Software Engineering

SE principles, Requirement Engineering

Dr. Amit Kumar Mondal,
Associate Professor, CSE KU

Beginning of Software Engineering

- The need for an engineering approach to developing software was first suggested at a NATO conference in 1968 (Nauer & Randell, 1969).

- An engineering approach to software development is characterized by
  - the application of scientific principles, methods, models, standards and theories
  - that make it possible to manage plan, model, design, implement, measure, analyze, maintain and evolve a software system.
- Ideally software engineering results in the economical production of quality software.

Current of Software Engineering

U.S. Department of Defense

News Sp

DOD Looking for Advanced Command, Control Solution

June 4, 2021 | By David Vergun , DOD News |

You have accessed part of a historical collection on defense.gov. Some of the information contained within may be outdated and links may not function. Please contact the DOD Webmaster with any questions.

The Joint All-Domain Command and Control's strategy goal is to link networks and sensors to warfighters with shared data in all domains — cyber, land, sea, air and space — across all of the military services and combatant commands in a secure manner and at great speed.

Marine Corps Lt. Gen. Dennis A. Crall, the Joint Staff's director of command, control, communications and computers — commonly called the J-6 — and the chief information

Current of Software Engineering

US Joint All Domain Command and Control (JADC2) Market

Published Date: Aug 2023 | Report Code: AS 8763

advance, the JADC2 Airborne Platform's ability to seamlessly integrate with other domains, including land, sea, space, and cyberspace, positions it as a linchpin for achieving comprehensive and effective joint command and control.

Based on Solution, the software segment is projected to grow at the highest CAGR during the forecast period.

The increasing prominence of JADC2 software is driven by its pivotal role in revolutionizing modern military operations. JADC2 Software serves as the digital nerve center, orchestrating seamless coordination across various domains, including land, air, sea, space, and cyberspace. In this age of information-driven warfare, quick and informed decision-making is the linchpin of success. JADC2 Software empowers commanders with real-time insights derived from vast data streams, enhancing situational awareness and enabling agile responses to dynamic threats.

Key Players-

Lockheed Martin Corporation (US), Bae Systems (UK), Raytheon Technologies Corporation (US), Northrop Grumman Corporation (US), L3Harris Technologies, Inc (US), and General Dynamics Corporation are some of the leading companies in the US JADC2 market. These companies have well-equipped manufacturing facilities and strong distribution networks across North America, Europe, Asia Pacific, Latin America, and Middle East & Africa.

Current of Software Engineering

AI companies will need to start reporting their safety tests to the US government

The Biden administration will start implementing a new requirement for the developers of major artificial intelligence systems to disclose their safety test results to the government

By JOSH BOAK Associated Press
January 29, 2024, 4:01 AM

Ben Buchanan, the White House special adviser on AI, said in an interview that the government wants "to know AI systems are safe before they're released to the public — the president has been very clear that companies need to meet that bar."

The software companies are committed to a set of categories for the safety tests, but companies do not yet have to comply with a common standard on the tests. The government's National Institute of Standards and Technology will develop a uniform framework for assessing safety, as part of the order Biden signed in October.

Software Engineering Knowledge [ch4]

- Many software practitioners think of software engineering knowledge almost exclusively as knowledge of specific technologies: Java, Perl, html, C, Linux, Windows NT, and so on. Knowledge of specific technology details is necessary to perform computer programming.

- software development knowledge has a 3-year half-life:
    - -half of what you need to know today will be obsolete within 3 years.
    - In the domain of technology-related knowledge, that's probably about right.
- But there is another kind of software development knowledge—a kind that I think of as "software engineering principles"—that does not have a three-year half-life.
- These software engineering principles are likely to serve a professional programmer throughout his or her career.

Development Team

Activities

Artifacts

Basic Principles

Source Code

Commit Message

Requirements/issue

Developers Discussion

Software Document

Release Notes
Change Logs

Software Engineering Core Principle [ch4]

- **Principle 1. *Be agile.*** Whether the process model you choose is prescriptive or agile,
  - the basic tenets of **agile development should govern your** approach.
  - Every aspect of the work you do **should emphasize economy of action**—keep your technical approach as simple as possible, keep the work products you produce as concise as possible, and make decisions locally whenever possible.

