

# SOFTWARE QUALITY

CPTS 583

---

Software Quality Assurance

-- *SQA scope, architecture, and approaches*

# Outline

- Scope of SQA
  - Problems addressed by SQA, versus quality control
- SQA architecture
  - Hierarchy of SQA
  - Roles in SQA
  - Tasks in SQA
  - Software standards
- Approaches to SQA
  - Software reviews
  - Verification and validation

# SQA overview

**SQA is:**

(1). A planned and systematic pattern of all actions necessary to **provide adequate confidence** that an item or product conforms to established **technical requirements**.

(2). A set of activities designed to **evaluate the process** by which the products are developed or manufactured.  
Contrast with *quality control*.

- **Beyond the development process**
- **Beyond technical aspects**

# SQA versus quality control

## What is quality control?

- a set of activities designed to **evaluate** the quality of a **developed or manufactured** product” (IEEE, 1991)

- SQA activities
  - Evaluate product
  - Evaluate software process
  - Beyond the process of development
- Quality control is **part of** quality assurance (SQA)

# SQA versus quality control

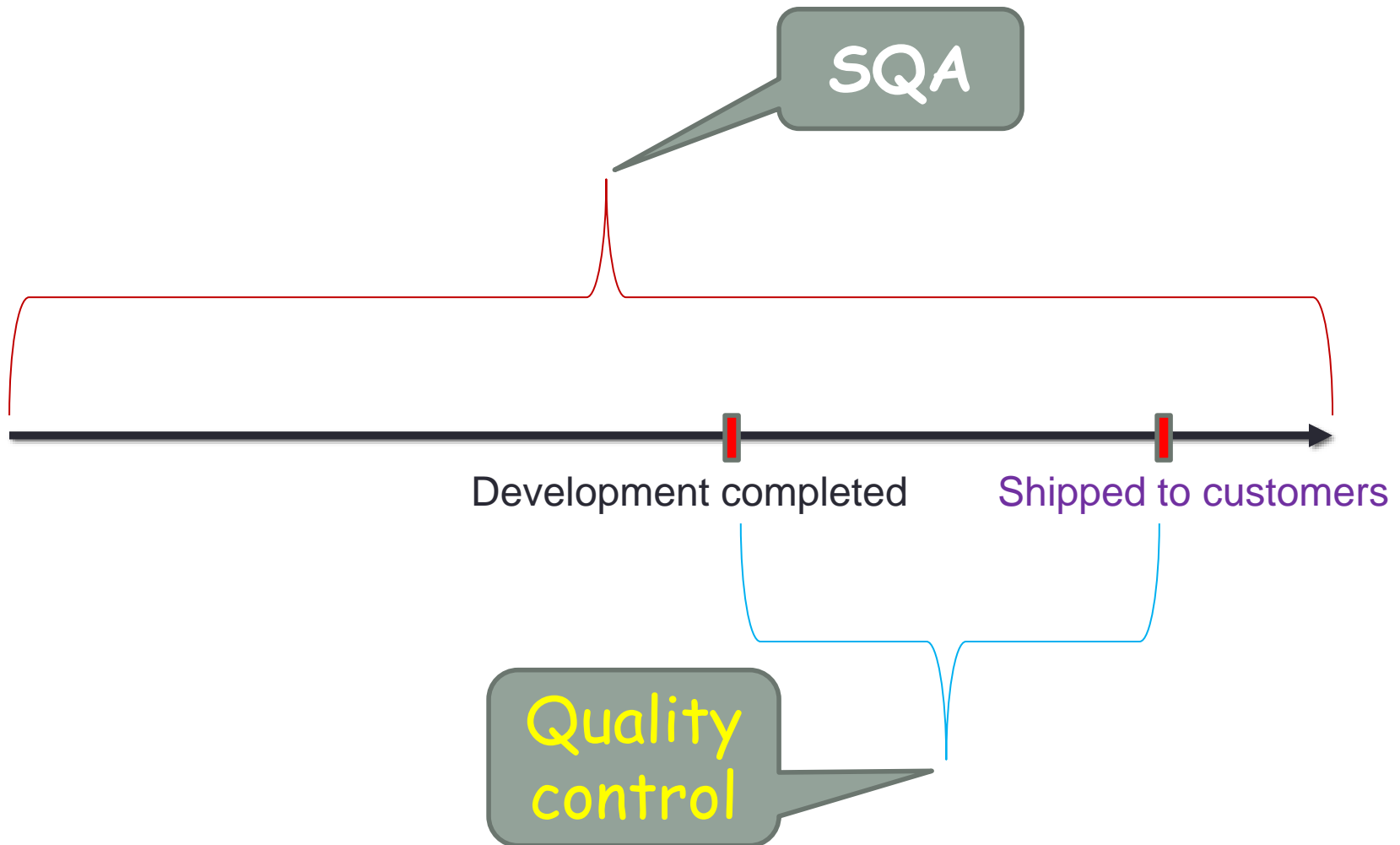
## SQA objectives

- Minimize the cost of guaranteeing quality
- Prevent causes of errors and correct them as early as possible
- Reduce the rate of producing products that do not qualify for shipment

