Software Engineering Fundamentals

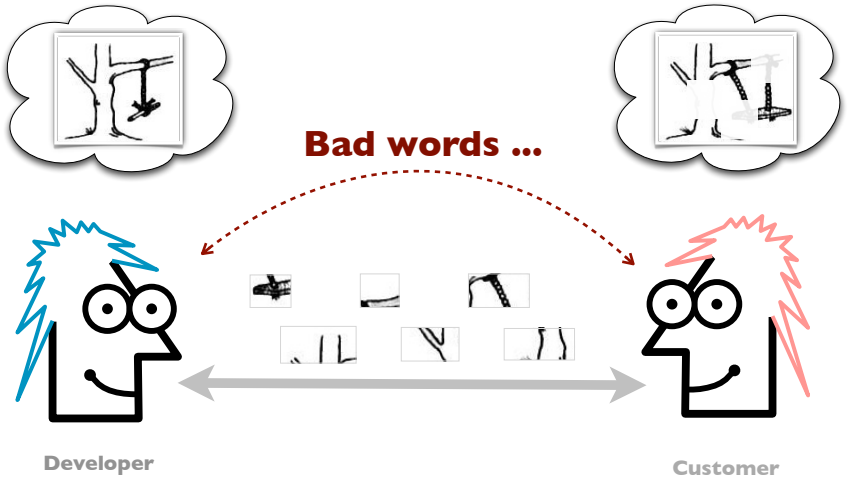
Requirements Engineering (Basics)

Dr. Petru Florin Mihancea

Based on:
I. Summerville - Software Engineering 8, Ch. 4 Software Processes
R. Pressman - Software Engineering, Ch. 10 System Engineering, Ch. 11 Analysis Concepts and Principles
M. Fowler - UML Distilled, Ch9. Use Cases

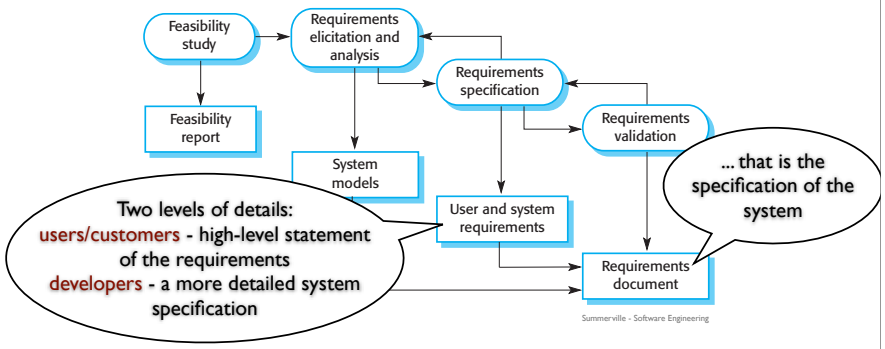
V20140315

Why?



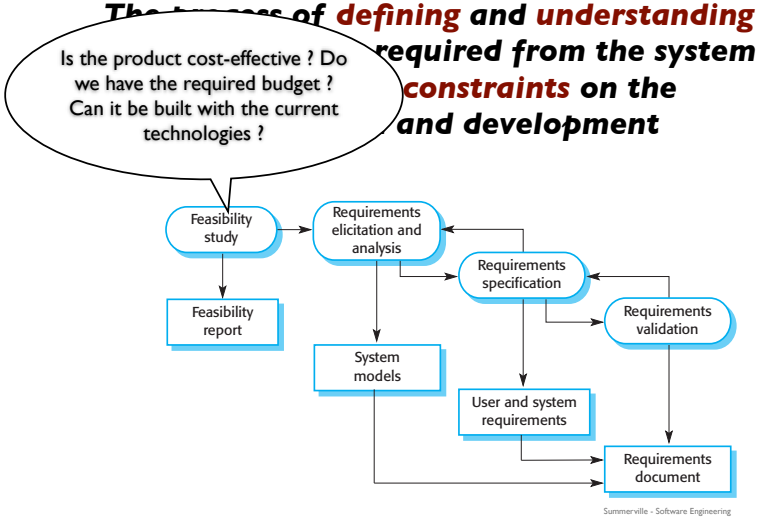
Requirements Engineering

The process of defining and understanding what services are required from the system and identifying the constraints on the system's operation and development



