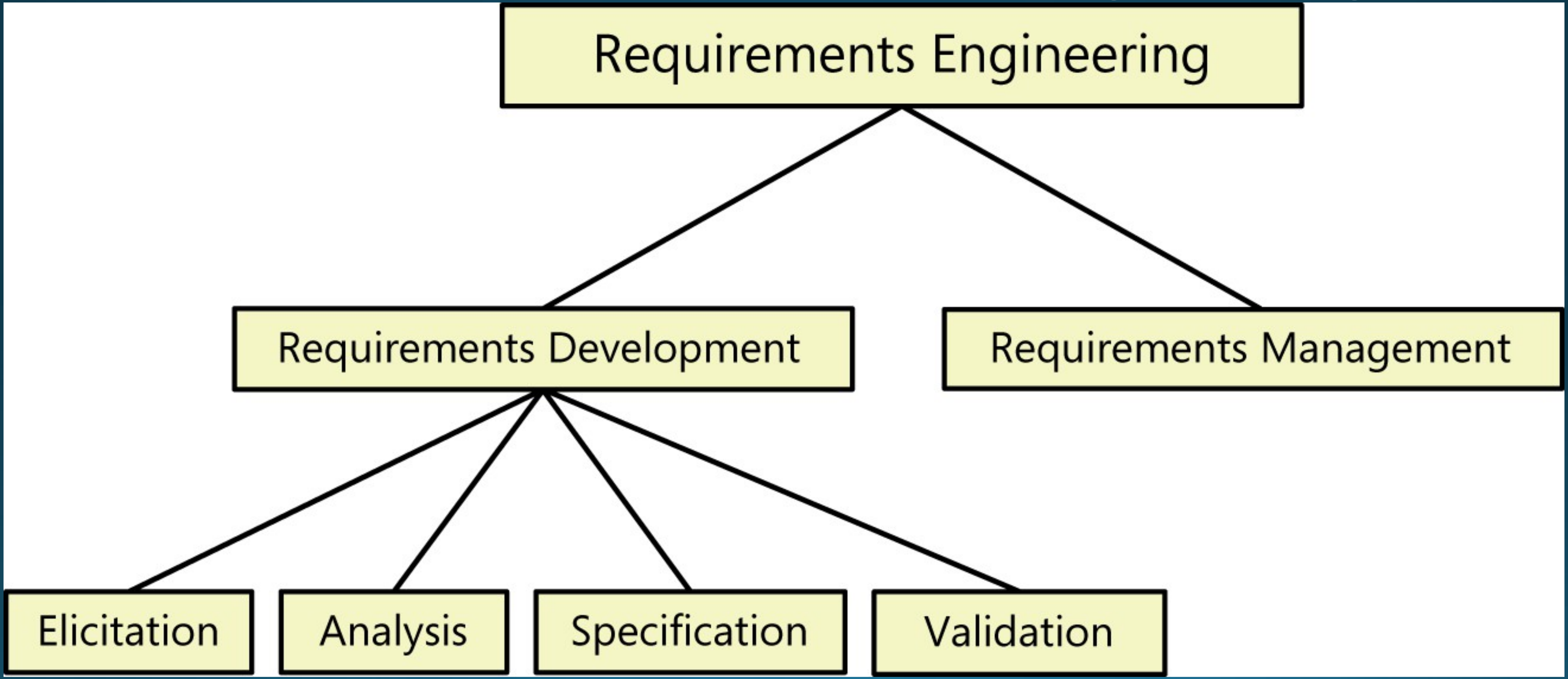
Requirements Development & Management

(L#3-4)

Dr Sumaira
Khan

Subdisciplines of Software Requirements Engineering



Elicitation

Elicitation encompasses all of the activities involved with discovering requirements through interviews, workshops, document analysis, prototyping, and others.

Analysis

Analyzing requirements involves reaching a richer and more precise understanding of each requirement and representing sets of requirements in multiple ways.

Specification

- Requirements specification involves representing and storing the collected requirements knowledge in a persistent and well-organized fashion, suitable for comprehension, review, and use by their intended audiences.