## Quality-control objectives

- Withhold products that do not qualify for shipment

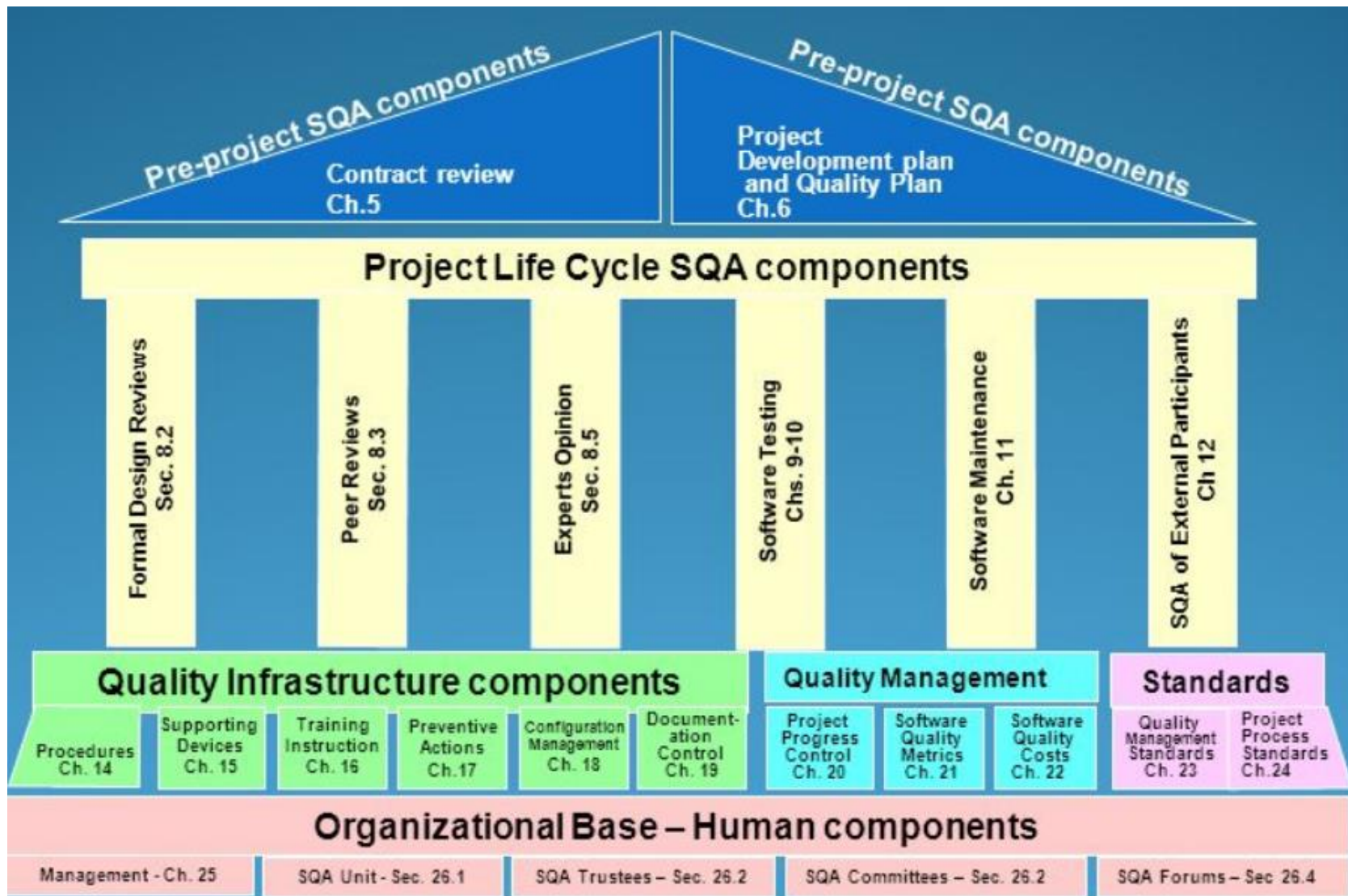
# SQA versus quality control



# SQA: the architecture

- An **umbrella** activity - throughout software process
- Encompasses
  - ✓ A quality management approach
  - ✓ Effective software engineering technology
  - ✓ Formal technical reviews
  - ✓ Multi-tier testing strategy
  - ✓ Control of software documentation and the changes made to it
  - ✓ A procedure to ensure compliance with software development standards
  - ✓ Measurement and reporting mechanism

# SQA: the architecture





# SQA: who does it

- SQA activities involve two groups



# SQA: what exactly to do

- SQA planning for a project
  - Evaluations to be performed
  - Audits and reviews to be performed
  - Standards that are applicable
  - Procedures for error reporting and tracking
  - Documents to be produced
  - Feedback to be given to developers