- **Principle 2. *Focus on quality at every step.*** The exit condition for every **process activity, action, and task should focus on the quality** of the work product that has been produced.

Software Engineering Core Principle [ch4]

- **Principle 3.** ***Be ready to adapt.*** Process is not a religious experience, and dogma has no place in it. When necessary,
  - adapt your approach to constraints imposed by the problem, the people, and the project itself.

- **Principle 4.** ***Build an effective team.*** Software engineering process and practice are important, but the **bottom line is people**. Build a self-organizing team that has mutual trust and respect.

Software Engineering Core Principle [ch4]

- **Principle 5.** ***Establish mechanisms for communication and coordination.*** Projects fail because **important information falls into the cracks and/or stakeholders fail** to coordinate their efforts to create a successful end product. These are management issues and they must be addressed.

- **Principle 6.** ***Manage change.*** The approach may be either formal or informal, but mechanisms must be established to manage the way changes are requested, assessed, approved, and implemented.

Software Engineering Core Principle [ch4]

- **Principle 7.** ***Assess risk.*** Lots of things can go wrong as software is being developed. It's essential that you establish contingency plans.

- **Principle 8.** ***Create work products that provide value for others.*** Create only those work products that provide value for other process activities, actions, or tasks. Every work product that is produced as part of software **engineering practice will be passed on to someone else**. A **list of required functions and features** will be passed along to the person (people) **who will develop a design**, the design will be passed along to those who **generate code**, and so on. Be sure that the work product imparts the necessary information without ambiguity or omission.

Requirement Engineering [ch5]

- Designing and building computer software is challenging, creative, and just plain fun.
- building software is so compelling that many software developers want to jump right in before they have a clear understanding of what is needed.
- Things will become clear as they build, that project stakeholders will be able to understand need only after examining early iterations of the software, that things change so rapidly that any attempt to understand requirements in detail is a waste of time
- Requirements engineering provides the appropriate mechanism for understanding what the customer wants, analyzing need, assessing feasibility, negotiating a reasonable solution, specifying the solution unambiguously, validating the specification, and managing the requirements as they are transformed into an operational system

Requirement Engineering Research

31st IEEE International
Requirements Engineering Conference
Hannover, Germany
September 4-8, 2023

Attending ▾ Program ▾ Tracks ▾ Organization ▾ Search Series ▾ Sign in Sign up

Requirements Engineering 2023 (series) /

Research Papers

Requirements Engineering 2023

About Program Accepted Papers Submission Instructions Formatting Instructions
Submission Q&A Call for Papers

Important Dates	AoE (UTC-12h)
Fri 30 Jun 2023	Camera Ready Submission

Requirement Engineering Research

Sponsors

You can reach some of the sponsor websites by clicking on their logo

Requiremen

- Requirements Elicitation and Prioritization
- Requirements Analysis and Specification
- Requirements Verification and Validation
- Requirements Management
- Pragmatic Requirements Engineering
- Theoretical Requirements Foundations
- Large-scale Requirements Engineering
- Agile Requirements Engineering
- Requirements Engineering and Continuous Integration/Continuous Delivery/DevOp
- Requirements, Configuration, Variability and Product Lines
- Requirements and Adaptive Systems
- Requirements Traceability and Dependencies
- Requirements Engineering and Human-Computer Interaction (HCI)
- Requirements in Open-Source Software Projects
- Requirements and Issue Tracking
- Requirements and Architecture
- Quality Engineering
- Product Management and Release Planning
- Global Requirements Engineering
- Online Feedback and User Review Analysis
- Domain-specific Requirements Engineering
- Requirements Engineering for Societal Challenges
- Requirements Engineering for Artificial Intelligence
- Requirements in System Engineering/System Science
- Requirements Engineering Education and Training
- Empirical Studies, Measurements and Prediction

