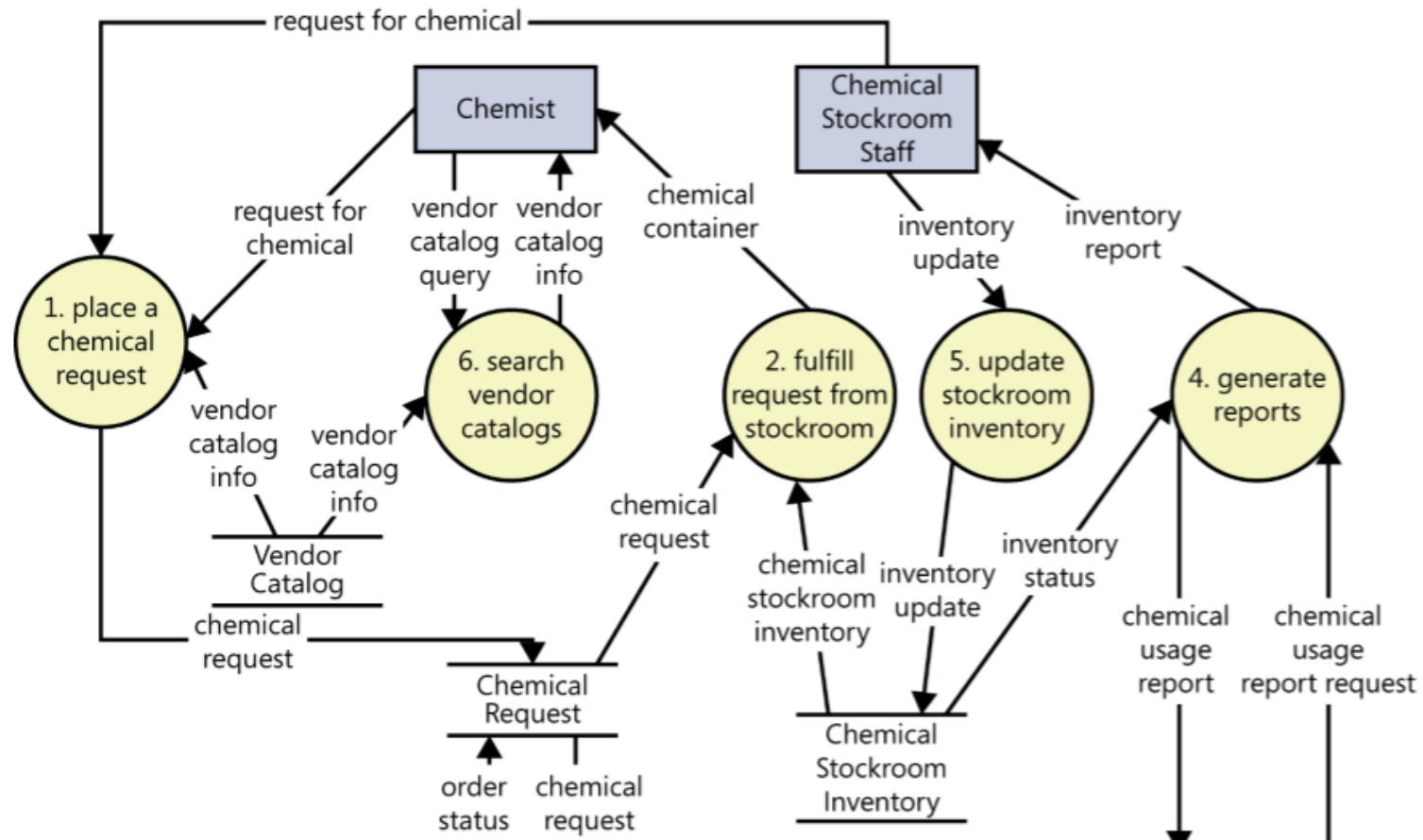
# USE CASE TEMPLATE AND EXAMPLE (CONT')

