Requirements Engineering

2019/2020

Course 1


We need the new airplane model ASAP to beat our competition.
CEO
Shouldn't we stop and get the product requirements first?
There's no time for that! We've been building planes for years; just get the new version out the door before the competition does!
CEO
Later...
Um, ... how do we get in?
?!

"The hardest single part of building a software system is deciding precisely what to build." *

*Frederick P. Brooks Jr., *The Mythical Man-Month: Essays on Software Engineering*. Addison-Wesley, Reading, MA, 1995.

Requirements Engineering (RE)

- It addresses the critical problem of identifying and designing the right software for the customer.
- It is considered as one of the most crucial stages in software design and development.
- Software requirements are the critical determinants of software quality.

Requirements Engineering

- Studies (Standish group report - US projects) - Chaos
- 1995 Report
  - 52.7% of projects cost 189% of their original budget estimates (only 42% of the original features were implemented).
  - 16.1% of all US software projects are developed on-schedule, on-budget and with all originally planned features, while 31.1% of projects are terminated before completion.
  - Main problems: poor requirements, low user involvement and unclear objectives
- 1999 Report
  - three of the top ten reasons for project failure: lack of user involvement, unstable requirements and poor project management.

Requirement Engineering

- 2001 Report :
  - unstable requirements and poor project management still among the primary reasons for project failure
- 1996 Europe survey:
  - 3800 organizations from over 17 countries in Europe
  - Most problems are in the area of requirements specifications (50%) and requirements management (50%)
- 2002 UK analysis
  - requirements problems accounted for 48% of all software problems encountered

Content

- Introduction: Basic concepts, terminology
- Requirements Elicitation
- Types of Requirements
- Requirements Specification
- Requirements Prioritization
- Requirements Management
- Requirements Negotiation
- Quality Assurance

Bibliography

1. A. Aurum, C. Wohlin –*Engineering and Managing Software Requirements*, Springer, 2005
2. B. Berenbach, D. Paulish a.o. – *Software & Systems Requirements Engineering: In practice*, McGraww Hill, 2009
3. E.Hull, K. Jackson, J. Dick – *Requirements Engineering*, Springer, 2005
4. R. Young – *The Requirement Engineering Handbook*, Artech House, 2004
5. C. Williams, M. Kaplan, T. Klinger, A. Paradkar, "*Toward Engineered, Useful Use Cases*"
6. *Behavior Driven Development* http://dannorth.net/introducing-bdd/
7. *Business Motivation Model* (BMM) http://www.omg.org/spec/BMM/
8. S. Robertson, M. Robertson, *Mastering the Requirements Process*, Addison Wesley, 3rd edition, 2013
9. Karl Wiegers, Joy Beatty, *Software Requirements (3rd Edition)*, Microsoft Press; 2013

Course - Final mark

- Written exam: 60%
- Project: 40%
- Both marks must be at least 5

Project

- You are responsible for the requirements engineering of a software system.
- You may work in teams (at most 4 students).
- You must also develop a prototype using the requirements obtained.
- Deadlines:
  - Requirements elicitation (E): **week 4** (October 22, 2019)
  - Requirements specification (RS): **week 8** (November 19, 2019)
  - Design and prototype (DP): **week 14** (January 14, 2020)

- Project evaluation: 30% E, 50% RS, 20% DP
- 3 points penalty for each missed deadline

Project Theme

- **Students' Internship**.
- Each member of a team has to discuss with at least 3 different stakeholders (end-users).
- Only one stakeholder should have computer science background.

Course 1 outline

- Definition
- Basic concepts
- Activities
- Artifacts

RE Definition (1)

- "Requirements engineering involves all lifecycle activities devoted to identification of user requirements, analysis of the requirements to derive additional requirements, documentation of the requirements as a specification, and validation of the documented requirements against user needs, as well as processes that support these activities." *
- It is a domain neutral discipline: software, hardware, and electromechanical systems.

* U.S. Department of Defense, Software Technology Strategy, December 1991.

RE Definition (2)

- The "science and discipline concerned with establishing and documenting software requirements." *

* Thayer, R. and Dorfman, M., Software Requirements Engineering, 2nd ed., IEEE Computer Society Press, 1997.