ategories for Research Papers

Requirement Engineering Research

Accepted Papers

★	Title
☆	A Comparative Evaluation of Requirement Template Systems **A:** Katharina Großer, **A:** Marina Rukavitsyna, **A:** Jan Jürjens DOI Pre-print File Attached
☆	A Data-Driven Approach for Finding Requirements Relevant Feedback from TikTok and YouTube **A:** Manish Sihag, **A:** Ze Shi Li, **A:** Amanda Dash, **A:** Nowshin Nawar Arony, **A:** Kezia Devathasan, **A:** Neil Ernst, **A:** Alexandra Albu, **A:** Daniela Damian
☆	All Eyes on Traceability: An Interview Study on Industry Practices and Eye Tracking Potential **A:** Maike Ahrens, **A:** Lukas Nagel File Attached
☆	Analysis and optimisation of SPL products using goal models **A:** Inmaculada Ayala, **A:** Mercedes Amor, **A:** Lidia Fuentes Pre-print File Attached
☆	An Experiment on the Effects of using Color to Visualize Requirements Analysis Tasks **A:** Yesugen Baatartogtokh, **A:** Irene Foster, **A:** Alicia M. Grubb Pre-print

Requirements/ Features

- A feature is a realized functional requirement of a software [13, 21].
- Such an example feature– *Should have the capability for id search*

As a Course coordinator,

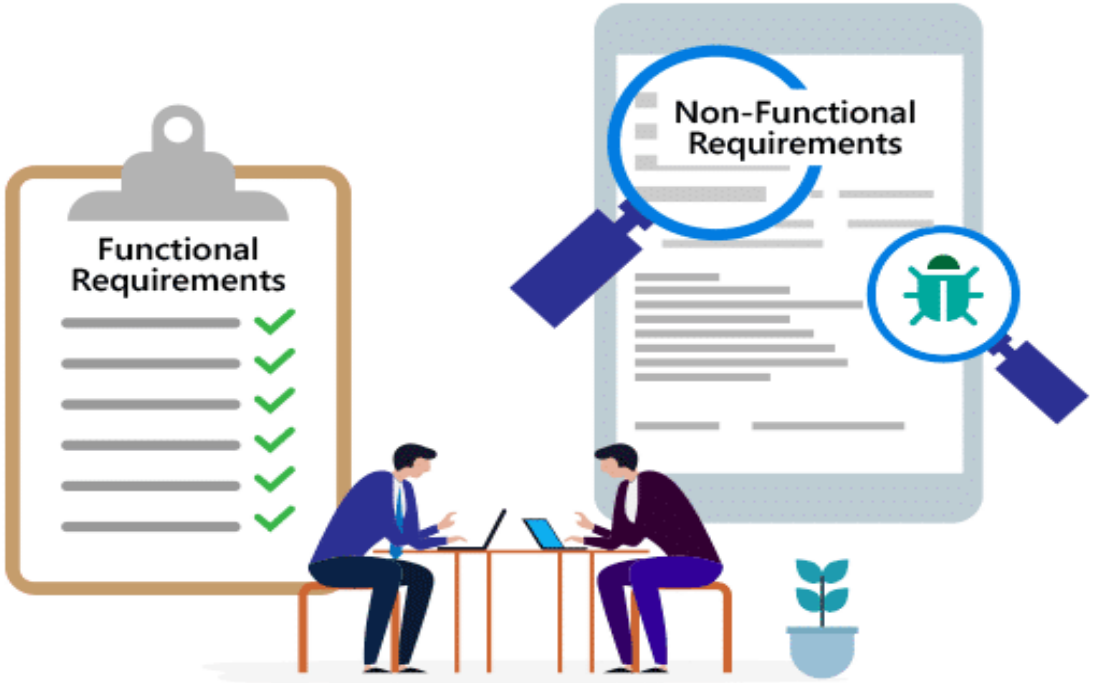
I need to search a student with id

So that I can see whether he/she registered a course.