Validation

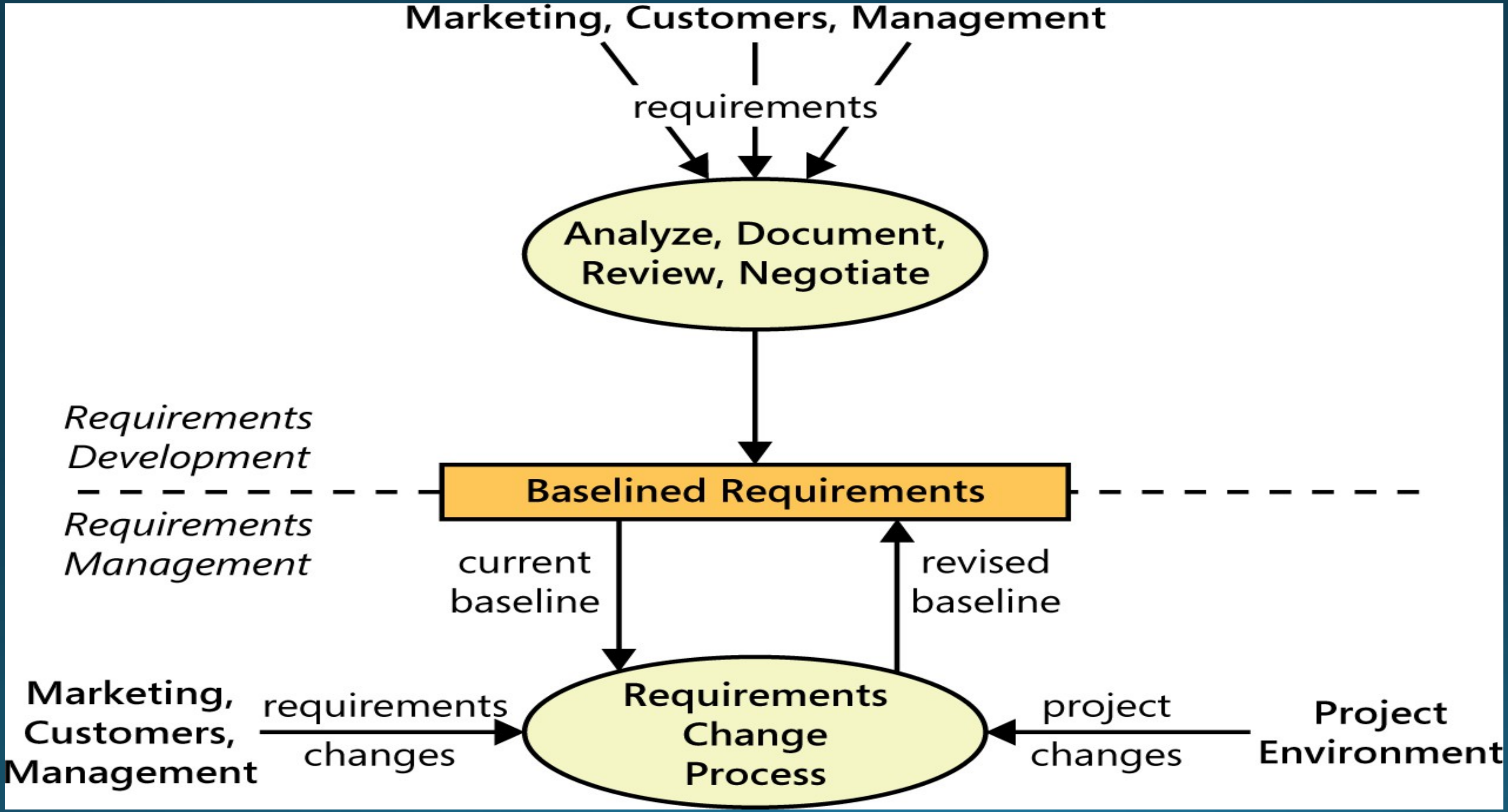
Requirements validation confirms that you have the correct set of requirements information that will enable developers and testers to build a solution that satisfies the business objectives.

Requirements Management

Includes:

- Defining requirements baseline
- Handling the requirements change
- Requirements Traceability
- Prioritization
- Negotiation

Boundary between Requirements Development and Requirements Management



Bad Requirements

- Insufficient user involvement
- Inaccurate planning
- Creeping user requirements
- Ambiguous/Incorrect/Incomplete requirements
- Gold plating
- Overlooked stakeholders

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 and users 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.

