Requirements engineering is systematic use of proven principles, techniques, languages, tools for the cost effective analysis, documentation, and on-going evolution of user needs and the specification of the external behavior of a system to satisfy those user needs. Notice that like all engineering disciplines, requirements engineering is not conducted in a sporadic, random or otherwise haphazard fashion, but instead is the systematic use proven approaches.

5.1-SW Requirements Analysis

Software requirements analysis may be divided into five areas of effort:

(i) Problem recognition,

(ii) Evaluation and analysis,

(iii) Modeling,

(iv) Specification, and

(v) Review.

Requirements analysis is a software engineering task that bridges the gap between system engineering and software design.

5.2-Requirements Elicitation for Software

Before the requirements can be analyzed, modeled, or specified they must be gathered through the elicitation process. A customer has a problem that may be amenable to a customer-based solution. A developer responds to the customer's request for help. Communication has begun. But, as we have already noted, the road from communication to understanding is often full of potholes.

5.2.1-Initiating the Process

Yet, communication must be initiated. Gause and Weinberg suggest that the analysis start by asking *context-free*. That is, a set of question that will lead to a basic understanding of the problem, the people who want a solution, the nature of the solution that is desired, and the effectiveness of the first encounter itself. The first set of context-free questions focuses on the customer, the overall goals, and the benefits.

The analyst might ask the following. These questions will help to "break the ice" and initiate the communication that is essential to successful analysis:

→These questions help to identify all stakeholders who will have interest in the software to be built.

- Who is behind the request for this work?
- Who will use the solution?
- What will be the economic benefit of a successful solution?
- Is there another source for the solution that you need?

→These questions enable the analyst to gain a better understanding of the problem and the customer to voice his or her perceptions about a solution.

- How would you characterize "good" output that would be generated by a successful solution?
- What problem(s) will this solution address?
- Can you show me (or describe) the environment in which the solution will be used?
- Will special performance issues or constraints affect the way the solution is approached?

→The final set of questions focuses on the effectiveness of the meeting.

- Are you the right person to answer these questions? Are your answers "official"?
- Are my questions relevant to the problem that you have?
- Am I asking too many questions?
- Can anyone else provide additional information?
- Should I be asking you anything else?

5.2.2-Facilitated Application Specification Techniques (FAST)

What makes a FAST different from the ordinary meeting? There are different meetings:

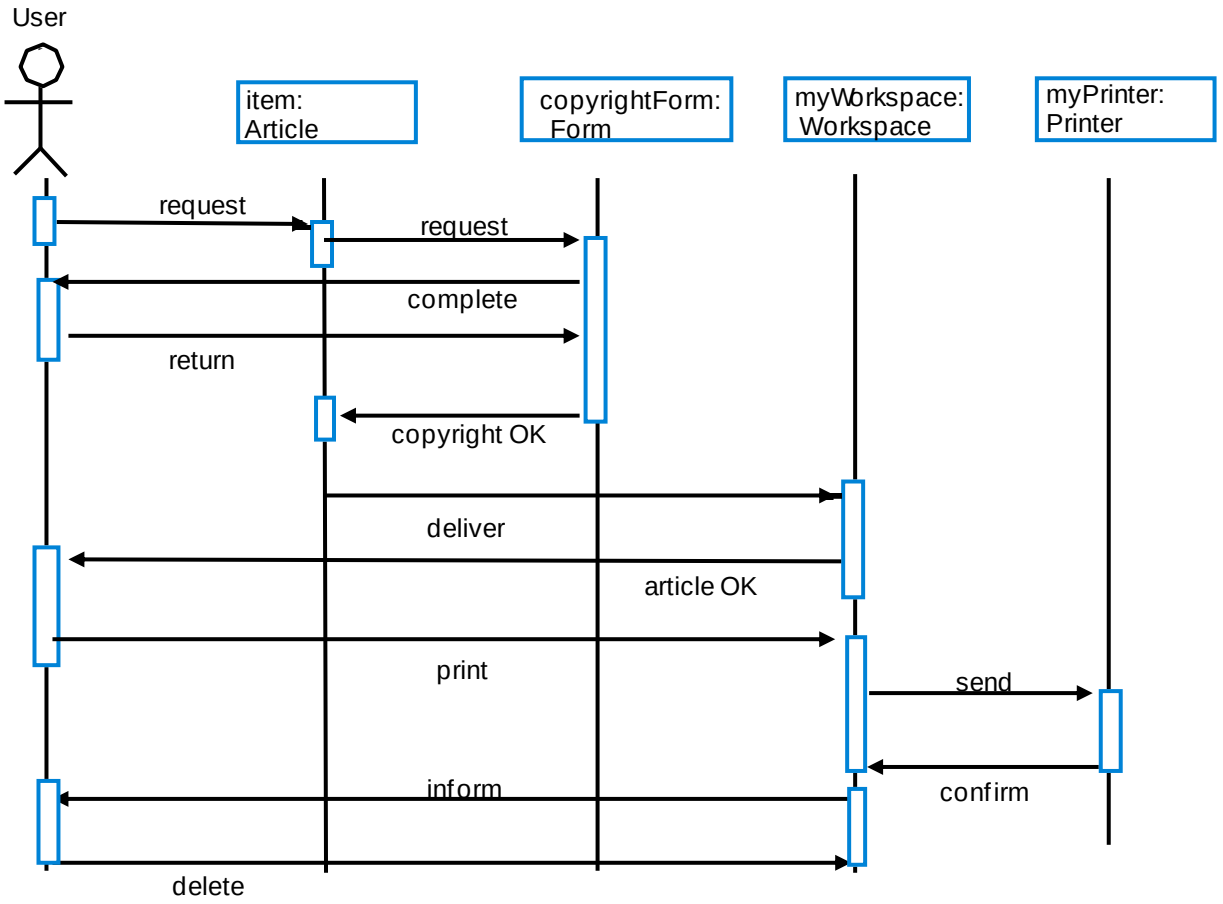
-First meeting: Attendees from both development and customer/user organization are invited to attend. The product request is distributed to all attendees before the meeting date.

-Second meeting: All developers are divided into small groups to meet individually to do specification on separate task.

-Final meeting: All groups of developer attend the meeting altogether to present their specs.

5.2.3-Use-Cases

- Use-cases are a scenario based technique in the UML which identify the actors in an interaction and which describe the interaction itself.
- A set of use cases should describe all possible interactions with the system.
- Sequence diagrams may be used to add detail to use-cases by showing the sequence of event processing in the system.



5.3-Analysis Principles

→Davis' Principles

"If a company wishes to let a contract for a large software development project, it must define its needs in a sufficiently abstract way that a solution is not pre-defined. The requirements must be written so that several contractors can bid for the contract, offering, perhaps, different ways of meeting the client organization's needs. Once a contract has been awarded, the contractor must write a system definition for the client in more detail so that the client understands and can validate what the software will do. Both of these documents may be called the *Requirements document* for the system"