- RE deals with all phases of a project or product life cycle from innovation to obsolescence (innovation → development → release→ maintenance→ obsolescence).

Requirements

- Requirements are descriptions of how a software product should perform.
- A requirement typically refers to some aspect of a new or enhanced product or service.
- Definition*

  (1) A condition or capability 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.

  A documented representation of a condition or capability as in (1) or (2).

*[IEEE-STD 610.12, *Standard Glossary of Software Engineering Terminology*, Institute of Electrical and Electronics Engineers, 1990]

Requirements

- A requirement is a necessary attribute in a system, a statement that identifies a capability, characteristic, or quality factor of a system in order for it to have value and utility to a customer or user.
- A requirement is a collection of needs arising from the user and various other stakeholders (general organization, community, government bodies and industry standards), all of which must be met.
- The requirements should be independent of design, showing “what” the system should do, rather than “how” it should be done.
- Stated vs real requirements:
  - Stated requirements are those provided by a customer at the beginning of a system or software development effort.
  - Real requirements are those that reflect the verified needs of users for a particular system or capability.

Stakeholders

- Are "… those participants in the development process together with any other individuals, groups or organizations whose actions can influence or be influenced by the development and use of the system whether directly or indirectly" *.

- Typical stakeholders
  - business owners
  - product managers
  - various types of users and administrators from the client side,
  - software team members from the software development side.

*Pouloudi A, Whitley EA (1997) *Stakeholder identification in inter-organizational systems: Gaining insights for drug use management systems*. European Journal of Information Systems, vol 6: pp. 1--14

Requirements truths

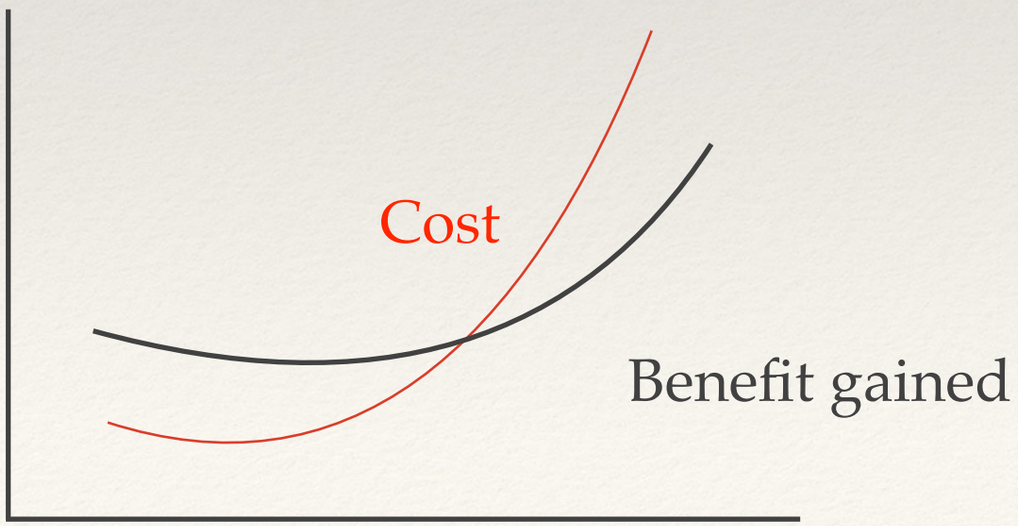
1. Requirements are not really about requirements.
    * They are what the software product, or hardware product, or service, etc is meant to do and to be.
    * They exist whether you discover them or not.
    * They exist whether you write them or not.
    * The product will be right only if it conforms to the requirements.

Requirements truths

2. If a software must be built, it must be optimally valuable to the owner.

   * The owner will not pay unless the product provides a benefit (some capability that was not available before, some business process becomes faster or cheaper, etc.)

   * To be optimally valuable, the product must provide a benefit that is in the proportion to the cost of the product.



Requirements truths

3. If the software is meant to satisfy a need, you have to know what that need is to build the right software.

   * You must understand the work of the owner's business and determine how that work should be carried out in the future.

   * Any kind of work: scientific, commercial, e-commerce, social networking, etc

   * You must come to the correct understanding of the requirements (the client agrees with), or the product/project will be deficient.

