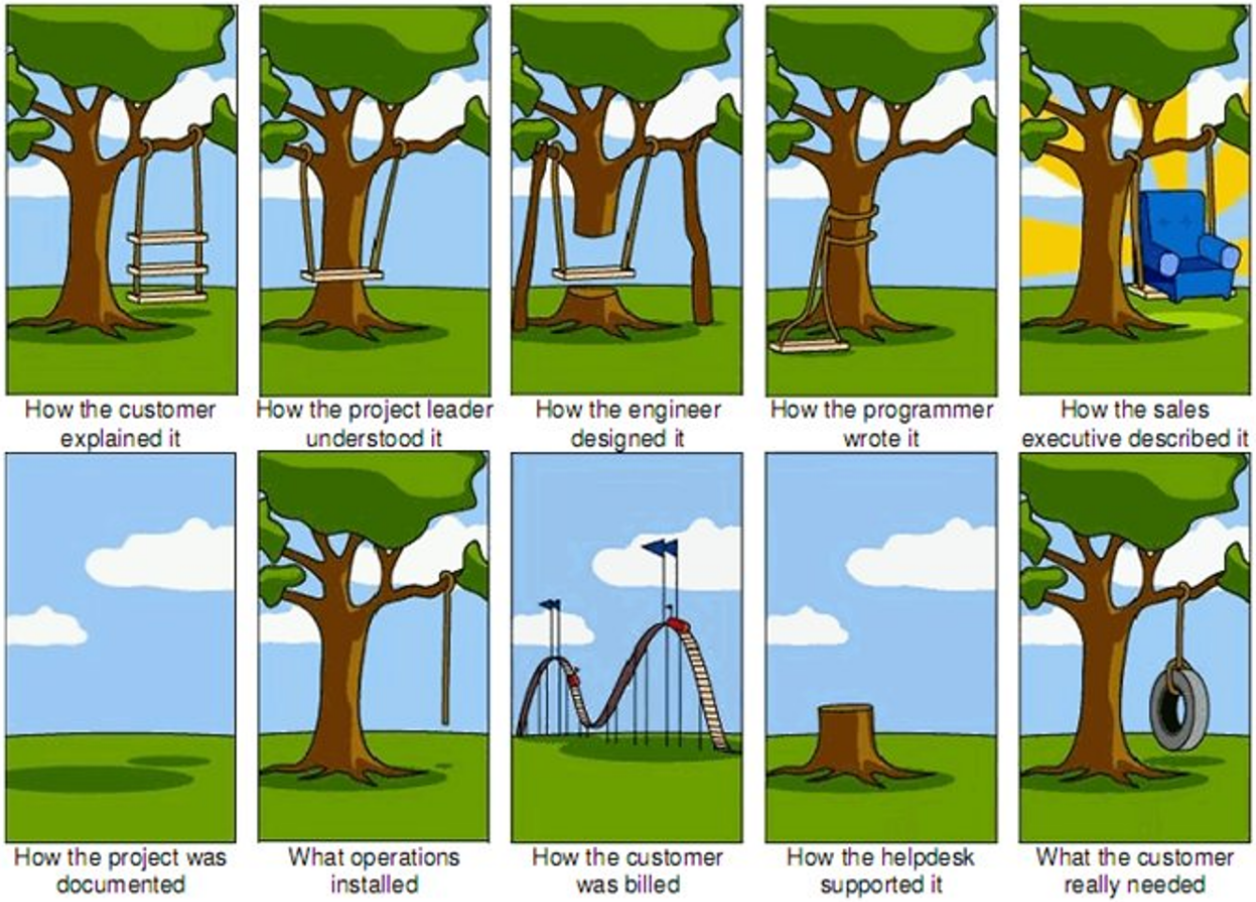
How to Make a Tree Swing?



Picture from `projectcartoon.com`

Introduction to Requirements Engineering

Matthieu Vergne

Nara Institute of Science and Technology

March 1st, 2017

Seminar Objectives

- Understand what is a requirement

Seminar Objectives

- Understand what is a requirement
- Understand what is Requirements Engineering (RE)

Seminar Objectives

- Understand what is a requirement
- Understand what is Requirements Engineering (RE)
- Understand how RE impacts/is impacted by software projects

Seminar Objectives

- Understand what is a requirement
- Understand what is Requirements Engineering (RE)
- Understand how RE impacts/is impacted by software projects
- Get a broad overview of existing RE techniques

Seminar Objectives

- Understand what is a requirement
- Understand what is Requirements Engineering (RE)
- Understand how RE impacts/is impacted by software projects
- Get a broad overview of existing RE techniques

This course is inspired from:

- RE course of Anna Perini, University of Trento, Italy (2014)
- Guide to the Software Engineering Body of Knowledge v3 (Bourque et al. [2014])

Seminar Objectives

- Understand what is a requirement
- Understand what is Requirements Engineering (RE)
- Understand how RE impacts/is impacted by software projects
- Get a broad overview of existing RE techniques

This course is inspired from:

- RE course of Anna Perini, University of Trento, Italy (2014)
- Guide to the Software Engineering Body of Knowledge v3 (Bourque et al. [2014])

Slides available on the Web:
`https://www.matthieu-vergne.fr/?page=teaching`

Outline

Outline

Definitions

IEEE Standards Board [1990]:

1. A condition or capacity needed by a user to solve a problem or achieve an objective.

Definitions

IEEE Standards Board [1990]:

1. A condition or capacity needed by a user to solve a problem or achieve an objective.
2. A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents.

Definitions

IEEE Standards Board [1990]:

1. A condition or capacity needed by a user to solve a problem or achieve an objective.
2. A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents.
3. A documented representation of a condition or capability as in (1) or (2).

Definitions

IEEE Standards Board [1990]:

1. A condition or capacity needed by a user to solve a problem or achieve an objective.
2. A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents.
3. A documented representation of a condition or capability as in (1) or (2).

SWEBOK v3 [2014]:

> *A software requirement is a property that must be exhibited by something in order to solve some problem in the real world.*

Definitions

IEEE Standards Board [1990]:

1. A condition or capacity needed by a user to solve a problem or achieve an objective.
2. A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents.
3. A documented representation of a condition or capability as in (1) or (2).

SWEBOK v3 [2014]:

> *A software requirement is a property that must be exhibited by something in order to solve some problem in the real world. (user? document?)*

Classifications

Various kinds of requirements:

Product Need or constraint on the product to be developed.

- *The software shall verify that a student meets all pre-requisites before he or she registers for a course.*

Classifications

Various kinds of requirements:

Product Need or constraint on the product to be developed.

- *The software shall verify that a student meets all pre-requisites before he or she registers for a course.*

Process Constraint on the development of the product.

- *The software shall be developed using Agile methods.*

Classifications

Various kinds of requirements:

Product Need or constraint on the product to be developed.

- *The software shall verify that a student meets all pre-requisites before he or she registers for a course.*

Process Constraint on the development of the product.

- *The software shall be developed using Agile methods.*

Functional Functions that the product is to execute.

- *formatting some text, modulating a signal*

Classifications

Various kinds of requirements:

Product Need or constraint on the product to be developed.

- *The software shall verify that a student meets all pre-requisites before he or she registers for a course.*

Process Constraint on the development of the product.

- *The software shall be developed using Agile methods.*

Functional Functions that the product is to execute.

- *formatting some text, modulating a signal*

Quality Constrain the solution.

- *performance, reliability, safety, security, maintainability, etc.*

Classifications

Various kinds of requirements:

Product Need or constraint on the product to be developed.

- *The software shall verify that a student meets all pre-requisites before he or she registers for a course.*

Process Constraint on the development of the product.

- *The software shall be developed using Agile methods.*

Functional Functions that the product is to execute.

- *formatting some text, modulating a signal*

Quality Constrain the solution.

- *performance, reliability, safety, security, maintainability, etc.*

System Requirements for the whole system.

- *Software + hardware + people + information + etc.*

Classifications

Various kinds of requirements:

Product Need or constraint on the product to be developed.

- *The software shall verify that a student meets all pre-requisites before he or she registers for a course.*

Process Constraint on the development of the product.

- *The software shall be developed using Agile methods.*

Functional Functions that the product is to execute.

- *formatting some text, modulating a signal*

Quality Constrain the solution.

- *performance, reliability, safety, security, maintainability, etc.*

System Requirements for the whole system.

- *Software + hardware + people + information + etc.*

Software Reqs for the software part, derived from the system reqs.

Classifications

Various kinds of requirements:

Product Need or constraint on the product to be developed.

- *The software shall verify that a student meets all pre-requisites before he or she registers for a course.*

Process Constraint on the development of the product.

- *The software shall be developed using Agile methods.*

Functional Functions that the product is to execute.

- *formatting some text, modulating a signal*

Quality Constrain the solution.

- *performance, reliability, safety, security, maintainability, etc.*

System Requirements for the whole system.

- *Software + hardware + people + information + etc.*

Software Reqs for the software part, derived from the system reqs.

Etc. ...

Classifications Usage

Requirements classifications are:

- Supports for identifying relevant requirements.

Classifications Usage

Requirements classifications are:

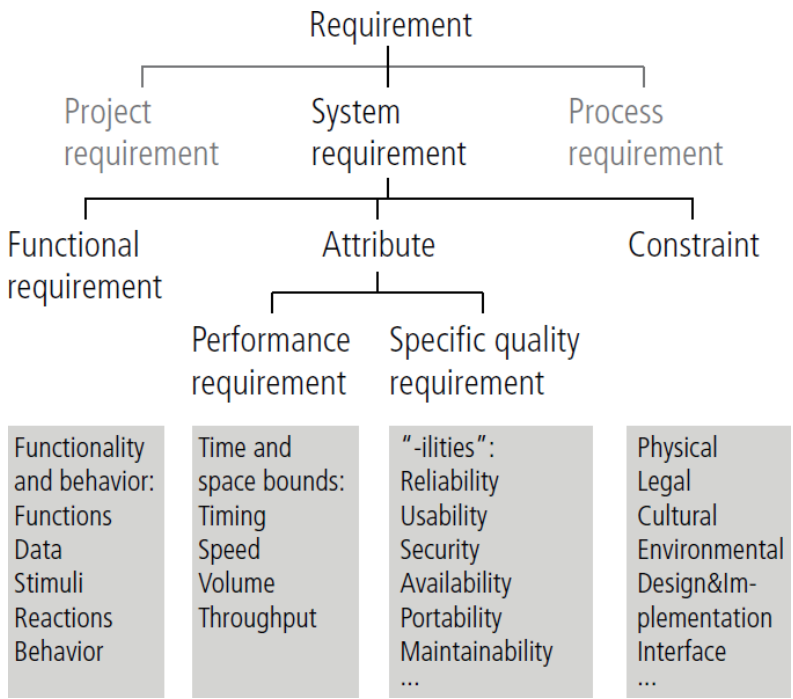
- Supports for identifying relevant requirements.
- Not strict rules to be followed.

Classifications Usage

Requirements classifications are:

- Supports for identifying relevant requirements.
- Not strict rules to be followed.

An example:
(Glinz [2007])



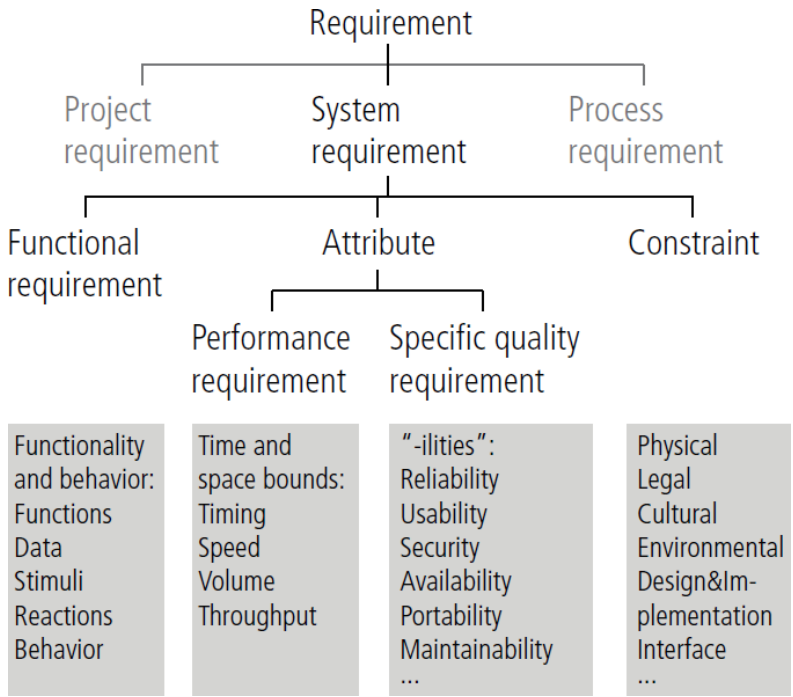
Classifications Usage

Requirements classifications are:

- Supports for identifying relevant requirements.
- Not strict rules to be followed.

An example:
(Glinz [2007])

All do not apply to every projects.



