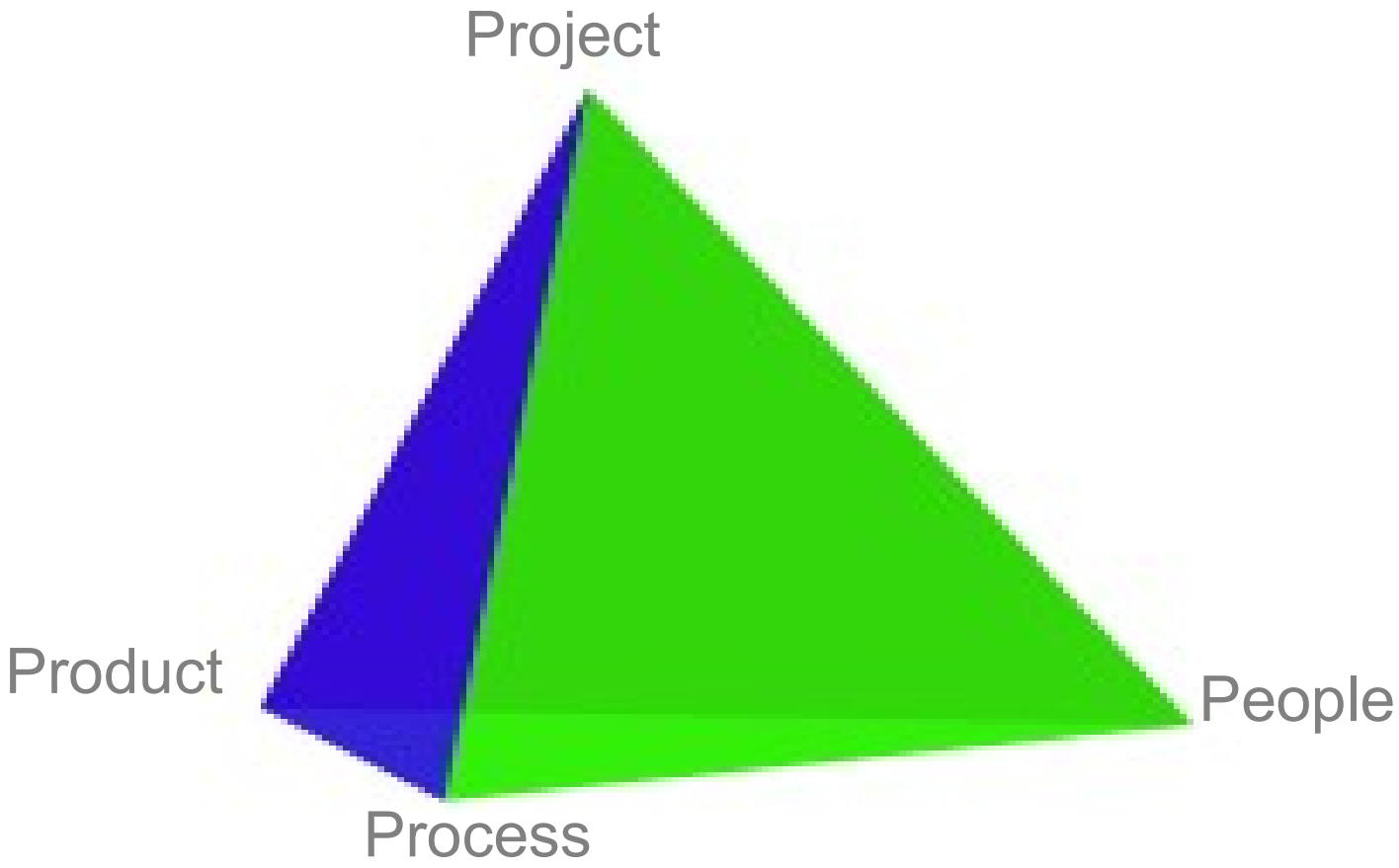
Software Project Management

The Core View

The 4 Ps



The Mythical Man Power

Can one person build the Eiffel tower?

- Facts:

☞ Construction started 7/1/ 1887

☞ Completed 22 months later

☞ 18000 pieces used to build the tower

☞ Between 150-300 workers were on the building site daily.

The Mythical Man Power

Can one person build the Eiffel tower?

- Answers:
  - In theory: yes
  - In Practice: No
- Why?
  - Let us do the math … calculate the person-hours required to complete the project:
    - *= 22 months*20 business days * ((150+300)/2 HC)*8hours*
    - *= 792000 working hours*
    - *= 4950 months*
    - *=412.5 years*
  - The one person has a capacity that can't be exceeded; only 2 hands and 2 legs.

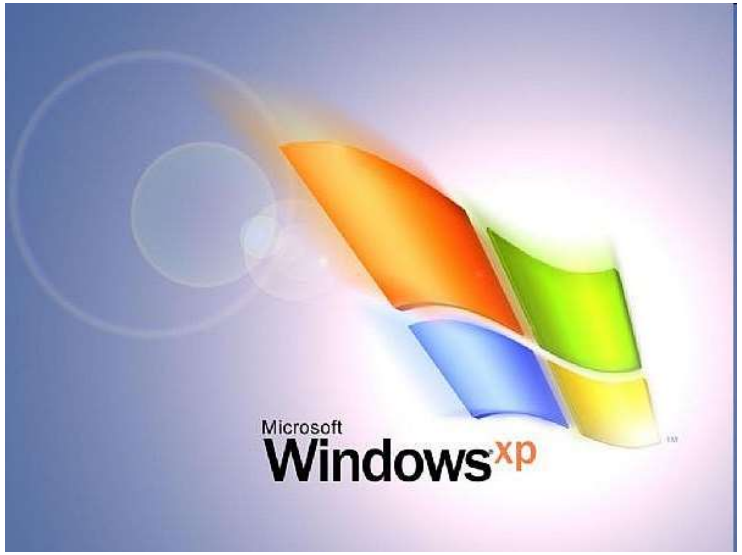
The Software Project vs. Eiffel Tower

Is the software project similar to Eiffel tower?

