Software Engineering

Software Quality Assurance

LEARNING OBJECTIVES

1. Understand the quality assurance process and the central process activities of quality assurance, quality planning and quality control.

2. Understand the importance and use of standards and metrics in the quality assurance process.

3. Understand the principles of software development process improvement and why process improvement is worthwhile.

SOFTWARE QUALITY ASSURANCE (SQA): OUTLINE

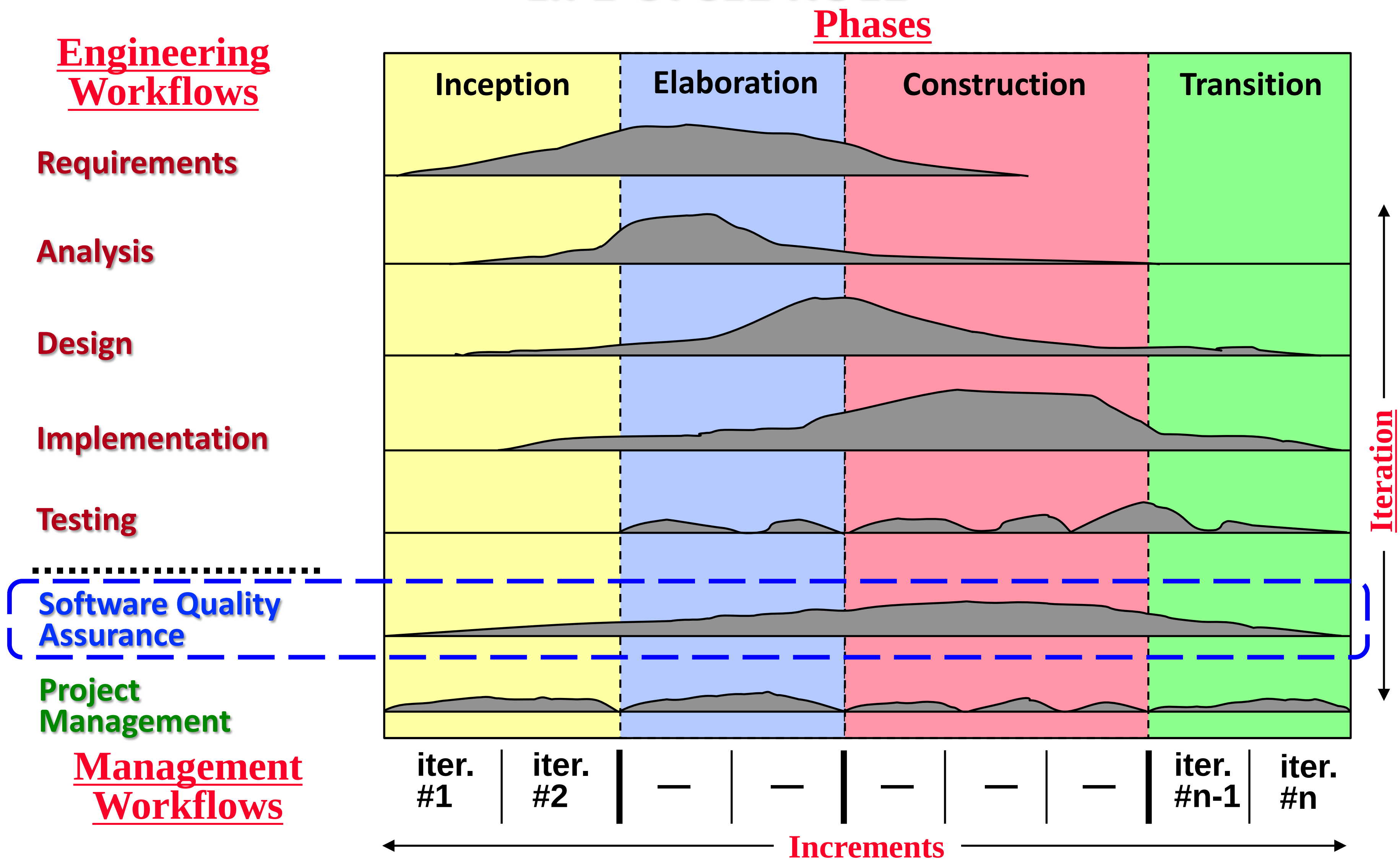
Software Quality Assurance

- Life Cycle Role
- Purpose and Importance

Achieving Software Quality

- SQA Activities
- Achieving Product Quality
- Achieving Project Quality
- Achieving Process Quality
- Achieving People Quality