Various Dimensions to Consider

- What is needed (requirement *per se*)
  - *Register participants for an event*

Various Dimensions to Consider

- What is needed (requirement *per se*)
  - *Register participants for an event*
- Who needs it
  - *Event organisers*

Various Dimensions to Consider

- What is needed (requirement *per se*)
  - *Register participants for an event*
- Who needs it
  - *Event organisers*
- Why it is needed
  - *Identify participants*
  - *Store contact information*
  - *Measure event interest*
  - *Evaluate food requirements*

Various Dimensions to Consider

- What is needed (requirement *per se*)
  - *Register participants for an event*
- Who needs it
  - *Event organisers*
- Why it is needed
  - *Identify participants*
  - *Store contact information*
  - *Measure event interest*
  - *Evaluate food requirements*
- How it is fulfilled
  - *Registration form on a website*
  - *Registration phone call*
  - *On-site registration desk*

Many Challenges

- What is needed (requirement *per se*)
  - *Could be ambiguous ("track" instead of "register")*
  - *Could be subjective (no reliable test procedure)*

Many Challenges

- What is needed (requirement *per se*)
  - *Could be ambiguous ("track" instead of "register")*
  - *Could be subjective (no reliable test procedure)*
- Who needs it
  - *Could focus on irrelevant people (not directly involved)*
  - *Could forget important people (some organisers)*
  - *Could forget authorities (regulators, law)*

Many Challenges

- What is needed (requirement *per se*)
  - *Could be ambiguous ("track" instead of "register")*
  - *Could be subjective (no reliable test procedure)*
- Who needs it
  - *Could focus on irrelevant people (not directly involved)*
  - *Could forget important people (some organisers)*
  - *Could forget authorities (regulators, law)*
- Why it is needed
  - *Could be unknown (no access to the decision team)*
  - *Could be wrong (failed guess)*
  - *Could be conflictual (disagreements)*

Many Challenges

- What is needed (requirement *per se*)
  - *Could be ambiguous ("track" instead of "register")*
  - *Could be subjective (no reliable test procedure)*
- Who needs it
  - *Could focus on irrelevant people (not directly involved)*
  - *Could forget important people (some organisers)*
  - *Could forget authorities (regulators, law)*
- Why it is needed
  - *Could be unknown (no access to the decision team)*
  - *Could be wrong (failed guess)*
  - *Could be conflictual (disagreements)*
- How it is fulfilled
  - *Could be unknown (lack expertise)*
  - *Could be inadapted (lack alternatives)*
  - *Could be infeasible (conflicts, law restrictions)*

Outline

What is RE?

Several detailed definitions depending on emphasis, but in short:

What is RE?

Several detailed definitions depending on emphasis, but in short:

Requirements Engineering

Systematic handling of requirements. (SWEBOK v3 [2014])

What is RE?

Several detailed definitions depending on emphasis, but in short:

Requirements Engineering

Systematic handling of requirements. (SWEBOK v3 [2014])

More precisely, we can speak about systematic methods to discover, model, improve, and exploit requirements.

RE in Software Projects - In Theory

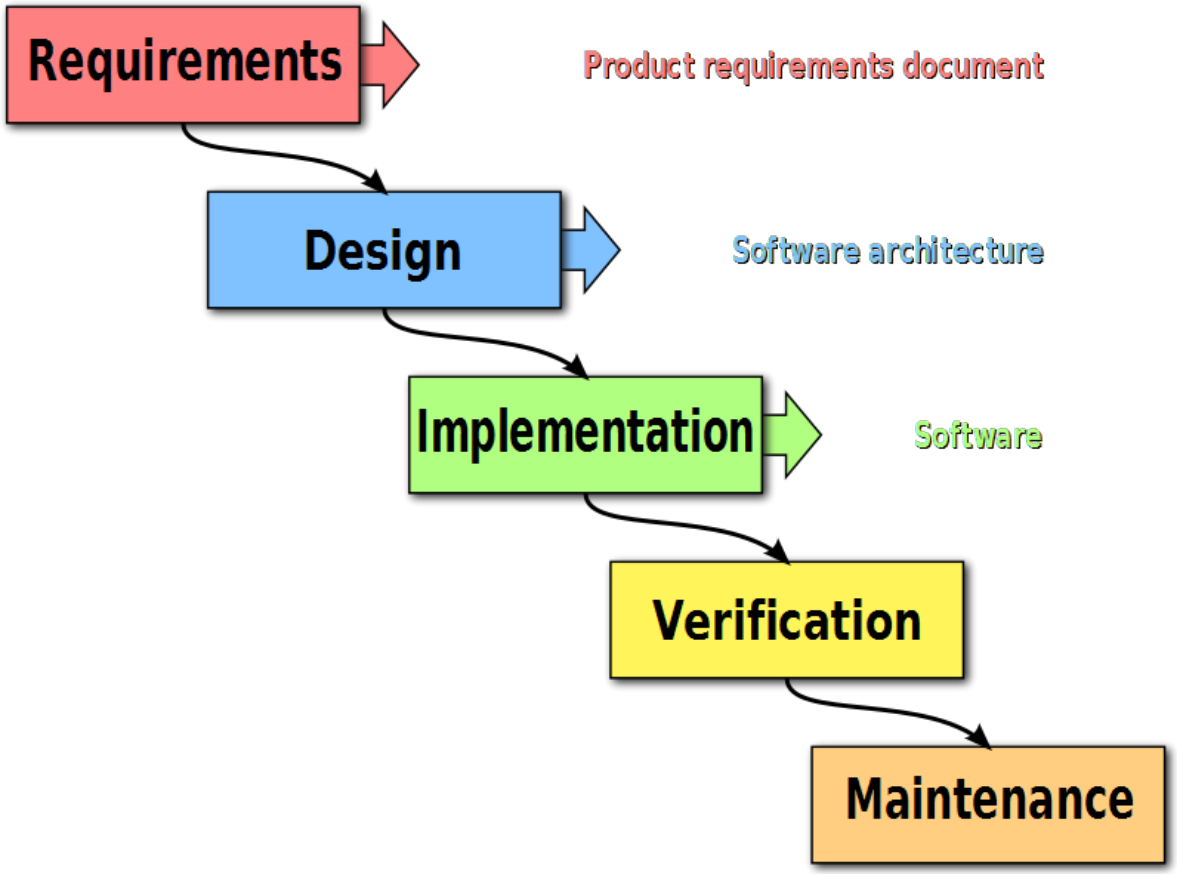
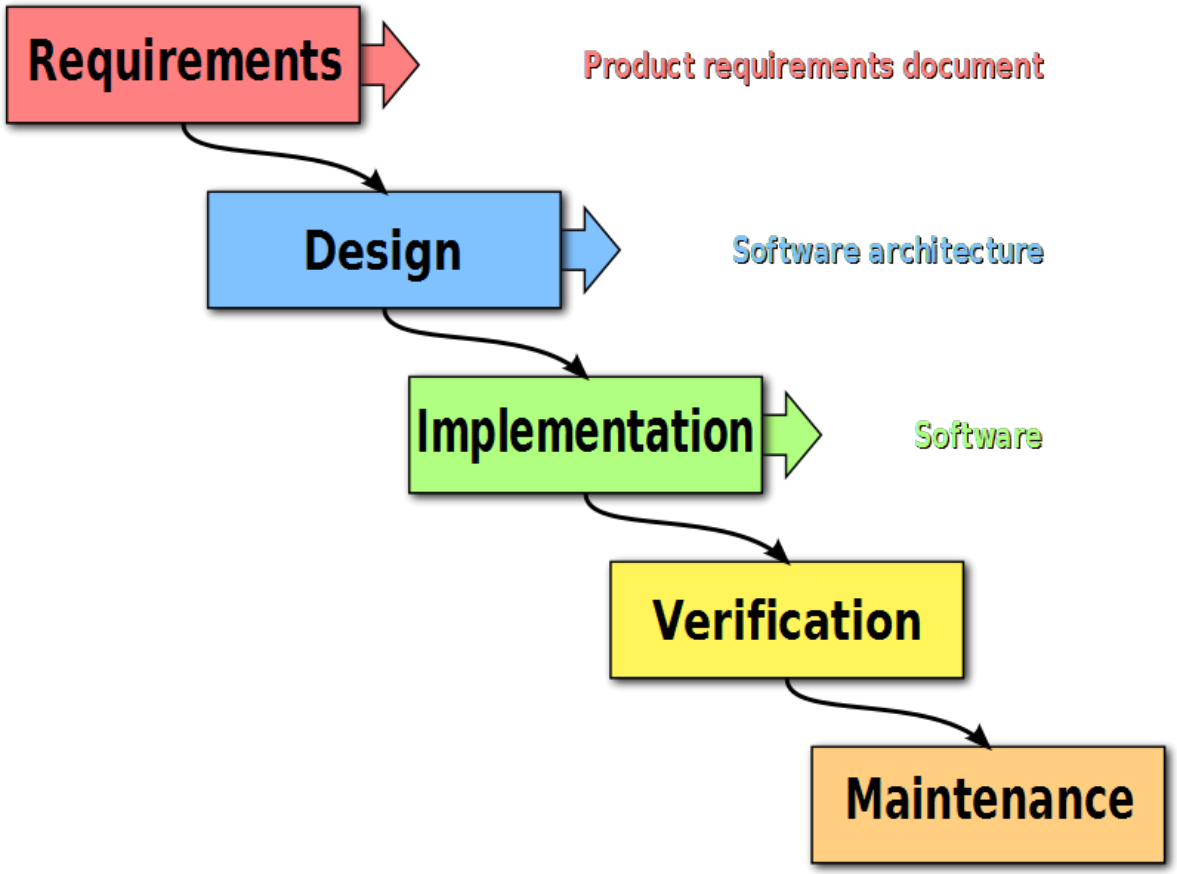


Diagram by Peter Kemp & Paul Smith (Wikimedia).

RE in Software Projects - In Theory



After first step, requirements don't change anymore. The rest of the work is about satisfying them as best as possible.

Diagram by Peter Kemp & Paul Smith (Wikimedia).

RE in Software Projects - Wake up!

Assumptions:

- All requirements are known after the first phase

RE in Software Projects - Wake up!

Assumptions:

- All requirements are known after the first phase
- Requirements are perfectly documented

RE in Software Projects - Wake up!

Assumptions:

- All requirements are known after the first phase
- Requirements are perfectly documented
- If tests do not pass, the developer is at fault

RE in Software Projects - Wake up!

Assumptions:

- All requirements are known after the first phase
- Requirements are perfectly documented
- If tests do not pass, the developer is at fault

Reality:

RE in Software Projects - Wake up!

Assumptions:

- All requirements are known after the first phase
- Requirements are perfectly documented
- If tests do not pass, the developer is at fault

Reality:

- New requirements discovered during implementation/verification

RE in Software Projects - Wake up!

Assumptions:

- All requirements are known after the first phase
- Requirements are perfectly documented
- If tests do not pass, the developer is at fault

Reality:

- New requirements discovered during implementation/verification
- All requirements cannot be clarified immediately

RE in Software Projects - Wake up!

Assumptions:

- All requirements are known after the first phase
- Requirements are perfectly documented
- If tests do not pass, the developer is at fault

Reality:

- New requirements discovered during implementation/verification
- All requirements cannot be clarified immediately
- Changes in requirements impact the rest of the chain

RE in Software Projects - Wake up!

Assumptions:

- All requirements are known after the first phase
- Requirements are perfectly documented
- If tests do not pass, the developer is at fault

Reality:

- New requirements discovered during implementation/verification
- All requirements cannot be clarified immediately
- Changes in requirements impact the rest of the chain
- Later the change, higher the cost