- Yes it is, you need:
  - Physical Resources
  - Human Resources:
    - ✓ Architect
    - ✓ Developers
    - ✓ Manager
    - ✓ Etc.
  - Budget
  - Plan
  - Quality

Software Product Size

- Windows Vista is said to have over 50 million lines of code, whereas XP was said to have around 40 million



Software Product Size

- Red Hat Linux version 7.1 (released April 2001) contained over 30 million SLOC

Software Product Size

- Mac OS X 10.4 has about 86 million SLOC

Software Product Size

- Debian 5.0 /GNU Linux has about 324 million SLOC

Software Product Size

- The Android operating system consists of 12 million lines of code SLOC

Software Product Size

- The Navigation system in the current S-class Mercedes-Benz requires over 20 million lines of code

Software Product Size

- The avionics system in the F-22 Raptor, the current U.S. Air Force frontline jet fighter, consists of about 1.7 million lines of software code.

- The F-35 Joint Strike Fighter requires about 5.7 million lines of code to operate its onboard systems.

- Boeing's new 787 Dreamliner requires about 6.5 million lines of software code to operate its avionics and onboard support systems.

Software Quality Management

- What is Software Quality Management?

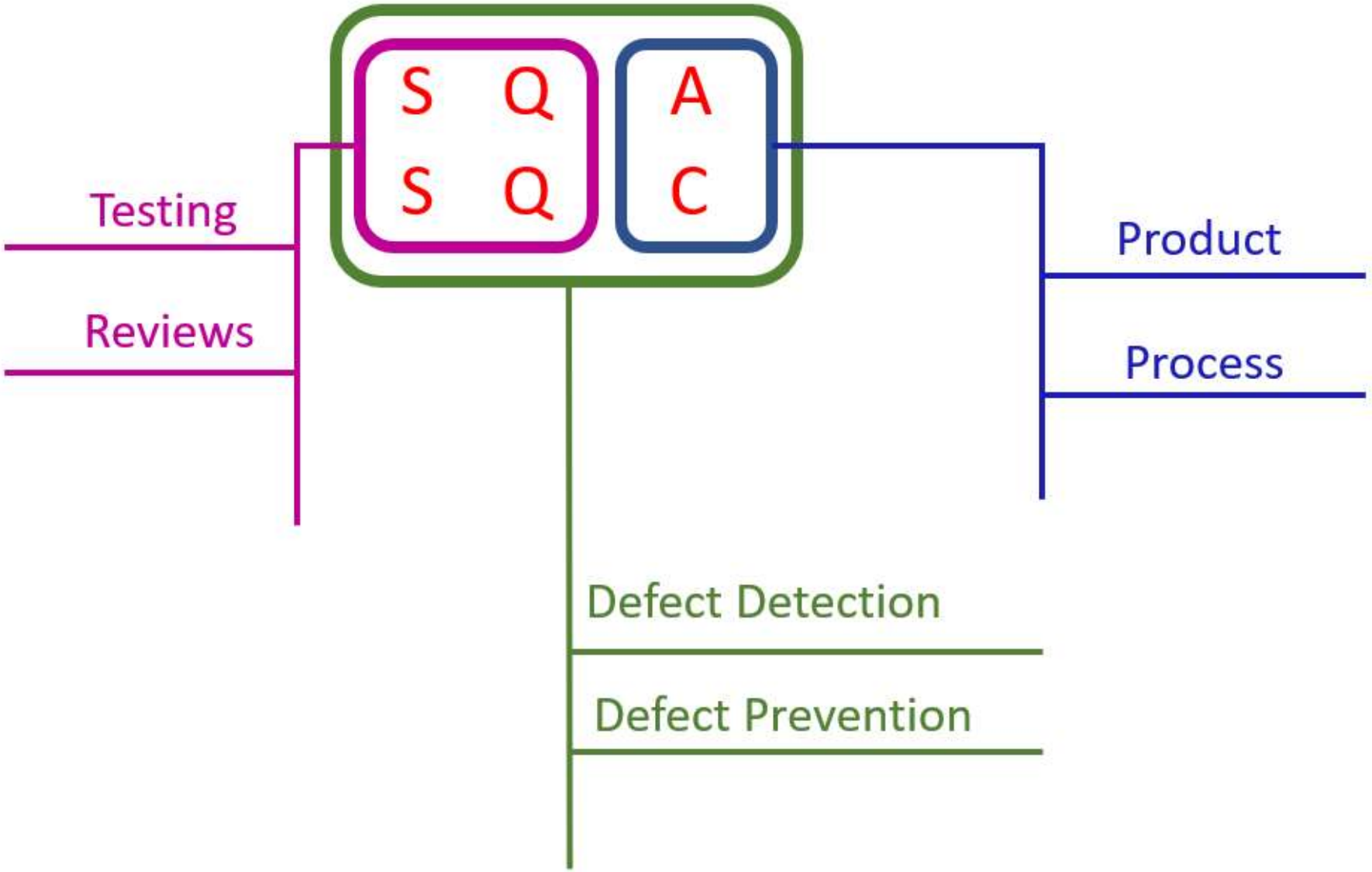
- Is Software Project Management related to Software Quality Management?