Requirements Engineering

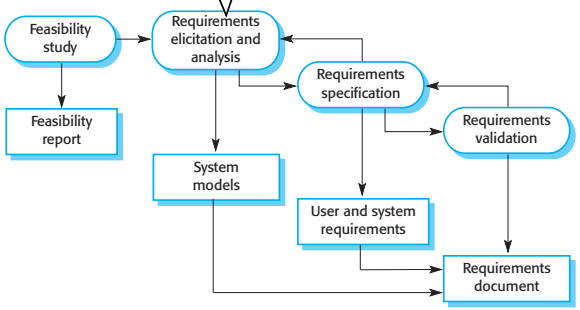
The process of defining and understanding required from the system constraints on the and development



Requirements Engineering

The process of defining and understanding what services are required from the system and identifying the constraints on the system's operation and development

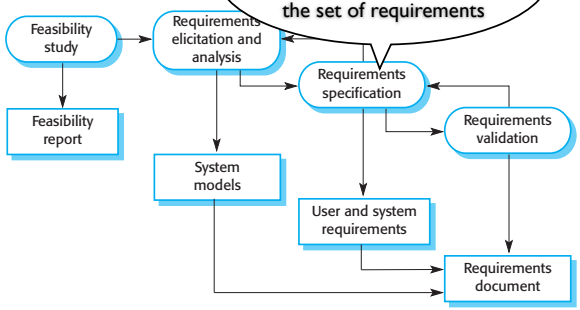
Deriving the requirements through observation and discussions with the potential users, etc.



Requirements Engineering

The process of defining and understanding what services are required from the system and identifying the constraints on the system's operation and development

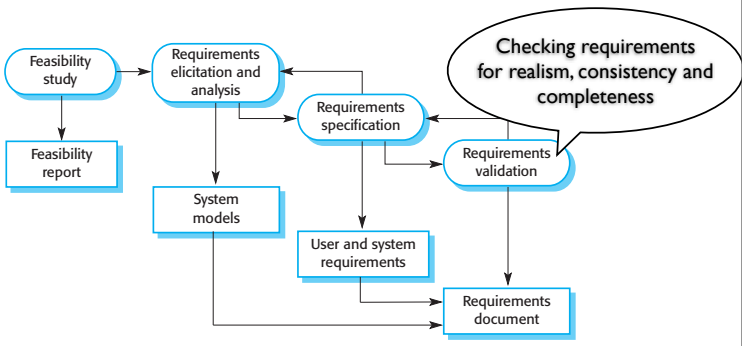
Translating the information from the analysis phase into a document that defines the set of requirements



Requirements Engineering

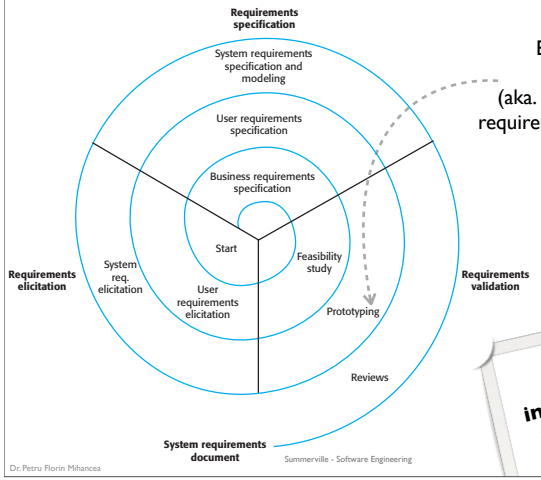
The process of defining and understanding what services are required from the system and identifying the constraints on the system's operation and development

Checking requirements for realism, consistency and completeness



Requirements Engineering

... a spiral perspective of the process accommodates approaches in which the requirements are developed to different levels of details



Extending this activity to include iterative development (aka. evolutionary prototyping) we enable requirements and the system to be developed together

In the case of agile methods, requirements are developed incrementally according to customer priorities and requirements comes from users who are part of the development team

1. Requirements Elicitation and Analysis

Ask users/customer during a meeting/interview

What are the objectives of the product?

What is to be accomplished?

How the product fits into the needs of the business?

How the system is going to be used on a day-by-day basis?

It looks simple ... :-)

But it is very hard ...

Problems of scope - customers/users specify unnecessary technical details that may confuse rather than clarify overall system objectives

Problems of understanding - customers/users are not completely sure about what is needed, might not have a full understanding of the problem domain, might have problems communicating their needs, might have ambiguous/ conflicting requirements, might omit something that is "obvious"

Problems of volatility - requirements change over time

We will see a little bit later how to proceed ...