SOFTWARE QUALITY ASSURANCE LIFE CYCLE ROLE



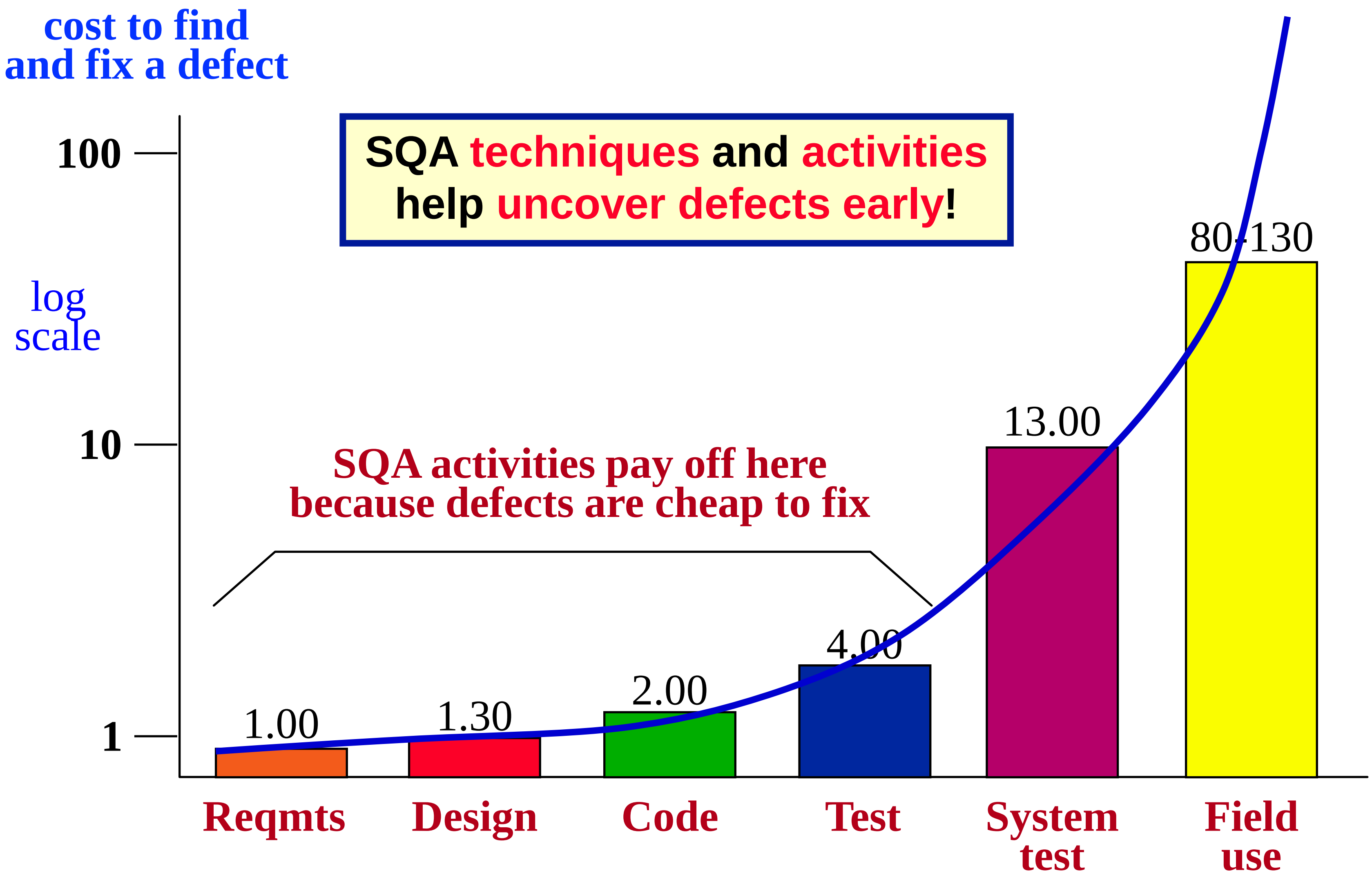
PURPOSE OF SOFTWARE QUALITY ASSURANCE

Quality assurance consists of those procedures, techniques, and tools applied by professionals to ensure that a product meets or exceeds pre-specified standards during it's development cycle. E.H. Bersoff

quality assurance	Defines organizational standards that lead to high quality software.	**WHAT**
requires ↓		
quality planning	Selects and tailors standards to a specific software product.	**HOW**
requires ↓		
quality control	Ensures standards are followed.	**MONITOR**