Requirements truths

4. There is an important difference between building a piece of software and solving a business problem. The former does not necessarily accomplish the latter.

    * In order for the software to be valuable to the business owner, it must solve the owner's business problem.

    * Any development effort must start with the problem, and not with a perceived solution.

Requirements truths

5. The requirements do not have to be written, but they have to become known to the builders.

    * Large specifications are not necessarily better:
        * few readers understand them
        * fewer have the patience to read it
    * The requirements need to be communicated to the development team.
    * Writing requirements down helps understanding them (ambiguity, clarification, correctness)

Requirements truths

6. The customer will not always give the right answer. Sometimes it is impossible for the customer to know what is right, sometimes he just doesn't know what he needs.

- Stakeholders have difficulties in expressing what they need.
- *Incremental improvements* problem:
  - existing system+a few improvements
  - no serious innovation =>failed product

Requirements truths

7. The requirements do not come about by chance. There needs to be some kind of orderly process for developing them.

    * Any important endeavor needs an orderly process.
    * Orderly processes comprise a set of tasks that achieve the intended result, but leave the order, emphasis, and degree of application to the person or team using the process.

Requirements truths

8. You can be as iterative as you want, but you still need to understand the business needs.

    * Iterative development do not make requirements redundant.
    * Any development technique has a need to discover the requirements as a prerequisite to serious development.
    * Requirements processes have been absorbed into development life cycles.

Requirements truths

9. There is no silver bullet. All the methods and tools will not compensate for poor thought and poor workmanship.
    - An orderly process is not a substitute for thinking.
    - The automated tools available help the requirements activity, but they should be seen as aids and not as substitutes for good requirements practices.

Requirements truths

10. Requirements, if they are to be implemented successfully, must be measurable and testable.

    * Requirements should be precise.
    * Humans are not always precise.
    * If you can measure the requirement using numbers instead of words, you can make the requirement testable.
    * If you cannot find a measurement for a requirement, then it is not a requirement.

Requirement characteristics

- A good requirement should be [IEEE Standard 830, IEEE Recommended Practice for Software Requirements Specifications, 1998]:
  - Feasible
  - Correct (valid)
  - Unambiguous
  - Verifiable
  - Modifiable
  - Consistent
  - Traceable
- Other project- or product-specific characteristics:
  - compliance with specific regulations,
  - meeting electrical safety requirements,
  - etc.

Feasible

- A requirement is feasible if an implementation of it on the planned platform is possible within the constraints of the program or project.
- A requirement is feasible if and only if it can be accomplished given the resources, budget, skills, schedule, and technology available to the project team.

- E.g. a requirement to handle 10,000 transactions per second
  - it might be feasible given current technologies,
  - it might not be feasible with the selected platform or database manager.

Valid / Correct

- A requirement is valid if and only if the requirement is one that the system shall (must) meet.
- Determination of validity is normally accomplished by review with the stakeholders who will be directly responsible for the success or failure of the product.
- “must” vs “nice to have” requirements:
  - must: they are actually needed to make the project or product a success.
  - nice to have: wishful thinking. They add cost without adding value. They can delay project completion.

Remark:

The IEEE Standard 830 uses the term “correct” (without error).

A “valid” requirement may be exactly what the customer wants, but it may still contain errors or be an inappropriate solution.

Unambiguous

- A requirement is unambiguous if it has only one interpretation.
- Natural language tends toward ambiguity.
- E.g.: “The data complex shall withstand a catastrophe (fire, flood).”
  - “The data complex shall withstand a catastrophe of type fire or flood”.
  - “The data complex shall withstand any catastrophe, two examples being fire and flood.”
  - Rephrased: “The data complex shall be capable of withstanding a severe fire. It shall also be capable of withstanding a flood.”

Verifiable

- A requirement is verifiable if the finished product or system can be tested to ensure that it meets the requirement.
- Product features are almost always abstract and thus not verifiable.
- E.g. “The car shall have power brakes.” (it does not have sufficient details)
  - “The car shall come to a full stop from 60 miles per hour within 5 seconds”.

Modifiable

- The characteristic modifiable refers to two or more interrelated requirements or a complete requirements specification.
- A requirements specification is modifiable if its structure and style are such that any changes to a requirement can be made easily, completely, and consistently while retaining the structure and style.
- The requirements specification should have a coherent, easy-to follow organization with no redundancy (e.g., the same text appearing more than once). It should keep requirements distinct rather than intermixed.