Part of  
the  
SQA  
plan for  
a DOE  
project

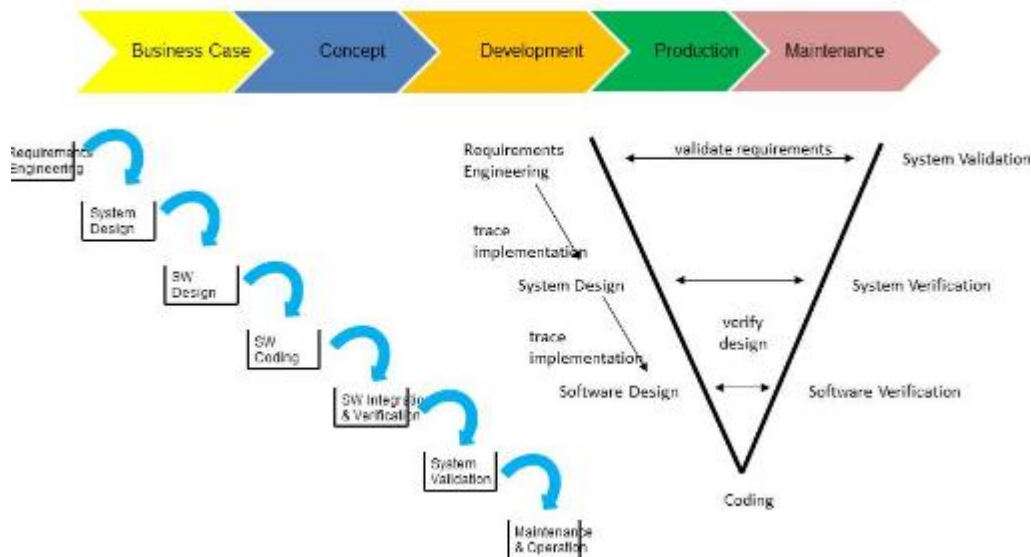
Stage	In-Stage Assess. Date	Stage Exit Date	Deliverable (date delivered)	Work Product (date completed)	QA Activity
Planning	03/03/97	03/07/97	Project Plan (02/24/97)	Project Schedule (02/24/97)	ISA Process - (review processes & audit contents)
Preparation	04/08/97	04/08/97	Functional Requirements Document Acquisition Plan Installation Plan	Revised Project Plan	ISA Process - (review processes & audit contents)
Software Design	05/29/97	06/05/97	Functional Design Document Security Plans Training Plan Sys. Test Plan Acceptance Plan Conf. Mgmt. Pln. Conversion Plan	Traceability matrix Revised Project Plan	ISA Process - (review processes & audit contents) Trace design components to requirements Trace requirements to design components
Programming and Integration	09/25/97	09/30/97	Site Installation Plan System documentation (draft)	Revised Project Plan	ISA Process - (review processes & audit contents)
System Testing and Acceptance	12/04/97	12/22/97	Test results System documentation (final) Operational system	Revised Project Plan	ISA Process - (review processes & audit contents)

# SQA: what exactly to do

- Development of project's software process description

Dev team selects a process model

- ✓ Organization policies
- ✓ Software standards
- ✓ Project plan



SQA team reviews the process description

# SQA: what exactly to do

- Verification of SE activities against the software process
  - Identifies, documents, track deviations
  - Verifies corrections
- Verification of software work products against the software process
  - Identifies, documents, track deviations
  - Verifies corrections
  - Periodically reports results to managers



# SQA: what exactly to do

- Documentation and reporting
  - Deviations against procedure
    - SE activities
    - Work products
  - Any non-compliances
    - To senior management



# Quality management in SQA

## Project level

- Apply specific process
- Check the process is followed
- Set a quality plan
  - Goals, metrics, etc.

---

## Organization level

- Establish framework of organizational process
- Establish standards

# Software standards



- Product standards
  - Applied to software **product**
  - Document standards (e.g., structure of requirements)
  - Documentation standards (e.g., comment header)
  - Coding standards (e.g., naming convention)
- Process standards
  - Applied to software **process**
  - Definition of specification
  - Design and validation process
  - Process support tools
  - Required documents



# Software standards

- Example standards

Product standards	Process standards
Design review form	Design review conduct
Requirements document structure	Submission of new code for system building
Method header format	Version release process
Java programming style	Project plan approval process
Project plan format	Change control process
Change request form	Test recording process

# Approaches to SQA



- Software reviews
  - A filter of software engineering process
- Purifying SE activities
  - Analysis, design, coding...
- Applied throughout process
- Uncovering errors/defects