RE in Software Projects - In Practice

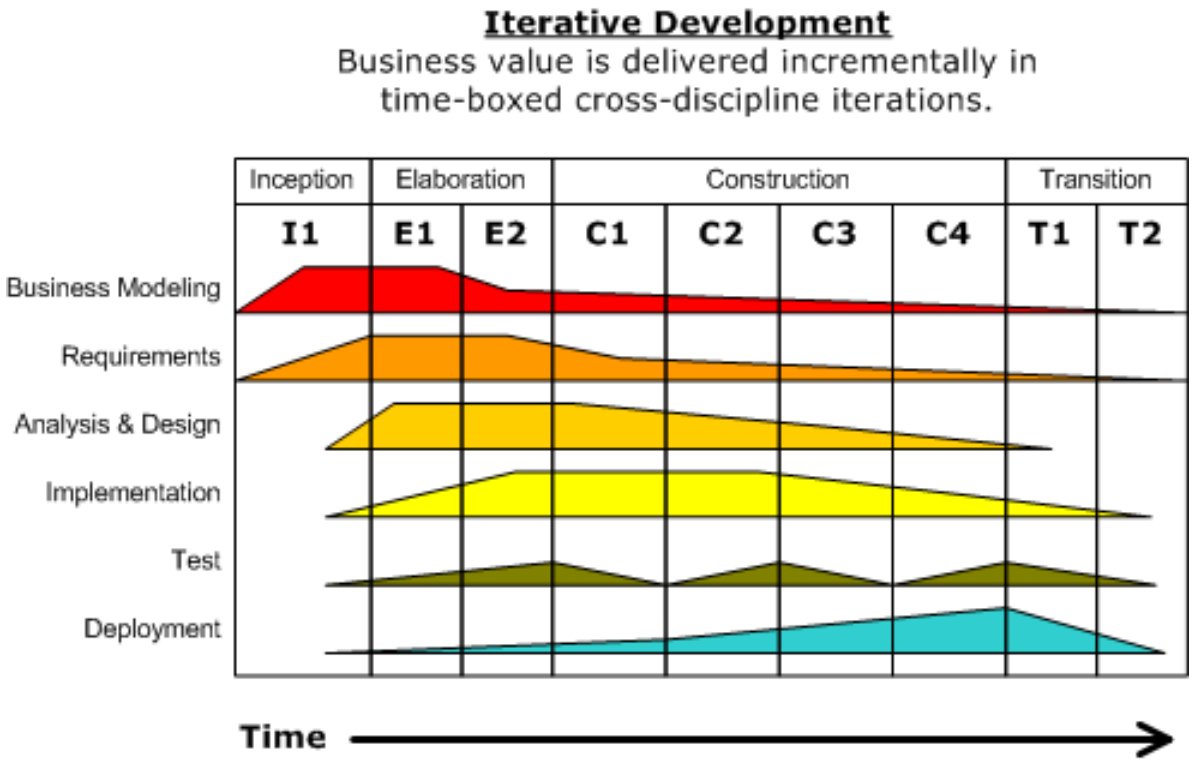
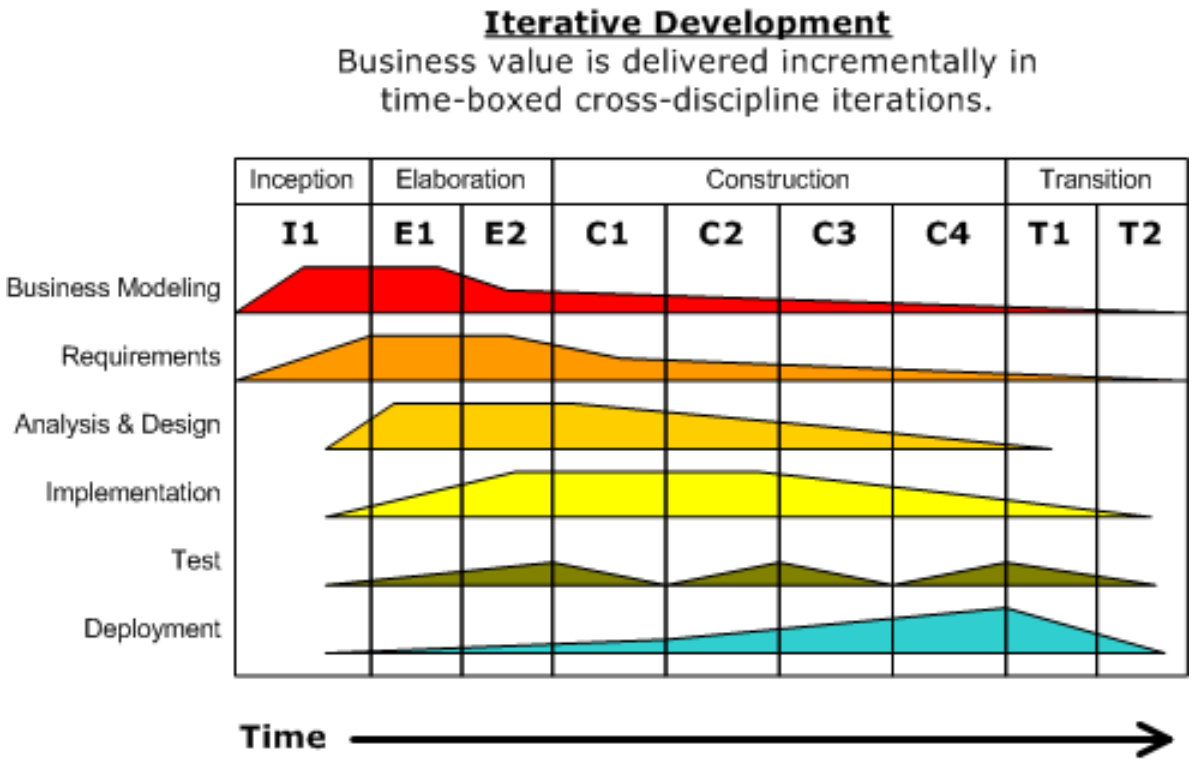


Diagram by Dutchguilder (Wikimedia).

RE in Software Projects - In Practice



In iterative and incremental software development, RE runs through the whole project.

Diagram by Dutchguilder (Wikimedia).

Successful RE is Knowledge-Intensive

Having requirements is not enough:

- Unclear requirements $\rightarrow$ Plan discussions

Successful RE is Knowledge-Intensive

Having requirements is not enough:

- Unclear requirements $\rightarrow$ Plan discussions
- Few requirements $\rightarrow$ Expect new requests

Successful RE is Knowledge-Intensive

Having requirements is not enough:

- Unclear requirements → Plan discussions
- Few requirements → Expect new requests
- Lot of requirements → Prioritise them

Successful RE is Knowledge-Intensive

Having requirements is not enough:

- Unclear requirements → Plan discussions
- Few requirements → Expect new requests
- Lot of requirements → Prioritise them
- Conflicts may occur → Agree on priority criteria

Successful RE is Knowledge-Intensive

Having requirements is not enough:

- Unclear requirements → Plan discussions
- Few requirements → Expect new requests
- Lot of requirements → Prioritise them
- Conflicts may occur → Agree on priority criteria
- Etc.

Successful RE is Knowledge-Intensive

Having requirements is not enough:

- Unclear requirements → Plan discussions
- Few requirements → Expect new requests
- Lot of requirements → Prioritise them
- Conflicts may occur → Agree on priority criteria
- Etc.

RE needs a lot of domain knowledge: find a balance between certainty and flexibility.

Certainty/Flexibility Depends on Projects

Some projects are more flexible by nature:

- Agile methods on a game is feasible (e.g. extension packs)

Certainty/Flexibility Depends on Projects

Some projects are more flexible by nature:

- Agile methods on a game is feasible (e.g. extension packs)
- But costly for a critical aircraft or hospital device

Certainty/Flexibility Depends on Projects

Some projects are more flexible by nature:

- Agile methods on a game is feasible (e.g. extension packs)
- But costly for a critical aircraft or hospital device

Some projects are more certain by nature:

Certainty/Flexibility Depends on Projects

Some projects are more flexible by nature:

- Agile methods on a game is feasible (e.g. extension packs)
- But costly for a critical aircraft or hospital device

Some projects are more certain by nature:

- An aircraft is constrained by physics and regulation

Certainty/Flexibility Depends on Projects

Some projects are more flexible by nature:

- Agile methods on a game is feasible (e.g. extension packs)
- But costly for a critical aircraft or hospital device

Some projects are more certain by nature:

- An aircraft is constrained by physics and regulation
- While a game is a fully creative process

Certainty/Flexibility Depends on Projects

Some projects are more flexible by nature:

- Agile methods on a game is feasible (e.g. extension packs)
- But costly for a critical aircraft or hospital device

Some projects are more certain by nature:

- An aircraft is constrained by physics and regulation
- While a game is a fully creative process

But it also depends on decision makers:

Certainty/Flexibility Depends on Projects

Some projects are more flexible by nature:

- Agile methods on a game is feasible (e.g. extension packs)
- But costly for a critical aircraft or hospital device

Some projects are more certain by nature:

- An aircraft is constrained by physics and regulation
- While a game is a fully creative process

But it also depends on decision makers:

- Is it really necessary?

Certainty/Flexibility Depends on Projects

Some projects are more flexible by nature:

- Agile methods on a game is feasible (e.g. extension packs)
- But costly for a critical aircraft or hospital device

Some projects are more certain by nature:

- An aircraft is constrained by physics and regulation
- While a game is a fully creative process

But it also depends on decision makers:

- Is it really necessary?
- For how much cost/benefice in money/time/...?

Certainty/Flexibility Depends on Projects

Some projects are more flexible by nature:

- Agile methods on a game is feasible (e.g. extension packs)
- But costly for a critical aircraft or hospital device

Some projects are more certain by nature:

- An aircraft is constrained by physics and regulation
- While a game is a fully creative process

But it also depends on decision makers:

- Is it really necessary?
- For how much cost/benefice in money/time/...?
- With which impact on our image?

Certainty/Flexibility Depends on Projects

Some projects are more flexible by nature:

- Agile methods on a game is feasible (e.g. extension packs)
- But costly for a critical aircraft or hospital device

Some projects are more certain by nature:

- An aircraft is constrained by physics and regulation
- While a game is a fully creative process

But it also depends on decision makers:

- Is it really necessary?
- For how much cost/benefice in money/time/...?
- With which impact on our image?
- Etc.

RE in a Nutshell

- RE is knowledge-intensive

RE in a Nutshell

- RE is knowledge-intensive
- RE is not a one shot activity, but a continuous one

RE in a Nutshell

- RE is knowledge-intensive
- RE is not a one shot activity, but a continuous one
- Requirements can be handled in many ways

RE in a Nutshell

- RE is knowledge-intensive
- RE is not a one shot activity, but a continuous one
- Requirements can be handled in many ways
- Requirements engineers should adapt to project and people

RE in a Nutshell

- RE is knowledge-intensive
- RE is not a one shot activity, but a continuous one
- Requirements can be handled in many ways
- Requirements engineers should adapt to project and people
- Doing so requires familiarity with various activities:
  - *Requirements elicitation*
  - *Requirements modelling*
  - *Requirements analysis*
  - *Requirements prioritisation*
  - *Requirements management*
  - *Etc.*

Outline

Outline

Requirements Elicitation

Goal: Gather new or revised requirements.

Requirements Elicitation

Goal: Gather new or revised requirements.

Main tasks:

Requirements Elicitation

Goal: Gather new or revised requirements.

Main tasks:

- Identify stakeholders

Requirements Elicitation

Goal: Gather new or revised requirements.

Main tasks:

- Identify stakeholders
- Understand their goals

Requirements Elicitation

Goal: Gather new or revised requirements.

Main tasks:

- Identify stakeholders
- Understand their goals
- Understand domain and environment (system-as-is)

Requirements Elicitation

Goal: Gather new or revised requirements.

Main tasks:

- Identify stakeholders
- Understand their goals
- Understand domain and environment (system-as-is)
- Draw requirements (system-to-be)

Requirements Elicitation

Goal: Gather new or revised requirements.

Main tasks:

- Identify stakeholders
- Understand their goals
- Understand domain and environment (system-as-is)
- Draw requirements (system-to-be)
- Document for reuse

Requirements Elicitation

Various stakeholders:

Requirements Elicitation

Various stakeholders:

- Customers

Requirements Elicitation

Various stakeholders:

- Customers
- Final users

Requirements Elicitation

Various stakeholders:

- Customers
- Final users
- Domain experts

Requirements Elicitation

Various stakeholders:

- Customers
- Final users
- Domain experts
- Regulatory authorities

Requirements Elicitation

Various stakeholders:

- Customers
- Final users
- Domain experts
- Regulatory authorities
- Developers

Requirements Elicitation

Various stakeholders:

- Customers
- Final users
- Domain experts
- Regulatory authorities
- Developers
- Etc.

Requirements Elicitation

Various stakeholders:

- Customers
- Final users
- Domain experts
- Regulatory authorities
- Developers
- Etc.

Other sources can help:

Requirements Elicitation

Various stakeholders:

- Customers
- Final users
- Domain experts
- Regulatory authorities
- Developers
- Etc.

Other sources can help:

- Existing specifications

Requirements Elicitation

Various stakeholders:

- Customers
- Final users
- Domain experts
- Regulatory authorities
- Developers
- Etc.

Other sources can help:

- Existing specifications
- Similar projects

Requirements Elicitation

Various stakeholders:

- Customers
- Final users
- Domain experts
- Regulatory authorities
- Developers
- Etc.

Other sources can help:

- Existing specifications
- Similar projects
- Standards

Requirements Elicitation

Various stakeholders:

- Customers
- Final users
- Domain experts
- Regulatory authorities
- Developers
- Etc.

Other sources can help:

- Existing specifications
- Similar projects
- Standards
- Etc.

Elicitation Technique - Interviews/Questionnaires

Goal: Gather information and opinions from isolated stakeholders.

Elicitation Technique - Interviews/Questionnaires

Goal: Gather information and opinions from isolated stakeholders.

Pros:

- Direct feedback with personal perspectives
- Can be close (strictly follow questions) or open (adapt questions on the fly)
- Isolation allows customisation

Elicitation Technique - Interviews/Questionnaires

Goal: Gather information and opinions from isolated stakeholders.

Pros:

- Direct feedback with personal perspectives
- Can be close (strictly follow questions) or open (adapt questions on the fly)
- Isolation allows customisation

Cons:

- Driven by interviewer more than stakeholders
- Does not exploit synergies (isolated interviews)
- Different interviews may contradict each other

Elicitation Technique - Meeting/Focus Group

Goal: Gather information and opinions from groups of stakeholders.

Elicitation Technique - Meeting/Focus Group

Goal: Gather information and opinions from groups of stakeholders.

Pros:

- Exploit synergies (e.g. brainstorming)
- Help to identify conflicts and agreements

Elicitation Technique - Meeting/Focus Group

Goal: Gather information and opinions from groups of stakeholders.

