Requirement Engineering

Recap

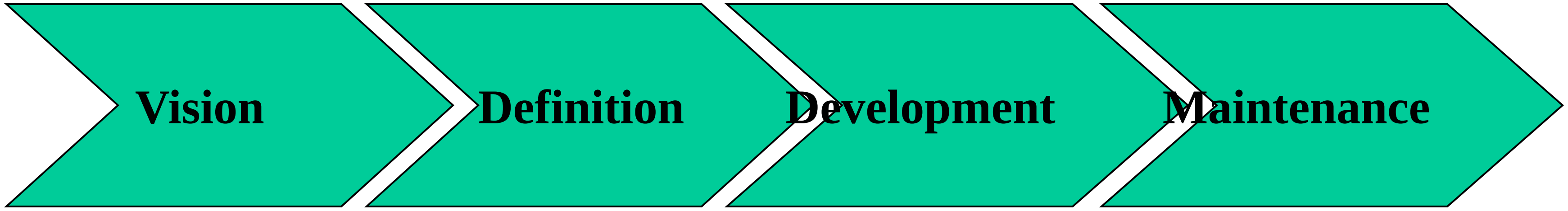
- Software development is a multi-activity process. It is not simply coding.
- Software construction and management
- Software Engineering Framework
- Software engineering phases
- Importance of Maintenance

Software Construction

- What is the problem to be solved?
- What are the characteristics of the entity that is used to solve the problem?
- How will the entity be realized?
- How will the entity be constructed?
- What approach will be used to uncover errors that were made in the design and construction of the entity?
- How will the entity be supported over the long term, when corrections, adaptations, and enhancements are requested by users of the entity?

Software Engineering Phases

1. Vision – focus on *why*
2. Definition – focus on *what*
2. Development – focus on *how*
3. Maintenance – focus on *change*



Requirement Engineering

- The entire system development process begins with requirement engineering.
- The process of establishing the system services and constraints is called requirement engineering.
- What vs. How

Software Requirements – Definition - IEEE

1 A condition or capability needed by 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 document.

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

Software Requirements – Definition - Jones

- The statement of needs by a user that triggers the development of a program or system - *Jones 1994*

Software Requirements – Definition – Davis

- A user need or necessary feature, function, or attribute of a system that can be sensed from a position external to that system - *Alan Davis 1993*

Software Requirements - Definition

- Requirements are ... A specification of what should be implemented. They are descriptions of how the system should behave, or of a system property or attribute. They may be a constraint on the development process of the system. - *Sommerville 1997*

Levels of Requirements

- Business Requirements
  - Represent high level objectives of the organization or customer requesting the system or product
  - Captured in a document describing the project vision and scope.
- User Requirements
  - Describes tasks the user must be able to accomplish

Levels of Requirements

- Functional Requirements
  - Define the software functionality the developers must build into the product to enable users to accomplish their tasks - thereby satisfying the business requirements.

Levels of Requirements

- Non-Functional Requirements
  - Include regulations, standards, and contracts to which the product must conform:
    - Description of external interfaces
    - design and implementation constraints
    - Quality and performance attributes.
  - Constraints are restrictions that are placed on the choices available to the developer for design and construction of the software product.

- All user requirements must align with business requirements
- Requirements do not include design or implementation details.
- Focus on what to build.

Kinds of requirements

- BR – user will be able to correct spelling errors in a document efficiently.
  - Spell checker is included as a feature
- UR – finding spelling errors in the document and decide whether to replace each misspelled word with the one chosen from a list of suggested words.
- FR
  - find and highlight misspelled words.
  - Display a dialog box with suggested replacements
  - Making global replacements
- NFR – It must be integrated into the existing word-processor which runs on windows platform.

