Requirements Engineering

2018/2019
Course 6

Course 6 outline

- Introduction to requirements traceability
- Requirements interdependencies
- Impact Analysis

Introduction

- Most individual requirements, developed during the requirements engineering process, cannot be treated in isolation during software development.
- Requirements are related to and affect each other in complex manners.
- Approx. 20% of requirements are truly singular (i.e., they are not related to or influence any other requirements).
- One requirement may affect other requirements when it:
    - constrains how they are designed and implemented,
    - affects their cost of implementation,
    - changes their customer satisfaction (increases/decreases).

Introduction

- Requirements interdependencies are not problematic per-se.
- They influence different development activities and decisions made during the software engineering process:
  - in release planning,
  - change management,
  - requirements design and implementation,
  - testing,
  - requirements reuse.

Examples:

- A change made to one requirement may change several other requirements. Ignoring the dependencies when assessing the impact of a change may result in higher costs.
- Release planning (an optimal set of requirements is selected for implementation in the next release of a software system):
  - It is not always possible to select the requirements with highest priority, due to requirements interdependencies. Implementing a high priority requirement may require that a requirement with low priority and high cost must be implemented first.
  - It is important to understand and know about these relationships in order to avoid selecting a set of requirements that must be changed later, and that may cause costly modifications of the software.

Introduction

- The aim of dealing with interdependencies is to improve development decisions.
- Managing interdependencies is about identifying, storing, and maintaining information about how requirements relate to and affect each other.
- It involves deciding which interdependency information is needed and how it should be presented.