Software Quality Assurance

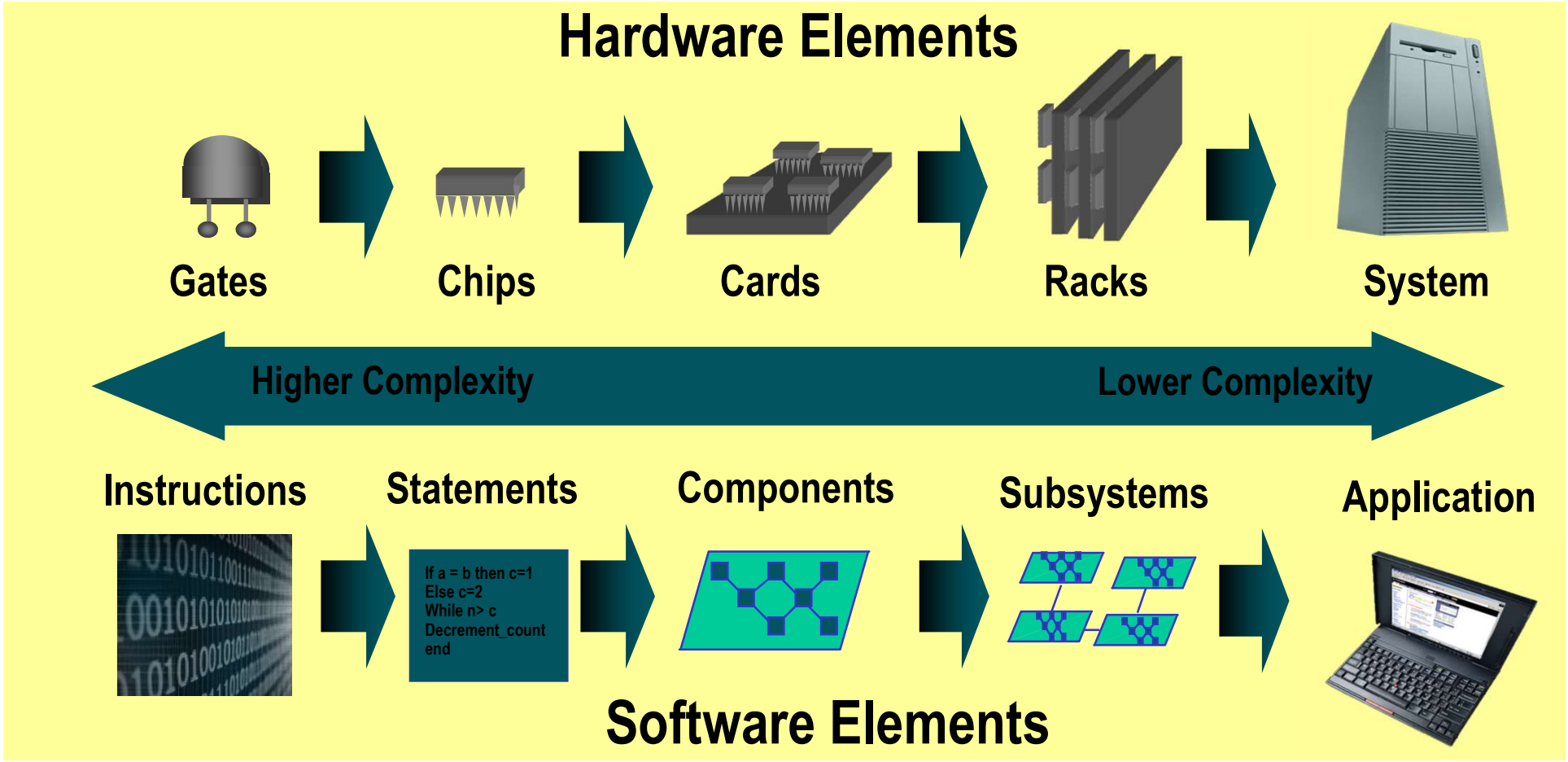
vs.

Software Quality Control

- Let us see …


S Q
S Q
A
C
Testing
Reviews
Product
Process
Defect Detection
Defect Prevention

Software vs. Hardware



Software Project

What is a Software Project?

A sequence of connected and related activities ***(requirement engineering, system engineering, coding, testing, documentation, controlling, …)*** that must be completed by a specific time, within budget, and according to specification.

Software Crisis

Many software-related failures: auto-pilot systems, air traffic control systems, banking systems, IRS.

- On January 15, 1990, the AT&T long-distance telephone network broke down, interrupting long-distance telephone services in US for over 8 hours.
- On June 4, 1996, the maiden flight of the new and improved Ariane 5 rocket exploded 37 seconds after lift-off.
- On June 8, 2001, a software problem caused the NYSE to shut down the entire trading floor for over an hour.

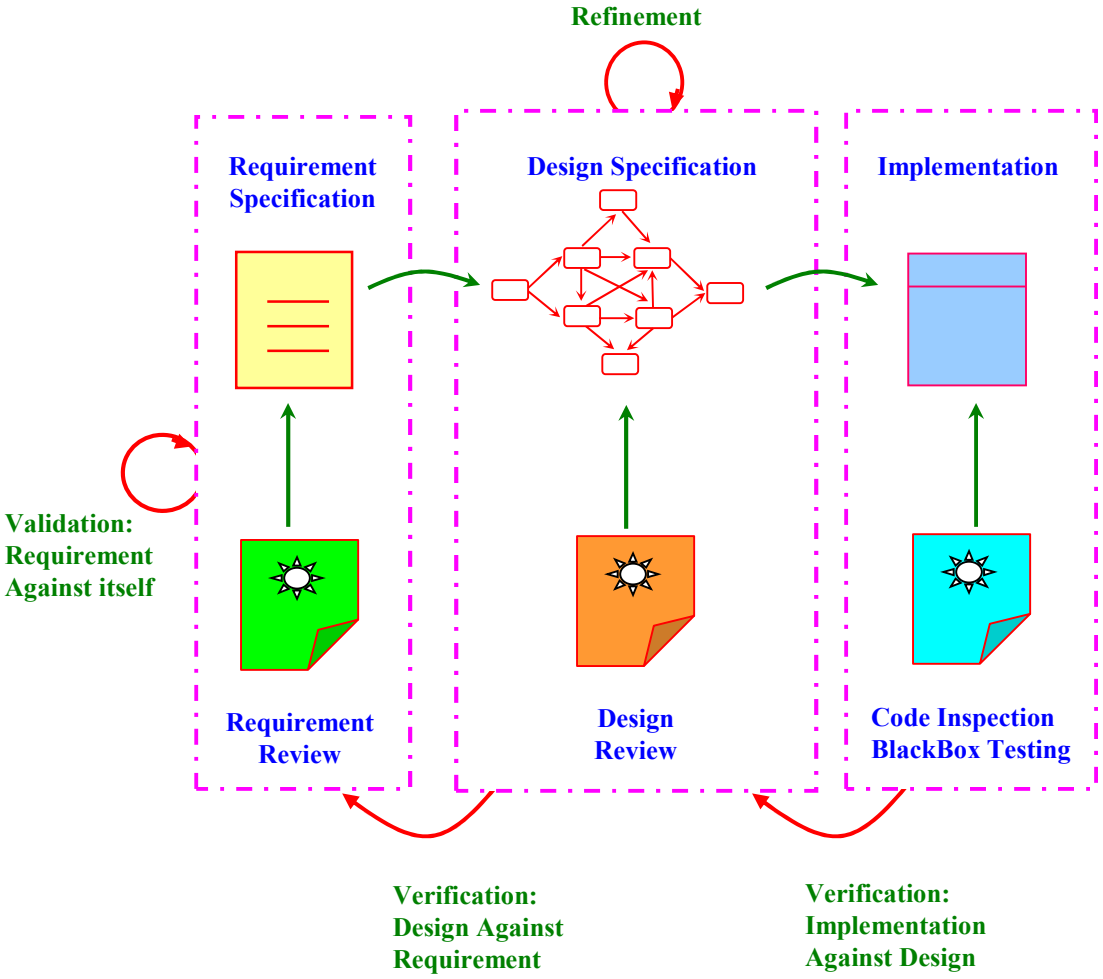
Software Project

What is the goal?