Pros:

- Exploit synergies (e.g. brainstorming)
- Help to identify conflicts and agreements

Cons:

- Discussions can be limited by power (leaders vs. subordinates)
- Discussions can be limited by personality (extroverts vs. introverts)
- Cannot involve too many stakeholders
- Hard to apply for timely/geographically spread teams

Elicitation Technique - Scenarios/Use Cases/Personas

Goal: Draw behavioural patterns.

Elicitation Technique - Scenarios/Use Cases/Personas

Goal: Draw behavioural patterns.

Pros:

- Helps to understand the system-as-is
- Easy to get from domain experts

Elicitation Technique - Scenarios/Use Cases/Personas

Goal: Draw behavioural patterns.

Pros:

- Helps to understand the system-as-is
- Easy to get from domain experts

Cons:

- Arguable for describing the system-to-be (bias)
- Not easy to identify inter-dependencies (isolated behaviours)

Elicitation Technique - Ethnographic Study

Goal: Analyse people's behaviours on site.

Elicitation Technique - Ethnographic Study

Goal: Analyse people's behaviours on site.

Pros:

- Direct observation of actual behaviours
- Can observe subtle behaviours and variants
- Can ask for explanations on the fly

Elicitation Technique - Ethnographic Study

Goal: Analyse people's behaviours on site.

Pros:

- Direct observation of actual behaviours
- Can observe subtle behaviours and variants
- Can ask for explanations on the fly

Cons:

- Impact on the workflow might be significant
- Require a lot of time
- Not applicable to all tasks (e.g. emergency)
- Rare but important behaviours might be missing

Summary

- Requirements elicitation is a hard task

Summary

- Requirements elicitation is a hard task
- Subject to communication failures
  - *Stakeholders can be unavailable*
  - *Stakeholders can overlook relevant details*
  - *Stakeholders can be unable to explain themselves*
  - *Stakeholders can disagree with each others*
  - *Stakeholders can be unwilling to share or lie*

Summary

- Requirements elicitation is a hard task
- Subject to communication failures
  - *Stakeholders can be unavailable*
  - *Stakeholders can overlook relevant details*
  - *Stakeholders can be unable to explain themselves*
  - *Stakeholders can disagree with each others*
  - *Stakeholders can be unwilling to share or lie*
- Other techniques not presented:
  - *Prototypes, user stories, competitors' analysis, data mining, etc.*

Summary

- Requirements elicitation is a hard task
- Subject to communication failures
  - *Stakeholders can be unavailable*
  - *Stakeholders can overlook relevant details*
  - *Stakeholders can be unable to explain themselves*
  - *Stakeholders can disagree with each others*
  - *Stakeholders can be unwilling to share or lie*
- Other techniques not presented:
  - *Prototypes, user stories, competitors' analysis, data mining, etc.*
- Different techniques can be complementary, but come with their own costs

Summary

- Requirements elicitation is a hard task
- Subject to communication failures
  - *Stakeholders can be unavailable*
  - *Stakeholders can overlook relevant details*
  - *Stakeholders can be unable to explain themselves*
  - *Stakeholders can disagree with each others*
  - *Stakeholders can be unwilling to share or lie*
- Other techniques not presented:
  - *Prototypes, user stories, competitors' analysis, data mining, etc.*
- Different techniques can be complementary, but come with their own costs
- Choosing the right techniques, applied with the right amounts, and combined in the right way is a skill by itself

Outline

Requirements Modelling

Goal: Represent relevant requirements properties.

Requirements Modelling

Goal: Represent relevant requirements properties.

Main tasks:

Requirements Modelling

Goal: Represent relevant requirements properties.

Main tasks:

- Identify domain terminology

Requirements Modelling

Goal: Represent relevant requirements properties.

Main tasks:

- Identify domain terminology
- Structure concepts and relationships
  - *System, users, goals, interactions, resources, etc.*

Requirements Modelling

Goal: Represent relevant requirements properties.

Main tasks:

- Identify domain terminology
- Structure concepts and relationships
  - *System*, *users*, *goals*, *interactions*, *resources*, *etc.*
- Select types/levels of formalisation

Requirements Modelling

Visual models:

Requirements Modelling

Visual models:

- Help to understand/explain the domain

Requirements Modelling

Visual models:

- Help to understand/explain the domain
- Help to communicate with stakeholders

Requirements Modelling

Visual models:

- Help to understand/explain the domain
- Help to communicate with stakeholders
- Help to build requirements documents

Requirements Modelling

Visual models:

- Help to understand/explain the domain
- Help to communicate with stakeholders
- Help to build requirements documents

Formal models:

Requirements Modelling

Visual models:

- Help to understand/explain the domain
- Help to communicate with stakeholders
- Help to build requirements documents

Formal models:

- Help to spot errors/inconsistencies

Requirements Modelling

Visual models:

- Help to understand/explain the domain
- Help to communicate with stakeholders
- Help to build requirements documents

Formal models:

- Help to spot errors/inconsistencies
- Help to scale requirements

Requirements Modelling

Visual models:

- Help to understand/explain the domain
- Help to communicate with stakeholders
- Help to build requirements documents

Formal models:

- Help to spot errors/inconsistencies
- Help to scale requirements
- Allows for automated reasoning

Requirements Modelling

Visual models:

- Help to understand/explain the domain
- Help to communicate with stakeholders
- Help to build requirements documents

Formal models:

- Help to spot errors/inconsistencies
- Help to scale requirements
- Allows for automated reasoning

A model can be both visual and formal.

Modelling Technique - Unified Modelling Language (UML)

Goal: Provide a standard way to visualise the design of a software system.

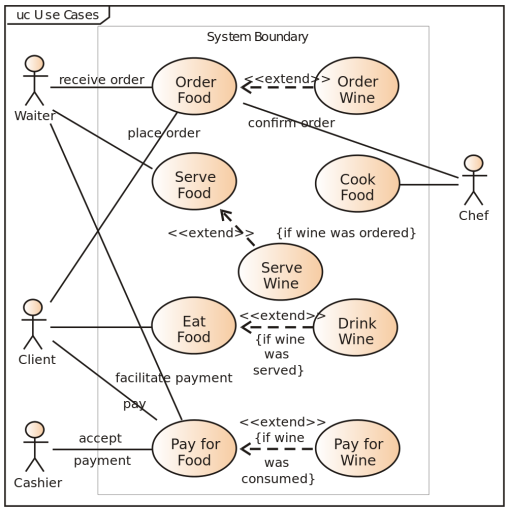
Source: `http://www.uml.org`, Picture: Kishorekumar 62 (Wikimedia)

Modelling Technique - Unified Modelling Language (UML)

Goal: Provide a standard way to visualise the design of a software system.

Pros:

- Cover several aspects of software design
  - *e.g. Use case diagrams can model goals:*
- Broadly used standard
- Many tools available



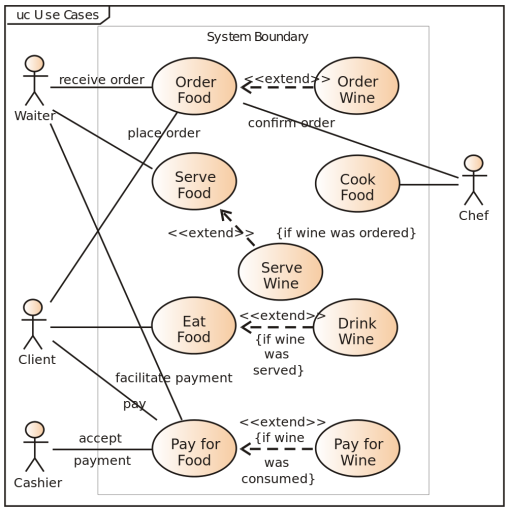
Source: `http://www.uml.org`, Picture: Kishorekumar 62 (Wikimedia)

Modelling Technique - Unified Modelling Language (UML)

Goal: Provide a standard way to visualise the design of a software system.

Pros:

- Cover several aspects of software design
  - *e.g. Use case diagrams can model goals:*
- Broadly used standard
- Many tools available



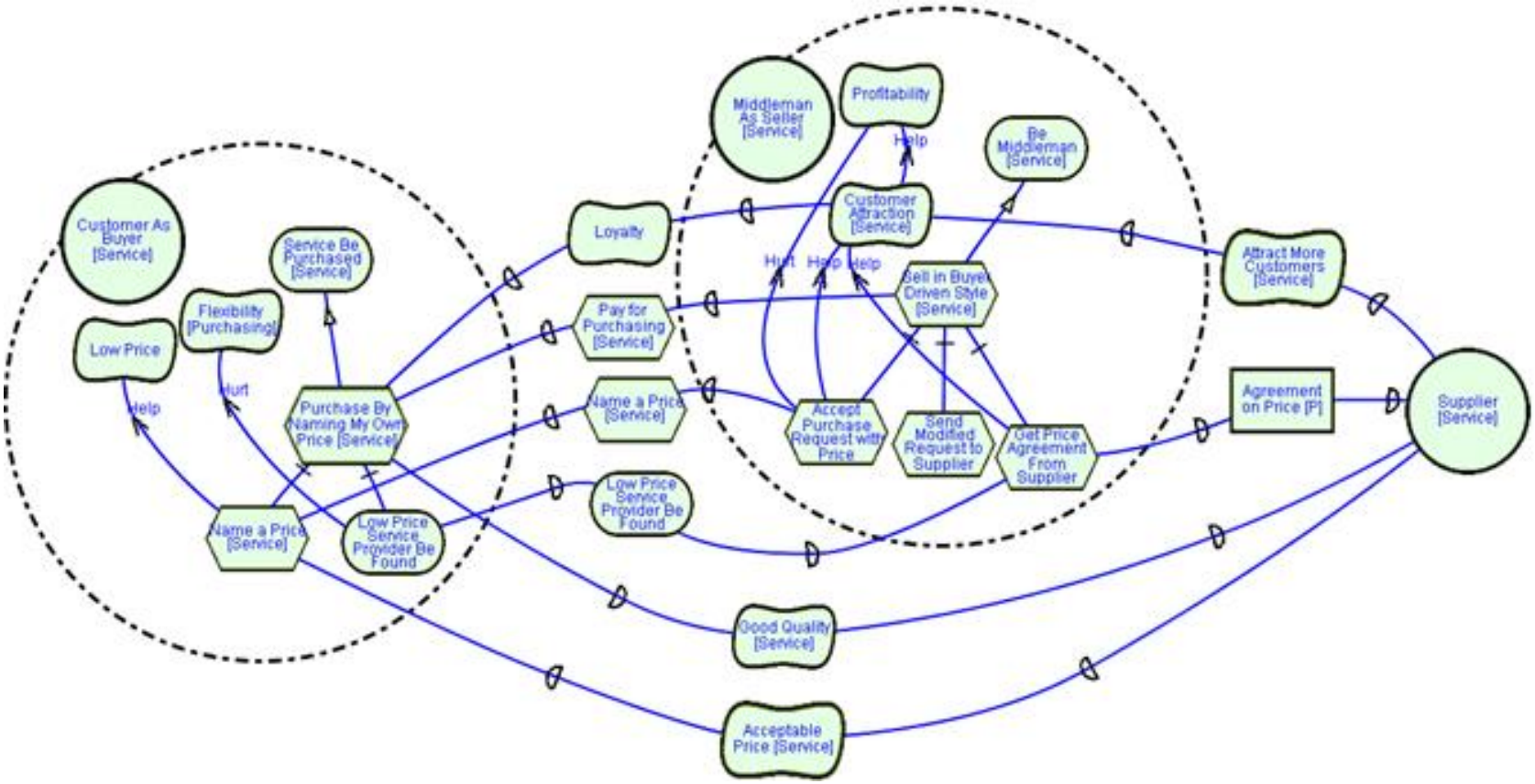
Cons:

- Designed for software modelling, not RE
- Miss relevant features for RE

Source: `http://www.uml.org`, Picture: Kishorekumar 62 (Wikimedia)

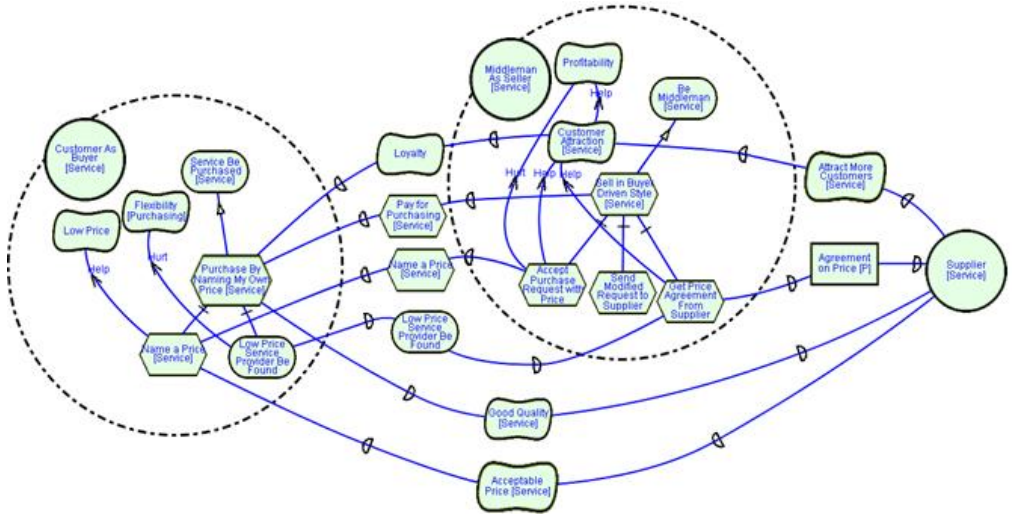
Modelling Technique - i* (i Star)

Goal: Represent stakeholders' goals and how they relate to each other and to the system.