# Approaches to SQA

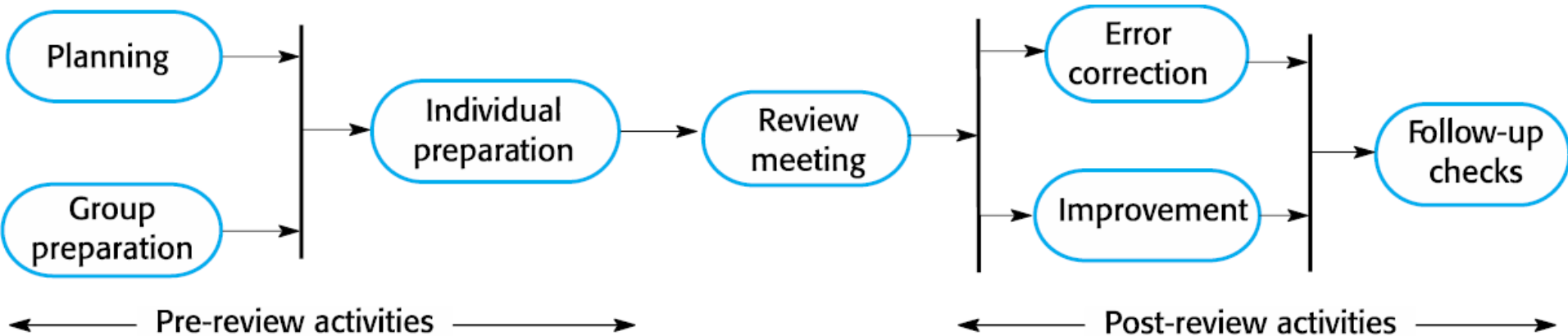
## Software reviews

- A diverse group of people
- Examines part/all of a system/process
- Point out improvements needed
- Confirm where improvement is not needed
- Approve/disapprove for moving to next stage



# Approaches to SQA

- Software reviews

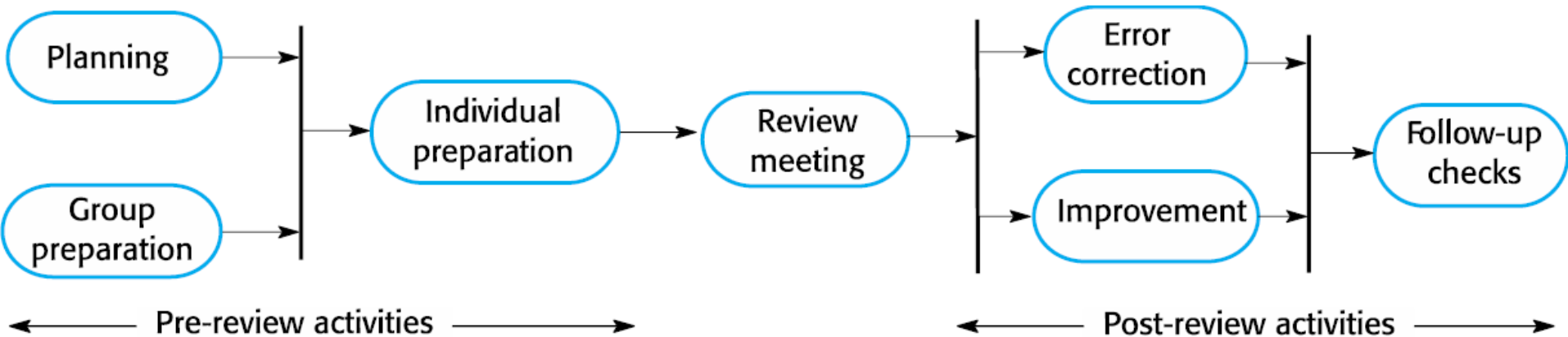


- **Pre-review activities**

- Planning
- Review preparation

# Approaches to SQA

- Software reviews

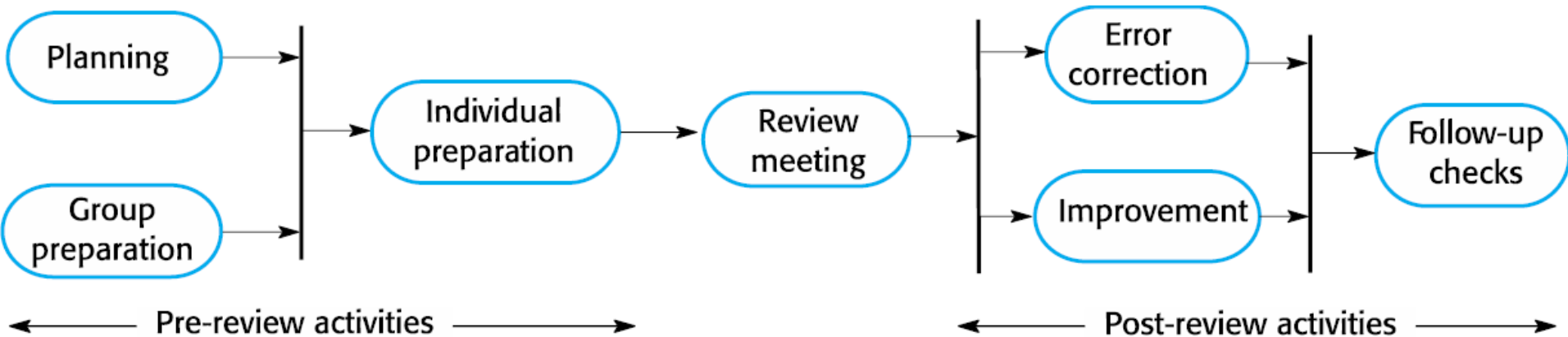


- The review meeting

- Author of documents/programs "Walk through" them with SQA team

# Approaches to SQA

- Software reviews



- **Post-review activities**

- Correcting errors found
- Improving the process to avoid making similar errors
- Following up to check if problems/issues found are fixed