Rule:

> Information in a set of requirements should be in one and only one place so that a change to a requirement does not require cascading changes to other requirements.

Consistent

- Consistency is a relationship among at least two requirements.
- A requirement is consistent if it does not contradict or is not in conflict with any external corporate documents or standards or other product or project requirements.
- Contradiction occurs when the set of external documents, standards, and other requirements result in ambiguity or a product is no longer feasible to build.
- E.g. conflicting requirements
  - Company standard requires all user interfaces to have a logo in upper right corner of the screen.
  - A user interface requirement specifies that the logo must be at the bottom center of the screen.

Traceable

- Requirements traceability is the ability to describe and follow the life of a requirement, in both a forward and backward direction, i.e., from its origins, through its development and specification, to its subsequent deployment and use, and through periods of ongoing refinement and iteration in any of these phases.
- A requirement is traceable if the source of the requirement can be identified, any product components that implement the requirement can easily be identified, and any test cases for checking that the requirement has been implemented can easily be identified.

Requirements Specification

- A requirements specification is a filtered set of requirements.
- "Given two requirements specifications, how would you quantitatively determine that one is better than the other?"
- Characteristics:
  - *feasible* -- if building the product specified is feasible given the state of technology, the budget, and time.
  - *unambiguous* -- if there is no pair-wise ambiguity in the specification.
  - *valid* -- if every requirement in it is valid.
  - *verifiable* -- if every requirement in it is verifiable.
  - *modifiable* -- if there is no redundancy, and changes to requirements are easily and consistently made; e.g., a change to one requirement does not require cascading changes to other requirements.
  - *consistent* -- if the requirement set is internally consistent.
  - *traceable* -- if every requirement in it can be traced back to its source and forward to test cases.
  - *concise* -- if the removal of any requirement changes the definition of the product or system.

Completeness

- A requirements specification is complete if it includes all relevant correct requirements, and sufficient information is available for the product to be built.
- A requirements specification is complete if it includes the following elements [IEEE Standard 830]:
  - Definition of the responses of the system or product to all realizable classes of input data in all realizable classes of situations. It is important to specify the responses to both valid and invalid input values and to use them in test cases.
  - Full labels and references to all figures, tables, and diagrams in the specification and definitions of all terms and units of measure.
  - Quantification of the nonfunctional requirements. Testable, agreed-on criteria must be established for each nonfunctional requirement.

RE Activities

- Requirements-related activities :
  - *Identifying the stakeholders*: Who has an interest in the system or in its possessing qualities that meet particular needs?
  - *Gaining an understanding of the customers' and users' needs for the planned system and their expectations of it*. (requirements elicitation)
  - *Identifying requirements*: Stating requirements in simple sentences and providing them as a set.
  - *Clarifying and restating the requirements*: It is done to ensure that they describe the customer's real needs and are in a form that can be understood and used by developers of the system.
  - *Analyzing the requirements*: It is done to ensure that they are well defined and that they conform to the criteria of a good requirement.
  - *Defining the requirements in a way that means the same thing to all of the stakeholders*: Each stakeholder group may have a significantly different perspective of the system and the system's requirements. Sometimes this requires investing significant time learning a special vocabulary or project lexicon.