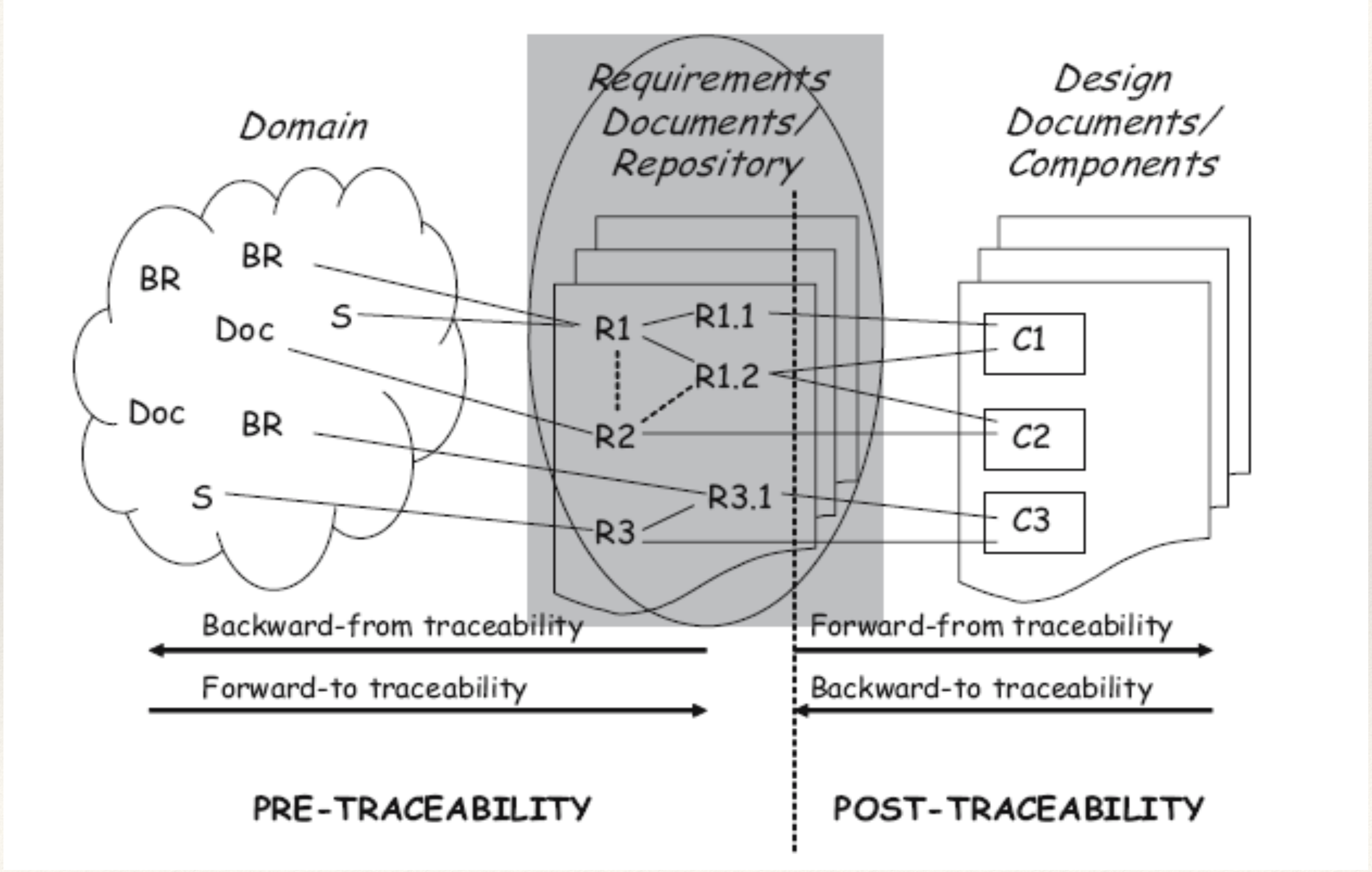
Requirements Traceability

- Different definitions for requirements traceability exist.
- “the ability to describe and follow the life of a requirement, in both forward and backward direction, ideally through the whole life cycle” (M. Jarke 1998, based on O. Gotel and A. Finkelstein 1994)
- Traceability is achieved through associating related information objects:
  - Requirements and related system components satisfying those requirements.
  - System objectives and requirements derived from those objectives.
  - Change proposals and requirements which they intend to change.
  - Test cases and the requirements whose fulfillment they intend to ensure.
  - System components and the resources needed to implement those requirements.
- Requirements traceability aids in developing quality software systems.
- Requirements interdependencies is viewed as a specific aspect of traceability.

Requirements Traceability

- Traceability information is needed as a basis for decisions and tasks in most phases of the software development process.
    - Within change integration, traceability information enables identification of the impact of a proposed change.
    - Identifying how requirements and other artifacts are affected by the change proposal allows more accurate cost and schedule analysis.
    - It motivates and explains the decisions and trade-offs made during development, and it is valuable for process improvement.
- It provides a possibility to ensure that all requirements are fulfilled by the system components and that no features have been added since all components or features within the system should be related to one or several requirements.
- Traceability improves development and maintenance and lowers costs.
