CPS 406 — Introduction to Software Engineering

Software Requirements Engineering and Requirements Elicitation

Summer 2020

Fateme Rajabiyazdi

http://rajabiyazdi.com

fatemeh.rajabiyazdi@mail.mcgill.ca

Outline

- Software Process Model Definition
- Requirements Engineering
- Requirements Elicitation & Analysis
- Requirements Elicitation Techniques
- Types of Requirements

Software Process Model

A model for the development of software

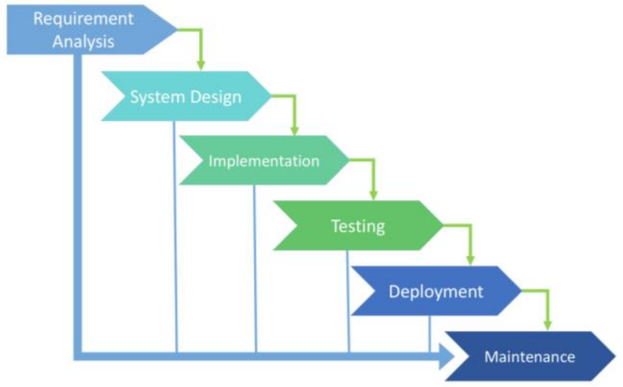
Software Process Model

A model for the development of software:

represents all the **activities** and **dependency relationships** necessary to develop a software system.