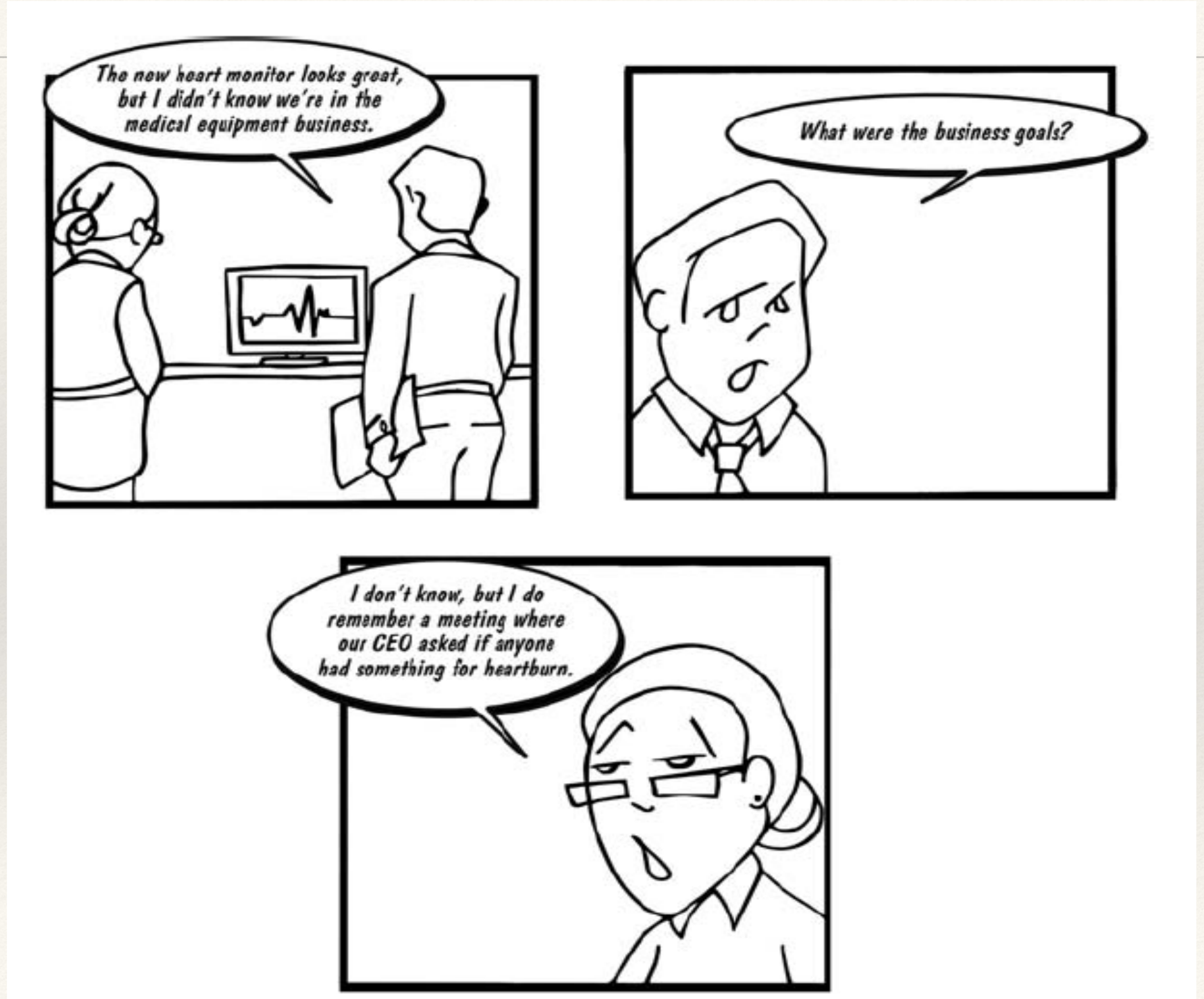
RE Activities

- *Specifying the requirements*: It requires including all of the precise details of each requirement so that it can be included in a specification document or other documentation, depending on the size of the project.
- *Prioritizing the requirements*: All requirements are not of equal importance to the customers and users of the planned system. Some are critical, some of relatively high priority, some of normal or average priority, and some even of lower priority. It is important to prioritize all of the requirements because of limited time and funding.
- *Deriving requirements*: There are some requirements that come about because of the design of a system, but do not provide a direct benefit to the end user. Eg. a requirement for disc storage might result from the need to store a lot of data.
- *Partitioning requirements*: You categorize requirements as those that can be met by hardware, software, training, and documentation.

RE Activities

- *Allocating requirements*: You allocate requirements to different subsystems and components of the system. The allocations may not always be satisfied by just one subsystem or component.
- *Tracking requirements*: You need the capability to trace or track where in the system each requirement is satisfied, so that you can verify that each requirement is being addressed. It is most often accomplished through use of an automated requirements tool.
- *Managing requirements*: You need to be able to add, delete, and modify requirements during all of the phases of system design, development, integration, testing, deployment, and operation.
- *Testing and verifying requirements*: It is the process of checking requirements, designs, code, test plans, and system products to ensure that the requirements are met.
- *Validating requirements*: It is the process for confirming that the real requirements are implemented in the delivered system. The order of validation of requirements should also be prioritized (limited funding).