ID and Name:	UC-4 Request a Chemical
Normal Flow:	<b>4.0 Request a Chemical from the Chemical Stockroom</b> <ol style="list-style-type: none"><li>1. Requester specifies the desired chemical.</li><li>2. System lists containers of the desired chemical that are in the chemical stockroom, if any.</li><li>3. System gives Requester the option to View Container History for any container.</li><li>4. Requester selects a specific container or asks to place a vendor order (see 4.1).</li><li>5. Requester enters other information to complete the request.</li><li>6. System stores the request and notifies the Chemical Stockroom.</li></ol>
Alternative Flows:	<b>4.1 Request a Chemical from a Vendor</b> <ol style="list-style-type: none"><li>1. Requester searches vendor catalogs for the chemical (see 4.1.E1).</li><li>2. System displays a list of vendors for the chemical with available container sizes, grades, and prices.</li><li>3. Requester selects a vendor, container size, grade, and number of containers.</li><li>4. Requester enters other information to complete the request.</li><li>5. System stores the request and notifies the Buyer.</li></ol>
Exceptions:	<b>4.1.E1 Chemical Is Not Commercially Available</b>

- 
- ✓ Draw a use-case diagram for the whole UWC 2.0 system

# DATA FLOW DIAGRAM

✓ How data moves through a system



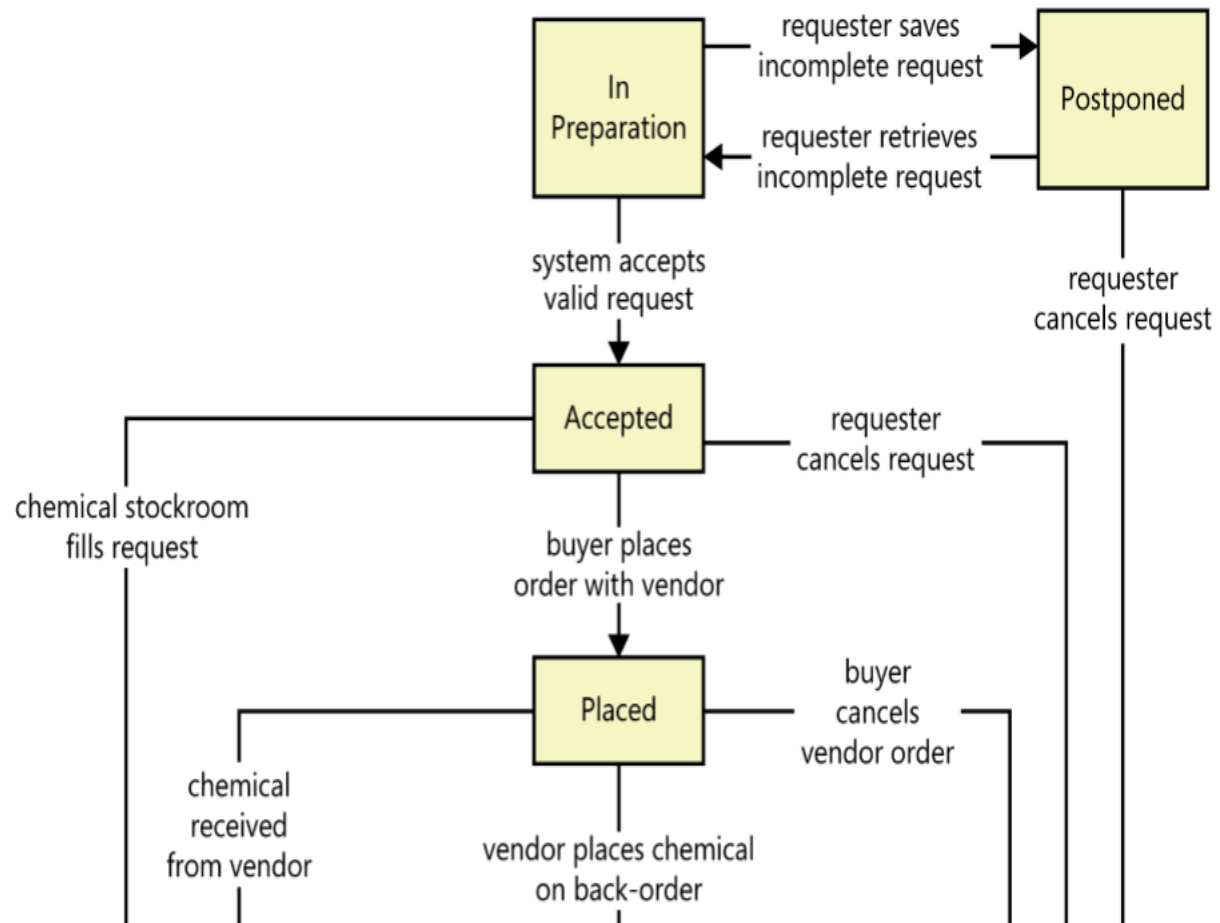
# DECISION TABLE/DECISION TREE

✓ Complex logics

Requirement Number					
Condition	1	2	3	4	5
User is authorized	F	T	T	T	T
Chemical is available	—	F	T	T	T
Chemical is hazardous	—	—	F	T	T
Requester is trained	—	—	—	F	T
Action					
Accept request			X		X
Reject request	X	X		X	

# STATE-TRANSITION DIAGRAM

- ✓ A set of complex state changes in natural language creates a high probability of overlooking a permitted state change or including a disallowed change.





# THE SOFTWARE REQUIREMENTS DOCUMENT

- ✓ The software requirements document is the official statement of what is required of the system developers.