# Approaches to SQA

## Software reviews

- Reviews for progress assessment
  - Applied on *product and process*
- **Inspections** for defect removal
  - Applied on *product*
- Quality reviews
  - Applied on *product and standards*



- ✓ Traditional face-to-face
- ✓ Distributed reviews

# Program inspections

- Examine system source (code) to discover defects
- No system execution needed
- Effective for discovering programming errors
- Not limited to programs
- Use a checklist (language dependent)
  - Initialization
  - Constant naming
  - Loop termination / array bounds





# Program inspections



## Checklist

Fault class	Inspection check
Data faults	<ul style="list-style-type: none"><li>• Are all program variables initialized before their values are used?</li><li>• Have all constants been named?</li><li>• Should the upper bound of arrays be equal to the size of the array or Size -1?</li><li>• If character strings are used, is a delimiter explicitly assigned?</li><li>• Is there any possibility of buffer overflow?</li></ul>
Control faults	<ul style="list-style-type: none"><li>• For each conditional statement, is the condition correct?</li><li>• Is each loop certain to terminate?</li><li>• Are compound statements correctly bracketed?</li><li>• In case statements, are all possible cases accounted for?</li><li>• If a break is required after each case in case statements, has it been included?</li></ul>
Input/output faults	<ul style="list-style-type: none"><li>• Are all input variables used?</li><li>• Are all output variables assigned a value before they are output?</li><li>• Can unexpected inputs cause corruption?</li></ul>

# Program inspections



Fault class		Inspection check
Interface faults		<ul style="list-style-type: none"><li>• Do all function and method calls have the correct number of parameters?</li><li>• Do formal and actual parameter types match?</li><li>• Are the parameters in the right order?</li><li>• If components access shared memory, do they have the same model of the shared memory structure?</li></ul>
Storage faults	management	<ul style="list-style-type: none"><li>• If a linked structure is modified, have all links been correctly reassigned?</li><li>• If dynamic storage is used, has space been allocated correctly?</li><li>• Is space explicitly deallocated after it is no longer required?</li></ul>
Exception faults	management	<ul style="list-style-type: none"><li>• Have all possible error conditions been taken into account?</li></ul>

# Contract review

- Contracts reflect the committal relationship between customer and supplier (software engineering team)
- Contract review: pre-project SQA
- Required by ISO 9001
- Check against
  - Unrealistic professional commitments
  - Schedule and budget failures



# Contract review

## Stage 1: proposal draft review

- ✓ **Customers requirements** clarified & documented
- ✓ **Alternative** approaches examined
- ✓ Customer-supplier **relationship** formally specified
- ✓ Development **risks** identified
- ✓ Project **resource** estimated
- ✓ **Timetable** prepared
- ✓ Firm's and customer's **capacity** examined
- ✓ Proprietary **rights** defined & protected



# Contract review

## Stage 2: contract draft review

- ✓ No unclarified issues remained
- ✓ Understandings (after proposal) documented
- ✓ No new changes
- ✓ No additions
- ✓ No omissions