***Continuous* quality improvement should be the *overall goal*.**

IMPORTANCE OF SOFTWARE QUALITY ASSURANCE

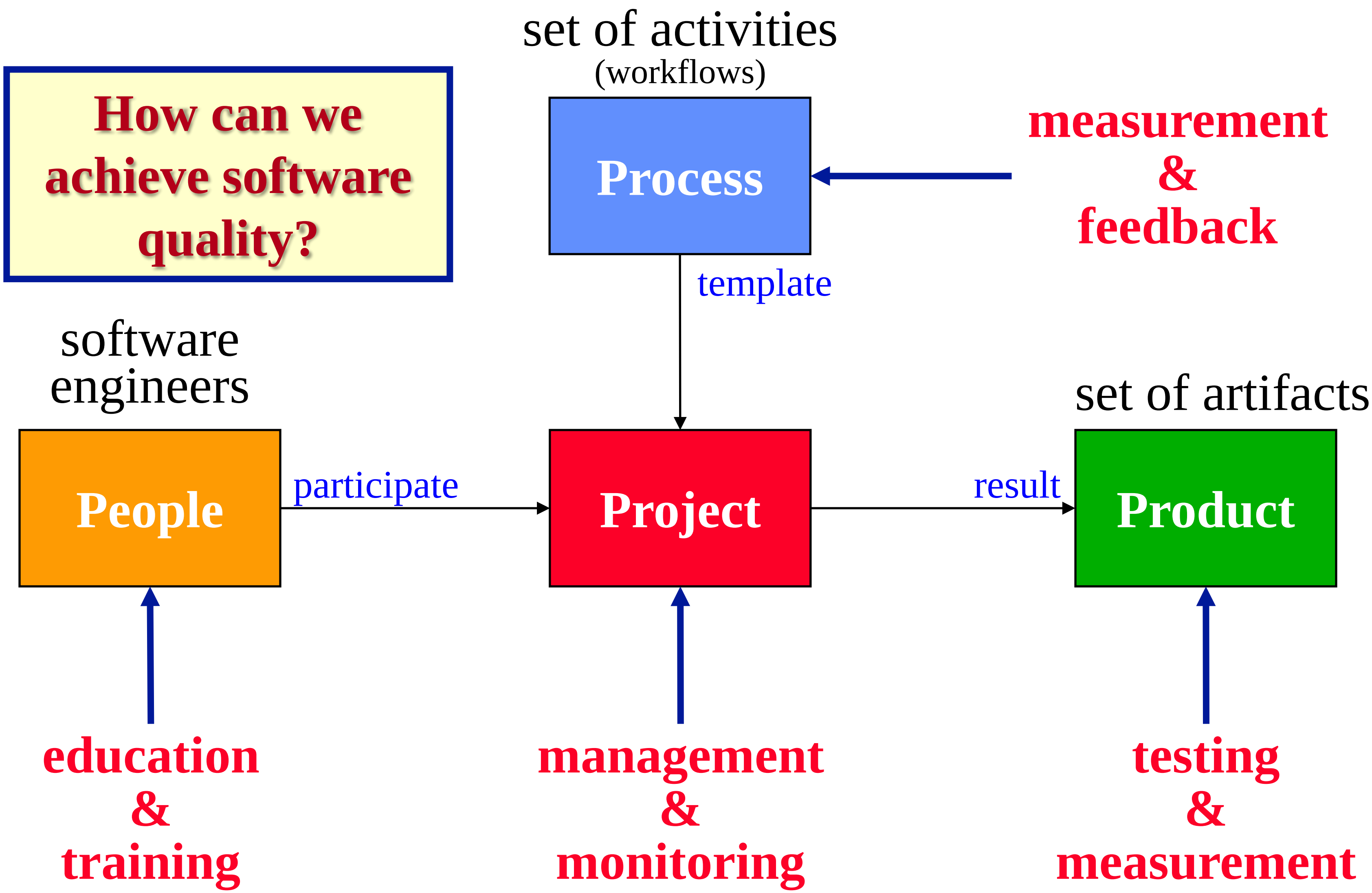


SOFTWARE QUALITY ASSURANCE (SQA): OUTLINE

- ✓ Software Quality Assurance
  - Life Cycle Role
  - Purpose and Importance

- ➔ **Achieving Software Quality**
  - SQA Activities
  - Achieving Product Quality
  - Achieving Project Quality
  - Achieving Process Quality
  - Achieving People Quality

ACHIEVING SOFTWARE QUALITY



ACHIEVING SOFTWARE QUALITY (cont'd)

1. We should have a set of quality attributes that a software product must meet.

   ☞ **There are design goals to achieve.**

2. We should be able to measure a quality attribute.

   Key point

   ☞ **There is a way to determine how well the product conforms to the design goals.**

3. We should track the values of the quality attributes.

   ☞ **So that it is possible to assess, over time, how well we are doing in achieving the design goals.**

4. We should use information about the quality of any developed software to improve the quality of future software products.