- Poor traceability results in loss of knowledge if individuals leave the project.
- Capturing and maintaining traces is an important activity during requirements engineering as well as other parts of software engineering.

Types of Requirements Traceability



Types of Requirements Traceability

- Requirements traceability can be divided into two types:
  - pre-traceability
  - post-traceability.
- Pre-traceability refers to the life of a requirement before it is included in the requirements specification.
- It enables a better understanding of requirements.
- Pre-traceability includes tracing the elicitation, definition, and evolution of the requirements.
- The requirements should be related to their origin e.g. stakeholder (S), business rule (BR), or previous documentation (Doc), but also to other associated requirements e.g. through requirements decomposition.
- Requirements pre-traceability manages evolution of a system.
- It enables identification of the parts of the specification that are affected by a particular change request.

Types of Requirements Traceability

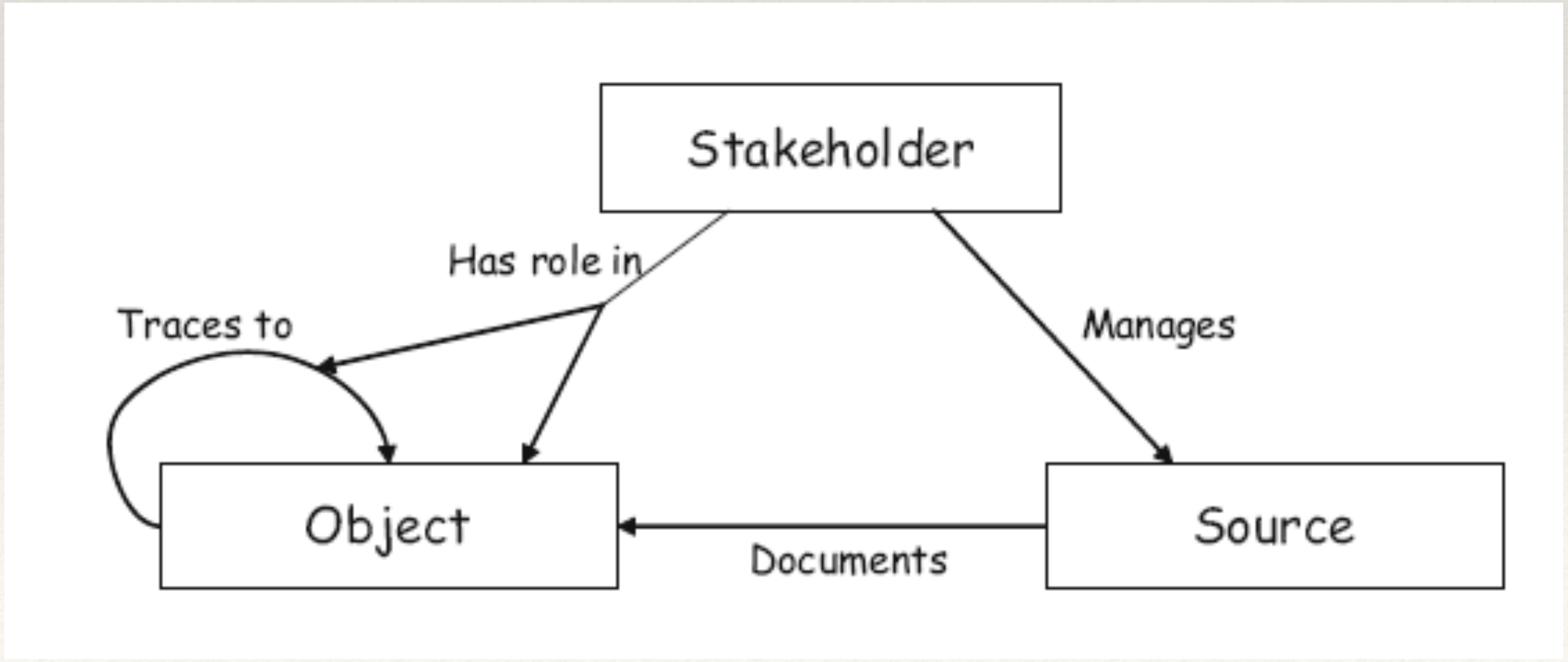
- Post-traceability refers to the life of a requirement after it was included in the requirements specification.
- It enables a better understanding and acceptance of the current system/ software.
- It is concerned with ensuring that all requirements are fulfilled by the system, through the design and implementation of the system, by relating the requirements to the component (C), which helps satisfying that particular requirement.
- No requirements should be lost and none added.
- It also involves relating requirements to test cases, which should be used to ensure that components fulfill those requirements.
- It enables identification of the impact that changes have on design and implementation.