# Contract review

## Influence factors

- Project magnitude
- Project complexity
- Staff expertise/experience

## Reviewers

- Members of proposal team
- Outside professionals
- Outside company/experts

## Difficulties

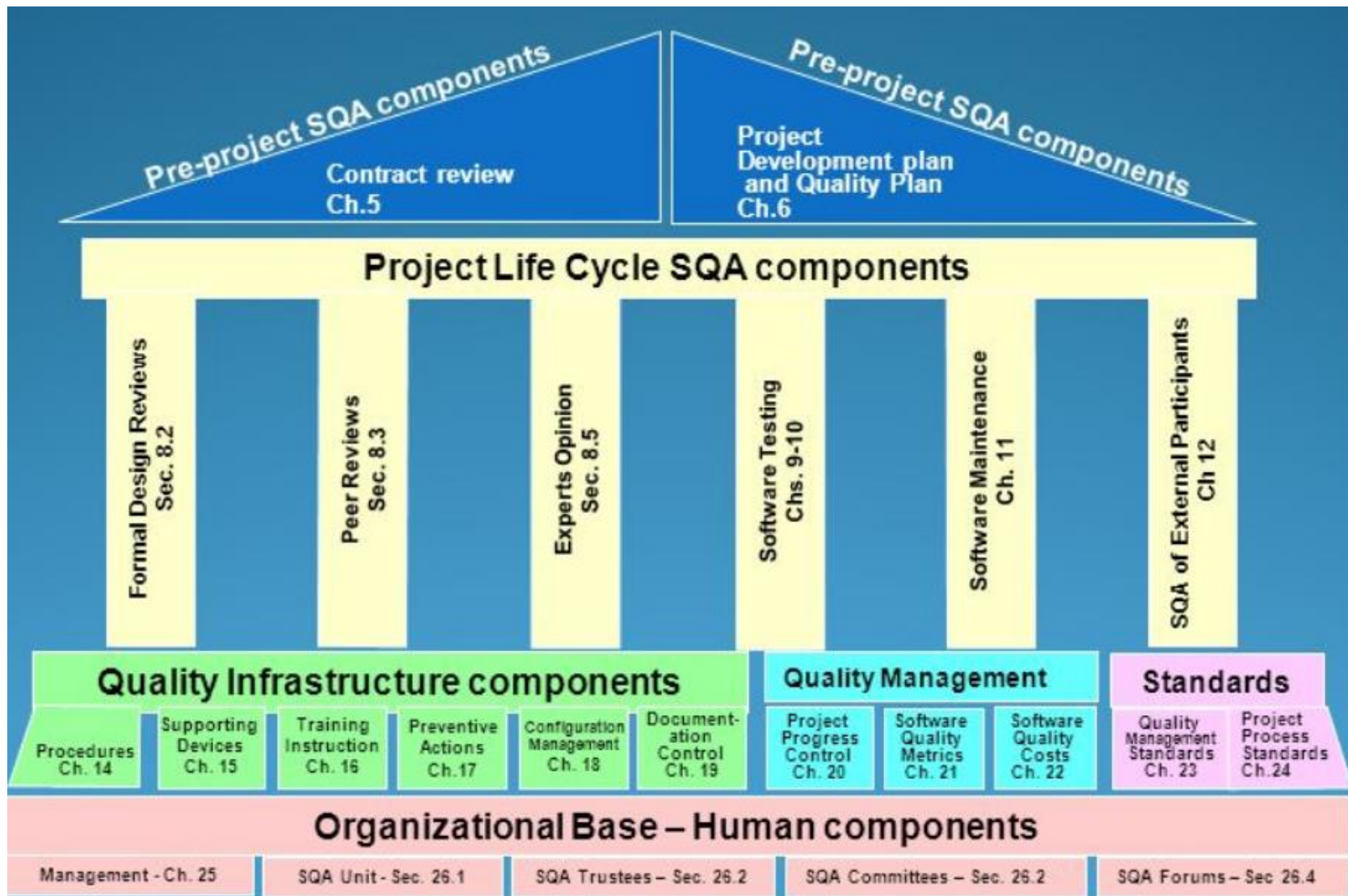
- Time pressure
- Workload
- Availability

## Good practices

- Pre-schedule the review
- Have a team to review
- Appoint a review leader



# SQA: the architecture



# Formal design review (FDR)

- Review software analysis & design
- In-project SQA component
- Detect errors in analysis and design
- Identify new risks
- Locate deviations (from template/convention)
- Approve analysis / design work products
- Facilitate exchange of knowledge about methods/tools/techniques
- Provide basis for future corrective actions