1. Requirements Elicitation and Analysis (2)

Categorize requirements

Organize them into related subsets

Explore each requirement in relation with each other

Examines them for consistency, omissions, ambiguity

Rank requirements based on the user needs

Negotiate "conflicting" requirements

Negotiate requirements to fit in the available budget

Understand the requirements (e.g., for specification)

- **build models to better understand the actual entity to be built**
- **build (throwaway) prototypes**

2. Requirements Specification

A specification can be a written document, a graphical model, a formal mathematical model, a collection of usage scenarios, a prototype or any combination of these

Possibilities

standard templates

- **a consistent and more understandable manner of presentation**

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.

Requires	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

Summerville - Software Engineering

2. Requirements Specification

A specification can be a written document, a graphical model, a formal mathematical model, a collection of usage scenarios, a prototype or any combination of these

Possibilities

standard templates

- **a consistent and more understandable manner of presentation**

document in natural language + graphical models

- **for large systems**

usage scenarios

- **may be all that is required for small and medium-sized system**

3. Requirements Validation

Examines the specification to ensure that all requirements have been stated unambiguously, that inconsistencies and omissions have been detected & corrected, etc.

Technical Reviews

the team includes engineers, customers, users, and other stakeholders and examine the specification (or small portions of specifications while it is developed)

Useful questions

Are requirements clearly stated ?

Can they be misinterpreted ?

Has the final statement of the requirement been examined by or against the original source ?

Is the requirement testable ?

Can we specify tests, to exercise the requirement ?

4. Requirements Management

Is a set of activities that helps the project team to identify, control and track requirements and changes to requirements at any time

a requirement identification mechanism must be established (e.g.,unique ids)

Traceability matrixes

Feature traceability

requirements vs. important observable features

Source traceability

the source of a requirement

Requirement dependencies

requirements affecting other requirements

Requirement	Specific aspect of the system or its environment								
	A01	A02	A03	A04	A05				Aii
R01			✓		✓				
R02	✓		✓						
R03	✓			✓					✓
R04		✓			✓				
R05	✓	✓		✓					

We can identify how a requirement change affects different aspects of the built system

How to proceed ...

Requirements Elicitation and Analysis

Initiating the process

FAST Meetings

Use Cases

1

Initiating the process

Initiating the process

The first meeting will be awkward

... like the first dating

neither person knows what/how to say, both are afraid to be misinterpreted but ... both want to be a success :-)

http://www.polyvore.com/cgi/search.cgi?wg.displayurl=satellitesofyou.tumblr.com

Initiating the process (2)

Ask context-free questions

Who is behind the request of this work ?

Who will use the solution ?

What will be the economic benefit of a successful solution ?

Is there another source for the solution you need ?

help identify stakeholders, measurable benefits of the product, alternative solutions to the required product

What problem(s) will this solution address ?

How would you characterize good output that would be given by the solution ?

Can you show me the environment in which the solution will be used ?

better understanding of the problem, the customer perception about the solution

Initiating the process (3)

Ask meta-questions

Are my questions relevant to the problem that you have ?

Can anyone else provide additional information ?

Should I be asking you anything else ?

Are you the right person to answer these question ?

identify the effectiveness of the meeting

Q&A meetings are not good for requirements elicitation

Should be used only as a first-contact helping you to plan another meeting with a different format e.g., the beginning of FAST

2

Facilitated Application Specification Techniques (FAST)

FAST Basics

Team-oriented approach

Customers & developers work together to identify the problem, to propose elements of the solution, to negotiate different approaches and to specify a preliminary set of requirements

Guidelines

neutral meeting site attended by both engineers and customers

establish rules for preparation and participation

a "facilitator" chairs the meeting

establish a definition mechanism (flip charts, wall stickers, etc.)

an agenda is suggested to address all important points but should be informal enough to encourage free flow of ideas

Has several variations

FAST Preparation

During process initiation (previous section)

Write 1-2 pages document called product request

Establish a place, date, time, and a facilitator for the FAST meeting

Both customers and developers are invited to attend

The product request is distributed to all participants (before the meeting)

> Our research indicates that the market for home security systems is growing at a rate of 40 percent per year. We would like to enter this market by building a microprocessor-based home security system that would protect against and/or recognize a variety of undesirable "situations" such as illegal entry, fire, flooding, and others. The product, tentatively called *SafeHome,* will use appropriate sensors to detect each situation, can be programmed by the homeowner, and will automatically telephone a monitoring agency when a situation is detected.