Apel et al. [28] –

- *A feature is a characteristic or end-user-visible behavior of a software system.*
- *Features are used in product-line engineering to specify and communicate commonalities and differences of the products between stakeholders,*
- *and to guide structure, reuse, and variation across all phases of the software life cycle*.
- In this definition, features are foremost a means to communicate visible characteristics or observable behaviors of a system. This definition also explains the role of features in the context of a software product line

Requirements/ Features



Requirements/ Features [ch4-Sommerville]

Software system requirements are often classified as functional requirements or nonfunctional requirements:

- *Functional requirements* These are statements of services the system should provide, how the system should react to particular inputs, and how the system should behave in particular situations. In some cases, the functional requirements *may also explicitly state what the system should not do* .

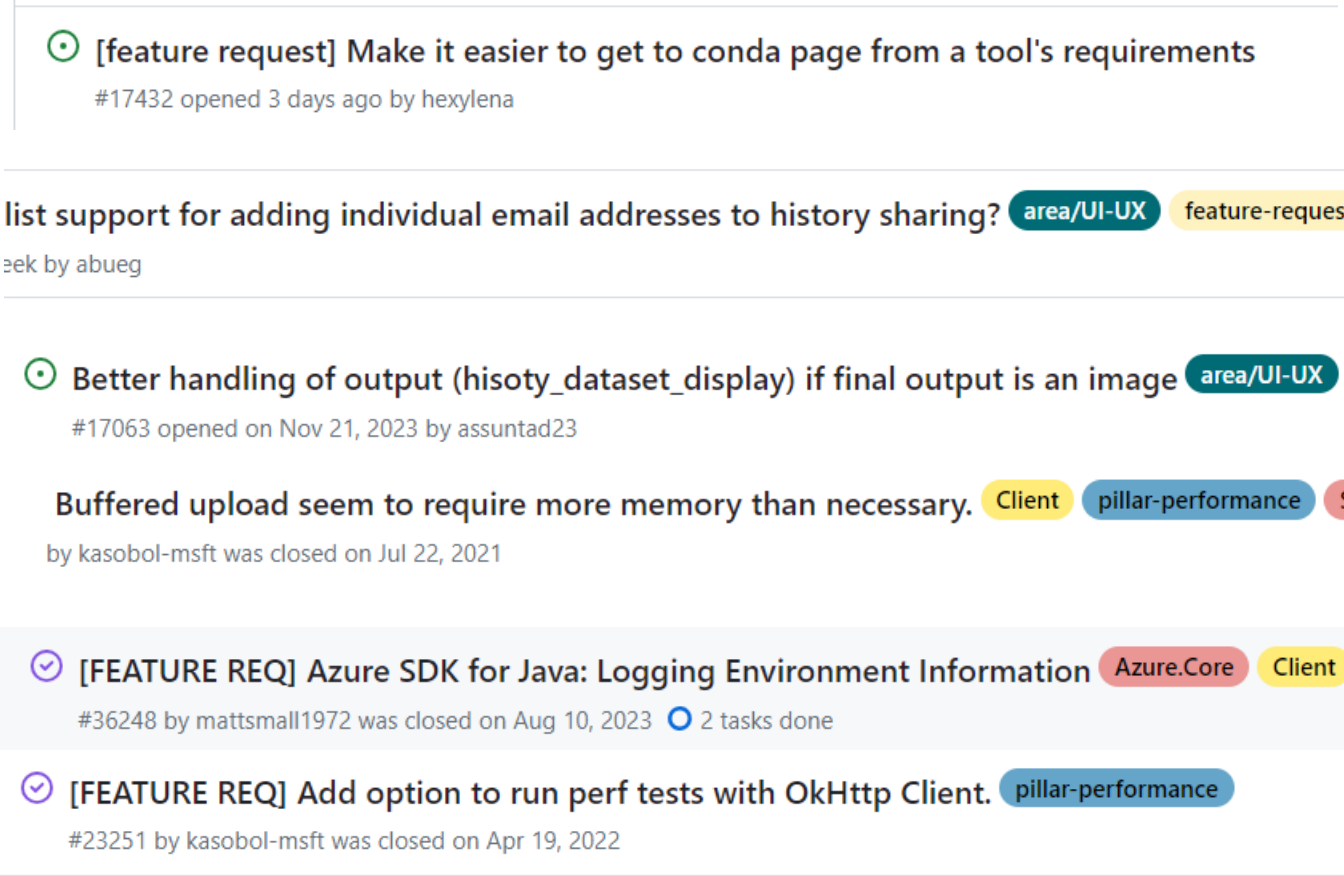
- *Quality/Non-functional requirements* These are constraints on the services or functions offered by the system. They include timing constraints, constraints on the development process, and constraints imposed by standards. Non-functional requirements often apply to the system as a whole, rather than individual system features or services.