Types of Requirements Traceability

- Traceability can also be divided into horizontal and vertical traceability which refers to whether the related information objects belong to the same type or not.
- Horizontal traceability deals with relating versions or variants of the same type of information, e.g. between requirements or between system components.
- Vertical traceability is concerned with tracing information between previous and subsequent phases in the development process i.e. between information objects of different types.
    - Eg. relating a requirement to the design made based on the requirement, and further to the system component that fulfills the requirement.

Requirements Traceability Meta-Model

- The source is the physical artifact where the information is maintained, e.g. requirements specification document, design document, memorandum, etc.
- The stakeholder is the agent involved in the management of traceability, e.g. the customer, system analyst, and project manager.
- Object refers to the type of information objects that should be related to each other, e.g. requirement, rationale, decision, and system component.

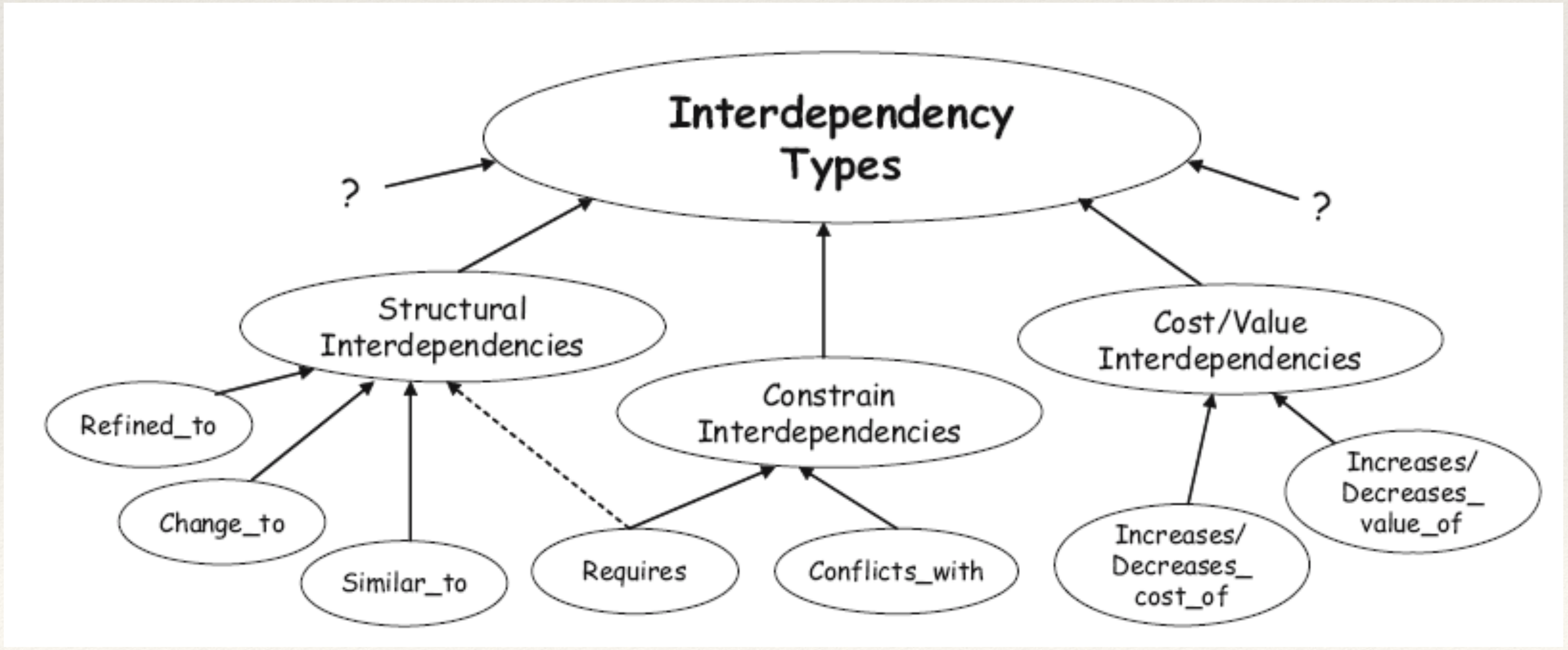


Requirements Traceability Meta-Model

- Source perspective emphasizes the document management as part of traceability, which is important because trace objects available in persistent sources constitute long-term traceability.
- Stakeholders perspective emphasizes the importance of different usage roles when designing and implementing a traceability system. It also provides the ability to define who is responsible for various products and decisions during the development process.
- The model can be used to represent several dimensions of traceability:
  - What information is represented.
  - Where it is represented and how.
  - Who are the stakeholders and what are their roles in the creation and use of the information.
  - Why certain object is created or modified.

Requirements Interdependency Types

- Focuses on objects of one type (requirements) and traceability among them.
- This implies pre-traceability and horizontal traceability.
- Requirements interdependencies is a specific issue in requirements traceability.
- There are several interdependency types identified in the literature:


Interdependency Types
?
?
Structural Interdependencies
Constrain Interdependencies
Cost/Value Interdependencies
Refined_to
Change_to
Similar_to
Requires
Conflicts_with
Increases/ Decreases_ cost_of
Increases/ Decreases_ value_of

Requirements Interdependency Types

- The types in this model can be further elaborated due to project-specific needs.
- The model may be improved, where more fundamental types are found.
- Even when the interdependency types appear to be fairly different they can be difficult to separate in practice.

Example:

The relationship between two requirements when one requirement should be implemented before the other.

- This can be described as a temporal dependency, i.e. one should be implemented before the other.
- On the other hand, it can also be viewed as the second requirement requiring the first one, i.e. that one cannot function without the other.

Structural Interdependencies

- Given a specific set of requirements, they can be organized in a structure where relationships are of a hierarchical as well as of a cross-structure nature.
- High-level business requirements are gradually decomposed into more detailed software requirements, forming a hierarchy.
- There can be structural relationships between requirements within different parts of the overall hierarchy.
- Types:
  - Refined_to
  - Change_to
  - Similar_to

Structural Interdependencies

- *Refined_to*. A higher-level requirement is refined by a number of more specific requirements.
- This dependency type is used to describe hierarchical structures, where more detailed requirements are related to their source requirements.
- These requirements provide further explanation, detail or clarification about the source requirement.
- The source requirement can be seen as an abstraction of the detailed requirements.
- If a detailed requirement is derived from a higher-level requirement, but is not a prerequisite for this requirement, the relationship is of the dependency type refined_to.

Structural Interdependencies

- Refined_to.

Example:

“The system should support a following up of the customer orders after their delivery,”

could be refined by the requirements stating:

- during a following up it should be possible to compare the cost of producing the products related to a given customer order with the manufacturing budgets for those products,
- the system should facilitate changing the manufacturing budgets when following up the products within a customer order.

Structural Interdependencies

- *Changes_to*. One requirement changes to another requirement if a new version of that requirement is developed which replaces the old one.
- This dependency type is used to describe the history of a requirement, i.e., how it has evolved over time since it enables to relate the different versions of a single requirement.
- A new version of a requirement may be developed for several reasons:
  - the result of making the requirement more comprehensive,
  - changing details within the requirement,
  - expressing it more formally.

Example:

- "It should take no longer than 10 seconds to perform a search for contact information" could be changed to
- "It should take no longer than 15 seconds to perform a search for contact information."

Structural Interdependencies

- *Similar_to*. One stated requirement is similar to or overlapping with one or more other requirements.
- This type describes situations where one requirement is:
  - similar to another in terms of how it is expressed
  - of a similar underlying idea of what the system is able to perform
- It can be used to describe similarities both within the requirements and their potential solutions.