   ☞ **There is feedback into the software development process.**

SOFTWARE QUALITY ASSURANCE (SQA): OUTLINE

✓ Software Quality Assurance

– Life Cycle Role

– Purpose and Importance

➔ **Achieving Software Quality**

☞ **SQA Activities**

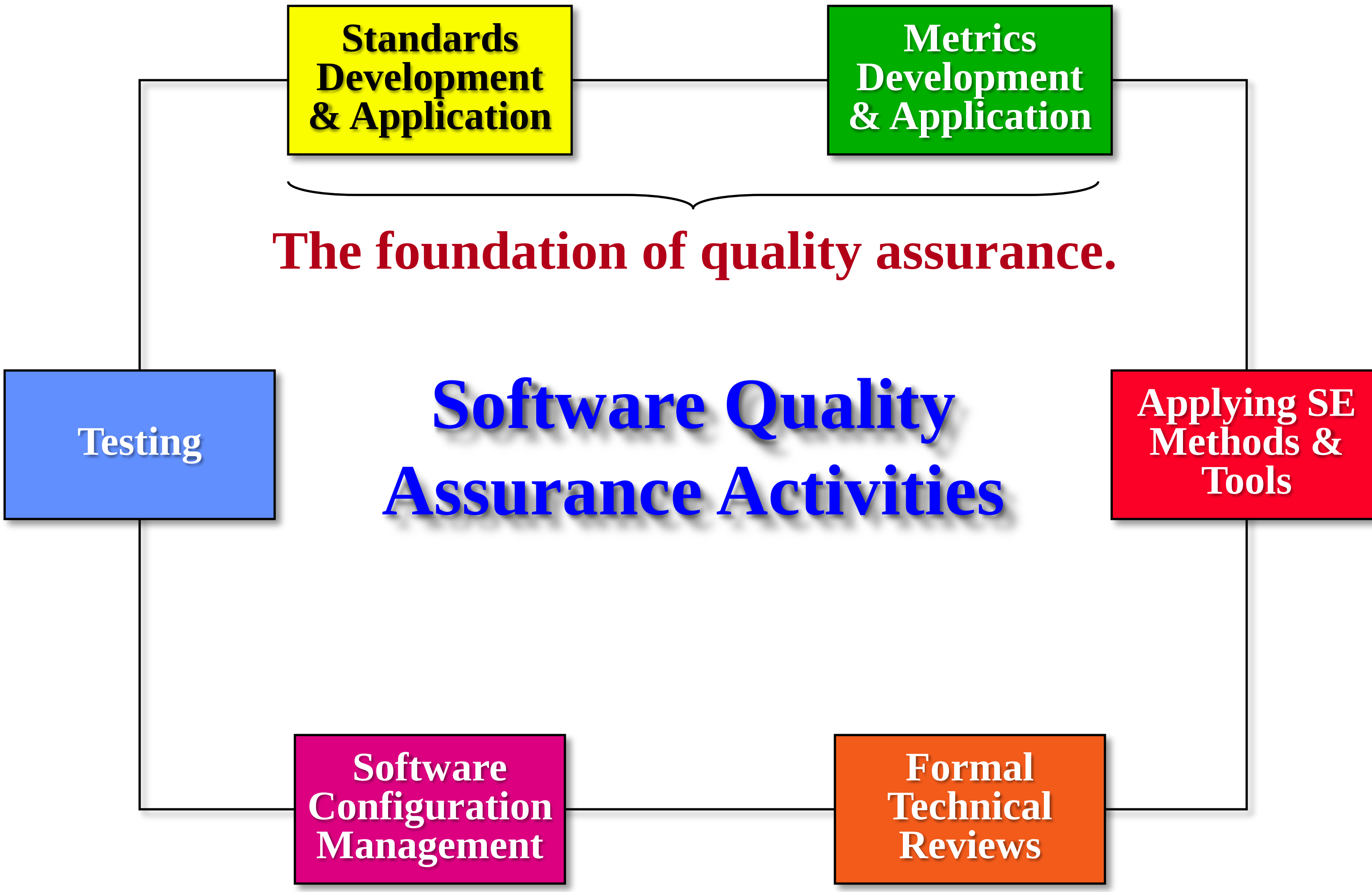
– Achieving Product Quality

– Achieving Project Quality

– Achieving Process Quality

– Achieving People Quality

SQA ACTIVITIES



SQA ACTIVITY: STANDARDS

A (technical) *standard* is a norm or requirement that establishes uniform engineering or technical criteria, methods, processes and practices.

The standards that are important for software engineering are:

1. **product standards** which are concerned with *what outcome is produced* and define the *characteristics that all product artifacts should exhibit* so as to have quality.

2. **process standards** which are concerned with *how the outcome is produced* and define *how the software process should be conducted* to ensure quality software.

SQA ACTIVITY: STANDARDS

Why are standards important for Software Quality Assurance?

1. They document the best (or most appropriate) practices.

   ☞ Helps avoid previous mistakes.

2. They provide a framework around which to implement quality control.

   ☞ Ensures that the best practices are properly followed.

3. They assist in ensuring the continuity of project work.

   ☞ Reduces learning effort when starting new work.

Standards Handbook needed!

Each project needs to decide which standards should be:
ignored; used as is; modified; created.

SQA ACTIVITY: METRICS

A *metric* is any type of measurement that relates to a software product, process or related artifact.

Why are metrics important for Software Quality Assurance?

1. Metrics can be used to control (i.e., plan and manage) the development process (e.g., effort expended, elapsed time, budget spent, etc.).

2. Metrics can be used to predict an associated product quality (e.g., cyclomatic complexity can predict ease of maintenance).