- ✓ 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 of WHAT the system should do rather than HOW it should do it.





# REQUIREMENT VALIDATION

# REQUIREMENTS VALIDATION

- ✓ Requirements error can be costly !

Cost to Fix Requirements Error (in Ratios)			
Development Discipline/Phase	Composite of Studies (NASA)*	NASA Software <sup>#</sup>	Davis' Composite <sup>+</sup>
Requirements	1 (baseline)	1 (baseline)	1 (baseline)
Design	5x	8x	2.5x – 5x
Code	10x	16x	5x – 10x
Test	50.5x	21x	Unit Test: 10x – 20x Acceptance Test: 25x – 50x
Post-Deployment	n/a	29x	100x – 200x
<p>* NASA reviewed McGibbon (2003), Pavlina (2003), Cigital (2003), Rothman (2002), Hoffman (2001), Rothman (2000), and Boehm (1981) and assessed the median of their data, presented here.</p> <p># NASA analyzed its own software projects to assess the relevance of reviewed studies.</p> <p># Leffingwell &amp; Widrig reviewed GTE, TRW, IBM, and Davis (2003), who himself reviewed several studies. The results of Davis' work are presented here.</p>			

# 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 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.

# REQUIREMENTS CHANGE



*"The client kept changing the requirements on a daily basis, so we decided to freeze them until the next release."*

# CHANGING REQUIREMENTS

- ✓ The business and technical environment of the system always changes after installation.
- ✓ The people who pay for a system and the users of that system are rarely the same people.
- ✓ Large systems usually have a diverse user community, with many users having different requirements and priorities that may be conflicting or contradictory.

# REQUIREMENTS MANAGEMENT

- ✓ Requirements 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.

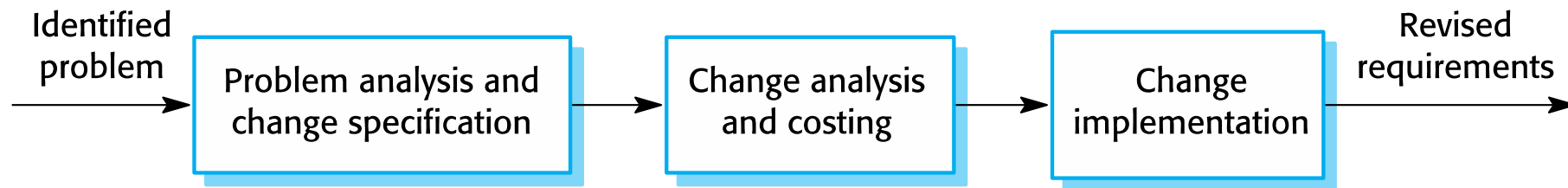
# REQUIREMENTS MANAGEMENT PLANNING

- ✓ Establishes the level of requirements management detail that is required.
  