RE Artifacts



RE Artifacts

- RE artifacts are used to support product design and project management decisions throughout the entire product life cycle.

- The quality and appropriateness of these artifacts is a key factor for successful system development.

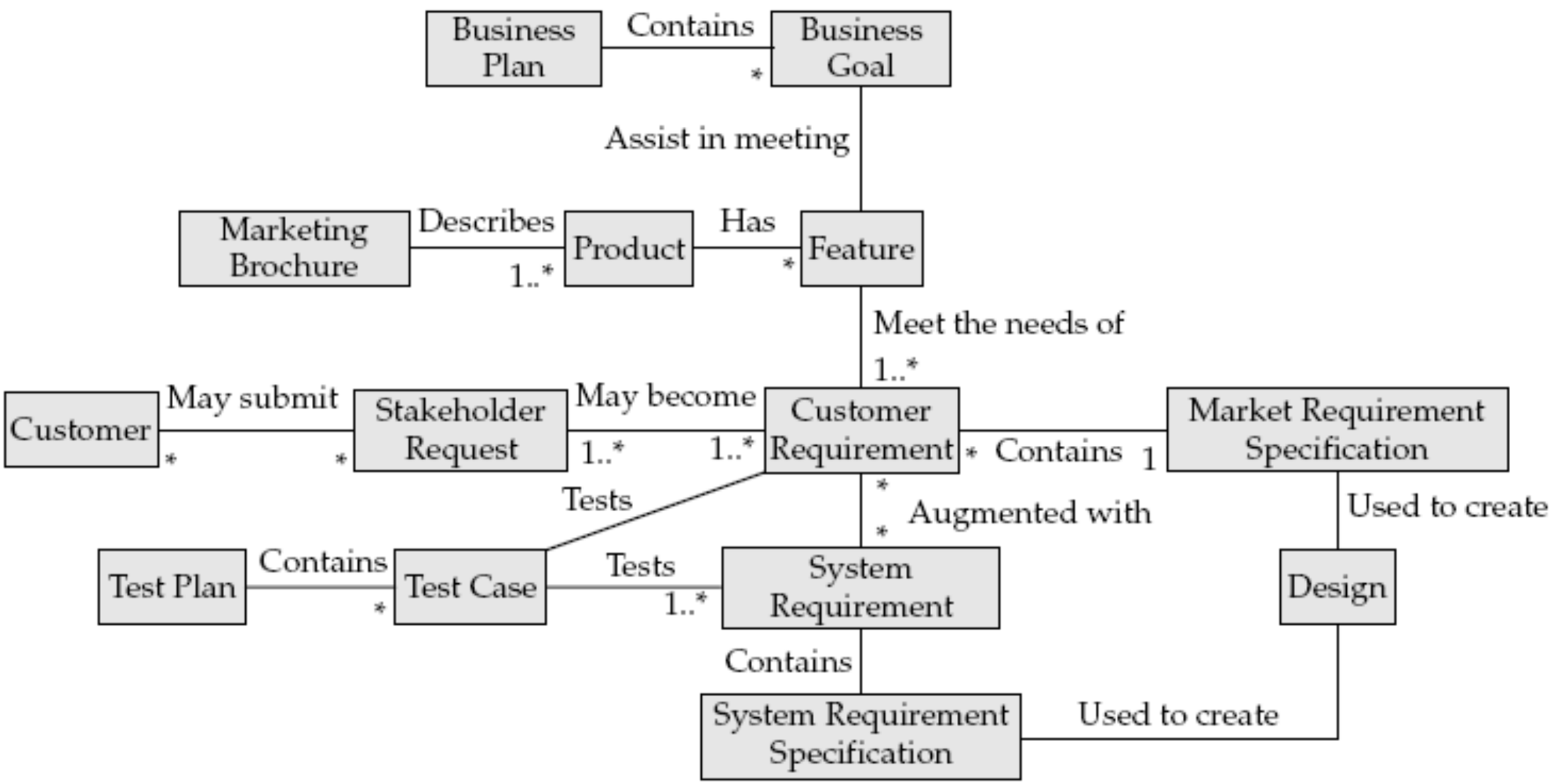
- RE artifacts (or work products), their interdependencies.