***Metrics* are the only objective way to measure quality attributes of software; otherwise it is all opinion and guess work.**

SOFTWARE QUALITY ASSURANCE (SQA): OUTLINE

- ✓ Software Quality Assurance
  - – Life Cycle Role
  - – Purpose and Importance

- ➔ **Achieving Software Quality**
  - ✓ SQA Activities
  - ☞ **Achieving Product Quality**
  - – Achieving Project Quality
  - – Achieving Process Quality
  - – Achieving People Quality

ACHIEVING PRODUCT QUALITY: DESIGN GOALS

- Recall that a design goal is an (external) *quality attribute* that we want the system to have such as:
  - safety
  - testability
  - reusability
  - robustness
  - understandability
  - usability
  - resilience
  - maintainability
  - portability
  - adaptability
  - modularity
  - learnability
  - security
  - reliability
  - efficiency

- Usually, we can only asses whether a system has one of these attributes (i.e., achieves its design goals) *after* it is completed!

  ☞*An external attribute cannot be measured directly from the software.*

- However, we would like to assess whether we are achieving a design goal during the system's development? **How to do it?**

Try to use internal attributes (things we can measure from the software) to assess (predict) external attributes (i.e., design goals).

ACHIEVING PRODUCT QUALITY: DESIGN GOALS (cont'd)

- Examples of possible relationships between external and internal attributes:

External (design goal)	Internal (measurable)
maintainability	# of parameters
reliability	cyclomatic complexity
portability	lines of code
usability	# of error messages
	length of user manual

Problems

1. **It is hard to formulate and validate relationships between internal attributes and design goals.**
2. **Quantitative software information for computing metrics must be collected, calibrated and interpreted.**

ACHIEVING PRODUCT QUALITY:
QUALITY METRICS FOR SYSTEM DESIGN

For a *design component*, a key design goal is maintainability.