Requirements Defects - 1

- All four categories of defects (errors of commission, **errors of omission**, **errors of clarity and ambiguity**, and errors of speed and capacity) are found in requirements
- If not prevented, requirements defects usually flow downstream into design, code, and user manuals

Requirements Defects - 2

- Historically, requirements defects are most expensive and troublesome to eliminate
- **Errors of omission** are most common among requirements defects
- Famous example is the Y2K problem

Example of minimal requirements

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

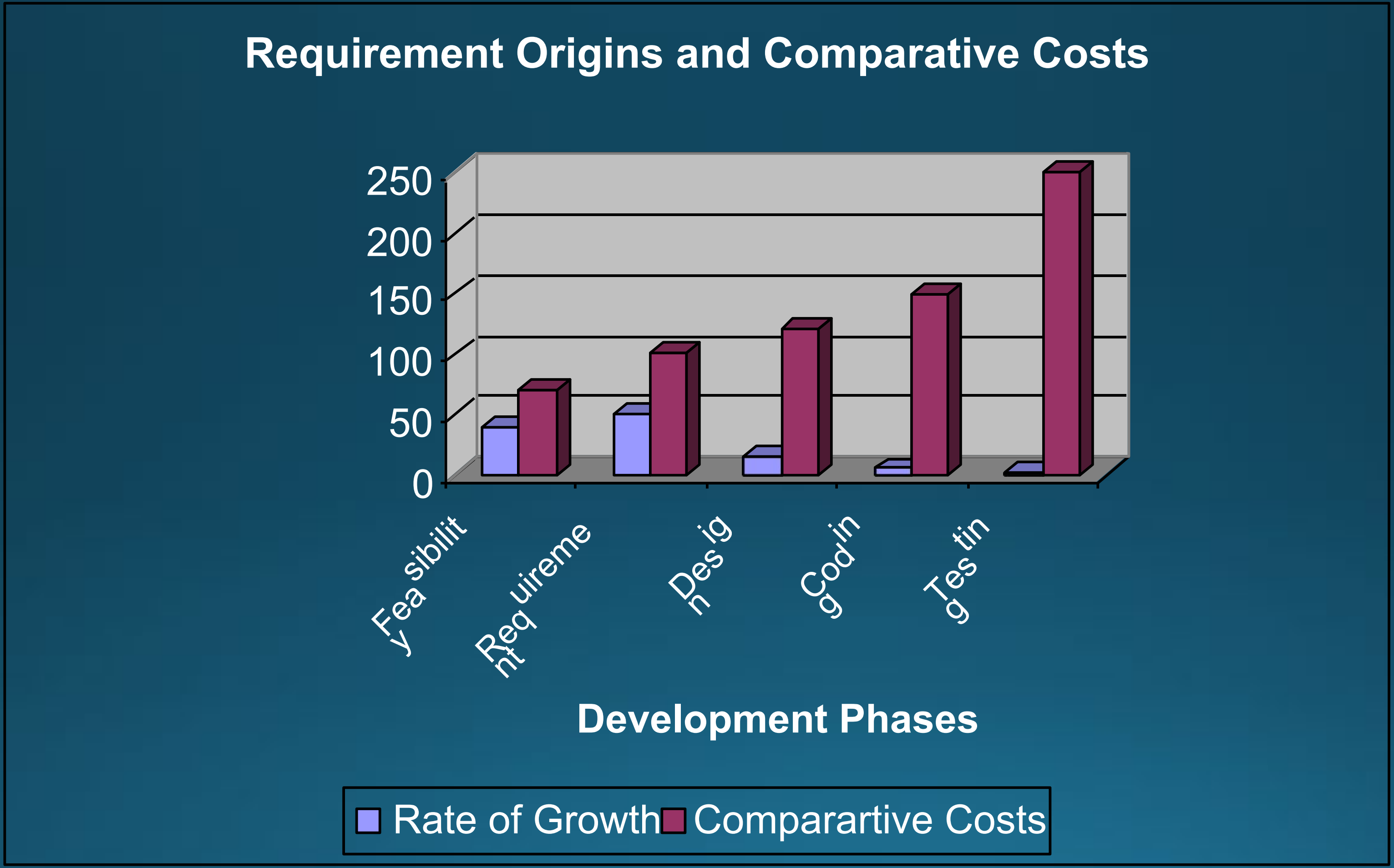
Changing/Creeping Requirements

- Requirements will change, no matter what
- A major issue in requirements engineering is the rate at which requirements change once the requirements phase has “officially” ended
- This rate is on average 3% per month in the subsequent design phase, and will go down after that
- This rate should come down to 1% per month during coding
- Ideally, this should come down to no changes in testing