Functional and Non-functional Requirements

Importance of the Software Requirement Process

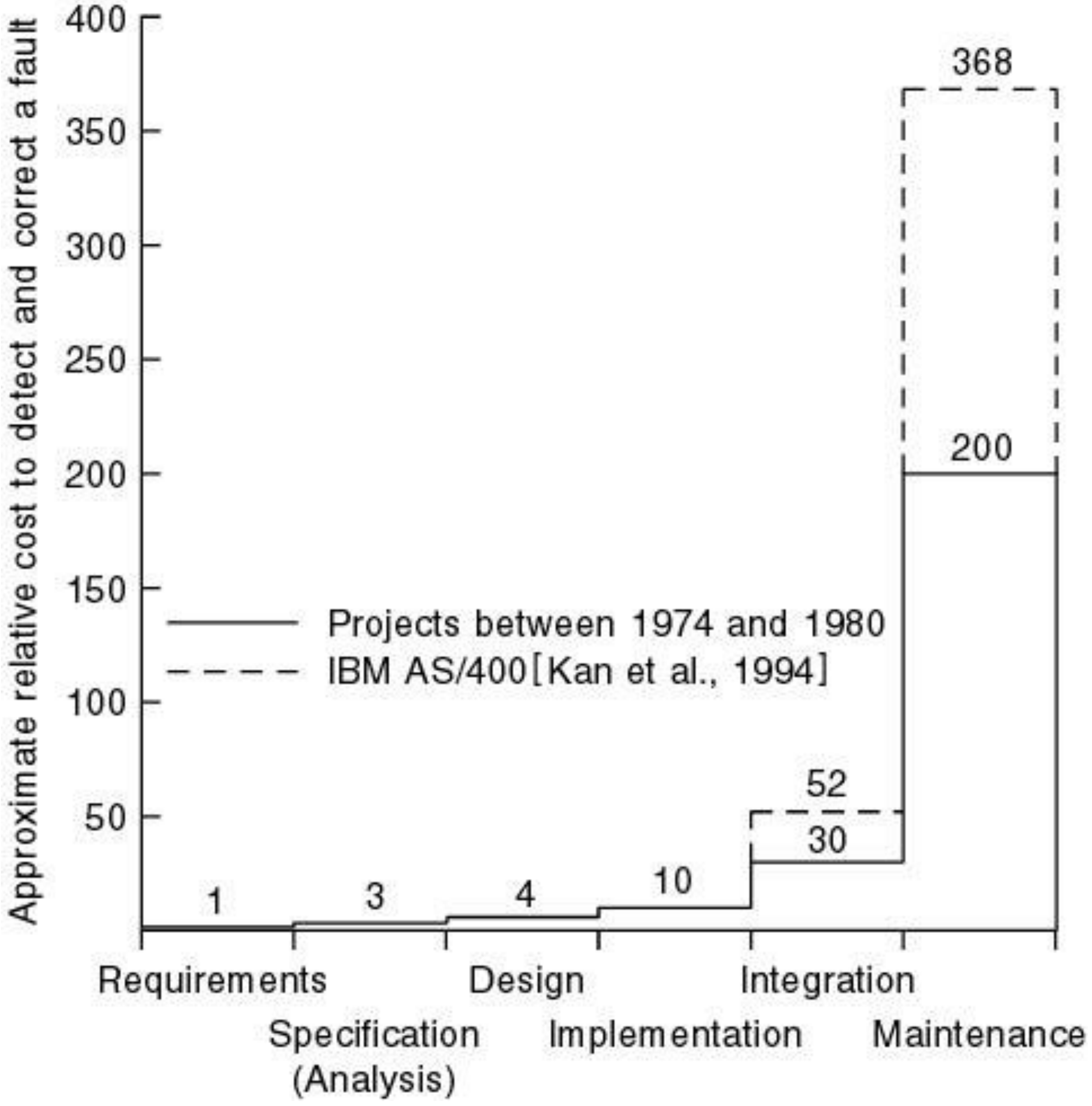
The hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing the detailed technical requirements, including all the interfaces to people, to machines, and to other software systems. No other part of the work so cripples the system if done wrong. No other part is more difficult to rectify later.

Fred Brooks - No Silver Bullet: Essence and Accidents of Software Engineering, 1987.

Importance of Requirements

- Many of the problems encountered in SW development are attributed to shortcoming in requirement gathering and documentation process.
- Building a house without requirements: no – building a software: yes
- 40-60% of all defects found in software projects can be traced back to poor requirements
- Interest of all stakeholders in a project is must