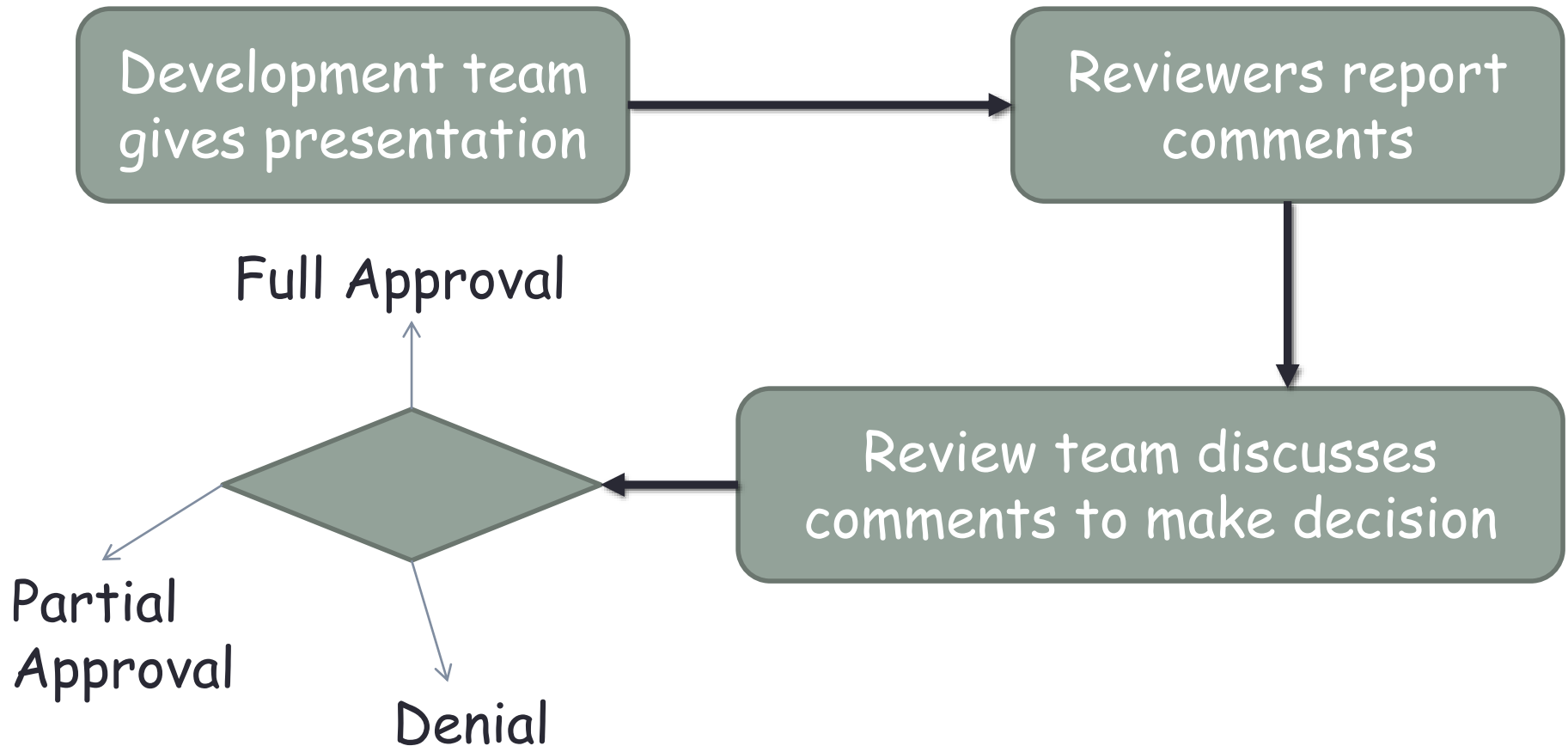
# Formal design review (FDR)

- 3~5 members
- Reviewer leader
  - Senior, external, not project leader
  - Schedule, coordinate, distribute design
- Members
  - Senior, from other departments
  - Review design resulting in comments, complete a checklist
- + Development team
  - Folks who produce the design
  - Present the design, address main issues



# Formal design review (FDR)

- Workflow of Review Session



# Formal design review (FDR)

## Post-review activities

- Review report
  - Summary of discussion
  - Decision
  - List of action items with completion dates
  - Name of reviewers responsible for following up
- Follow up
  - Verify action items carried out
  - Document the follow-up



- ☐ Too short - just decision
- ☐ missing action items
- ☐ No information regarding followup

# Formal design review (FDR)

## Design Review Checklist

**Checklist Description:** This checklist captures common elements that should be present in any design. It is presented during the Design Review process to stimulate thought, guide brainstorming, and to ensure the design being outlined contains all proper design considerations. As the project architecture, system, and application design is being reviewed, assess the design considerations that apply to your subject matter expertise and business/technical needs.

**Project Name:**

**Review Date:**

**Assessment and Recommendations:**

- ☐ Approved without revision
- ☐ Approved with revisions (see Notes)
- ☐ Not approved

**Notes:**

**Reviewer:**

**Signature:**

**Artifacts Reviewed:**

- ☐ Technical Design Specification
- ☐ Implementation Plan

- ☐ Conceptual Architecture Review Checklist
- ☐ Requirements Traceability Matrix

# Formal design review (FDR)

General Design		Comments
<input type="checkbox"/>	Does the design support both product and project goals?	
<input type="checkbox"/>	Is the design feasible from a technology, cost, and schedule standpoint?	
<input type="checkbox"/>	Have known design risks been identified, analyzed, and planned for or mitigated?	
<input type="checkbox"/>	Are the methodologies, notations, etc. used to create and capture the design appropriate?	
<input type="checkbox"/>	If possible, were proven past designs reused?	
<input type="checkbox"/>	Does the design support proceeding to the next development step?	
<input type="checkbox"/>	Have proper fallback consideration been made?	
Design Considerations		Comments
<input type="checkbox"/>	Does the design have conceptual integrity (i.e., does the whole design tie together)?	
<input type="checkbox"/>	Can the design be implemented within technology and environmental constraints?	
<input type="checkbox"/>	Does the design use standard techniques and avoid exotic, hard-to-understand elements?	
<input type="checkbox"/>	Is the design unjustifiably complex?	
<input type="checkbox"/>	Is the design lean (i.e., are all of its parts strictly necessary)?	
<input type="checkbox"/>	Does the design create reusable components if appropriate?	
<input type="checkbox"/>	Will the design be easy to port to another environment if appropriate?	


# Peer Reviews

## Versus FDR

- Participants
  - Project leader's equals and department members
  - FDR: superior to project leaders
- Authority/objective
  - Identifying errors/deviations from standards
  - FDR: approve design, determine moving to next phase or not

## Implementation

- Inspection
- Walkthrough



Peer review is  
efficient and effective  
in practice!