Software Process Models

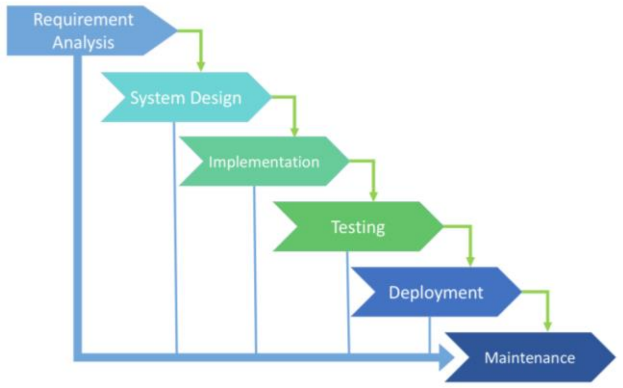
Software Process Models

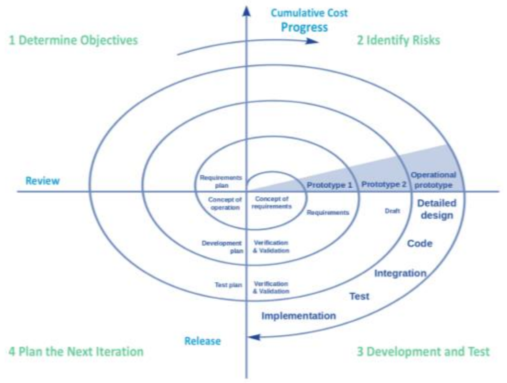


Waterfall Model

Image source: https://existek.com/

Software Process Models



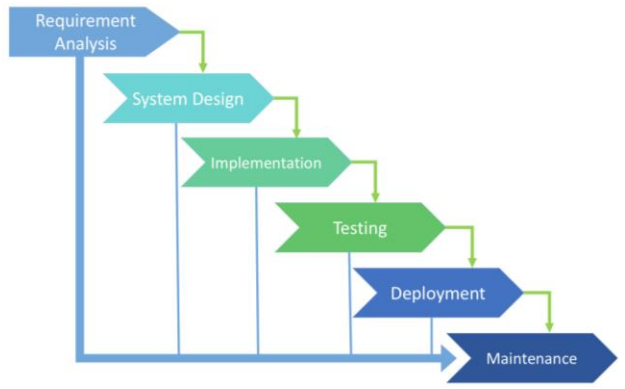


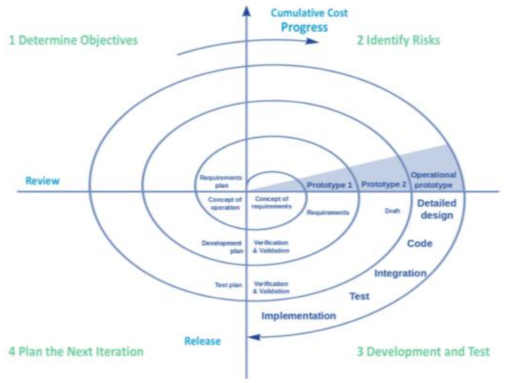
Waterfall Model

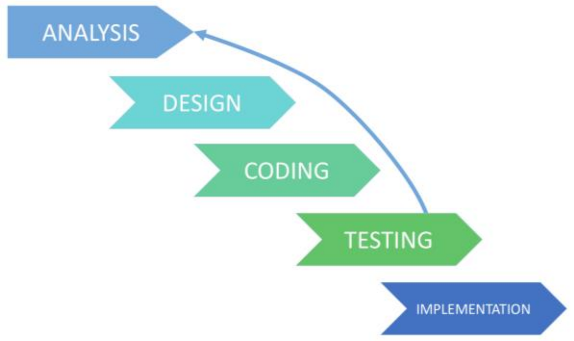
Spiral Model

Image source: https://existek.com/

Software Process Models







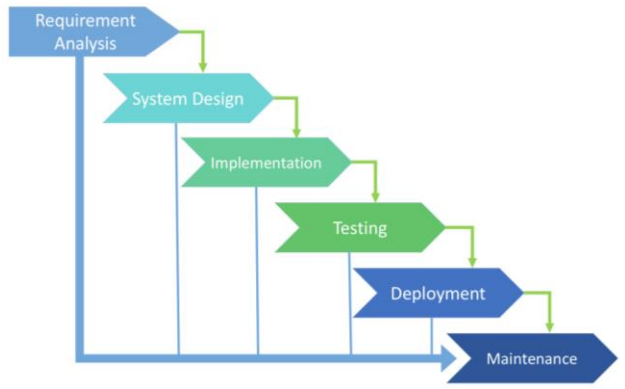
Waterfall Model

Spiral Model

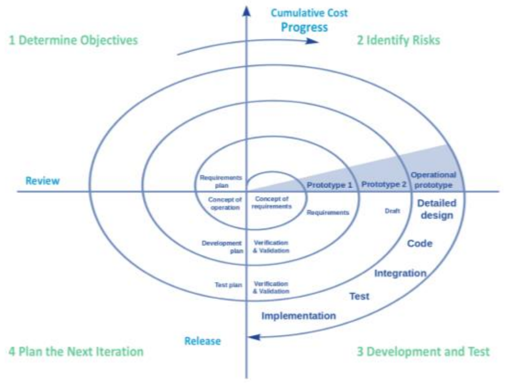
Iterative Model

Image source: https://existek.com/

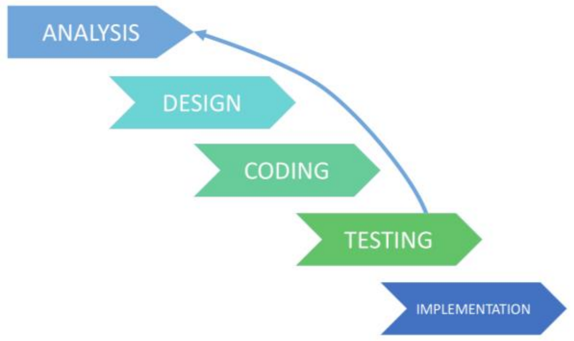
Software Process Models



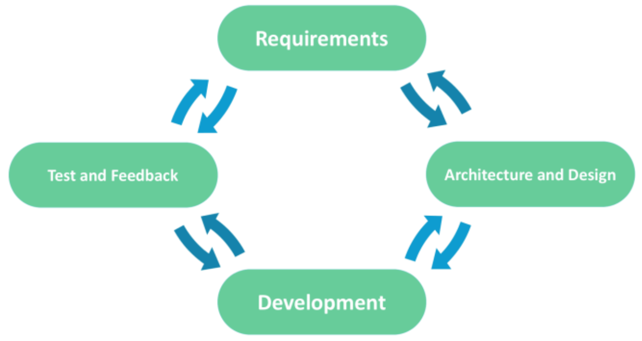
Waterfall Model



Spiral Model



Iterative Model



Agile Model

Image source: https://existek.com/

Software Development Activities

Four basic process activities of software development are:

- Specification
- Development
- Validation
- Evolution

Software Development Activities

Four basic process activities of software development are:

- **Specification = requirements engineering**
- Development
- Validation
- Evolution

Specification

Imagine that you want to build a house.

Be two-stored, have a red floor, and several windows.

http://www.sweethome3d.com/

Specification

The same applies to software development.

A **detailed vision** and **specifications** of the project is required to help software engineers, analysts and project managers create the system.

Requirements Engineering

The process of **defining the requirements** for the system under construction.

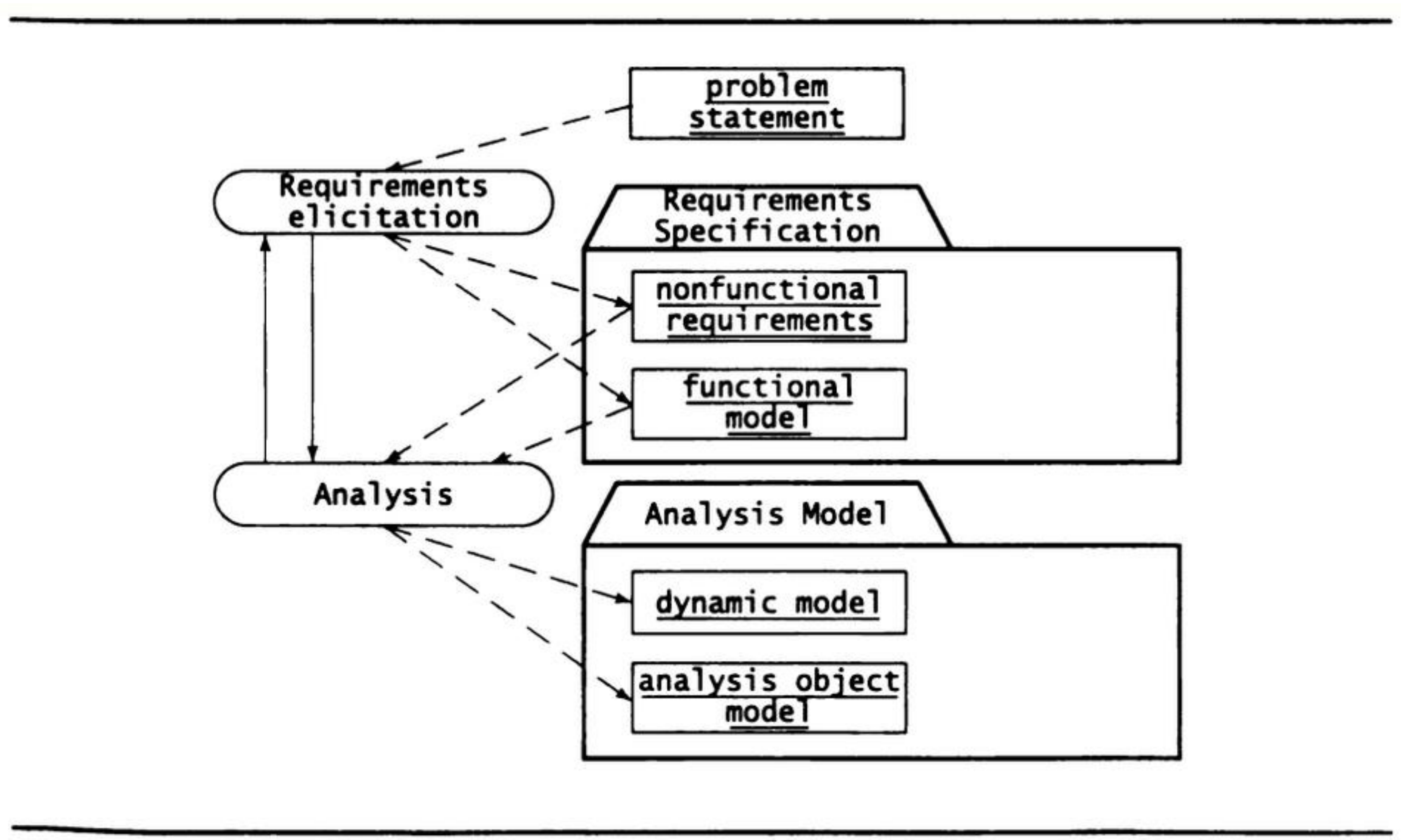
Requirements Engineering

Requirements engineering has **two main activities:**

Elicitation: results in **requirements specification** that the customer understands.