Example:

- "The system shall support the management of library items."
- "The system shall provide means to handle books and journals within the library."
- They are similar since both books and journals could be considered as library items.

Constraining Interdependencies

- They describe how requirements can constrain each other or be dependent on each other.
- The following types fall into this category:
    - Requires,
    - Conflicts_with
- *Require*. The fulfillment of one requirement depends on the fulfillment of another requirement.
- If one requirement is to be included into the system, it requires another requirement to be included as well.
- It can also be used to describe hierarchical relations between two requirements of a stronger nature than Refined_to.
- Requires means that one or more detailed requirements are required, i.e. not optional, in order to fulfill a requirement on a higher level.

Example:

If the system should be able to include emailing and web access, a network connection is required.

Constraining Interdependencies

- *Conflicts_with*. A requirement is in conflict with another requirement if they cannot exist at the same time or if increasing the satisfaction of one requirement decreases the satisfaction of another requirement.
- This type includes situations where:
  - it is impossible to implement both requirements,
  - requirements have a negative influence on each other's achievement and a trade-off between the requirements must be made.

Example:

  - "All personnel should be able to search for information about both products and customers"
  - "Only personnel with security status A should be able to search for customers classified as military related"
  - They contradict each other, and cannot be simultaneously satisfied.

Cost/Value Interdependencies

- Cost/value interdependencies are concerned with the costs involved in implementing a requirement in relation to the value that the fulfillment of that requirement will provide to the perceived customer/user.
- The following types fall into this category:
  - Increases/Decreases_cost_of,
  - Increases/Decreases_value_of
- *Increases/Decreases_cost_of.* If one requirement is chosen for implementation, then the cost of implementing another requirement increases or decreases.
- It relates requirements that influence the implementation cost of each other, e.g., by making it more expensive or cheaper to implement another requirement.

Example:

A requirement stating that no response time should be longer than 5s increases the cost of implementing other requirements.

Cost/Value Interdependencies

- *Increases/Decreases_value_of*. If one requirement is chosen for implementation, then the value to the customer of another requirement increases or decreases.
- It focuses on the effect relations between requirements have on the customer value.
- Some requirements may have a positive influence on the customer value of each other, while others have a negative influence (by making functionality more complex).

Example:

The satisfaction of including an agenda in a mobile phone increases if one can synchronize it with agendas on PCs.

Impact Analysis

- Change is an inescapable property of any software, for a number of reasons.
- Factors that can impose changes to requirements are:
  - The problem that the system is supposed to solve changes, (e.g., economic or political reasons).
  - The users change their minds about what they want the system to do.
  - The environment in which the system resides changes.
  - The new system is developed and released leading users to discover new requirements.
- Problems arise if requirements and changes to requirements are not managed:
  - Errors during requirements elicitation lead to a number of requirements changes during subsequent phases
  - Failure to create a practical change management process may mean that:
    - changes cannot be timely handled, or
    - changes are implemented without control

Impact Analysis

- A change must be assessed with regard to the impact on cost and functionality, potential to destabilize the system, and impact on external stakeholders.
- If the potential to destabilization is not considered, the system will deteriorate.
- When a change is introduced in the code, it may affect tests, design, architecture. Requirements are typically ignored.
- To manage hierarchically, changes should be introduced top-down.
- A challenge in software development is the constant change of requirements.
- Change management and impact analysis systematically manage changes, and assess the effect of change requests.

Impact Analysis

- Impact analysis is the activity of identifying what needs to be modified in order to make a change, or to determine the consequences on the system if the change is implemented.
- Definition: "the activity of identifying the potential consequences, including side effects and ripple effects, of a change, or estimating what needs to be modified to accomplish a change before it has been made" (Bohner, S.A., Arnold, R.S., Software change impact analysis, 1996)
- Most research on impact analysis is related to software maintenance.
- Impact analysis is an important part of requirements engineering since changes to software often are initiated by changes to the requirements.
- The output from impact analysis can be used as a basis for estimating the cost associated with a change.
- The cost of the change can be used to decide whether or not to implement it depending on its cost/benefit ratio.