Pressman - Software Engineering

Product Request Example

FAST Preparation (2)

Each attendee is asked to prepare

a list of objects that are

- part of the environment surrounding the system
- produced by the system
- used by the system to perform its functions

a list of services (functions) that manipulate/interact with those objects

a list of constraints (e.g., costs, size) and performance criteria (e.g., speed)

For the previous *SafeHome* example

objects: smoke/window/door/motion sensors, an alarm, an event (i.e., sensor activation), a control panel, a phone line, etc.

services: setting the alarm, monitoring sensors, dialing a phone number, etc. (note that the services act on the objects)

constraints: must cost less than 80$ per piece

FAST

1. Everybody should agree that the product is justified (i.e., feasible)

2. Each participant presents his lists

ideally, each entry should be capable of being manipulated separately (e.g., using post-it note)

3. A combined list (of each type) is built by the team

eliminate duplicates, add new ideas appeared during the discussions, but nothing is deleted

4. Discuss in order to produce a consensus list

shortened, lengthened, or reworded to better describe the product

5. Several smaller teams are created

each works to develop a mini-specification for one or more entries from the lists; another approach would be the creation of use cases

FAST (2)

6. Each sub-team presents its mini-specifications

additions, deletions, further elaborations may occur

new objects, services, constraints may be found during mini-specification that must be added to the initial list

7. A list of validation criteria is produced by each attendee

they are discussed in order to produce a consensus list of criteria

8. Some attendees are assigned to write the complete draft specification produced during FAST

3

Use Cases

Definition

Capture
functional requirements
by describing the users interactions with the system

They are a view of a system behavior as it appears to the outside users

Concepts

Actor

A role played by a user with respect to the system

Same actor
different persons

ophir.wordpress.com/2009/05/

Concepts

Actor

A **role** played by a user
with respect to the system

Same person

different actors

ophir.wordpress.com/2009/05/

Concepts

Actor

A **role** played by a user
with respect to the system

External to the system

Concepts

Actor

A **role** played by a user
with respect to the system

Interacts with
the system

http://histechreport.com/

Concepts

Actor

A **role** played by a user
with respect to the system

It can be an **external**
system

http://www.socialsignal.com

Concepts

Actor

A **role** played by a user
with respect to the system

Has an **unique name** (usually a noun)

Primary actor
interacts to achieve the required system function

Secondary actor
supports the system so that the primary actors can do their work

Concepts

It can be expressed in a narrative way and are **close** to user stories & features

Scenario

A **sequence of interactions**
between an **actor** and the **system**

Customer

1. Customer selects the books
2. System asks for delivery address
3. Customer provides this information
4. Systems asks for credit card data
5. Customer gives this information
6. System validates the credit card via VISA services
7. System informs delivery department

Online Book Store

Concepts

A use case is **NOT** an interaction step !!!

Use Case

set of scenarios

tied together by a common

goal of an actor

Use case description

Buy books ← the use case name, usually verb+noun

Primary actor: Customer

Secondary actors: Delivery department, VISA validation system

...

Main success scenario

1. Customer selects the books
2. System asks for delivery address
3. Customer provides this information
4. System asks for credit card data
5. Customer gives this information
6. System validates the credit card via VISA services
7. System informs delivery department

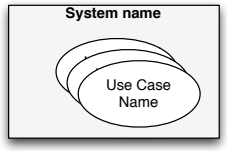
Extension

In 6 validation fails
1. Customer reenters card infos (then back to 6) or cancels

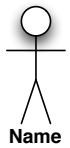
UML

Use Case Diagram

A **model** of the system capturing
its **external behavior**
and emphasizes ...

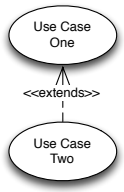


the system boundaries



the actors

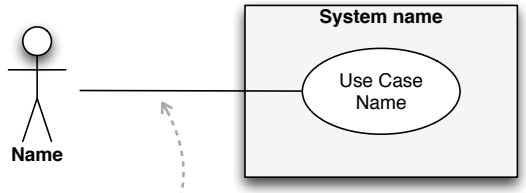
the use cases



relations between these elements

UML

Use Case Diagram
Association

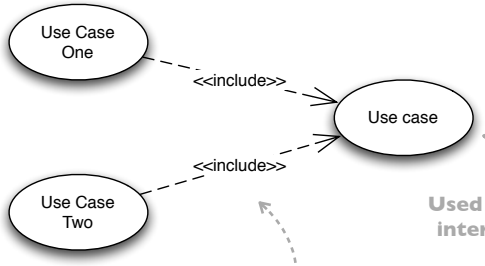


The participation of an actor in a use case i.e., instance of an actor and instance of a use case communicate with each other

UML

Do NOT use it to decompose use cases into interaction steps or for functional decomposition !!!