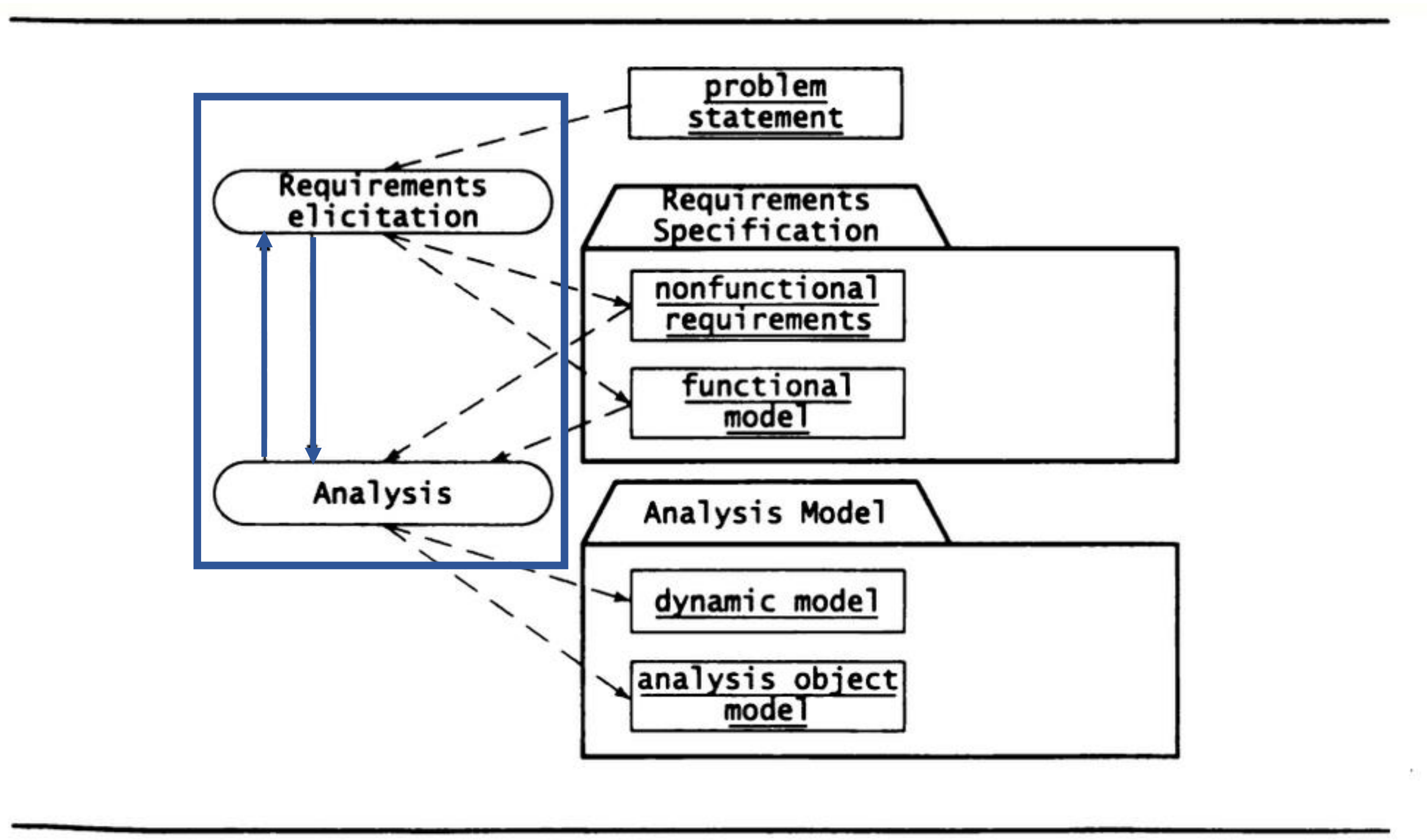
Analysis: results in **analysis model** that developer can unambiguously understand.

Requirement Elicitation and Analysis



Object-Oriented Software Engineering: Using UML, Patterns, and Java. Bernd Bruegge & Allen H Dutoit. 2009

Requirement Elicitation and Analysis



Object-Oriented Software Engineering: Using UML, Patterns, and Java. Bernd Bruegge & Allen H Dutoit. 2009

Requirements Elicitation (What)

Focuses on describing the **purpose of the system**.

Determines the coverage and **boundary of the system**.

Separates requirements according to **level of priority**.

Requirements Elicitation is Hard

People with different backgrounds must collaborate to bridge the gap:

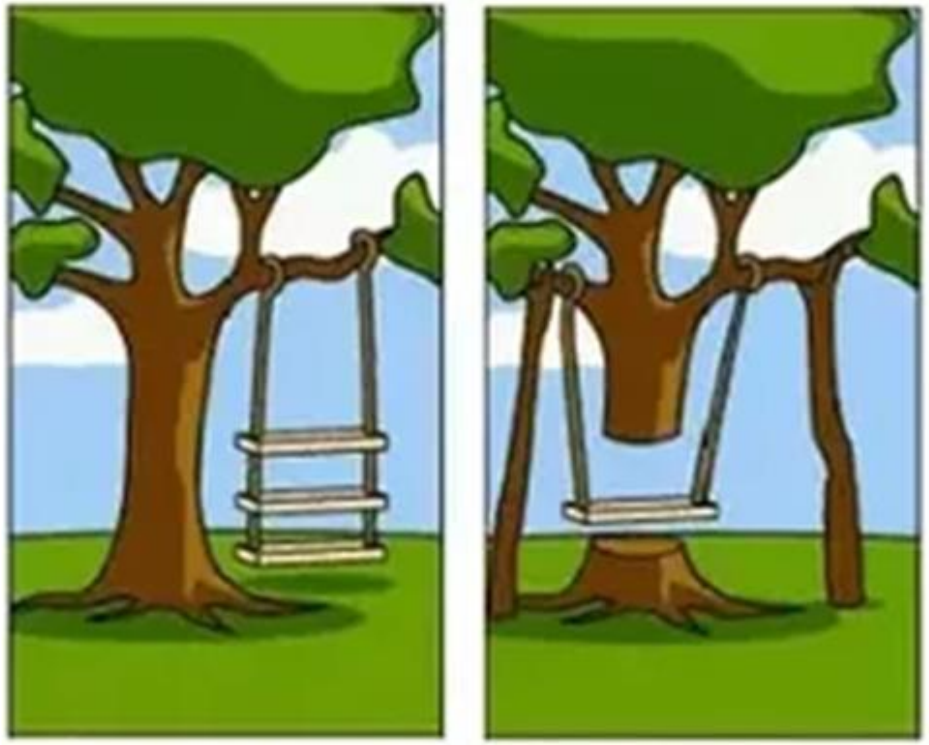
Client and end users who have **application domain knowledge**.

Software Engineers who have **solution domain knowledge**.

Requirements Elicitation is Hard

How the customer explained it.