Elements of an Artifact Model

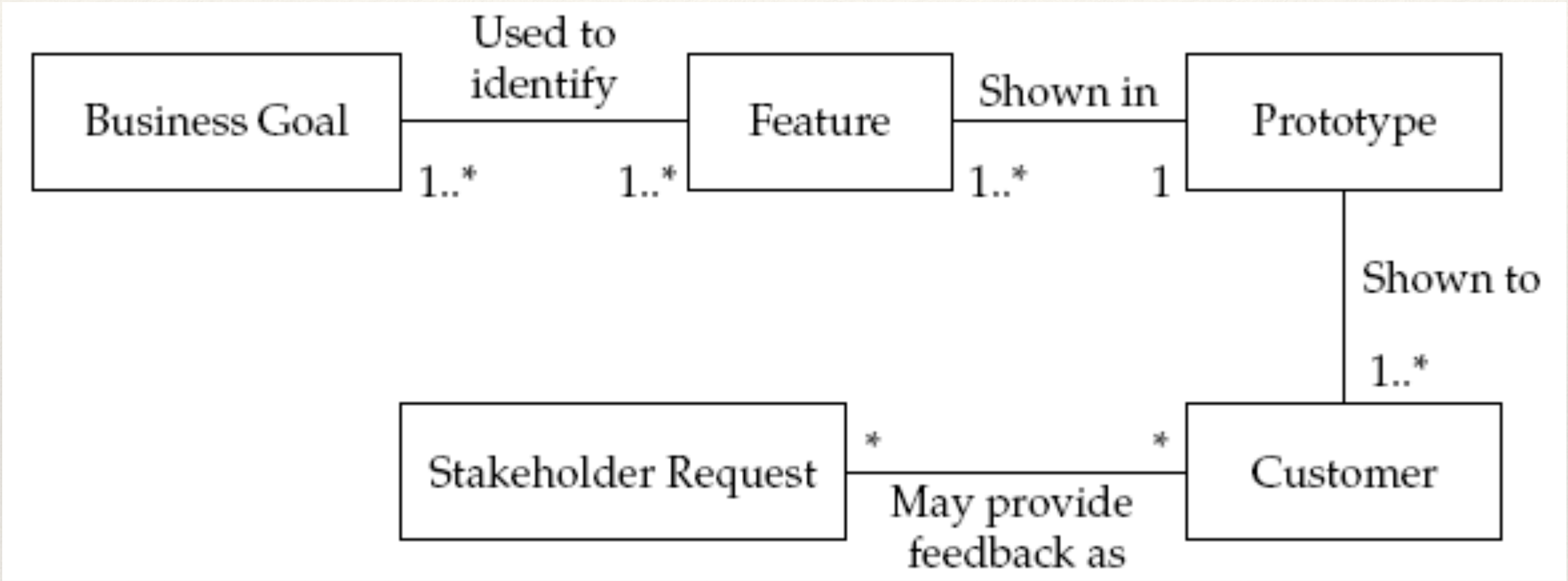
RE artifacts model consists of:

- *Artifact* -- A rectangular box with the name of the artifact. The definition of each artifact should be in a glossary or taxonomy accompanying the model.
- *Association* -- A line connecting two artifacts. The line indicates that there is a relationship between the artifacts. Every association must be labeled to indicate the relationship between the artifacts.
- *Cardinality* -- The cardinality indicates quantities. Any numbering convention can be used if appropriately defined. The UML 3 notation is typically used. If the cardinality is not specified on an association, then unity is implied.

Elements of an Artifact Model



Project-specific models



RE Process

- *RE activities*: elicitation, interpretation and structuring (analysis and documentation), negotiation, verification and validation, change management and requirements tracing.
- There are several process models available to describe the requirements engineering process.
- The process itself is often depicted in different forms, including linear, incremental, non-linear and spiral models.
- Studies have found that projects were generally handled by following a linear model, with some iteration of activities.
- There is no standard requirement engineering process defined, yet.
- Depends on the projects, companies, etc.