Modelling Technique - i* (i Star)

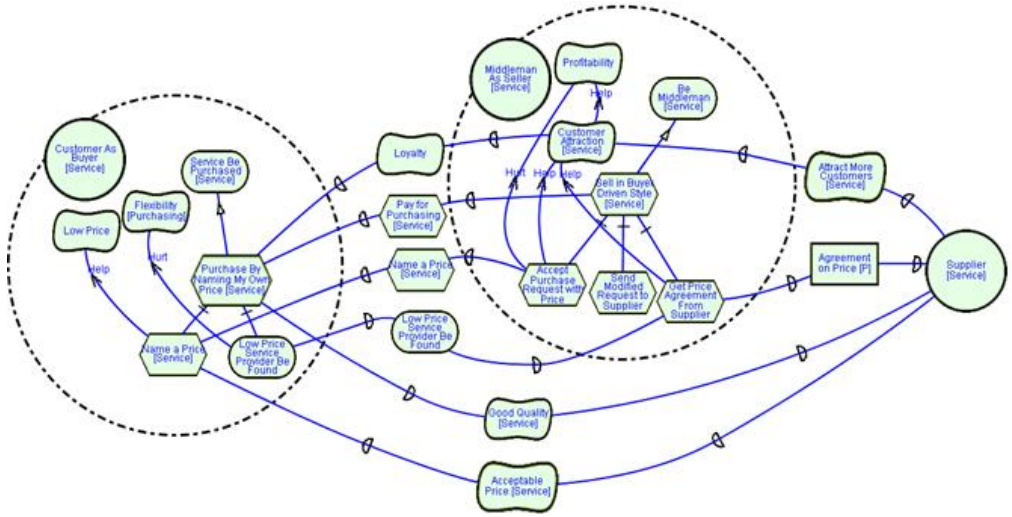
Goal: Represent stakeholders' goals and how they relate to each other and to the system.



Source: i* Wiki (`http://istar.rwth-aachen.de`)

Modelling Technique - i* (i Star)

Goal: Represent stakeholders' goals and how they relate to each other and to the system.



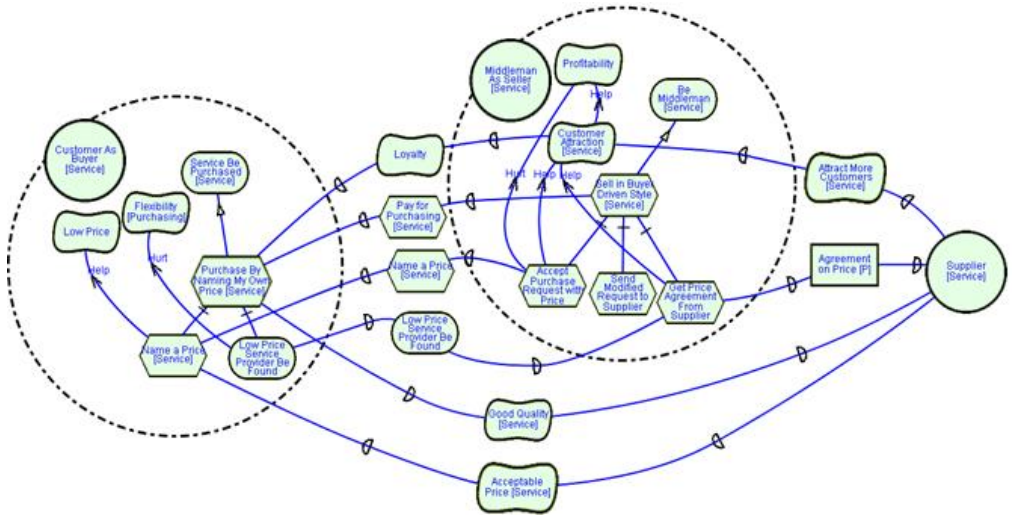
Pros: Offer a more comprehensive representation of stakeholders and their interactions with the system

- *Actors*, *goals*, *tasks*, *resources*, *dependencies*, *etc.*

Source: i* Wiki (`http://istar.rwth-aachen.de`)

Modelling Technique - i* (i Star)

Goal: Represent stakeholders' goals and how they relate to each other and to the system.



Pros: Offer a more comprehensive representation of stakeholders and their interactions with the system

- *Actors*, *goals*, *tasks*, *resources*, *dependencies*, *etc.*

Cons: Limited to general aspects (mitigated by variants)

Source: i* Wiki (`http://istar.rwth-aachen.de`)

Modelling Technique - Gherkin

Goal: Enforce firm, unambiguous requirements.

```
Given the car is a <type>
When we calculate the price
Then the price should be $<price>

Examples:
 |     type |      price |
 | Ferrari | 1,000,000 |
```

Modelling Technique - Gherkin

Goal: Enforce firm, unambiguous requirements.

```
Given the car is a <type>
When we calculate the price
Then the price should be $<price>

Examples:
 |    type |     price |
 | Ferrari | 1,000,000 |
```

Modelling Technique - Gherkin

Goal: Enforce firm, unambiguous requirements.

```
Given the car is a <type>
When we calculate the price
Then the price should be $<price>

Examples:
 |    type |     price |
 | Ferrari | 1,000,000 |
```

Pros:

- Balance formal and informal specifications
- Feature acceptance test generation
  - *Behaviour Driven Development, Test Driven Development*
- Support many natural + programming languages

Modelling Technique - Gherkin

Goal: Enforce firm, unambiguous requirements.

```
Given the car is a <type>
When we calculate the price
Then the price should be $<price>

Examples:
 |    type |     price |
 | Ferrari | 1,000,000 |
```

Pros:

- Balance formal and informal specifications
- Feature acceptance test generation
  - *Behaviour Driven Development, Test Driven Development*
- Support many natural + programming languages

Cons:

- Not RE method, tool only
- Not applicable to vague reqs.

Modelling Technique - Boolean Equations

Goal: Represent requirements as logical formula to satisfy.

$customer(Alice)$

$seller(Bob)$

$product(Car)$

$\forall c, customer(c) \Rightarrow \exists (p, s), seller(s) \wedge product(p) \wedge buy(c, p, s)$

Modelling Technique - Boolean Equations

Goal: Represent requirements as logical formula to satisfy.

$$customer(Alice)$$
$$seller(Bob)$$
$$product(Car)$$
$$\forall c, customer(c) \Rightarrow \exists(p, s), seller(s) \wedge product(p) \wedge buy(c, p, s)$$

Modelling Technique - Boolean Equations

Goal: Represent requirements as logical formula to satisfy.

$$customer(Alice)$$
$$seller(Bob)$$
$$product(Car)$$
$$\forall c, customer(c) \Rightarrow \exists(p, s), seller(s) \wedge product(p) \wedge buy(c, p, s)$$

Pros:

- Allow to use formal reasoning (e.g. SAT solvers)
- Exact meaning (no ambiguity)

Modelling Technique - Boolean Equations

Goal: Represent requirements as logical formula to satisfy.

$$customer(Alice)$$
$$seller(Bob)$$
$$product(Car)$$
$$\forall c, customer(c) \Rightarrow \exists (p, s), seller(s) \wedge product(p) \wedge buy(c, p, s)$$

Pros:

- Allow to use formal reasoning (e.g. SAT solvers)
- Exact meaning (no ambiguity)

Cons:

- Hard to understand for non-logicians
- Not applicable to vague requirements

Summary

- Different Models emphasise different properties

Summary

- Different Models emphasise different properties
- Various techniques exist, not only the ones presented:
  - *Data Flow Diagram and RML, KAOS, extensions of i* (e.g. Tropos for multi-agents, Nòmos for laws, Zanshin [残心] for self-adaptation), ReqIF & OSLC standards, etc.*

Outline

Requirements Analysis

Goal: Validate requirements through their model.

Requirements Analysis

Goal: Validate requirements through their model.

Main tasks:

Requirements Analysis

Goal: Validate requirements through their model.

Main tasks:

- Identify & fix inconsistencies

Requirements Analysis

Goal: Validate requirements through their model.

Main tasks:

- Identify & fix inconsistencies
- Identify & fix incompleteness

Requirements Analysis

Goal: Validate requirements through their model.

Main tasks:

- Identify & fix inconsistencies
- Identify & fix incompleteness
- Identify & fix ambiguities

Requirements Analysis

Goal: Validate requirements through their model.

Main tasks:

- Identify & fix inconsistencies
- Identify & fix incompleteness
- Identify & fix ambiguities
- Risk analysis

Requirements Analysis

Goal: Validate requirements through their model.

Main tasks:

- Identify & fix inconsistencies
- Identify & fix incompleteness
- Identify & fix ambiguities
- Risk analysis
- Evaluate alternatives

Requirements Analysis

Informal analysis:

Requirements Analysis

Informal analysis:

- Fit with natural language and visual models

Requirements Analysis

Informal analysis:

- Fit with natural language and visual models
- Exploit the flexibility of human reasoning

Requirements Analysis

Informal analysis:

- Fit with natural language and visual models
- Exploit the flexibility of human reasoning
- Good for qualitative analysis

Requirements Analysis

Informal analysis:

- Fit with natural language and visual models
- Exploit the flexibility of human reasoning
- Good for qualitative analysis
- Help for formal modelling/analysis

Requirements Analysis

Informal analysis:

- Fit with natural language and visual models
- Exploit the flexibility of human reasoning
- Good for qualitative analysis
- Help for formal modelling/analysis

Formal analysis:

Requirements Analysis

Informal analysis:

- Fit with natural language and visual models
- Exploit the flexibility of human reasoning
- Good for qualitative analysis
- Help for formal modelling/analysis

Formal analysis:

- Fit with formal models

Requirements Analysis

Informal analysis:

- Fit with natural language and visual models
- Exploit the flexibility of human reasoning
- Good for qualitative analysis
- Help for formal modelling/analysis

Formal analysis:

- Fit with formal models
- Exploit the exactitude of formal reasoning

Requirements Analysis

Informal analysis:

- Fit with natural language and visual models
- Exploit the flexibility of human reasoning
- Good for qualitative analysis
- Help for formal modelling/analysis

Formal analysis:

- Fit with formal models
- Exploit the exactitude of formal reasoning
- Good for quantitative analysis

Analysis Technique - Manual Analysis

Goal: Rely on stakeholders to analyse and refine the models.

Analysis Technique - Manual Analysis

Goal: Rely on stakeholders to analyse and refine the models.

Pros:

- Build on expertise of the stakeholders
- Requirements engineer acts as a facilitator
- Good when the models are still small

Analysis Technique - Manual Analysis

Goal: Rely on stakeholders to analyse and refine the models.

Pros:

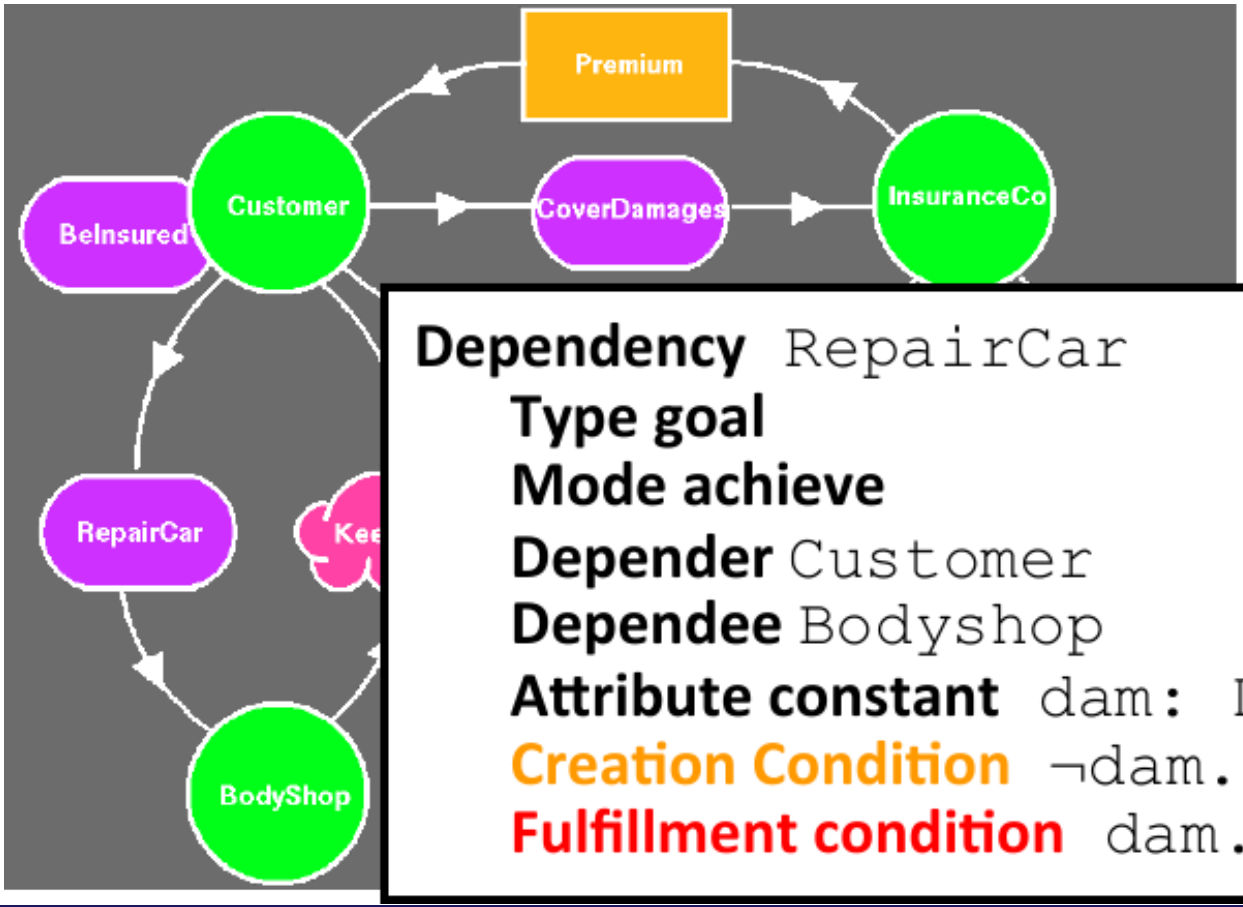
- Build on expertise of the stakeholders
- Requirements engineer acts as a facilitator
- Good when the models are still small