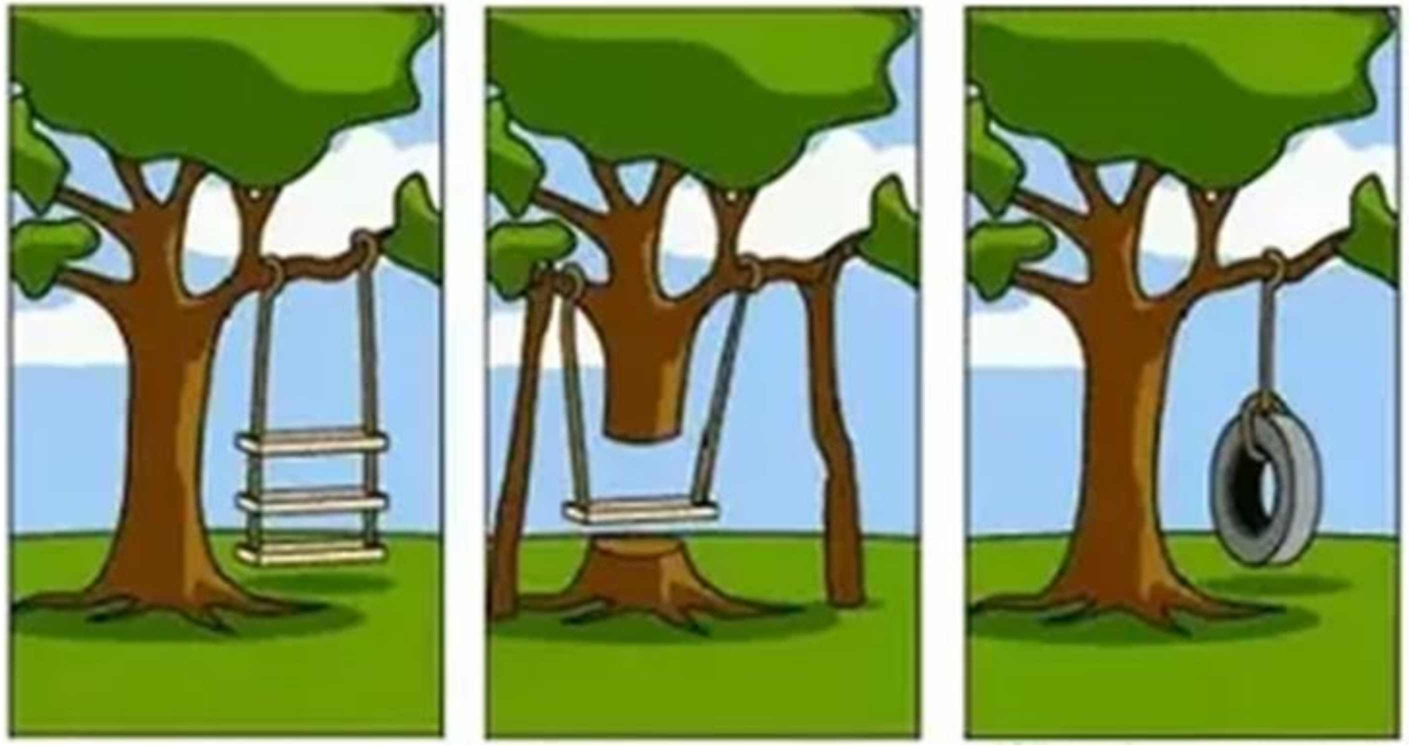
Requirements Elicitation is Hard



How the customer explained it.

How the engineer designed it.

Requirements Elicitation is Hard



How the customer explained it.

How the engineer designed it.

What the customer really needed.

Requirements Elicitation: Challenges

Requirements Elicitation: Challenges

Missing real-world scenarios

http://en.patmat.cz/

Requirements Elicitation: Challenges

Missing real-world scenarios

Leaving out unintended features

http://en.patmat.cz/

Requirements Elicitation: Challenges

Missing real-world scenarios

Leaving out unintended features

Setting ambiguous specifications

http://en.patmat.cz/

Ex. Heathrow Terminal 5 Opening 2008

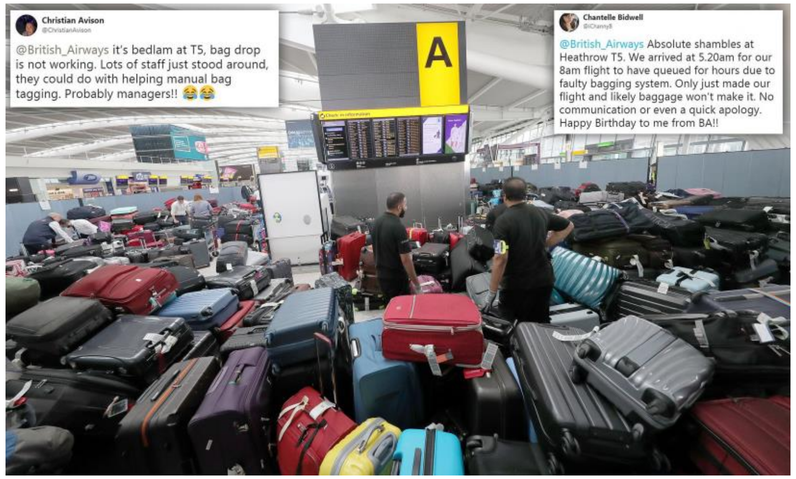
Staff tested the brand new check-in baggage handling system.

Engineers tested the system with over 12,000 test pieces of luggage.

It worked flawlessly on all test runs.

Ex. Heathrow Terminal 5 Opening 2008

The following 10 days after opening, over 42,000 bags failed to travel with their owners, and over 500 flights were cancelled.



Christian Avison
@ChristianAvison

@British_Airways it's bedlam at T5, bag drop is not working. Lots of staff just stood around, they could do with helping manual bag tagging. Probably managers!! 😂😂

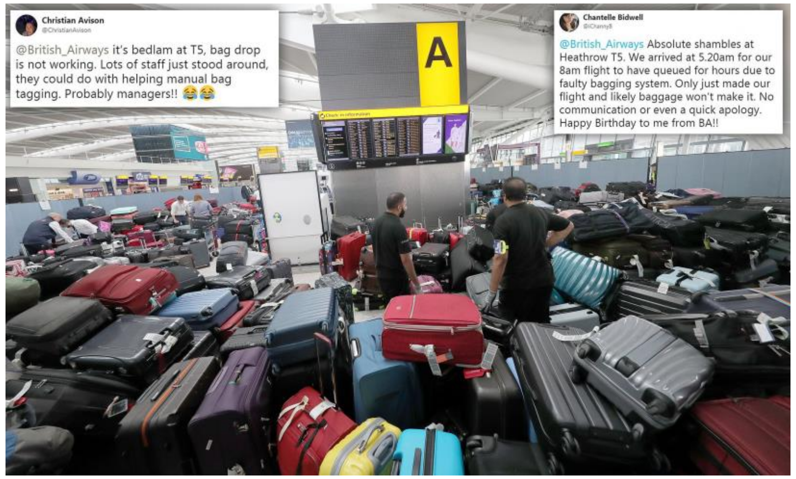
Chantelle Bidwell
@iChanny8

@British_Airways Absolute shambles at Heathrow T5. We arrived at 5.20am for our 8am flight to have queued for hours due to faulty bagging system. Only just made our flight and likely baggage won't make it. No communication or even a quick apology. Happy Birthday to me from BA!!

https://www.dailymail.co.uk

Ex. Heathrow Terminal 5 Opening 2008

The following 10 days after opening, over 42,000 bags failed to travel with their owners, and over 500 flights were cancelled.



Christian Avison
@ChristianAvison

@British_Airways it's bedlam at T5, bag drop is not working. Lots of staff just stood around, they could do with helping manual bag tagging. Probably managers!! 😂😂

Chantelle Bidwell
@iChanny8

@British_Airways Absolute shambles at Heathrow T5. We arrived at 5.20am for our 8am flight to have queued for hours due to faulty bagging system. Only just made our flight and likely baggage won't make it. No communication or even a quick apology. Happy Birthday to me from BA!!

https://www.dailymail.co.uk

Check-in staff were adding luggage to the system, which was designed to handle 12,000 bags an hour.

Baggage workers were not removing them quickly enough at the other end!

Ex. London Underground Train

London underground train leaves station without a driver!

What happened?

A passenger door was stuck and did not close.

The driver left his train to close the passenger door.

He left the driver door open!

Ex. London Underground Train

He relied on the specification that said the train does not move if at least one door is open.