Requirements/ Features [ch4-Sommerville]

- *Functional requirements*
  - 1. A user shall be able to search the appointments lists for all clinics.
  - 2. The system shall generate each day, for each clinic, a list of patients who are expected to attend appointments that day.
  - 3. Each staff member using the system shall be uniquely identified by his or her eight-digit employee number.

- *Quality/Non-functional requirements*
  - *The system should be easy to use by medical staff and should be organized in such a way that user errors are minimized.*

Galaxy , Azure Features



Requirement Engineering [ch5]

- It encompasses seven distinct tasks:
  - inception, elicitation, elaboration,
  - negotiation, specification,
  - validation, and management

Requirement Engineering [ch5]

Inception.

- *How does a software project get started?*
- *Is there a single event that becomes the catalyst for a new computer-based system or product, or does the need evolve over time?*

- Most projects begin when a business need is identified or a potential new market or service is discovered.
- Stakeholders from the business community define a business case for the idea, try to identify the breadth and depth of the market, do a rough feasibility analysis, and identify a working description of the project's scope.

Requirement Engineering [ch5]

Inception.

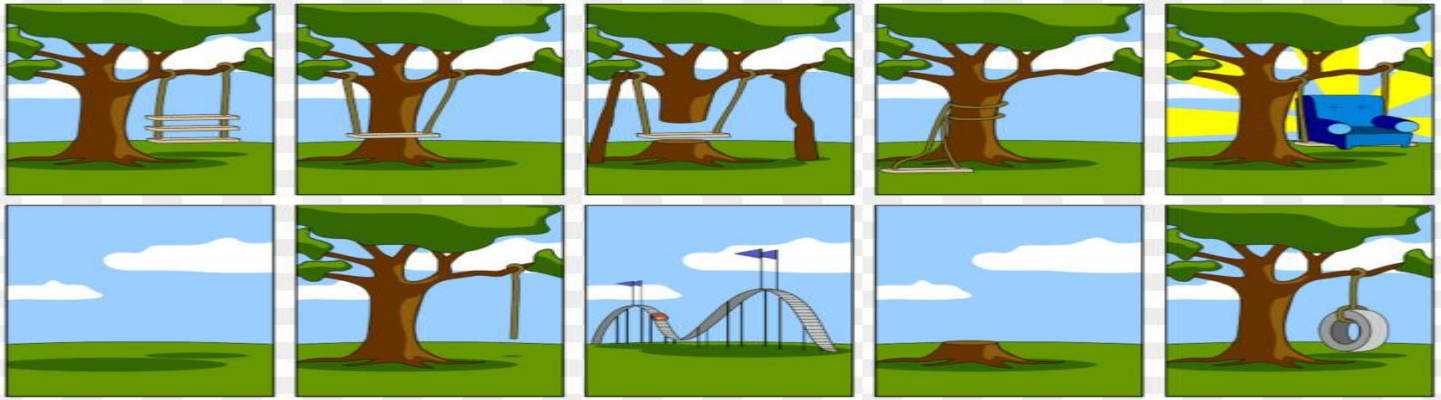
- At project inception, you establish a basic understanding of the problem, the people who want a solution, the nature of the solution that is desired, and the effectiveness of preliminary communication and collaboration between the other stakeholders and the software team.

Requirement Engineering [ch5]

Elicitation. It certainly seems simple enough—

- ask the customer, the users, and others what the objectives for the system or product are,
- what is to be accomplished,
- how the system or product fits into the needs of the business, and finally, how the system or product is to be used on a day-to-day basis.