Cons:

- Lack systematicity (no guarantee of success)
- Hard to maintain when the models become complex

Analysis Technique - Formal Tropos

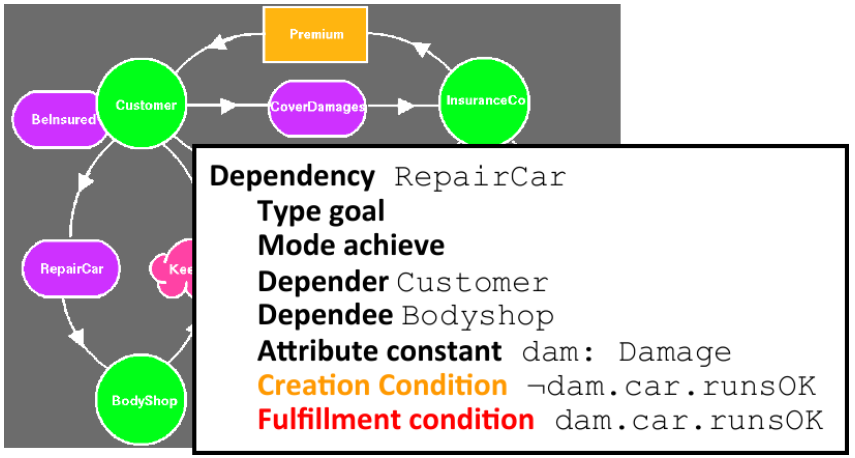
Goal: Translate goal-models into logical formulae for formal analysis.



Dependency RepairCar
Type goal
Mode achieve
Depender Customer
Dependee Bodyshop
Attribute constant dam: Damage
Creation Condition ¬dam.car.runsOK
Fulfillment condition dam.car.runsOK

Analysis Technique - Formal Tropos

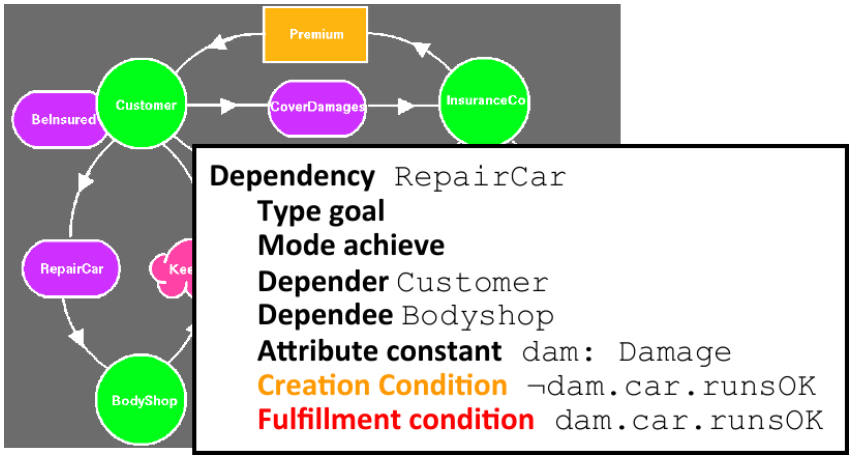
Goal: Translate goal-models into logical formulae for formal analysis.



Pictures: reworked from Anna Perini

Analysis Technique - Formal Tropos

Goal: Translate goal-models into logical formulae for formal analysis.



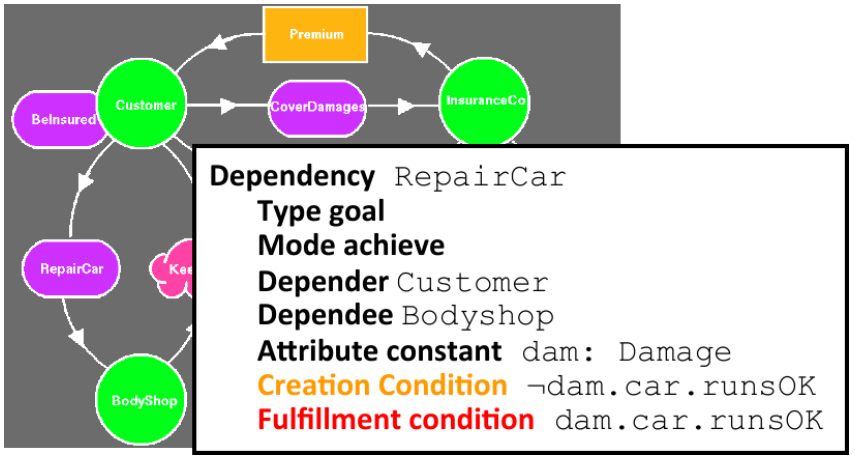
Pros:

- Consistency proven or conflicts spotted
- Interesting properties can be proven too
- Can play with the model to analyse different scenarios

Pictures: reworked from Anna Perini

Analysis Technique - Formal Tropos

Goal: Translate goal-models into logical formulae for formal analysis.



Pros:

- Consistency proven or conflicts spotted
- Interesting properties can be proven too
- Can play with the model to analyse different scenarios

Cons:

- Not applicable to vague requirements
- Formula hard to understand for non-logicians

Pictures: reworked from Anna Perini

Summary

- Analysis depends on models

Summary

- Analysis depends on models
- Carefully choose models for supporting analysis

Outline

Requirements Prioritisation

Goal: Select the most relevant requirements

Requirements Prioritisation

Goal: Select the most relevant requirements

Main tasks:

Requirements Prioritisation

Goal: Select the most relevant requirements

Main tasks:

- Identify prioritisation criteria (e.g. cost/benefice, risks, popularity)

Requirements Prioritisation

Goal: Select the most relevant requirements

Main tasks:

- Identify prioritisation criteria (e.g. cost/benefice, risks, popularity)
- Identify constraints (e.g. deadlines, requirements dependencies)

Requirements Prioritisation

Goal: Select the most relevant requirements

Main tasks:

- Identify prioritisation criteria (e.g. cost/benefice, risks, popularity)
- Identify constraints (e.g. deadlines, requirements dependencies)
- Order requirements by priority

Requirements Prioritisation

Goal: Select the most relevant requirements

Main tasks:

- Identify prioritisation criteria (e.g. cost/benefice, risks, popularity)
- Identify constraints (e.g. deadlines, requirements dependencies)
- Order requirements by priority
- Filter out unwanted requirements

Requirements Prioritisation

Mono vs. multi-objectives:

Requirements Prioritisation

Mono vs. multi-objectives:

- Mono: All criteria are reduced to 1, usually with a weighted average.

Requirements Prioritisation

Mono vs. multi-objectives:

- Mono: All criteria are reduced to 1, usually with a weighted average.
- Multi: Each criterion is evaluated separately, the best solutions form a Pareto front

Requirements Prioritisation

Mono vs. multi-objectives:

- Mono: All criteria are reduced to 1, usually with a weighted average.
- Multi: Each criterion is evaluated separately, the best solutions form a Pareto front

Human vs. automated decision:

Requirements Prioritisation

Mono vs. multi-objectives:

- Mono: All criteria are reduced to 1, usually with a weighted average.
- Multi: Each criterion is evaluated separately, the best solutions form a Pareto front

Human vs. automated decision:

- Human: Exploit qualitative judgement and expertise of decision makers

Requirements Prioritisation

Mono vs. multi-objectives:

- Mono: All criteria are reduced to 1, usually with a weighted average.
- Multi: Each criterion is evaluated separately, the best solutions form a Pareto front

Human vs. automated decision:

- Human: Exploit qualitative judgement and expertise of decision makers
- Automated: Exploit formalised quantitative evaluation

Requirements Prioritisation

Mono vs. multi-objectives:

- Mono: All criteria are reduced to 1, usually with a weighted average.
- Multi: Each criterion is evaluated separately, the best solutions form a Pareto front

Human vs. automated decision:

- Human: Exploit qualitative judgement and expertise of decision makers
- Automated: Exploit formalised quantitative evaluation
- Hybrid: Automated for preliminary + human for final decision

Requirements Prioritisation

Pairwise vs. order:

Requirements Prioritisation

Pairwise vs. order:

- Pairwise: Local decision comparing successively pairs of requirements

Requirements Prioritisation

Pairwise vs. order:

- Pairwise: Local decision comparing successively pairs of requirements
- Order: Global decision

Requirements Prioritisation

Pairwise vs. order:

- Pairwise: Local decision comparing successively pairs of requirements
- Order: Global decision

Various methods:

Requirements Prioritisation

Pairwise vs. order:

- Pairwise: Local decision comparing successively pairs of requirements
- Order: Global decision

Various methods:

- Fully manual: Rely on stakeholders expertise.

Requirements Prioritisation

Pairwise vs. order:

- Pairwise: Local decision comparing successively pairs of requirements
- Order: Global decision

Various methods:

- Fully manual: Rely on stakeholders expertise.
- Machine learning: Evaluate new solutions based on already known ones.

Requirements Prioritisation

Pairwise vs. order:

- Pairwise: Local decision comparing successively pairs of requirements
- Order: Global decision

Various methods:

- Fully manual: Rely on stakeholders expertise.
- Machine learning: Evaluate new solutions based on already known ones.
- Search algorithms: Iteratively improve a (set of) solution(s) evaluated through a (set of) fitness function(s).

Requirements Prioritisation

Pairwise vs. order:

- Pairwise: Local decision comparing successively pairs of requirements
- Order: Global decision

Various methods:

- Fully manual: Rely on stakeholders expertise.
- Machine learning: Evaluate new solutions based on already known ones.
- Search algorithms: Iteratively improve a (set of) solution(s) evaluated through a (set of) fitness function(s).
- Constraints satisfaction: Find solution(s) satisfying Boolean formulae.

Prioritisation Technique - MoSCoW Prioritisation

Goal: Have stakeholders rank all requirements in 4 levels – must have, should have, could have, and won't have.

Prioritisation Technique - MoSCoW Prioritisation

Goal: Have stakeholders rank all requirements in 4 levels – must have, should have, could have, and won't have.

Pros:

- Levels are simple to understand for stakeholders
- Levels focus on practical priorities

Prioritisation Technique - MoSCoW Prioritisation

Goal: Have stakeholders rank all requirements in 4 levels – must have, should have, could have, and won't have.

Pros:

- Levels are simple to understand for stakeholders
- Levels focus on practical priorities

Cons:

- The ranking is extremely scarce for many requirements
- Lack of rationale behind each choice

Prioritisation Technique - Bubble Sort

Goal: Iteratively improve global order by swapping pairs of requirements.

Prioritisation Technique - Bubble Sort

Goal: Iteratively improve global order by swapping pairs of requirements.

Pros:

- Simple to implement
- Simple to understand

Prioritisation Technique - Bubble Sort

Goal: Iteratively improve global order by swapping pairs of requirements.

Pros:

- Simple to implement
- Simple to understand

Cons:

- Require a lot of pair comparisons ($O(n^2)$)
- No way to tell that a requirement is "far" below/above another

Prioritisation Technique - Analytic Hierarchy Process (AHP)

Goal: Build a matrix of requirements comparisons to infer their global order.

Prioritisation Technique - Analytic Hierarchy Process (AHP)

Goal: Build a matrix of requirements comparisons to infer their global order.

Pros:

- Comparisons are weighted, thus more informative
- Estimations of the missing values can be computed to ask only the most relevant comparisons

Prioritisation Technique - Analytic Hierarchy Process (AHP)

Goal: Build a matrix of requirements comparisons to infer their global order.

Pros:

- Comparisons are weighted, thus more informative
- Estimations of the missing values can be computed to ask only the most relevant comparisons

Cons:

- Although improved, remain costly in terms of human effort.

Prioritisation Technique - Case Based Ranking (CBRank)

Goal: Learn from previous comparisons to infer remaining comparisons.