When he shut the passenger door, the train left without him.

https://www.independent.co.uk

Ex. London Underground Train

He relied on the specification that said the train does not move if at least one door is open.

When he shut the passenger door, the train left without him.

https://www.independent.co.uk

The driver door was not treated as a door in the source code!

Requirements Elicitation (How)

Requirements Elicitation (How)

Interviews

Formal or informal interviews with stakeholders

Interviews

Formal or informal interviews with stakeholders:

- **Open-ended** interviews

  Exploratory

- **Closed-ended** interviews

  Prepare list of questions

Interviews

Formal or informal interviews with stakeholders

- **Open-ended** interviews

  Exploratory

- **Closed-ended** interviews

  Prepare list of questions

In practice interviews are a mixed of these!

Interviews

It is hard to elicit domain knowledge during interviews:

- **Software engineering** requirement team use **domain terminology**.
- **Stakeholders** may find it **difficult to explain** or think it is not worth mentioning.

Interviews

It is hard to elicit domain knowledge during interviews:

- **Software engineering** requirement team use **domain terminology**.
- **Stakeholders** may find it **difficult to explain** or think it is not worth mentioning.

Information from interviews can be used in supplement to other forms of requirement elicitations such as …..

Requirements Elicitation (How)

Interviews
Formal or informal interviews with stakeholders

Observation/Ethnography
Observing users and the environment

Ethnography

Observing end users in their **operational environment**.

Attempts to discover social, human, and political factors, which may also impact requirements.

Ethnography

Observing end users in their **operational environment**.

Attempts to discover social, human, and political factors, which may also impact requirements.

Often reveal critical process details that are missed by other requirements elicitation techniques.

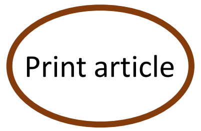
Requirements Elicitation (How)

Interviews

Formal or informal interviews with stakeholders

Observation/Ethnography

Observing users and the environment



Scenarios

Use of the system as a series of interactions

Scenarios

Describe the **use of the system** as a **series of interactions** between an end user and the system.

It usually also contains details about the work place, social situations and resource constraints.

A scenario can include text, video, pictures and story boards.

Types of Scenarios

As-is scenario

- Describes a current situation.

Visionary scenario

- Describes a future system.
- Often cannot be done by the user or developer alone.

Types of Scenarios

Evaluation scenario

- User tasks against which the system is to be evaluated.

Training scenario

- Step by step instructions that guide a novice user through a system elicitation.

How Should We Find Scenarios?

How Should We Find Scenarios?

If the **system exists**, don’t wait for information!

- What is obvious does not need to be said.

If the **system does not exist**, don’t expect the client to be verbal!

- Client understands problem domain, not the solution domain.

How Should We Find Scenarios?

If the **system exists**, don’t wait for information!

- The clients think what is obvious does not need to be said.

If the **system does not exist**, don’t expect the client to be verbal!

- Client understands problem domain, not the solution domain.

Engage in a conversation with the client!

How Should We Find Scenarios?

Engage in a dialectic approach!

- You help the client to formulate the requirements.
- The client helps you to understand the requirements.

http://www.jaburgwilk.com

Heuristics for Finding Scenarios

Ask yourself or the client the following questions:

- What are the primary tasks that the system needs to perform?
- What data will the user create, store, change, remove or add in the system?
- What external changes does the system need to know about?
- What changes or events will the users need to be informed about?

Heuristics for Finding Scenarios

Ask yourself or the client the following questions:

- What are the primary tasks that the system needs to perform?
- What data will the user create, store, change, remove or add in the system?
- What external changes does the system need to know about?
- What changes or events will the users need to be informed about?

Don’t rely on questions *and* questionnaires alone!

Heuristics for Finding Scenarios

Insist on task observation if the system already exists.

Ask to speak to the end user, not just to the client.

Expect resistance and try to overcome it.

Scenario Example: Article Printing

Initial assumption:

The user has logged on to the online library system and has located the journal containing the copy of the article.

www.youtube.com Channel: Pendleton Community Public Library

Scenario Example: Article Printing

Scenario:

The user selects the article to be downloaded. The system prompts the user to provide subscription information for the journal or indicate method of payment for the article. Payment can be made by credit card or by quoting an organizational account number. The PDF version of the article is downloaded on the user’s computer. The user is informed that it is available. The user is asked to select a printer and a copy of the article is printed.

Scenario Example: Article Printing

What can go wrong:

The user subscription information is entered incorrectly.

The payment may be rejected by the system.

The printer may be disconnected from the system.

….

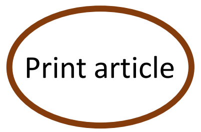
Requirements Elicitation (How)

Interviews

Formal or informal interviews with stakeholders

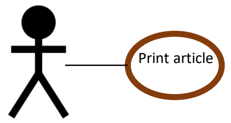
Observation/Ethnography

Observing users and the environment



Scenarios

Use of the system as a series of interactions



Use cases

Abstractions that describe a class of scenarios

Use Cases

Description of a **sequence of interactions** between a system and external **actors** to complete a given task.

Actors – any agent that interact with the system to achieve a useful goal (e.g., people, other software systems, hardware).

Use Cases

In general, a use case should cover the full sequence of steps from the beginning of a task until the end.

A use case should describe the user's interaction with the system,

not the computations the system performs.

Heuristics for Finding Use Cases

Select a narrow **vertical slice** of the system (i.e., one scenario)

- Discuss it in detail with the user to understand the user’s style of interaction.
- Discuss the chosen scenario with many users.

Select a **horizontal slice** of the system (i.e., many scenarios)

- Discuss the scope of the system with the user.

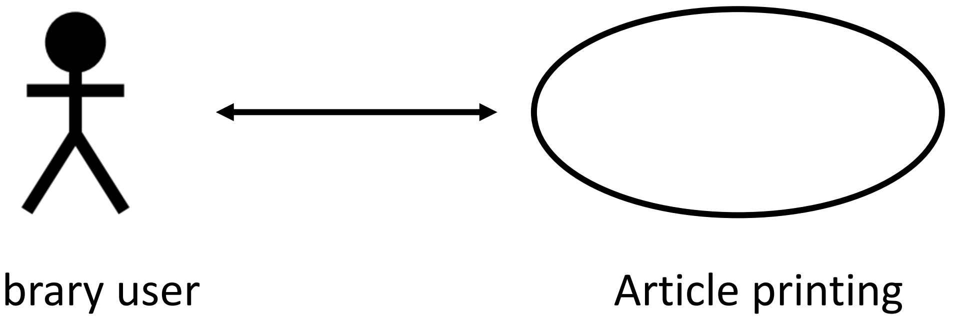
Heuristics for Finding Use Cases

Use illustrative prototypes (e.g., mock-ups) as visual support.