- Experience tells us that maintainability should be related to the complexity of a design component.
- Complexity, in turn, is related to quality attributes such as:
  - cohesion
  - coupling
  - understandability
  - adaptability

Some of these attributes also cannot be measured directly!

So, what can we measure to determine the complexity of a design component and thus it's maintainability?

ACHIEVING PRODUCT QUALITY:
QUALITY METRICS FOR SYSTEM DESIGN

1. **Structural fan-in/fan-out**

   fan-in – the number of calls to a component by other components

   fan-out – the number of components called by a component

   ☞ **high fan-in ⇒ high coupling**

   ☞ **high fan-out ⇒ calling component has high complexity**

2. **Informational fan-in/fan-out**

   – Consider also the number of parameters passed plus the number of accesses to shared data structures.

   $$\text{complexity} = \text{component-length} * (\text{fan-in} * \text{fan-out})^2$$

   ☞ **This metric has been validated in the Unix system.**

   ☞ **It is a useful predictor of effort required for implementation.**

ACHIEVING PRODUCT QUALITY:
QUALITY METRICS FOR SYSTEM DESIGN

3. IEEE Standard 982.1-1988

- Considers properties of:
  - **subsystems** (number of subsystems and degree of coupling)
  - **database** (number of attributes and classes)

☞ **Compute a design structure quality index—DSQI → (0-1).**

☞ **Used to compare with past designs; if DSQI is too low, further design work and review may be required.**

- We can also consider changes made throughout the product's lifetime and compute how stable it is (i.e., how many changes have been made in subsystems in the current release).

☞ **Define a software maturity index—SMI → (0-1).**

☞ **As SMI approaches 1, the product begins to stabilize.**

ACHIEVING PRODUCT QUALITY:
QUALITY METRICS FOR IMPLEMENTATION

For an *implementation component (i.e., code)*, some key design goals are reliability and ease of implementation.

Some approaches to measure reliability and/or difficulty:

1. **Halstead's Software Science**

   Looks at the number of operators and operands in a component and calculates values for component volume, *V (in bits)*, component difficulty, *D*, and effort, *E*, required to implement the component.

2. **McCabe's Complexity Metric**

   Looks at **control flow** in a component and calculates **cyclomatic complexity**.

3. **Lines of code (LOC)**

4. **Length of identifiers**

5. **Depth of conditional nesting**

Standards should be established to avoid complex components and/or highlight problem components.

ACHIEVING PRODUCT QUALITY:
FORMAL APPROACHES

1. **Proving programs/specifications correct**
   - Logically prove that requirements have been correctly transformed into programs (e.g., prove assertions about programs).
2. **Statistical Quality Assurance**
   - Categorize and determine the causes of software defects.
   - Use **80-20 rule**: 80% of defects can be traced to 20% of causes.
   - Isolate and correct the 20% of causes, which fixes 80% of defects.

☞ **The development effort is directed to things that cause the majority of defects.**

3. **The Cleanroom Process**
   - A combination of the above two approaches that can be used to produce extremely reliable software.

SOFTWARE QUALITY ASSURANCE (SQA): OUTLINE

- ✓ Software Quality Assurance
  - – Life Cycle Role
  - – Purpose and Importance

- ➔ **Achieving Software Quality**
  - ✓ SQA Activities
  - ✓ Achieving Product Quality
  - ☞ **Achieving Project Quality**
  - – Achieving Process Quality
  - – Achieving People Quality