Impact Analysis and Req. Dependencies

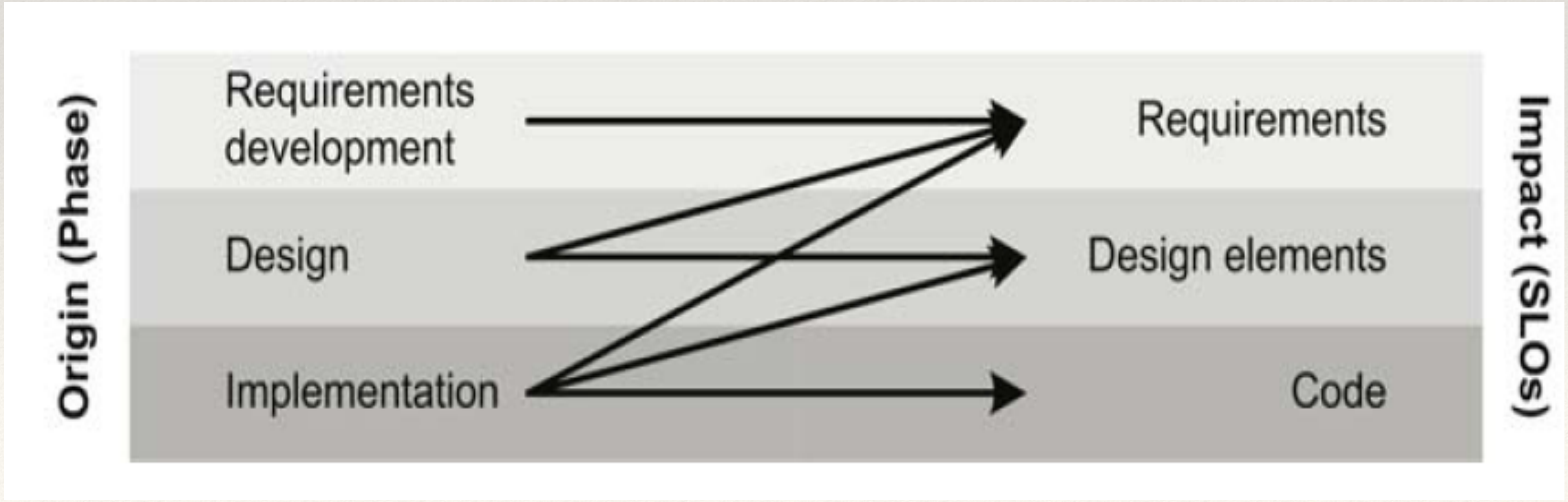
- Requirements interdependencies are useful for impact analysis since they show the evolution of requirements.
- They also show the major assumptions behind a requirement, by relating it to the original source (indicates the importance of the requirement).
- Requirements interdependencies show if/how requirements affect each other.
- This eases the accuracy of impact analysis since one can identify other requirements that need to be changed due to a change request.
- Testing is about ensuring that all the requirements of the system are met.
- The order in which test cases are executed is crucial (some functionality cannot be tested before others are verified).
- Functionality, on which other functionality is based, should be tested first.
- These relationships can be discovered based on requirements dependencies.

Impact Analysis Concepts

- Software life-cycle objects (SLOs) also called software products, or working products. An SLO is an artifact produced during a project, such as a requirement, an architectural component, a class and so on.
    - SLOs are connected to each other through a web of relationships.
    - Relationships can be both between SLOs of the same type, and between SLOs of different types.
    - E.g. two requirements can be interconnected to signify that they are related to each other.
    - A requirement can also be connected to an architectural component to signify that the component implements the requirement.
- Impact analysis is often carried out by analyzing the relationships between various entities in the system.
- Two types of analysis: dependency analysis and traceability analysis.

Impact Analysis Concepts

- *Dependency analysis*: detailed relationships among program entities (i.e., variables or functions) are extracted from source code.
- *Traceability analysis*: is the analysis of relationships that have been identified during development among all types of SLOs.
- Traceability analysis is suitable for analyzing relationships among requirements, architectural components, documentation and so on.



Impact Analysis Concepts

- Sets of impact:
- The System Set represents the set of all SLOs in the system – all the other sets are subsets of this set.
- The *Starting Impact Set* (SIS) represents the set of objects that are initially thought to be changed. The SIS typically serves as input to impact analysis approaches that are used for finding the Estimated Impact Set.
- The *Estimated Impact Set* (EIS) always includes the SIS and can therefore be seen as an expansion of the SIS.
  - The expansion results from the application of change propagation rules to the internal object model repeatedly until all objects that may be affected are discovered.
  - Ideally, the SIS and EIS should be the same, meaning that the impact is restricted to what was initially thought to be changed.
- The *Actual Impact Set* (AIS), contains those SLOs that have been affected once the change has been implemented.
  - In the best-case scenario, the AIS and EIS are the same, meaning that the impact estimation was perfect.