Prioritisation Technique - Case Based Ranking (CBRank)

Goal: Learn from previous comparisons to infer remaining comparisons.

Pros:

- Machine learning reduces human effort
- Tend to be more efficient than AHP (Avesani et al. [2005])

Prioritisation Technique - Case Based Ranking (CBRank)

Goal: Learn from previous comparisons to infer remaining comparisons.

Pros:

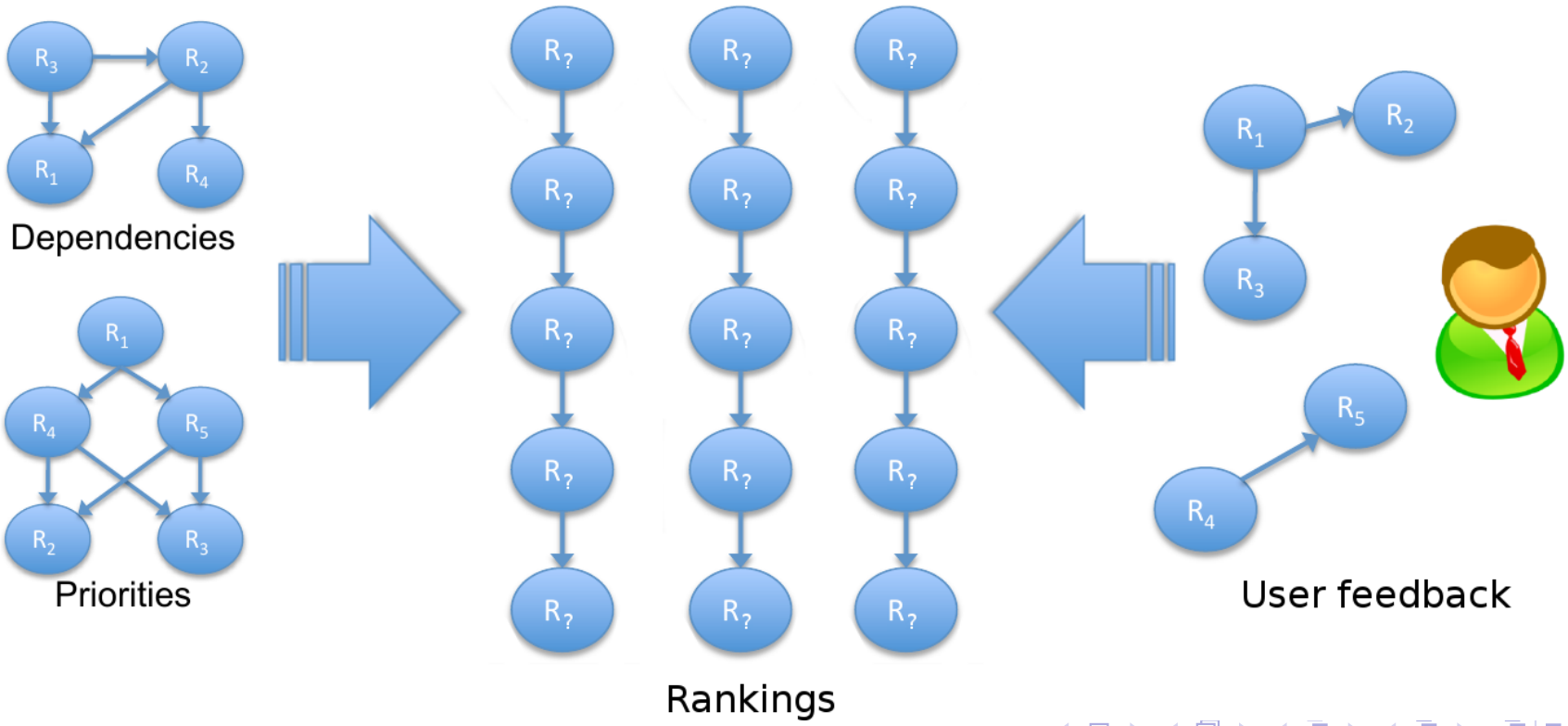
- Machine learning reduces human effort
- Tend to be more efficient than AHP (Avesani et al. [2005])

Cons:

- Critics on learning with anecdotal evidences

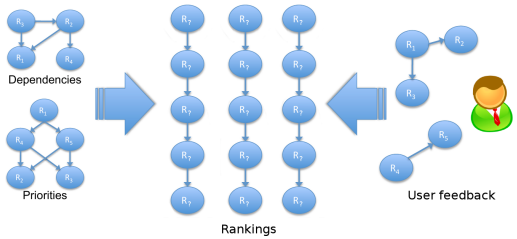
Prioritisation Technique - Interactive Genetic Algorithms

Goal: Iteratively improve a set of rankings, by recombinations and random mutations, to comply with pre-determined + elicited constraints.



Prioritisation Technique - Interactive Genetic Algorithms

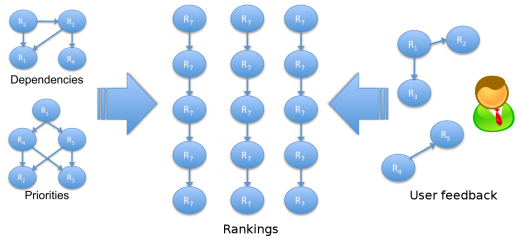
Goal: Iteratively improve a set of rankings, by recombinations and random mutations, to comply with pre-determined + elicited constraints.



Pictures: reworked from Angelo Susi

Prioritisation Technique - Interactive Genetic Algorithms

Goal: Iteratively improve a set of rankings, by recombinations and random mutations, to comply with pre-determined + elicited constraints.



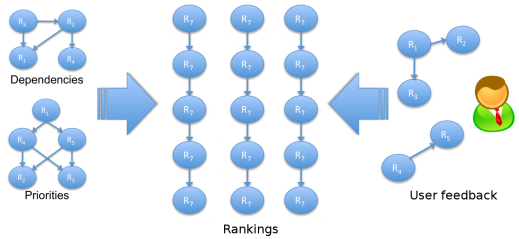
Pros:

- Recombinations and mutations motivated by Darwin evolution theory
- The resulting rankings can provide a various set of good alternatives

Pictures: reworked from Angelo Susi

Prioritisation Technique - Interactive Genetic Algorithms

Goal: Iteratively improve a set of rankings, by recombinations and random mutations, to comply with pre-determined + elicited constraints.



Pros:

- Recombinations and mutations motivated by Darwin evolution theory
- The resulting rankings can provide a various set of good alternatives

Cons:

- Solutions can converge to local optima only
- Set of solutions more costly than single one

Pictures: reworked from Angelo Susi

Summary

- Requirements prioritisation usually involves human qualitative judgement

Summary

- Requirements prioritisation usually involves human qualitative judgement
- Human efforts makes scalability problems common to all approaches

Summary

- Requirements prioritisation usually involves human qualitative judgement
- Human efforts makes scalability problems common to all approaches
- Other techniques exist
  - *Cost-value approach, Binary Search Tree, Planning Game, etc.*

Outline

Requirements Management

Goal: Ensure requirements access and update

Requirements Management

Goal: Ensure requirements access and update

Main tasks:

Requirements Management

Goal: Ensure requirements access and update

Main tasks:

- Store and retrieve requirements

Requirements Management

Goal: Ensure requirements access and update

Main tasks:

- Store and retrieve requirements
- Relate requirements to other artefacts

Requirements Management

Goal: Ensure requirements access and update

Main tasks:

- Store and retrieve requirements
- Relate requirements to other artefacts
- Support updates of requirements and their relations

Management Technique - Dynamic Renumbering

Goal: Identify requirements from their location in requirement documents.

Management Technique - Dynamic Renumbering

Goal: Identify requirements from their location in requirement documents.

Pros:

- Align automatic and manual retrieval
- Provide implicit classification of requirements

Management Technique - Dynamic Renumbering

Goal: Identify requirements from their location in requirement documents.

Pros:

- Align automatic and manual retrieval
- Provide implicit classification of requirements

Cons:

- Hard to manage without automated tools

Management Technique - Database Record Identification

Goal: Exploit IDs in database.

Management Technique - Database Record Identification

Goal: Exploit IDs in database.

Pros:

- Constant, no need to update
- Naturally generated by automated tools

Management Technique - Database Record Identification

Goal: Exploit IDs in database.

Pros:

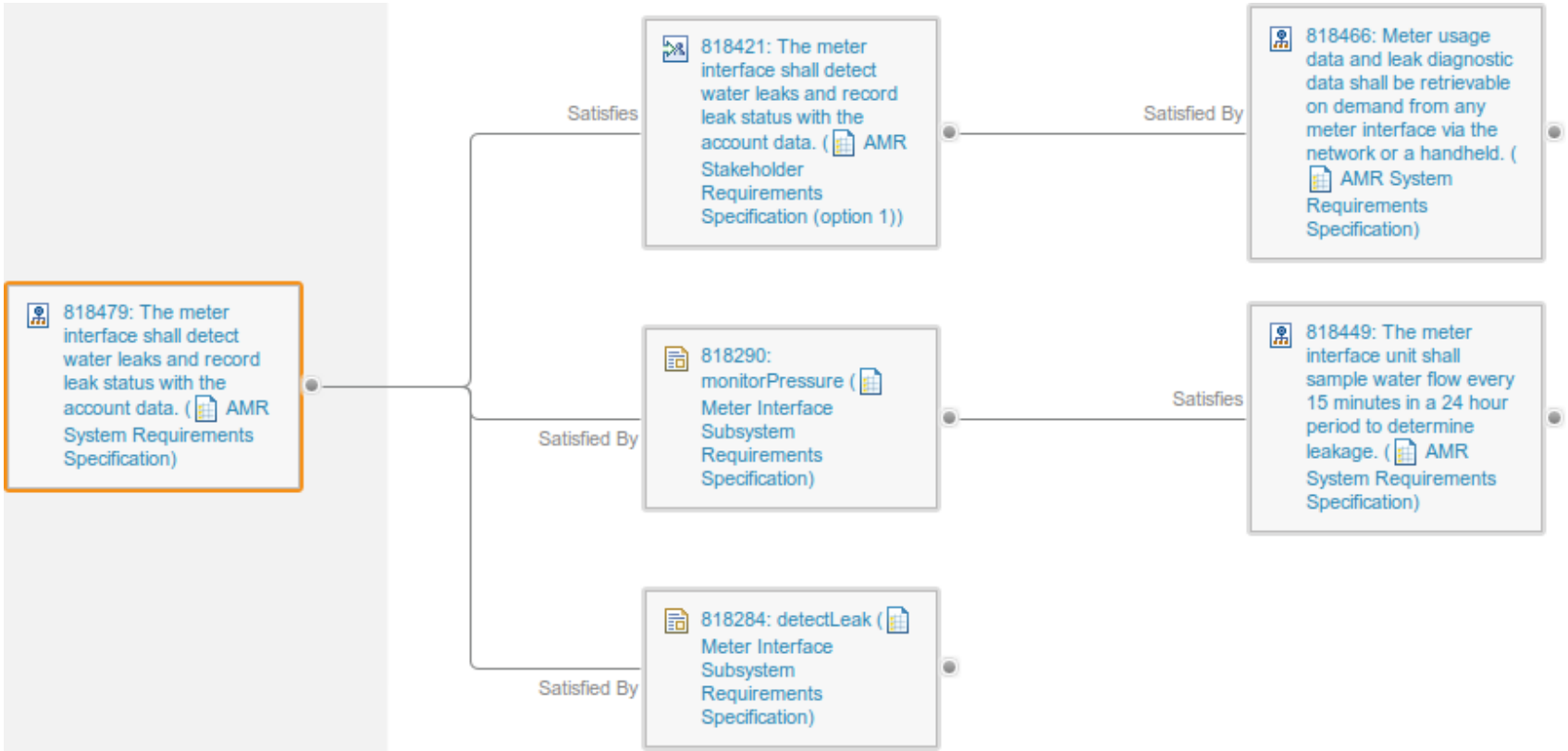
- Constant, no need to update
- Naturally generated by automated tools

Cons:

- Hard to generate meaningful document from automatically generated indexes

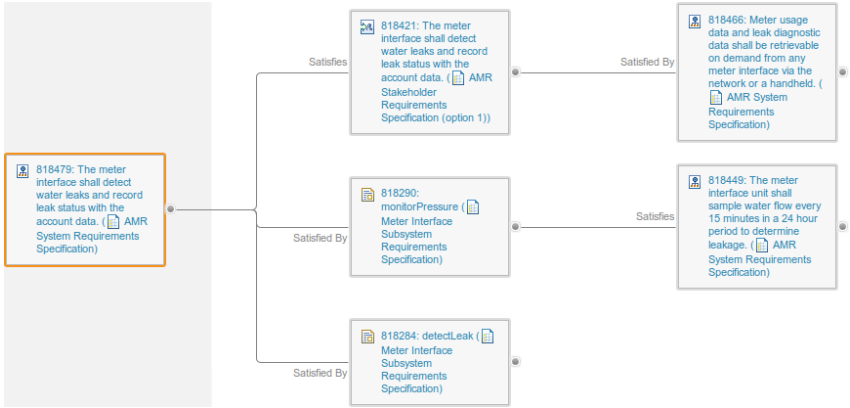
Management Technique - Traceability

Goal: Link requirements to other requirements and derived resources.