ACHIEVING PROJECT QUALITY: REVIEWS

***Reviews* are the primary method for achieving project quality.**

Requirements capture — requirements walkthroughs

Analysis — analysis walkthroughs

Design — design walkthroughs

Implementation — code walkthroughs

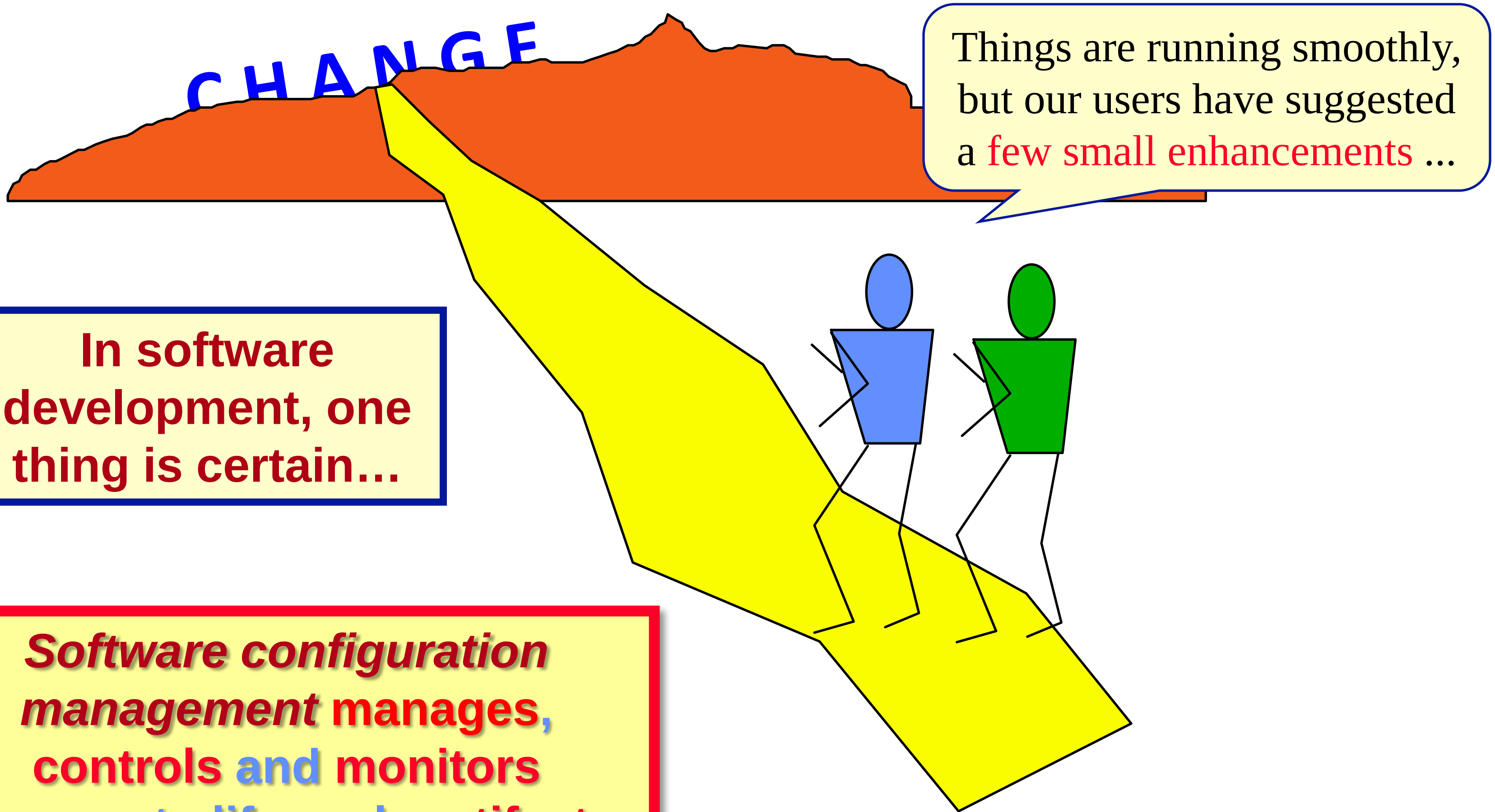
Testing — test plan review

Leads to early discovery of defects.

Requirements, Analysis and Design introduce 50-60% of all defects.

Formal technical reviews can uncover 75% of these!

ACHIEVING PROJECT QUALITY:
SOFTWARE CONFIGURATION MANAGEMENT (SCM)



In software development, one thing is certain…

***Software configuration management* manages, controls and monitors changes to life cycle artifacts.**

SOFTWARE QUALITY ASSURANCE (SQA): OUTLINE

✓ Software Quality Assurance
- Life Cycle Role
- Purpose and Importance

➔ **Achieving Software Quality**
✓ SQA Activities
✓ Achieving Product Quality
✓ Achieving Project Quality
☞ **Achieving Process Quality**
- Achieving People Quality

ACHIEVING PROCESS QUALITY

How important is the process in software development?