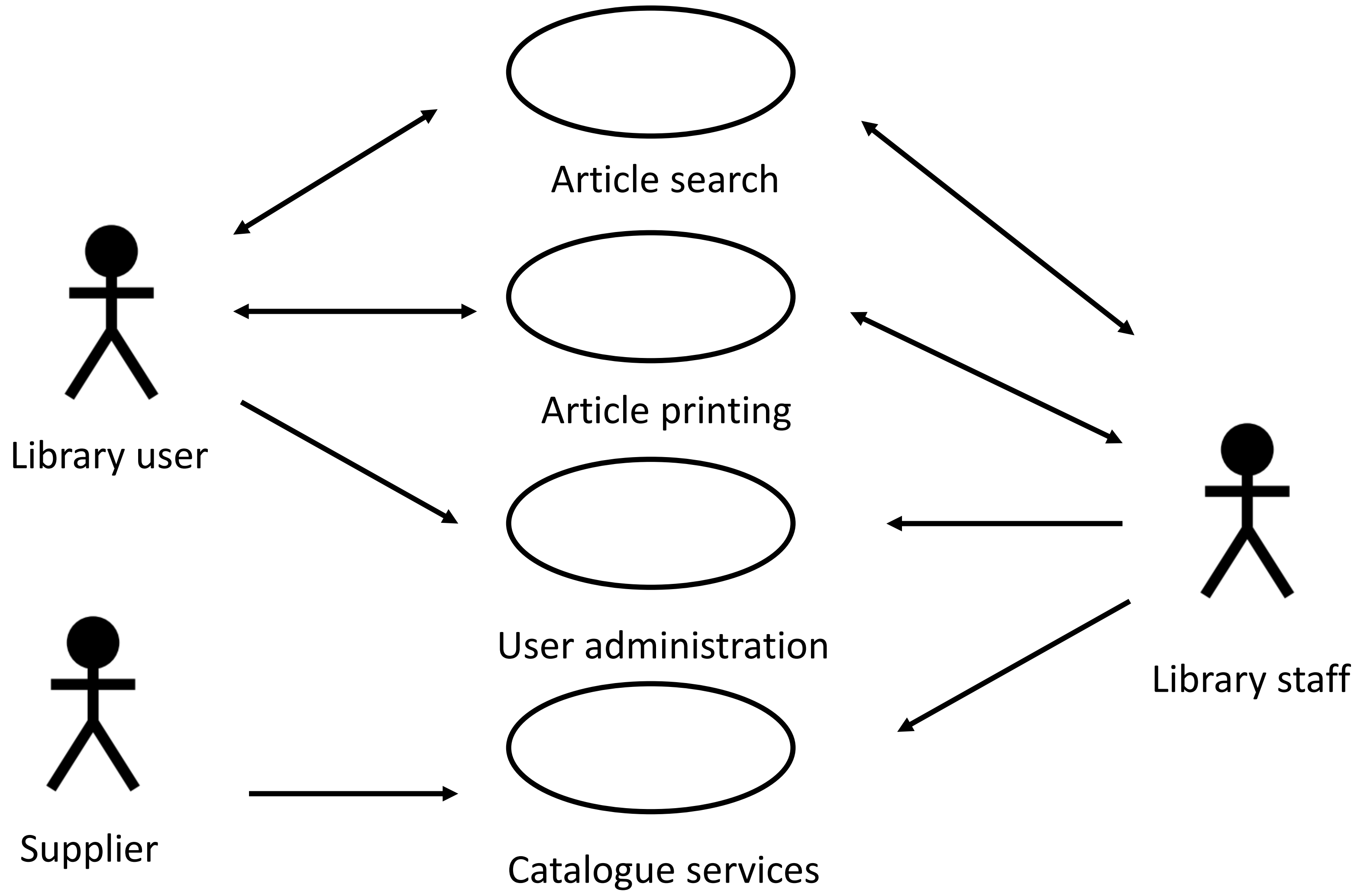
Find out what the user does via:

- Task observation
- Questionnaires

Use Case Example: Article Printing



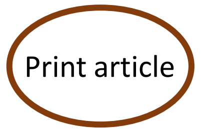
Use Case Example: Library System



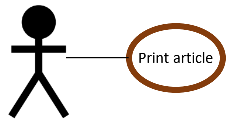
Requirements Elicitation (How)

Interviews
Formal or informal interviews with stakeholders

Observation/Ethnography
Observing users and the environment



Scenarios
Use of the system as a series of interactions



Use cases
Abstractions that describe a class of scenarios

Types of Requirements

- **Functional requirements**

- **Non-Functional requirements**

- **Domain requirements**

Types of Requirements

- **Functional requirements**

  Interactions between the system & environment independent from the implementation.

- **Non-Functional requirements**

  Aspects not directly related to functional behavior of the system.

- **Domain requirements**

  Characteristics and constraints of the domain.

Functional vs. Non-Functional

Functional requirements (What)

- Have to do with the functionality of the system, what the system does with the computation.
- Ex. Elevator shall take ppl to the floor they select.

Non-Functional requirements (How)

- Refer to system non-functional properties; system qualities, they must be objective and quantifiable.
- Ex. Elevator must take 10 floors in less than one minute!

Qualities of Requirements

- Clear, Unambiguous, Understandable
- Realistic
- Valid
- Verifiable
- Consistent
- Complete

Activity

Let’s look at a few non-functional requirement examples!

Cast your vote:

Good! Unclear! Not measurable!

Activity

Link: www.PollEv.com

Enter this: **fatemerajabi419**

The iOS application must support iPhone devices.

- Good
- Unclear
- Not measurable

Total Results: 0

The website must process a customer order in <100 ms.

Good

Unclear

Not measurable

Total Results: 0

The system will crash no more than once per 10000 transactions.

Good

Unclear

Not measurable

Total Results: 0

All text must be pink.

- Good
- Unclear
- Not measurable

Total Results: 0

NF Requirements Specifying Metrics

Property	Measure
Speed	
Size	
Ease of use	
Reliability	
Robustness	
Portability	

NF Requirements Specifying Metrics

Property	Measure
Speed	Process transactions per second/Event response time/Screen refresh time
Size	
Ease of use	
Reliability	
Robustness	
Portability	

NF Requirements Specifying Metrics

Property	Measure
Speed	Process transactions per second/Event response time/Screen refresh time
Size	K Bytes/Number of RAM chips
Ease of use	
Reliability	
Robustness	
Portability	

NF Requirements Specifying Metrics

Property	Measure
Speed	Process transactions per second/Event response time/Screen refresh time
Size	K Bytes/Number of RAM chips
Ease of use	Training time/Number of help frames/User error rate
Reliability	
Robustness	
Portability	

NF Requirements Specifying Metrics

Property	Measure
Speed	Process transactions per second/Event response time/Screen refresh time
Size	K Bytes/Number of RAM chips
Ease of use	Training time/Number of help frames/User error rate
Reliability	Mean time between failure/Probability of unavailability/Rate of failure occurrence
Robustness	
Portability	

NF Requirements Specifying Metrics

Property	Measure
Speed	Process transactions per second/Event response time/Screen refresh time
Size	K Bytes/Number of RAM chips
Ease of use	Training time/Number of help frames/User error rate
Reliability	Mean time between failure/Probability of unavailability/Rate of failure occurrence
Robustness	Time to restart after failure/Percentage of events causing failure
Portability	