Impact Analysis Concepts

- Two forms of information are also necessary in order to determine the impact of a change:
  - information about the dependencies between objects (references between them),
  - knowledge about how changes propagate from object to object via dependencies and traceability links (expressed in terms of rules or algorithms).
- Two types of change: primary and secondary.
- *Primary change* (or direct impact), corresponds to the SLOs that are identified by analyzing how the effects of a proposed change affect the system.
  - It is difficult to automate as it is mainly based on human expertise.
- *Secondary change* (or indirect impact) can take two forms:
  - Side effects
  - Ripple effects

Impact Analysis Concepts

- *Side effects* are unintended behaviors resulting from the modifications needed to implement the change.
  - Side effects affect both the stability and function of the system and must be avoided.
- *Ripple effects* are effects on some parts of the system caused by making changes to other parts.
  - Ripple effects cannot be avoided, since they are the consequence of the system's structure and implementation.
  - They must, however, be identified and accounted for when the change is implemented.

Strategies for Impact Analysis

- Are divided into two categories: automatable and manual.
- Automatable strategies are in some sense algorithmic in their nature.
  - They have the ability to provide very fine-grained impact estimation in an automated fashion.
  - They require the presence of a detailed infrastructure and result at times in too many false positives. The prerequisite is a structured specification of the system (the specification is consistent and complete, and includes some semantic information).
  - Used to assess the Estimated Impact Set (EIS)
  - Examples: traceability and dependency analysis, and slicing
- Manual strategies are best performed by human beings (as opposed to tools).
  - They require less infrastructure, but may be coarser in their impact estimation than the automatable ones.
  - They are less precise in their predictions of impact.
  - Used for assessing the Starting Impact Set (SIS)
  - Examples: using design documentation and interviewing knowledgeable developers
- The manual strategies are potentially easier to adopt and work with because they require less structured input and no new forms of SLOs need to be developed.

Impact Analysis

- The change process consists of:

1. Problem analysis and change specification
2. Change analysis and costing:
    a. Check change request validity
    b. Find directly affected requirements
    c. Find dependent requirements
    d. Propose requirements changes
    e. Assess costs of change
    f. Assess cost acceptability
3. Change implementation

- Impact Analysis is performed in steps 2b, 2c and 2e

Impact Analysis Metrics

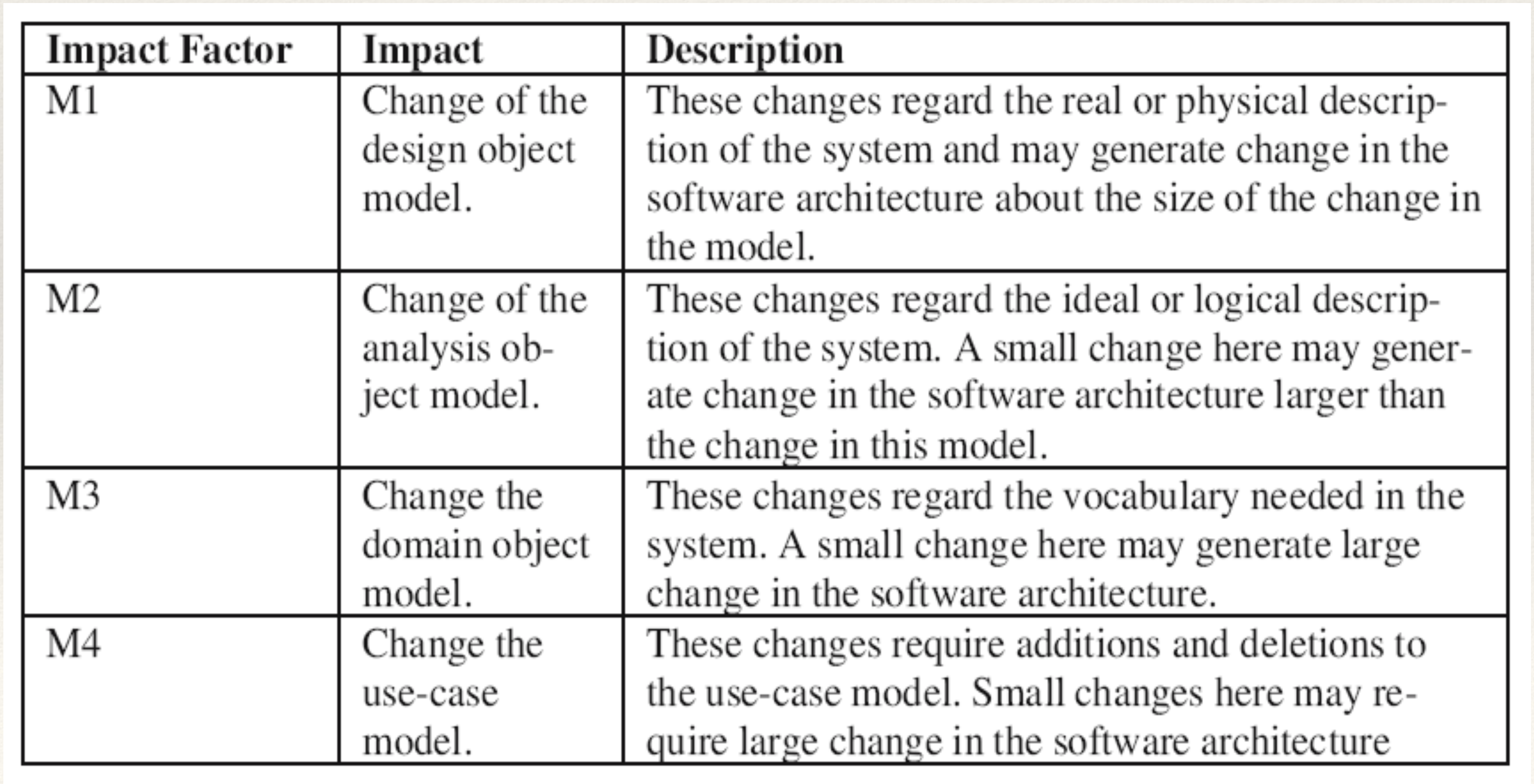
- Metrics are useful in impact analysis for various reasons:
  - They can be used to measure and quantify change caused by a new or changed requirement at the point of the impact analysis activity.
  - They can be used to evaluate the impact analysis process itself once the changes have been implemented.
- Metrics for Quantifying Change Impact
  - Are based on the SLOs that are predicted to be changed as an effect of new or changed requirements.
  - Indicators of how severe the change is can be used.
  - They can be used to estimate the cost of a proposed change or a new requirement.
  - The more requirements and other SLOs that are affected, the more widespread they are and the more complex the proposed change is, the more expensive the new or changed requirement will be.
  - Requirements that are costly in this sense but provide little value can be filtered out for the benefit of requirements that provide more value but to a smaller cost.