Management Technique - Traceability

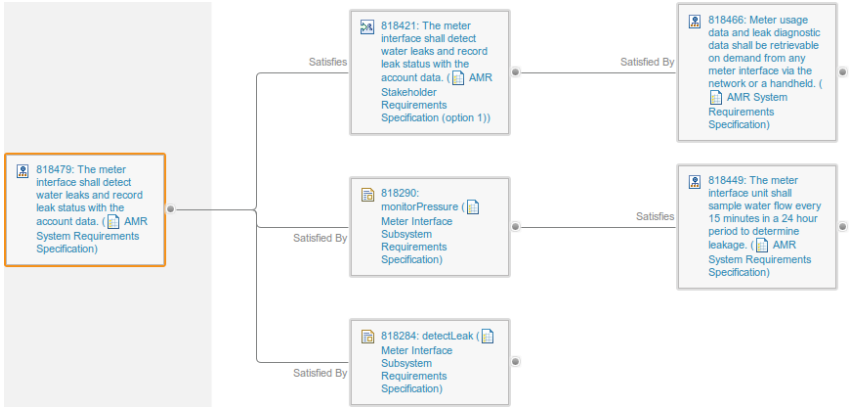
Goal: Link requirements to other requirements and derived resources.



Picture: IBM Rational DOORS (`https://www.doorsng.com/`)

Management Technique - Traceability

Goal: Link requirements to other requirements and derived resources.



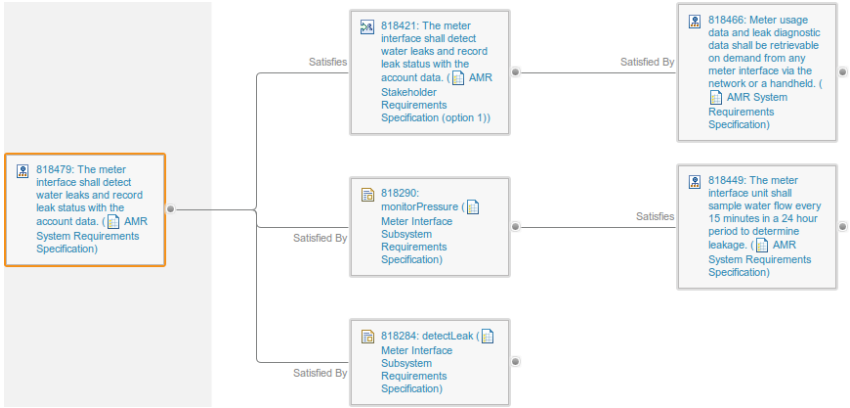
Pros:

- Help to update related requirements (horizontal traceability)
  - *Requirements dependencies, derived requirements*
- Help update other resources (vertical traceability)
  - *Interviews → requirements → tests & docs*

Picture: IBM Rational DOORS (`https://www.doorsng.com/`)

Management Technique - Traceability

Goal: Link requirements to other requirements and derived resources.



Pros:

- Help to update related requirements (horizontal traceability)
  - *Requirements dependencies, derived requirements*
- Help update other resources (vertical traceability)
  - *Interviews → requirements → tests & docs*

Cons:

- Complexity grows with number of requirements
- Hard to handle without automated tools (e.g. IBM Rational DOORS)

Picture: IBM Rational DOORS (`https://www.doorsng.com/`)

Summary

- Usually simple storage and retrieval.

Summary

- Usually simple storage and retrieval.
- May start with spreadsheets when simple

Summary

- Usually simple storage and retrieval.
- May start with spreadsheets when simple
- Quickly need advanced tools for deep management

Outline

Summary

We have seen:

- What is a requirement

Summary

We have seen:

- What is a requirement
- What is Requirements Engineering (RE)

Summary

We have seen:

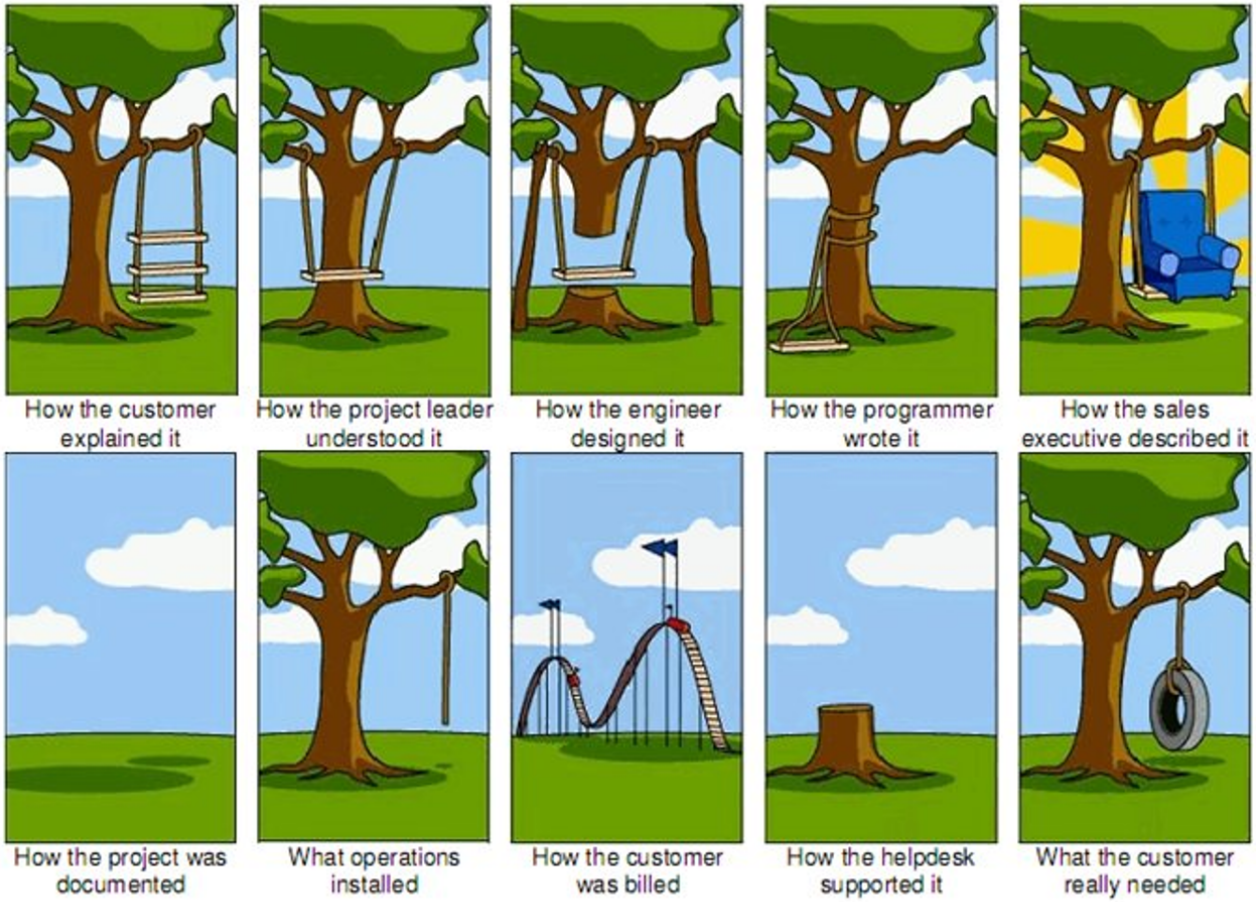
- What is a requirement
- What is Requirements Engineering (RE)
- RE and software development are tightly related

Summary

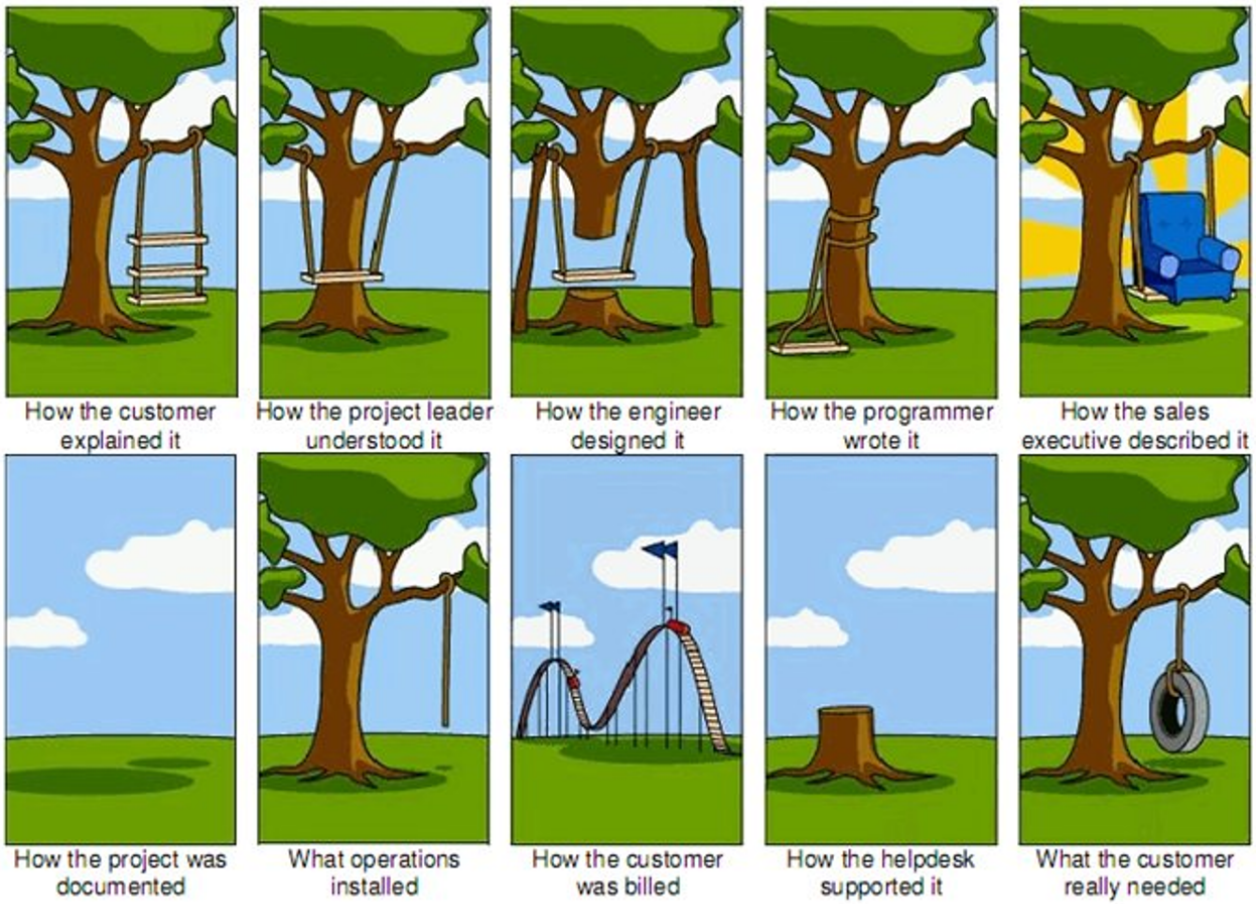
We have seen:

- What is a requirement
- What is Requirements Engineering (RE)
- RE and software development are tightly related
- Various RE techniques for different tasks

How to Make a Tree Swing?



How to Make a Tree Swing?



By having everyone agree (on the requirements)!

Going Further

Books about RE:

- Requirements engineering: fundamentals, principles, and techniques (Pohl [2010])
  - *Teaching material:* *https://re-buch.de/en/teaching-material/*
- Requirements engineering: from system goals to UML models and software specifications (Lamsweerde [2009])

Thanks for your attention.

Questions?

P. Avesani, C. Bazzanella, A. Perini, and A. Susi. Facing Scalability Issues in Requirements Prioritization with Machine Learning Techniques. In Proceedings of the 13th IEEE International Conference on Requirements Engineering, pages 297–306, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2425-7. doi: 10.1109/RE.2005.30. URL http://dl.acm.org/citation.cfm?id=1099549.1100639.

P. Bourque, R. E. Fairley, and IEEE Computer Society. Guide to the software engineering body of knowledge. 2014. URL http://www.computer.org/portal/web/swebok/swebokv3. OCLC: 926093687.

M. Glinz. On Non-Functional Requirements. pages 21–26. IEEE, Oct. 2007. ISBN 978-0-7695-2935-6. doi: 10.1109/RE.2007.45. URL http://ieeexplore.ieee.org/document/4384163/.

IEEE Standards Board. IEEE standard glossary of software engineering terminology. Institute of Electrical and Electronics Engineers, New York, N.Y, 1990. ISBN 978-1-55937-067-7.

A. v. Lamsweerde. Requirements engineering : from system goals to UML models and software specifications. Wiley ; John Wiley [distributor], Hoboken, N.J.; Chichester, 2009. ISBN 978-0-470-01270-3 0-470-01270-6. URL http://eu.wiley.com/WileyCDA/WileyTitle/productCd-EHEP000863.html.

K. Pohl. Requirements engineering: fundamentals, principles, and techniques. Springer, Heidelberg ; New York, 2010. ISBN 978-3-642-12577-5. OCLC: ocn642291082.