Use Case Diagram
Include

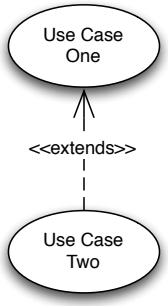


Used to factor out common interaction steps between two use cases

A relationship from a base use case to an inclusion use case, specifying that the behavior from the inclusion use case is inserted into the behavior defined for the base use case

UML

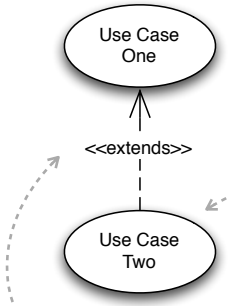
Use Case Diagram
Extends



It is not generalization!

UML

Use Case Diagram
Extends

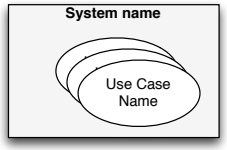


In other words, these interaction steps are inserted in the extended scenario in some conditions (e.g., special cases, extension steps)

A relation from an extension use case to a base use case specifying how the behavior for the extension use case can be inserted into the behavior defined for the base use case

Use Case Diagram

Other elements



System boundary: represents the boundaries between the the system and the actors who interact with the system

between use cases

Generalization: a taxonomic relationship between a more general use case and a more specific use case (e.g., the specific use case completely replaces one or more course of actions from the more general use case)

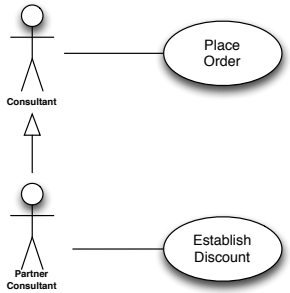
taxonomy - the classification of something (e.g., the taxonomy of these fossils), a scheme of classification (e.g., a taxonomy of smell

Macosx-Dictionary

Use Case Diagram

Other elements

generalization between actors



Classifying Use Cases

Sea-Level Use Case

interactions between a primary actor and the system

delivers something of value to the actor

Fish-Level Use Case

they exist only because they are inserted in sea-level use cases

Kite-Level Use Case

business use cases (not system use cases) show how a business/organization responds to a customer or event and how sea-level use cases fit into wider business interactions

Identifying Actors

They are NOT part of the system

they interact with the system from the outside by sending and/or receiving information from the system

Good questions

W.Ambler - Object Primer

Who obtains information from this system ?
Who provides information to the system ?
What other systems interact with this system ?
Where does the system get information ?
Who will supply, use, or remove information from the system ?

If a person fulfills a combination of roles, each role is an actor, but the combination is not

Identifying Use Cases

Satisfies a goal of an actor

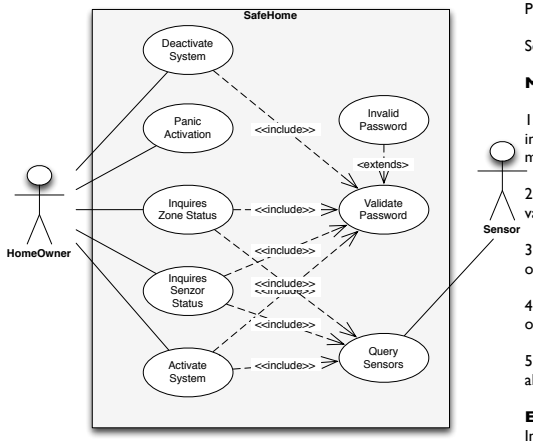
must provide something of value to an actor

Good questions W.Ambler - Object Primer

What are users in this role trying to accomplish ?
What does this actor need to be able to do ?
What are the main tasks of users in this role ?
What does this actor need to be informed of by the system ?
What does this actor need to inform the system about ?

Do NOT try to break use cases into sub-use cases and sub-sub-use cases using functional decomposition
it is a good way to loose a lot of time :)

In the *SafeHome* System (partial view)



Activate System

Primary actor: HomeOwner

Secondary actors: Sensor

Main usage scenario

1. Make system ready for activation (a not ready indicator means that a sensor is open and the owner must physically close the doors, windows, etc.)

2. User introduces the password and the system validates it

3. Selects *stay* (inside motion sensors are deactivated) or *away* (all sensors are activated)

4. The system is activated and a red alarm light can be observed by the owner

5. Continuously monitor the sensors and starts the alarm in the case a sensor is open

Extensions
In step 2 validation fails
1. the control panel beeps once and reset itself for additional input