Defects and Creeping Requirements

- Studies have shown that very significant percentage of delivered defects can be traced back to creeping user requirements
- This realization can only be made, if defect tracking, requirements traceability, defect removal efficiency, and defect rates are all monitored for software projects

Cost of Requirement Creep



Damage Control of Creeping Requirements

- Following quality assurance mechanisms can limit the damage done by creeping requirements
  - Formal change management procedures
  - State-of-the-art configuration control tools
  - Formal design and code inspections

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

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

System Goal

“The system should be easy to use by experienced controllers and should be organized in such a way that the user errors are minimized.”

Verifiable Non-Functional Requirement

- "Experienced controllers should be able to use all the system functions after a total of two hours training. After this training, the average number of errors made by experienced users shall not exceed two per day."

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.

Validating Requirements

Requirements need to be validated for

- consistency
- testability
- performance issues
- conformance to local/national laws
- conformance to ethical issues
- conformance to company policies
- availability of technology

Prevention vs Removal

- For requirements errors, prevention is usually more effective than removal
- JAD, QFD, and prototyping are more effective in defect prevention
- Requirements inspections and prototyping play an important role in defect removal

Requirements Bill of Rights for Software Customers

1. Expect analysts to speak your language.
2. Expect analysts to learn about your business and your objectives for the system.
3. Expect analysts to structure the information you present during requirements elicitation into a written software requirements specification.
4. Have analysts explain all work products created from the requirements process.
5. Expect analysts and developers to treat you with respect and to maintain a collaborative and professional

Requirements Bill of Rights for Software Customers

6. Have analysts and developers provide ideas and alternatives both for your requirements and for implementation of the product.
7. Describe characteristics of the product that will make it easy and enjoyable to
   - use.
8. Be given opportunities to adjust your requirements to permit reuse of existing software components.
9. Receive good-faith estimates of the costs, impacts, and trade-offs when you
   - request a change in the requirements.
10. Receive a system that meets your functional and quality needs, to

Requirements Bill Of Responsibilities For Software Customers

1. Educate analysts and developers about your business and define business jargon.
2. Spend the time that it takes to provide requirements, clarify them, and iteratively flesh them out.
3. Be specific and precise when providing input about the system’s requirements.
4. Make timely decisions about requirements when requested to do so.
5. Respect a developer’s assessment of the cost and feasibility of requirements.

Requirements Bill Of Responsibilities For Software Customers

6. In collaboration with the developers, set priorities for functional requirements, system feature, or use cases.
7. Review requirements documents and evaluate prototypes.
8. Communicate changes to the requirements as soon as you know about them.
9. Follow the development organization's process for requesting requirements changes.
10. Respect the processes the analysts use for requirements engineering.

Sign-Off

- “Reaching agreement on the requirements for the product to be built.”
- “I was presented with a piece of paper that had my name typed on it, so I signed because otherwise developers wouldn’t start coding.”
- “Sure, I signed off on the requirements, but I did not have time to read them all. I trusted you guys – you let me down.”

Sign-Off

- When a change request is made to DM:

  "But you signed off on these requirements, so that's what we are building. If you wanted something else, you should have said so."

- Don't use sign-off as a weapon. Use it as a project milestone, with a clear, shared understanding of the activities that lead to sign-off and its implications for future changes.

Base-lining Process for Stakeholders Confidence

- Customer management - project scope won’t explode, because customers manage the scope change decisions.
- User rep - development will work with them
- Development Management – keep the project focused on achieving the objectives
- Requirement Analyst – they can manage changes to the project

Benefits from a high-quality requirements process

- Fewer defects in requirements and in the delivered product
- Reduced development rework
- Faster development and delivery
- Fewer unnecessary and unused features
- Lower enhancement costs
- Fewer miscommunications
- Reduced scope creep
- Reduced project chaos
- Higher customer and team member satisfaction
- Products that do what they’re supposed to do