- Software development has some unique factors that affect the product quality *independent* of the process used:
  - software is designed, not manufactured.
  - software development is creative, not mechanical.
  - external factors (novelty, competitive advantage) may impact quality.
  - individual skills and experience can have a significant influence.

☞ **Sometimes people & technology are more important than process.**
(People quality/expertise, development tools may have more impact on product quality than the process used.)

☞ **Insufficient resources will always adversely affect product quality regardless of the process used.**

☞ **A detailed software development process is usually not transferable since it is highly organization-specific.**

ACHIEVING PROCESS QUALITY: ISO 9001/9000-3

ISO 9000-3 **is intended to help a client assess the process management capabilities of a software organization.**

- ISO 9000 specifies actions to be taken when any system (not necessarily a software system) has quality goals and constraints.
- ISO 9001 (clause 4.2) requires an organization to have a documented quality system including a quality manual, plans, procedures and instructions.

  ☞ **The quality manual defines and documents an organization's quality process.**

- ISO 9000-3 explains how the quality system should be integrated throughout the software development process.

  ☞ **It specifies generic procedures that should be in place to have a quality process.**

ACHIEVING PROCESS QUALITY: SEI PROCESS CAPABILITY MATURITY MODEL (CMM)

SEI-CMM **is intended to help a software organization assess and improve their software development processes.**

Level 1: Initial process (ad hoc)
No formal procedures, no cost estimates, no project plans, no management mechanism to ensure procedures are followed.

Level 2: Repeatable process (intuitive) (focus on management)
Basic project controls; intuitive methods used.

Level 3: Defined process (qualitative) (focus on management + engineering)
Development process defined and institutionalized. Training provided.

Level 4: Managed process (quantitative) (add metrics)
Measured process; process database established.

Level 5: Optimizing process (add feedback)
Improvement feedback; rigorous defect-cause analysis and prevention.

SOFTWARE QUALITY ASSURANCE (SQA): OUTLINE

- ✓ Software Quality Assurance
  - – Life Cycle Role
  - – Purpose and Importance

- ➔ **Achieving Software Quality**
  - ✓ SQA Activities
  - ✓ Achieving Product Quality
  - ✓ Achieving Project Quality
  - ✓ Achieving Process Quality
  - ☞ **Achieving People Quality**

ACHIEVING PEOPLE QUALITY:
PEOPLE CAPABILITY MATURITY MODEL (PCMM)

PCMM **is intended to assess and improve knowledge and skill of people.**

Level 1: Initial

No technical or management training provided; staff talent is not a critical resource; knowledge and skills stagnate; no organizational loyalty.

Level 2: Repeatable

Focus on developing basic work practices; staff recruiting, growth and development important; training to fill skill "gaps"; performance evaluated.

Level 3: Defined

Focus on tailoring work practices to organization's business; strategic plan to locate and develop required talent; skills-based compensation.

Level 4: Managed

Focus on increasing competence in critical skills; mentoring; team-building; quantitative competence goals; evaluation of effectiveness of work practices.

Level 5: Optimizing

Focus on improving team and individual skills; **use of best practices**.

SOFTWARE QUALITY ASSURANCE: RETROSPECTIVE

- An organization should have a **quality manual** which documents its software quality assurance procedures.
- Each project should have a **quality plan** which sets out the **quality attributes (design goals)** that are most important for that project and how they will be assessed.
- An organization should have well defined **standards** for its **software development process** and the accompanying **artifacts.**
- Mechanisms (processes) should be established that **monitor compliance** with all quality requirements of the organization.
- **Reviews** are the *primary means* of carrying out software quality assurance.
- Where practical, **metrics** can be used to *highlight anomalous parts* of the software that may have quality problems.

SOFTWARE QUALITY ASSURANCE: SUMMARY

Quality software does not just happen!

- Software quality assurance needs to be built into the software development process.
- Developing quality software requires:
  - Management support and involvement.
  - Standards that everyone follows.
  - Software metrics gathering and use.
  - Commitment to following the standards even when things get rough!

Testing is an important part of quality assurance, but its not all there is to obtaining a quality software product.