Impact Analysis Metrics

- *Change impact* can be measured based on the set of requirements that is affected by the change.
    - Eg. the number of requirements affected by a change can be counted based on this set.
    - The affected requirements' complexity often determines how severe the change is and can be measured in various ways (i.e., the size of each requirement in terms of function points and the dependencies of each requirement on other requirements).
    - For other SLOs, the metrics are similar.
    - For architecture and design, measures of impact include the number of affected components, the number of affected classes or modules, and number of affected methods or functions.
    - For source code, low-level items such as affected lines of code can be measured and the level of complexity for components, classes, and methods can be measured using standard metrics such as cyclomatic complexity and regular object-oriented metrics.

Impact Analysis Metrics

- The impact factor is useful to determine how severe or costly a change is.
- It is based on empirical findings in which it was determined that changes to different types of SLOs can be used as an indicator of the extent of the change.
- The higher the impact factor, the more severe the change.



Impact Factor	Impact	Description
M1	Change of the design object model.	These changes regard the real or physical description of the system and may generate change in the software architecture about the size of the change in the model.
M2	Change of the analysis object model.	These changes regard the ideal or logical description of the system. A small change here may generate change in the software architecture larger than the change in this model.
M3	Change the domain object model.	These changes regard the vocabulary needed in the system. A small change here may generate large change in the software architecture.
M4	Change the use-case model.	These changes require additions and deletions to the use-case model. Small changes here may require large change in the software architecture

Metrics for Evaluation of Impact Analysis

- Various classifications on the impact sets:
- They are relations between the cardinalities of the impact sets, and can be seen as indicators of the effectiveness of the impact analysis approach employed.
- Bohner and Arnold:
- #SIS / #EIS, i.e. the number of SLOs initially thought to be affected over the number of SLOs estimated to be affected (primary change and secondary change).
  - A ratio close to 1 is desired, as it indicates that the impact is restricted to the SLOs in SIS.
  - A ratio much less than 1 indicates that many SLOs are targeted for indirect impact, which means that it will be time-consuming to check them.
- #EIS / #System, i.e. the number of SLOs estimated to be affected over the number of SLOs in the system.
  - The desired ratio is much less than 1, as it indicates that the changes are restricted to a small part of the system.
  - A ratio close to 1 would indicate either a faulty impact analysis approach or a system with extreme ripple effects.

Metrics for Evaluation of Impact Analysis

- #EIS / #AIS, i.e. the number of SLOs estimated to be affected over the number of SLOs actually affected.
  - The desired ratio is 1, as it indicates that the impact was perfectly estimated.
  - In reality, it is likely that the ratio is smaller than 1, indicating that the approach failed to estimate all impacts.
  - Two special cases are if AIS and EIS only partly overlap or do not overlap at all, which also would indicate a failure of the impact analysis approach.
- Fasolino and Visaggio: based on the cardinalities of the impact sets
  - Inclusiveness, Amplification, ChangeRate, S-Ration
- Lam and Shankararaman: are not related to the impact sets
  - Quality deviation, Defect count, Dependency count