Requirement Engineering [ch5]



But it isn’t simple—Christel and Kang [Cri92] identify a number of problems as elicitation occurs--

- **Problems of scope.** The boundary of the system is ill-defined or the customers/users specify unnecessary technical detail that may confuse overall system objectives.

Requirement Engineering [ch5]

- **Problems of understanding.** The customers/users are not completely sure of what is needed,
  - have a poor understanding of the capabilities and limitations of their computing environment,
  - don't have a full understanding of the problem domain, have trouble communicating needs to the system engineer,
  - omit information that is believed to be "obvious," specify requirements that conflict with the needs of other customers/users,
  - or specify requirements that are ambiguous or untestable.

Problems of volatility. The requirements change over time.

- To help overcome these problems, you must approach requirements gathering in an organized manner.

Requirement Engineering [ch5]

- **Elaboration.** The information obtained from the customer during inception and elicitation is expanded and refined during elaboration.
  - ➢ focuses on developing a refined requirements model that identifies various aspects of software function, behavior, and information.
- Elaboration is driven by the creation and refinement of user scenarios that describe how the end user will interact with the system. Each user scenario is parsed to extract analysis classes—business domain entities that are visible to the end user.
- The attributes of each analysis class are defined, and the services that are required by each class are identified.
- The relationships and collaboration between classes are identified, and a variety of supplementary diagrams are produced

Requirement Engineering [ch5]

Negotiation. It isn't unusual for customers and users to ask for more than can be achieved, given limited business resources.

- It's also relatively common for different customers or users to propose conflicting requirements, arguing that their version is "essential for our special needs."
- You have to reconcile these conflicts through a process of negotiation.
- Customers, users, and other stakeholders are asked to rank requirements and then discuss conflicts in priority.

- Using an iterative approach that prioritizes requirements, assesses their cost and risk, and addresses internal conflicts, requirements are eliminated, combined, and/or modified so that each party achieves some measure of satisfaction.

Requirement Engineering [ch5]