Balance the main three (other 2 constraints scope and resource) … in order to:

- Stay within the budget
- Deliver on time to gain market share
- Exceed customer satisfaction

Verification and Validation



Software Project Expenditures

Activity	Cost
Management	5%
Requirements	5%
Design	10%
Code and Unit Testing	30%
Integration and Acceptance testing	40%
Deployment	5%
Environment	5%
Total	100%

The 80% Benchmark

80%	Of the engineering is consumed by 20% of the requirements
80%	Of the software cost is consumed by 20% of the components
80%	Of the errors are caused by 20% of the components
80%	Of software scrap and rework is caused by 20% of the errors
80%	Of the resources are consumed by 20% of the components
80%	Of the engineering is accomplished by 20% of the tools
80%	Of the progress is made by 20% of the people

Conventional Software Management Performance

Barry Boehm's "Industrial Software Metrics Top 10 List":

1. Finding and fixing a software problem after delivery costs 100 times more than finding and fixing the problem in early design phases
2. You can compress software development schedules 25% of nominal, but no more
3. Fore every $1 you spend on development, you will spend $2 on maintenance
4. Software development and maintenance costs are primarily a function of source lines of code.
5. Variations among people account for the biggest difference in software productivity; hire good people to succeed.

Conventional Software Management Performance

6. The overall ratio of software to hardware costs is still growing.
7. Only about 15%of software development effort is devoted to programming
8. Software systems and products typically cost 3 times as much per SLOC as individual software programs. Software system products (system of systems) costs 9 times as much
9. Walkthroughs catch 60% of the errors
10. 80% of the contributions comes from 20% of the contributors.

Improving Software Economics

Cost Model Parameters	Trends
Size Abstraction and component-based development technologies	•Higher Order Languages •Object-Oriented OAD/OOP Reuse •Commercial Components
Process Methods and techniques	•Iterative development •Process maturity models •Architecture-first development
Personnel People factors	•Training and Personnel skill development •Teamwork •Win-win cultures
Environment Automation technologies and tools	•Integrated tool (visual modeling, compiler, editor, debugger ..) •Automation of coding, documents, testing
Quality Performance, reliability, accuracy	•Hardware platform performance •Statistical Quality Control

Software Project

How to achieve this goal?

Follow the documented processes when executing the following phases of the software project:

1. Define - Scope the project
2. Plan - Develop the project plan
3. Execute - Launch the plan
4. Monitor - Monitor/ control project progress
5. Close - Close out the project

Software Project

What is the software development process?

A process **is a set of documented** procedures, methods, practices, and tools used to produce a software product.

The process will answer the following:

- What to do? Tasks/activities
- How to do it? Procedure/practice
- When to do it? Sequence of activities
- What are the artifacts? (input/output)

Software Project

What is the difference between a process and Procedure/Methodology?

- The Programmer knows the procedure to create a Java program, based on the language syntax, compile the program, and then run it
- The Designer applies some OOD methodology when writing the detailed design document

Software Project

- But both the programmer and designer may produce some artifacts that are
  - ❑ Not compliant with the requirements
  - ❑ Hard to maintain
  - ❑ Hard to comprehend, not following consistent writing style
  - ❑ Not with acceptable quality
  - ❑ Not within acceptable milestones

The problem: we didn't know
the micro-steps of what/how/when/where it happened !

Software Project

If the programmer and designer follow the **process**, then the artifacts they produce will be

- ✓ Predictable
- ✓ Based on the requirements
- ✓ Easy to maintain and control
- ✓ consistent with the writing style
- ✓ Of acceptable quality
- ✓ within acceptable milestones

By following the process, we will be able to know precisely what/how/when/where it happened !

Software Project

The **process is**

- ✓ Not for Heroes
- ✓ For average technical staff to use
- ✓ For technical staff with software skills but limited capacity
- ✓ Not to increase productivity; in fact it may decrease it

The internet browser didn't require a hero to produce, it required a skilled team with a plan

Software Processes

Software Process is an overloaded term

- **Metaprocess**: an organization's policies, procedures, and practices for pursuing a software-intensive line of business; the focus is on organizational economics, an dlong-term strategies.
- **Macroproces**: the project's policies, procedures, and practices for producing a complete software product within certain cost, schedule, and quality constraints.
- **Microprocess**: a project team's policies, procedures, and practices for achieving an artifact of the software process.

Attributes of Software Processes

Attributes	Metaprocess	Macroprocess	Microprocess
Subject	•Line of business	•Project	•Iteration
Objectives	•Business profitability •Competitiveness	•Project profitability •Risk Management •Project budget, schedule, quality	•Resource management •Risk resolution •Milestone budget, schedule, quality
Audience	•Customers •Organizational management	•Software project managers •Software engineers	•Subproject managers •Software engineers
Metrics	•Project predictability •Revenue, market share	•On budget, on schedule •Major milestone success •Project scrap and rework	•On budget, on schedule •Major milestone progress •Release/iteration scrap and rework
Concerns	•Bureaucracy vs. Standardization	•Quality vs. Financial Performance	•Content vs. schedule
Time Scales	6 to 12 months	1 to many years	1 to 6 months

Software Project

Quality Engineering Principle:

The quality of the software system is controlled by the quality of the process used to produce that software.

Quality Management Principle:

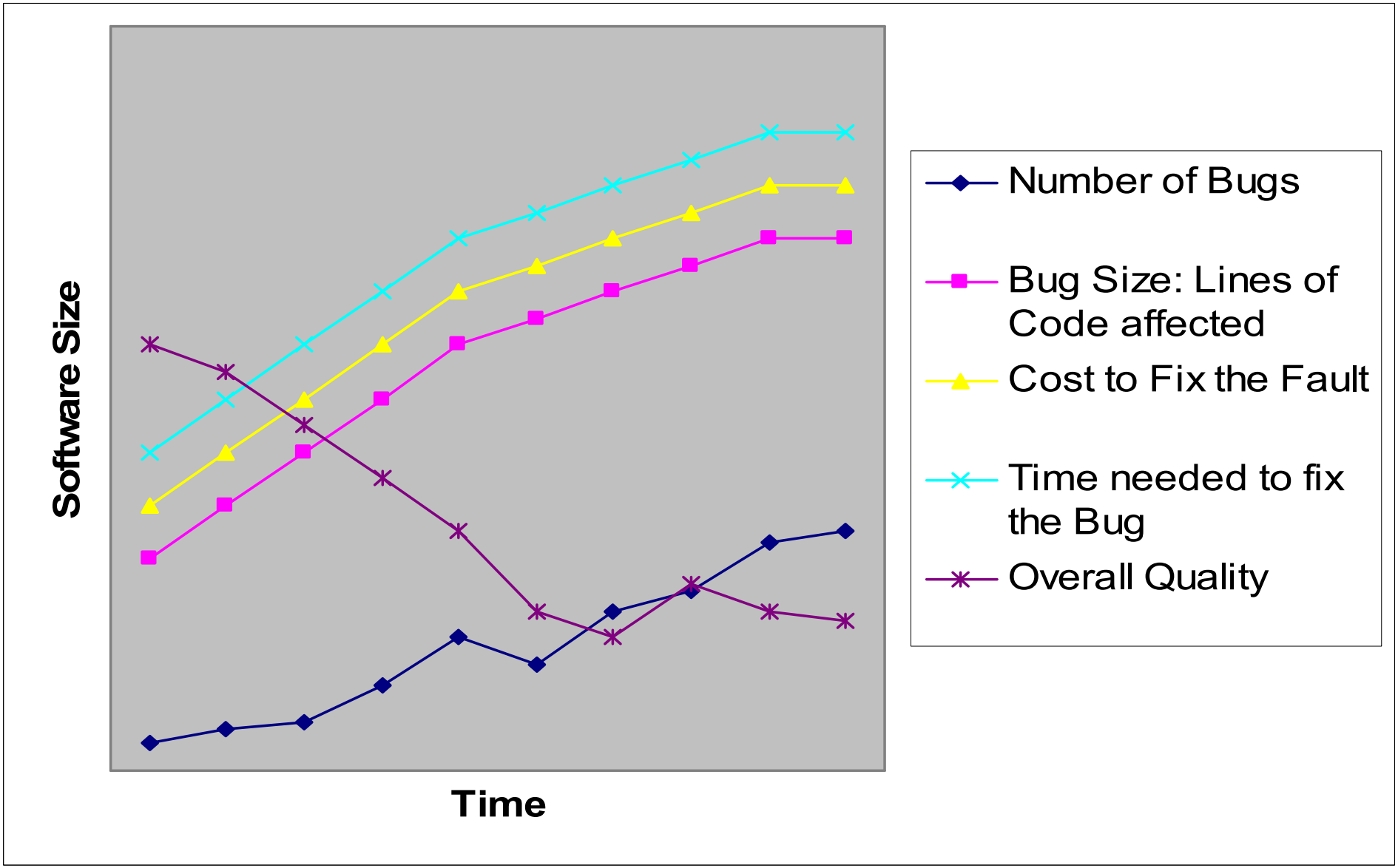
- Document the process
- Measure the process
- Improve the process based on the measurement

Hardware vs. Software

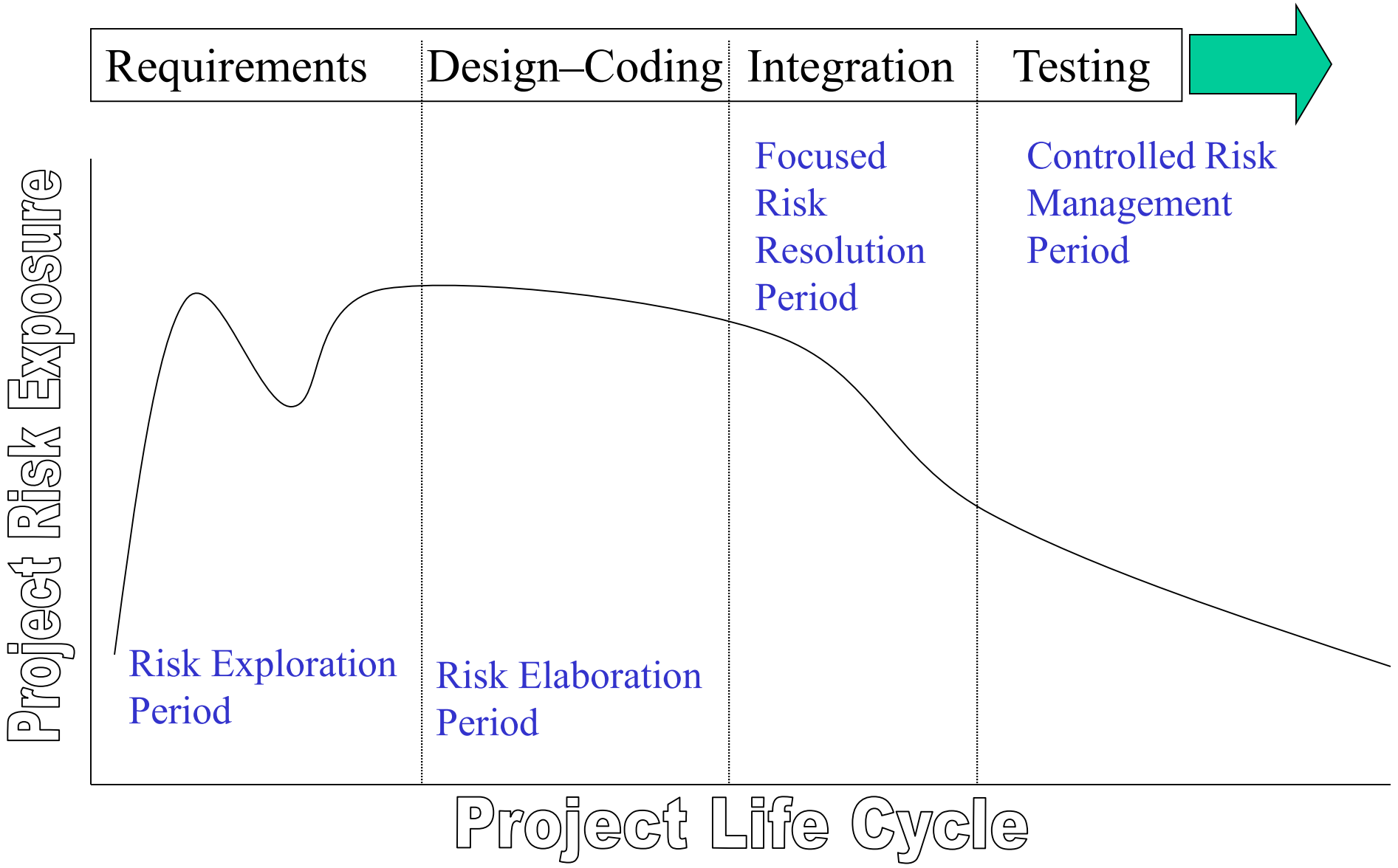
- Both Hardware and Software are becoming component-oriented; assemble system from basic components through plug-and-play
- Hardware: Getting smaller, cheaper, and faster