NF Requirements Specifying Metrics

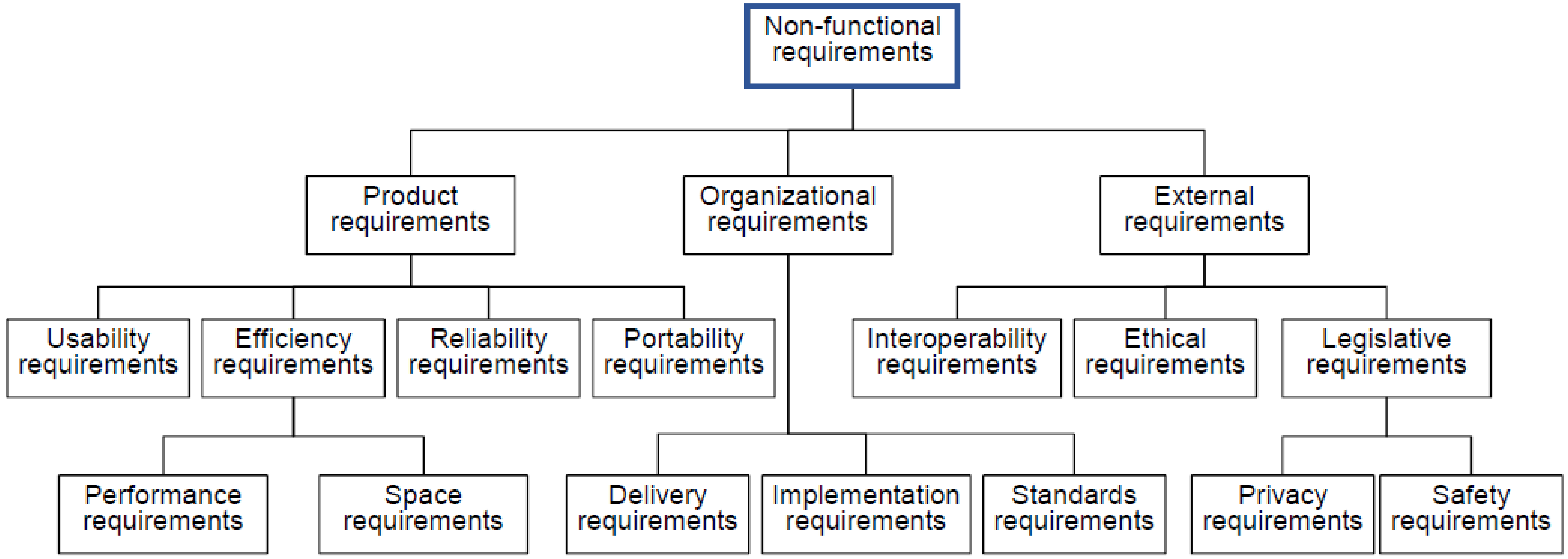
Property	Measure
Speed	Process transactions per second/Event response time/Screen refresh time
Size	K Bytes/Number of RAM chips
Ease of use	Training time/Number of help frames/User error rate
Reliability	Mean time between failure/Probability of unavailability/Rate of failure occurrence
Robustness	Time to restart after failure/Percentage of events causing failure
Portability	Percentage of target dependent statements/Number of target systems

NF Requirements Specifying Metrics

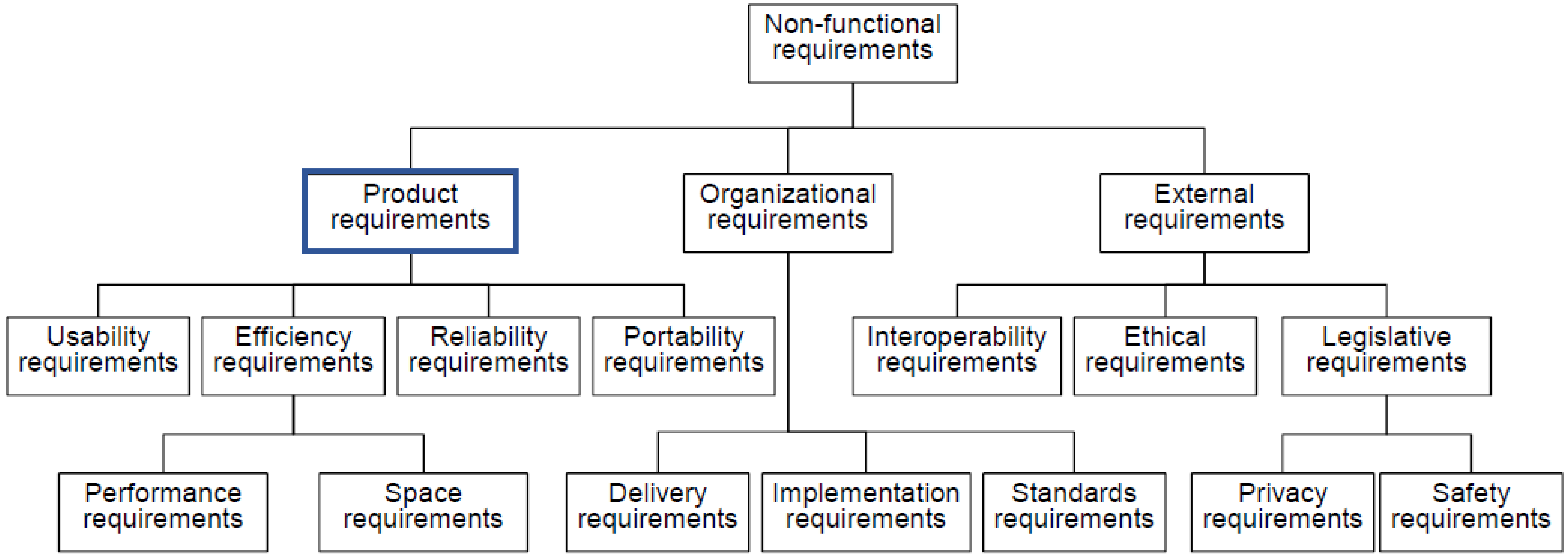
Property	Measure
Speed	Process transactions per second/Event response time/Screen refresh time
Size	K Bytes/Number of RAM chips
Ease of use	Training time/Number of help frames/User error rate
Reliability	Mean time between failure/Probability of unavailability/Rate of failure occurrence
Robustness	Time to restart after failure/Percentage of events causing failure
Portability	Percentage of target dependent statements/Number of target systems

Notice how each metric is a quantifiable amount, a number to be verified!