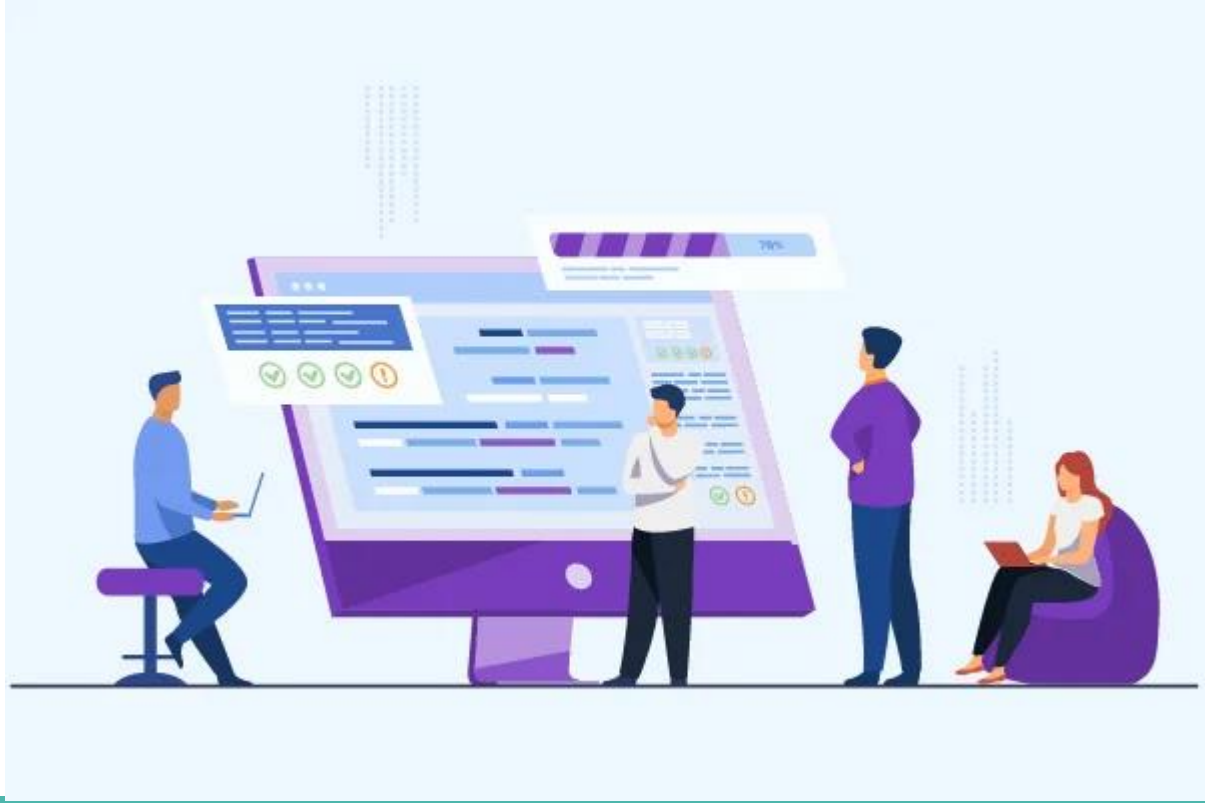
- **Specification.**

Requirement Engineering [ch5]

- **Specification.**
  - *specification* means different things to different people.
  - It can be written document, a set of graphical models, a formal mathematical model,
  - A collection of usage scenarios, a prototype, or any combination of these.
- Some suggest that a ***"standard template"*** should be developed and used for a specification -- leads to requirements in a consistent and more understandable manner.
- For large systems, a written document, combining natural language descriptions and graphical models may be the best approach. However, usage scenarios are required for smaller products or systems that reside within well-understood technical environments.

Requirement Engineering [ch5]

Validation.



Requirement Engineering [ch5]

Validation. Requirements validation examines the specification[5] to ensure

- that all software requirements have been stated unambiguously;
- that inconsistencies, omissions,
- and errors have been detected and corrected;
- and that the work products conform to the standards established for the process, the project, and the product.

- The primary requirements validation mechanism is the technical review
- The review **team** that validates requirements includes software engineers, customers, users, and other stakeholders who examine the specification looking for errors in content or interpretation, areas where clarification may be required, missing information, inconsistencies (*a major problem when large products or systems are engineered*), conflicting requirements, or unrealistic requirements.

Requirement Engineering [ch5]

- **Requirements management.** Requirements for computer-based systems change, and the desire to change requirements persists throughout the life of the system.
- Requirements management is a set of activities that help the project team identify, control, and track requirements and changes to requirements at any time as the project proceeds.
- Many of these activities are identical to the software configuration management (SCM).

Requirement Engineering [ch5]

SafeHome

Conducting a Requirements Gathering Meeting

The scene: A meeting room. The first requirements gathering meeting is in progress.

The players: Jamie Lazar, software team member; Vinod Raman, software team member; Ed Robbins, software team member; Doug Miller, software engineering manager; three members of marketing; a product engineering representative; and a facilitator.

The conversation:

Facilitator (pointing at whiteboard): So that's the current list of objects and services for the home security function.

Marketing person: That about covers it from our point of view.

Vinod: Didn't someone mention that they wanted all *SafeHome* functionality to be accessible via the Internet? That would include the home security function, no?

Marketing person: Yes, that's right . . . we'll have to add that functionality and the appropriate objects.

Facilitator: Does that also add some constraints?

Jamie: It does, both technical and legal.

Production rep: Meaning?

Jamie: We better make sure an outsider can't hack into the system, disarm it, and rob the place or worse. Heavy liability on our part.

Doug: Very true.

Marketing: But we still need that . . . just be sure to stop an outsider from getting in.

Ed: That's easier said than done and . . .

Facilitator (interrupting): I don't want to debate this issue now. Let's note it as an action item and proceed.

(Doug, serving as the recorder for the meeting, makes an appropriate note.)

Facilitator: I have a feeling there's still more to consider here.

(The group spends the next 20 minutes refining and expanding the details of the home security function.)