- Software: Getting larger
  - Is it really getting more expensive?
    - Customers demand more features
    - Regression testing cost becomes a real burden on the budget
    - The Overall ratio of software to hardware costs is still growing.
  - Is it really getting slower?
    - More feature interactions may cause a slow-down
    - Processor speed doubles every 18 months

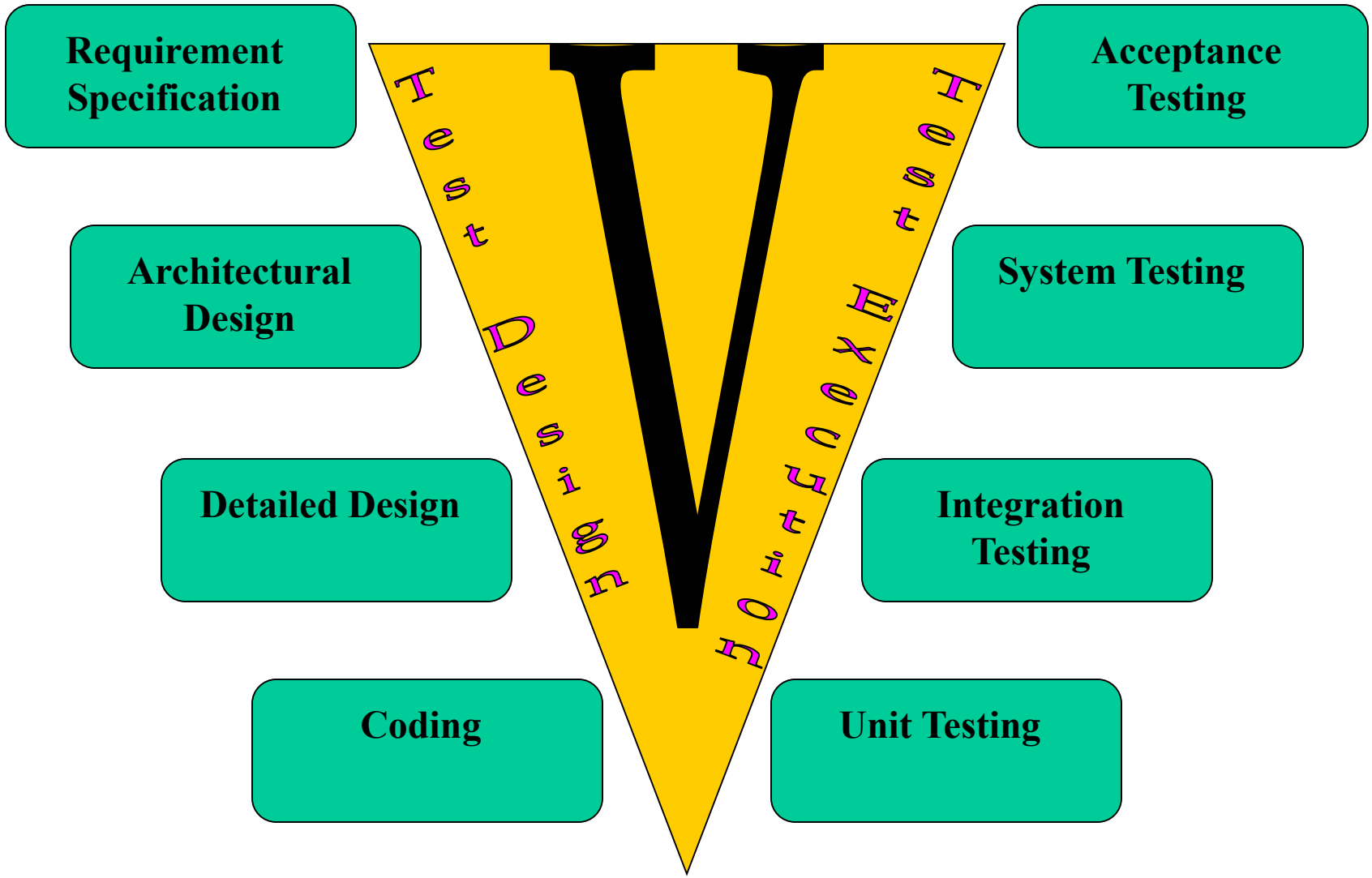
Software Trends



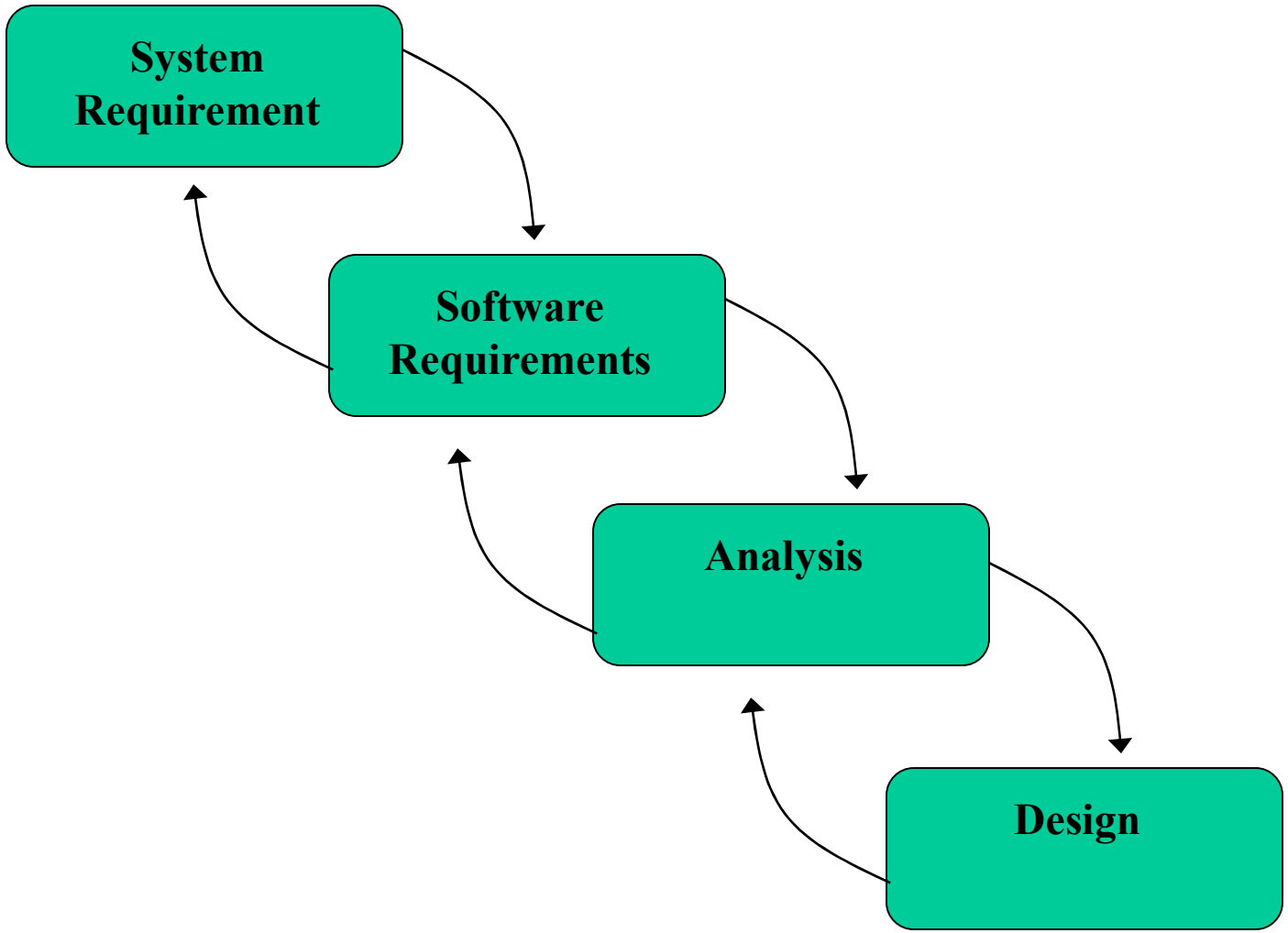
Risks of the Software Projects



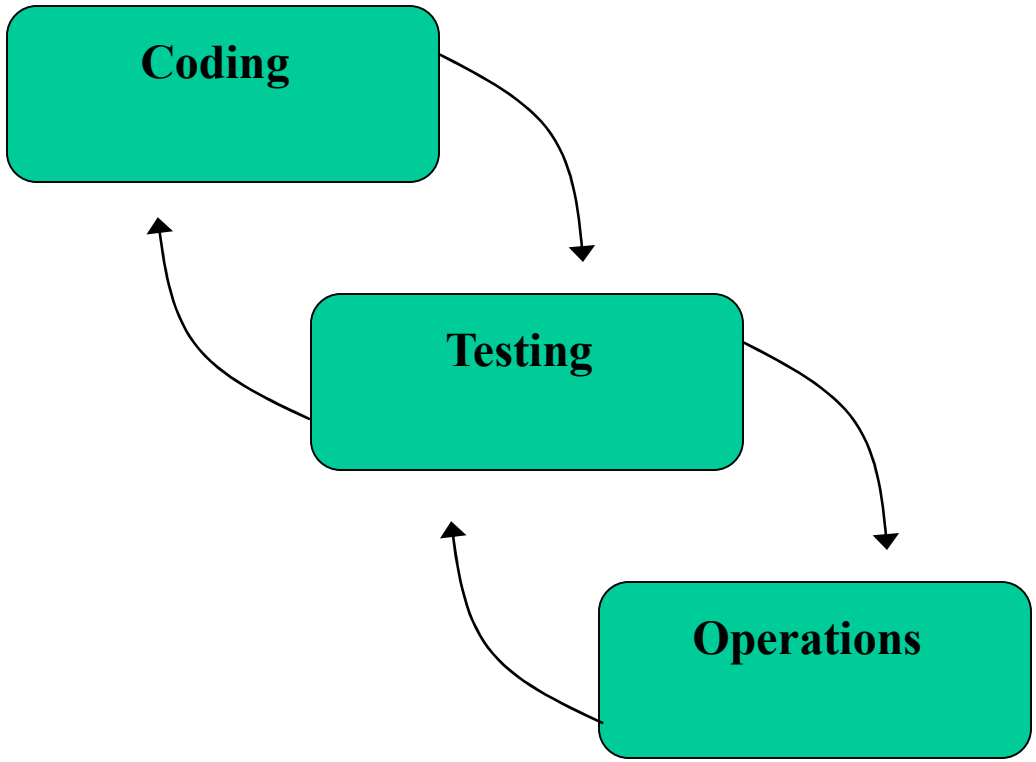
The V-Model of the Software Development



Waterfall Model of Software Development



Waterfall Model of Software Development



Software System Stakeholders

Customer	Manager	Architect	Developer	Tester
Requirements Cost Schedule Performance Reliability Security	Cost Schedule Requirements Process Resources	Maintainability Portability Feasibility Reusability Extensibility Flexibility The ***ilities***	Components Connectors Class/Pattern Data flow Reuse Flexibility Extensibility	Functionality Requirements Regression Tools Simulators

Requirement Driven Approach

The ATM machine's Requirement & Specification Document, RSD, may have the following requirements:

<Req 1> The system shall provide a text field that will allow the user to enter the pin number