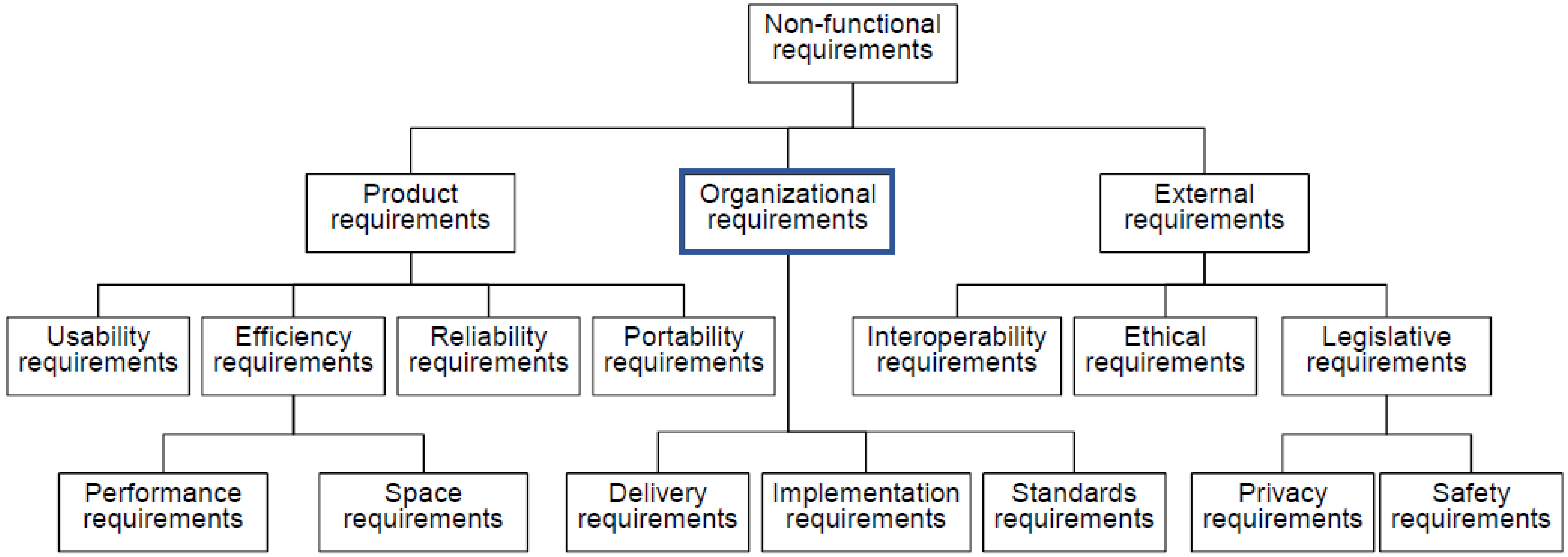
Classification of NF Requirements



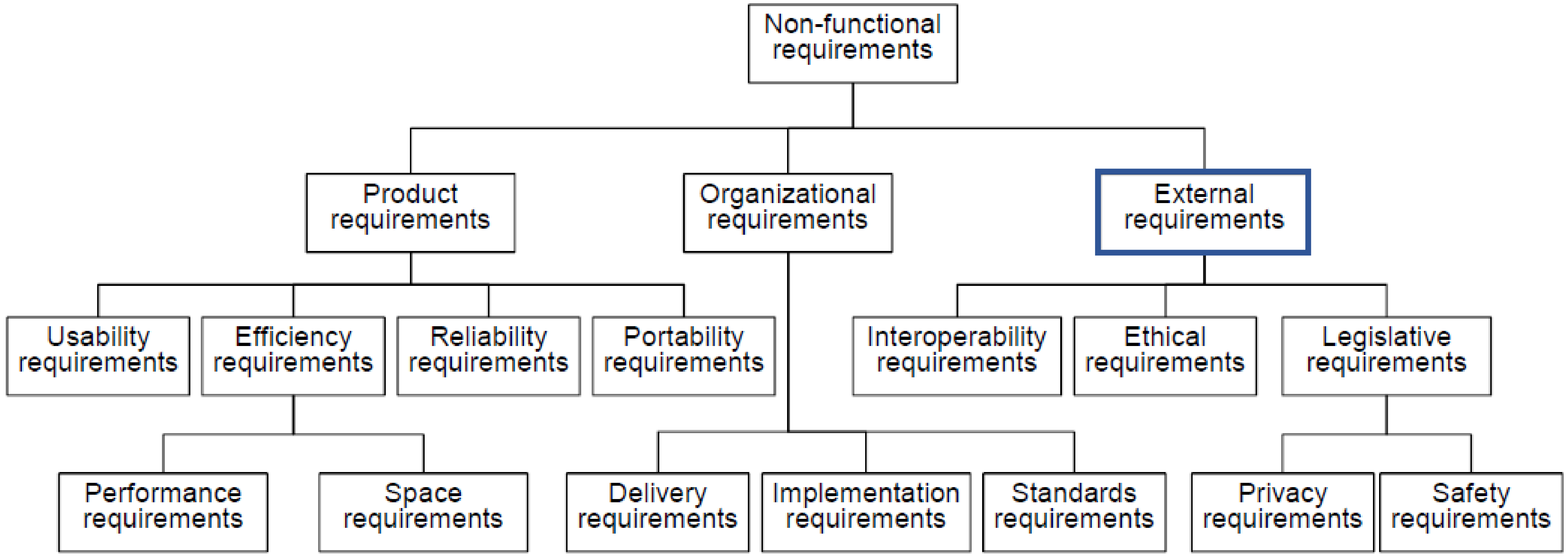
Classification of NF Requirements



Classification of NF Requirements



Classification of NF Requirements



Summary

Software process model represents all the activities and dependency relationships necessary to develop a software system.

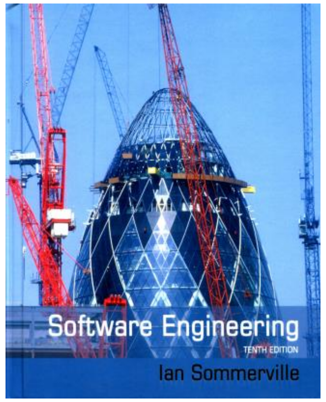
Requirements engineering is the process of defining the system requirements.

Requirements elicitation and analysis are two main activities of requirement engineering.

Requirements elicitation techniques include interviews, observations, scenarios, and use cases.

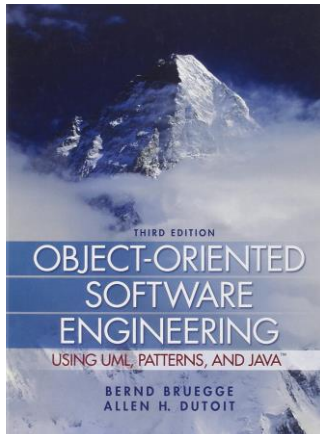
Types of requirements are functional, non-functional, and domain requirements.

Further Readings



Software engineering. Ian Sommerville. 2015

- Chapter 4. Software Processes
- Chapter 6. Software Requirements
- Chapter 7. Requirement Engineering Processes


THIRD EDITION
OBJECT-ORIENTED SOFTWARE ENGINEERING
USING UML, PATTERNS, AND JAVA
BERND BRUEGGE
ALLEN H. DUTOIT

Object-oriented software engineering: using UML, Patterns and Java. Bernd Bruegge & Allen H Dutoit. 2009

- Chapter 4. Requirements Elicitation

CPS 406 — Introduction to Software Engineering

Software Requirements Engineering and Requirements Elicitation

Summer 2020

Fateme Rajabiyazdi

http://rajabiyazdi.com

fatemeh.rajabiyazdi@mail.mcgill.ca