Importance of the Software Requirement Process

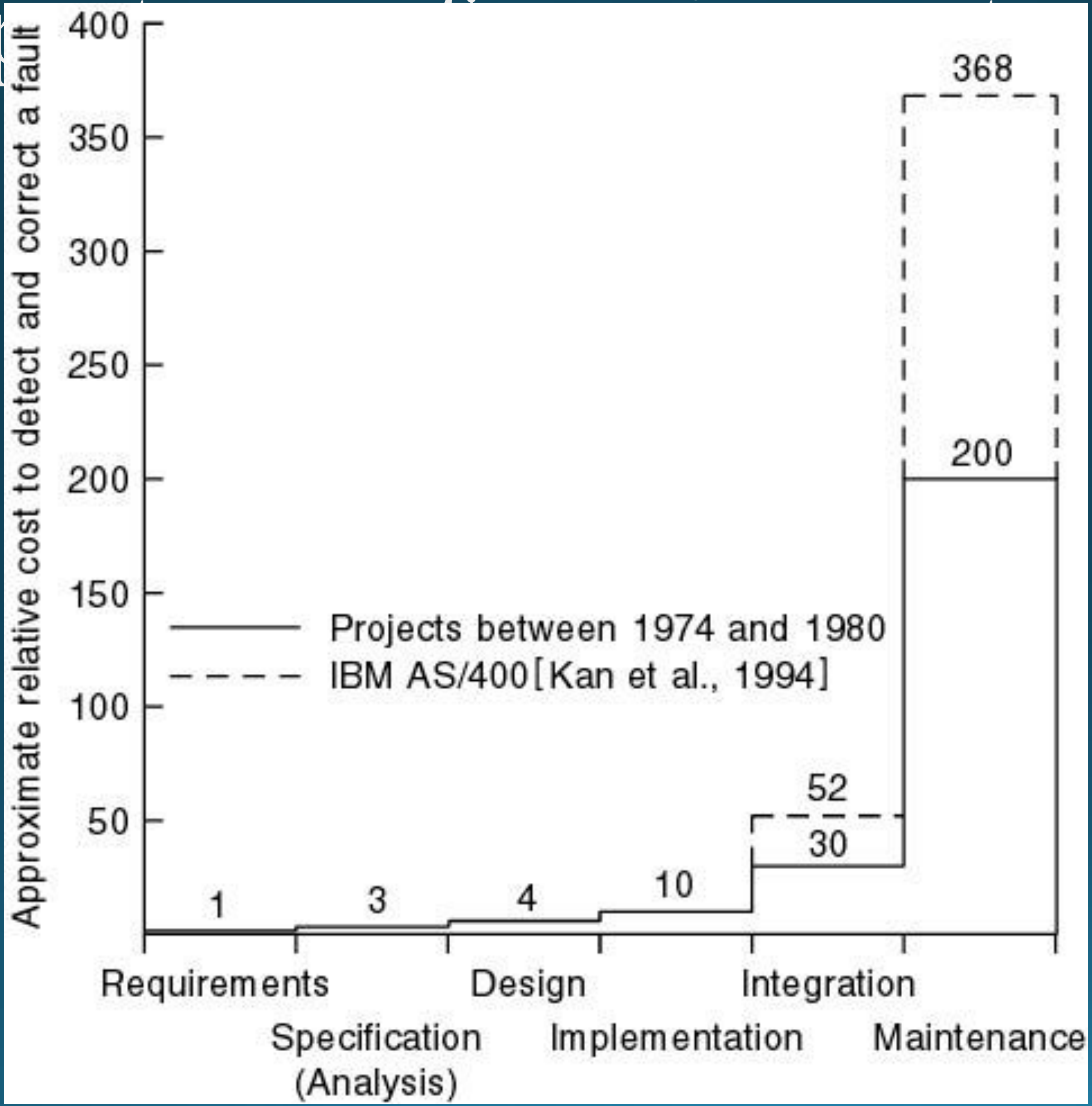
The hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing the detailed technical requirements, including all the interfaces to people, to machines, and to other software systems. No other part of the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

Imp


Approximate relative cost to detect and correct a fault
400
350
300
250
200
150
100
50
368
200
52
30
10
4
3
1
Projects between 1974 and 1980
IBM AS/400[Kan et al., 1994]
Requirements
Specification (Analysis)
Design
Implementation
Integration
Maintenance

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

References

- Software Requirements by Karl Wiegers, 2013
- Requirements Engg Processes and Techniques by Kotonyo & Sommerville