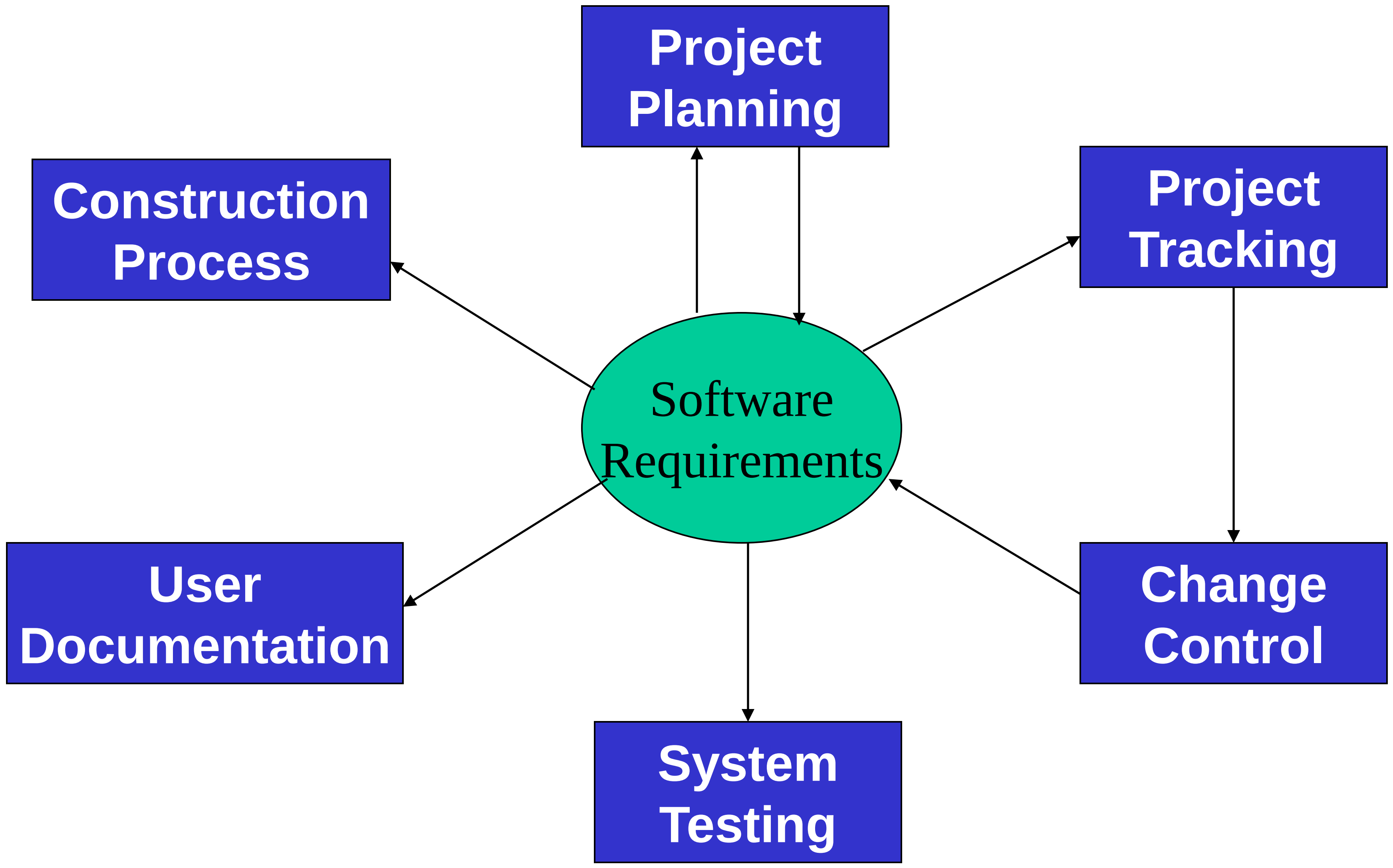
Importance of requirements



Benefits of high quality requirement process

- Boehm(1981) – correcting an error after development costs 68 times more
- Other studies suggest that it can be as high as 200 times.
- Hence requirements are the most critical success factor for any project

Role of Requirements



Some Risks From Inadequate Requirement Process

- -Insufficient user involvement leads to unacceptable products.
- -Creeping user requirements contribute to overruns and degrade product quality.
- -Ambiguous requirements lead to ill-spent time and rework.
- Gold-plating by developers adds unnecessary features.
- -Minimal specifications lead to missing key requirements.
- Overlooking the needs of certain user classes (stake holders) leads to dissatisfied customers.
- -Incompletely defined requirements make accurate project planning and tracking impossible.

Example of minimal requirements

- We need a flow control and source control engineering tool.
- Worked perfectly but
  - There was no print functionality

Ambiguous requirements

- Ambiguity – arising from natural language
  - Sommerville – no output is better than wrong output.
  - Rooko mut jane do
- Leads to different expectations
- Waste of time and effort

Ambiguous requirements

- Multiple readers arrive at different understanding
  - The operator identity consists of the operator name and password; the password consists of six digits. It should be displayed on the security VDU and deposited in the login file when an operator logs into the system

Ambiguous Requirements

- System should be user friendly
- System should have a good user interface
- It should perform operations efficiently
- It should respond to queries quickly

Contradiction

- All programs must be written in Ada
- The program must fit in the memory of the embedded micro-controller
- Problem: Code generated by the Ada compiler was of large foot print – it would not fit.

Contradiction

- System must monitor all temperatures in a chemical reactor.
- System should only monitor and log temperatures below -20^0 C and above 400^0 C.