- ✓ Requirements management decisions:
  - Requirements identification
  - A change management process
  - Traceability policies
  - Tool support

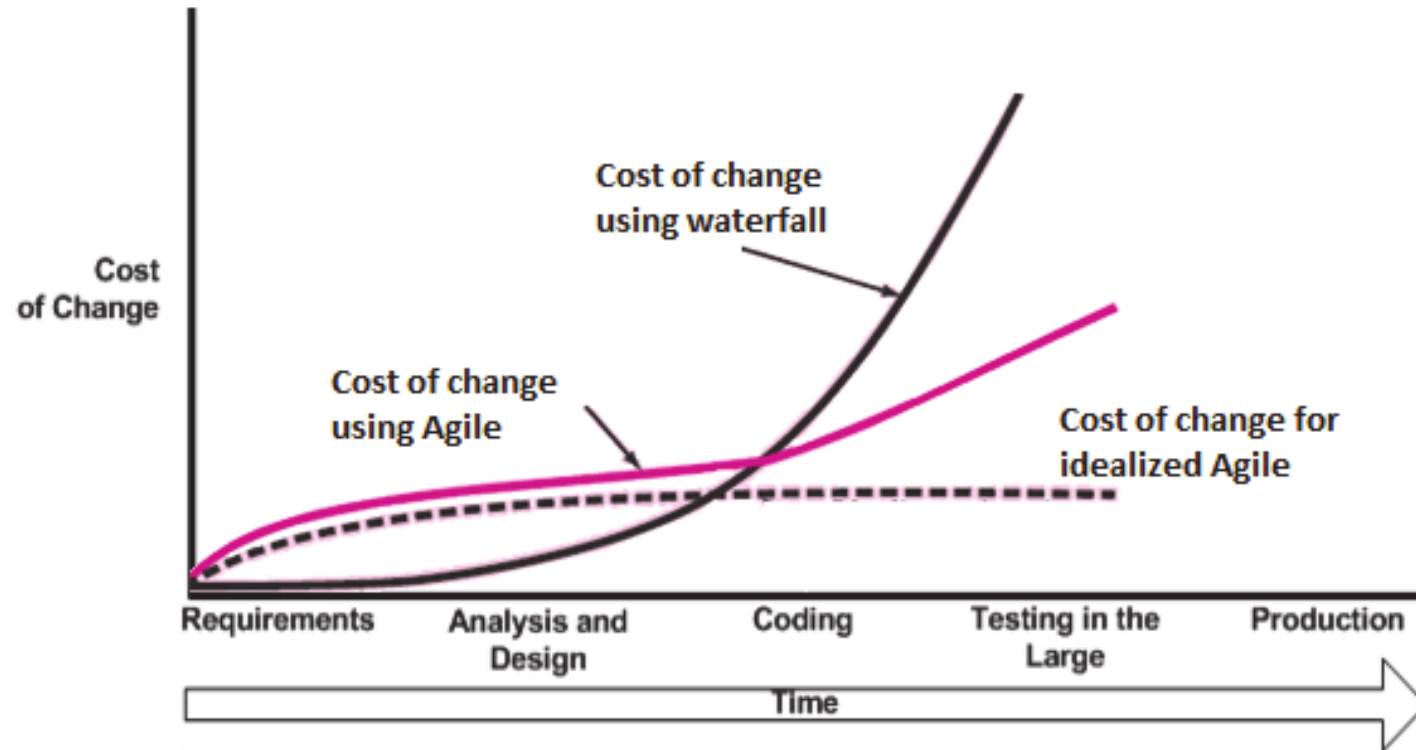


# REQUIREMENTS CHANGE MANAGEMENT

- ✓ Deciding if a requirements change should be accepted
  - Problem analysis and change specification
  - Change analysis and costing
  - Change implementation



# RESPONSE TO CHANGE: AGILE APPROACH!



# SUMMARY

- ✓ Requirements: what the system should do and constraints on its operation and implementation.
- ✓ Functional requirements = the services
- ✓ Non-functional requirements = constraints (development & use)
  - apply to the system as a whole.
- ✓ The software requirements document (i.e. SRS) is an agreed statement of the system requirements.
- ✓ The RE process is an iterative process
  - requirements elicitation, specification and validation.

## SUMMARY (CONT.)

- ✓ Requirements elicitation and analysis = iterative process
  - requirements discovery, classification and organization, negotiation and requirements documentation.
- ✓ Techniques for requirements elicitation
  - interviews, scenarios, use-cases and ethnography, etc.
- ✓ Requirements validation = checking the requirements
  - for validity, consistency, completeness, realism and verifiability.
- ✓ Business, organizational and technical changes inevitably
  - => changes to the requirements for a software system.
- ✓ Requirements management = managing and controlling the requirement changes.