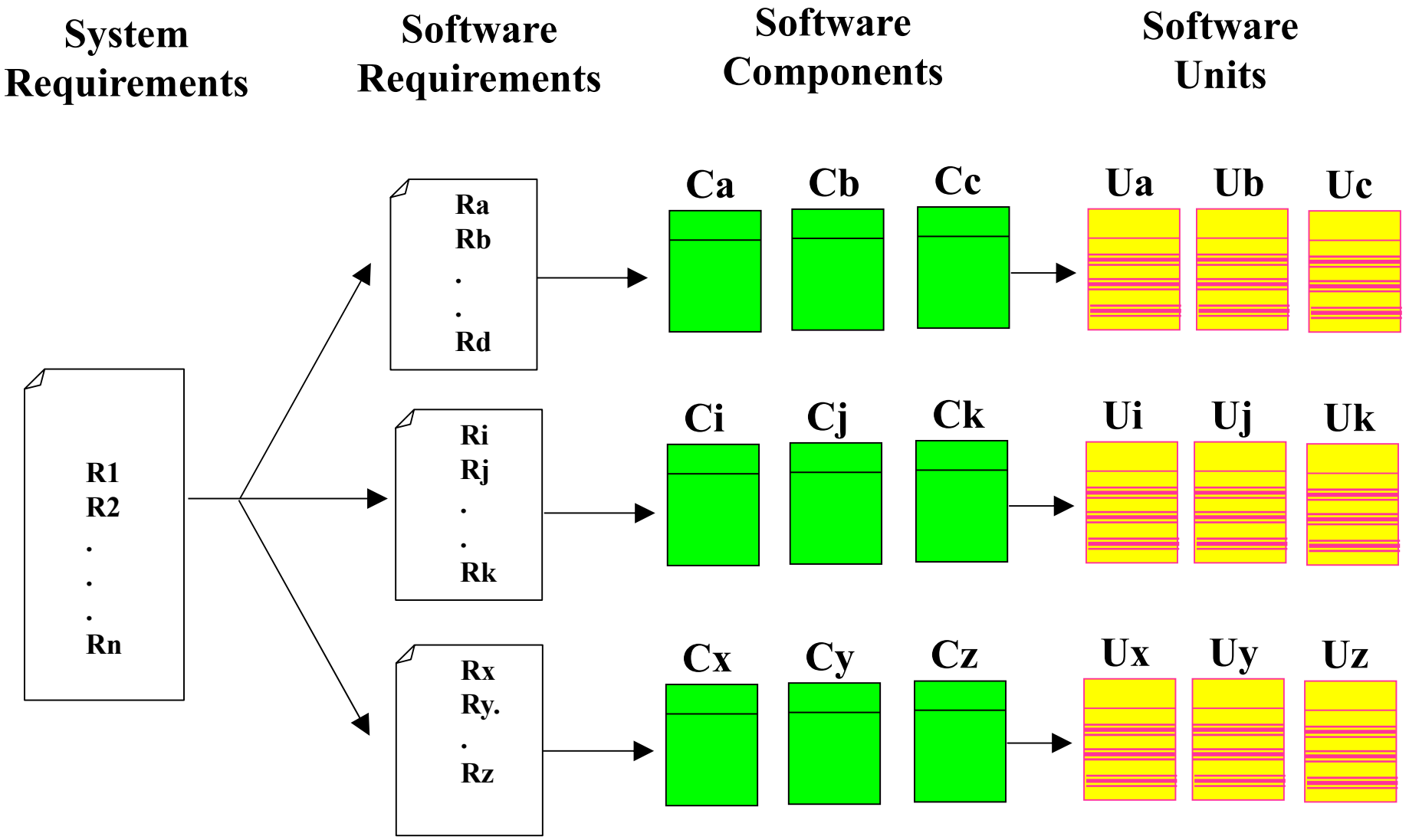
<Req 2> If the user entered a valid pin, the system shall provide a menu for the user in order to select a transaction

<Req 3> ….

<Req 4> …

<Req 5> …

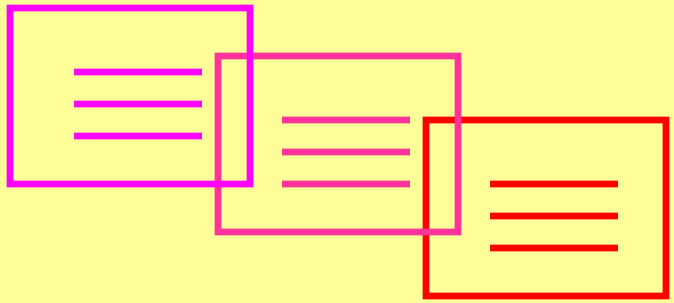
Requirement Driven Approach



Software Requirements and Specifications

Software Requirements and specification shall have the following properties:

- Abstract
- Concise
- Complete
- Unambiguous
- Maintainable
- Comprehensible
- Cost-Effective



Software Requirements and Specifications

- Can we automate the process of mapping informal specs into formal specs?
- Can we get rid off the prose requirements?
- How about a semi-automated mapping process?

The focus here is to find a semi-automated mapping process from the prose requirements to the formal mathematical model

Software Requirements and Specifications

- Mathematical models are used to prove correctness and precision of requirements, while software engineering principles are used to produce the actual products

Kathryn L. Heninger (worked for DoD + D. Pranas) listed 3 Principles for requirement document design:

1. State questions before trying to answer them
2. Separate concerns
3. Be as formal as possible

Software Requirements and Specifications

- Kathryn L. Heninger developed the following objectives for the existing flight software for the Navy's A-7 requirement document.

1. Specify external behavior only
2. Specify constraints on the implementation
   (Be alarmed to this one, at some point we said We want to have implementation-free specifications)
3. Be easy to change
4. Serve as a reference tool
5. Record forethought about the life cycle of the system
6. Characterize acceptable responses to undesired events

Science vs. Engineering

- Science focuses on ***facts+theories+formulas+proofs*** about the world
- Engineering emphasizes on how to apply the ***facts+theories+formulas*** ***in the*** design of the products
- When we talk about software, recognize the:
  - Impact of engineering on software development
  - Impact of people on software development

Science vs. Engineering

As a professional software engineer, prepare an answer for this question:
Do you write the specification of the car based on the road that you will use?

Science vs. Engineering

- Computer Science:
  - Is NOT about teaching tools; you should learn tools on your own
  - Is NOT ONLY how to use a programming language: syntax+semantic. Many programming languages come and go without us being aware that they even existed. What happened to Algol68, Pascal, …? However y=mx+b is still here and will continue to be there.
  - Is about automata theory, language theory
  - Discrete mathematics, Algorithms
  - Is about formal specifications and proof of correctness; programs and their mathematical descriptions are two different things.

Software Engineering vs. Software Project Management

- Software Engineering applies the sound scientific theories and principles to produce **software products**
- Software Project Management is the art to define, plan, execute, and monitor the activities that will bring **software products** to existence.

Programming as Engineering

- David Parnas in his famous commentary paper "Teaching Programming as Engineering" listed the following as the Mathematics needed for professional programming:

1. *Finite State Machines*
2. *Set, Functions, Relations, Composition*
3. *Mathematical Logic Based on Finite Sets*
4. *Programs as "Initial states"*
5. *Programs as Descriptions of State Sequences*
6. *Programs Described by functions from starting states to stopping state*
7. *Tabular Descriptions of Functions and Relations*