# Peer Reviews

## Inspection

- Formal peer review
- Target corrective actions
- Contribute more to SQA
- Review leader
- Author
- Specialized professionals
  - Designer, coder, tester

## Walkthrough

- Informal peer review
- Comment on the document
- Review leader
- Author
- Specialized professionals
  - Standard enforcer, maintenance expert, user representative

# Peer Reviews

- The review procedure

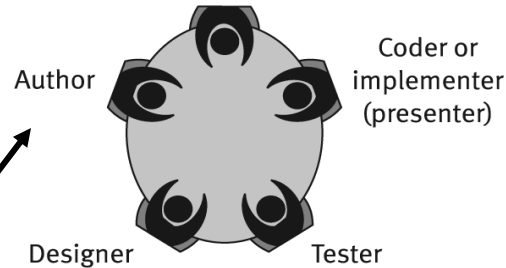
author is not the presenter

Much more formal

## PARTICIPANTS

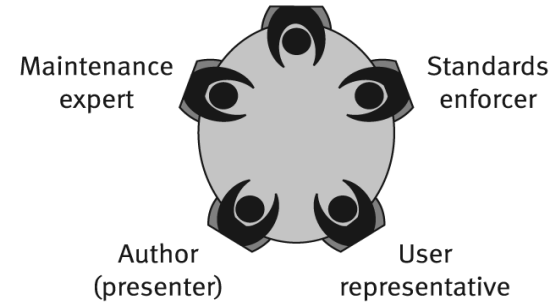
### Inspection

Moderator (scribe)

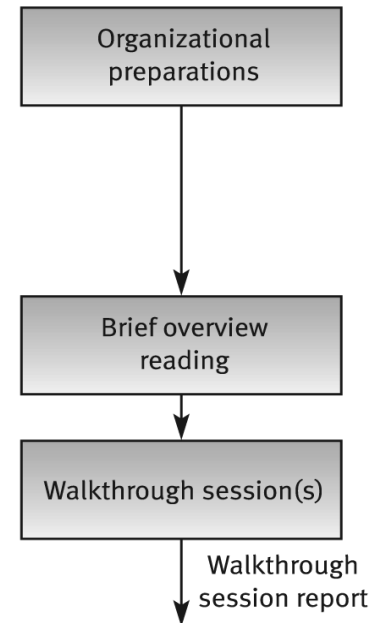
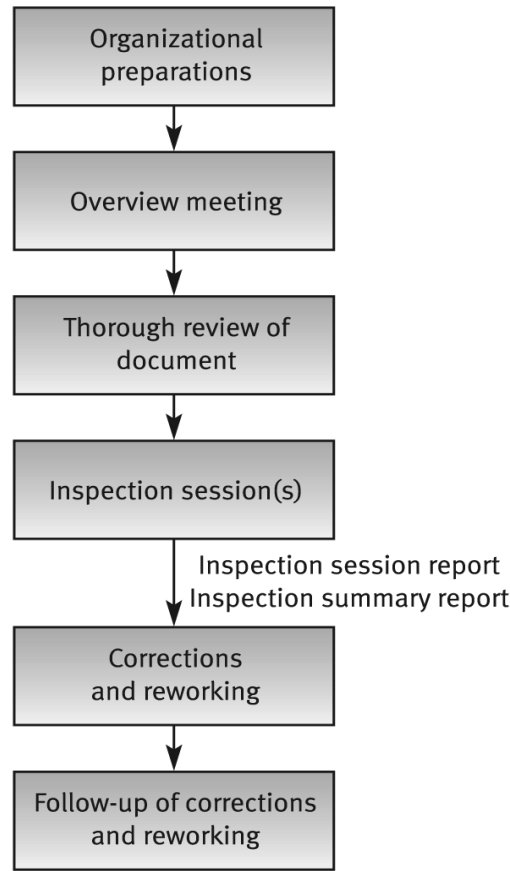


### Walkthrough

Coordinator (scribe)



## PROCESS





# Efficiency of Peer Reviews

- Detection efficiency
  - Average number of hours spent on each defect detected
- Defect detection density
  - Average number of defects detected from each page of the document being reviewed
- Effectiveness
  - $$\frac{\text{\#defects detected by peer review}}{\text{\# total defects detected by developers by all means}}$$

# Efficiency of Peer Reviews

## Example case in industry

- At Fujitsu, a study 1977-1982
- On Effectiveness of design review, code inspection, and testing
- Defects are measured in number per 1,000 lines of code in the scope of maintenance
- In the first six months of system use
- Major Finding
  - substantial improvement in software quality associated with an **increased share of code inspection and design reviews** and a reduced share of software testing

# Efficiency of Peer Reviews

Year	Defect detection method			Defects per 1000 lines of maintained code
	Test %	Design review %	Inspection %	
1977	85	---	15	0.19
1978	80	5	15	0.13
1979	70	10	20	0.06
1980	60	15	25	0.05
1981	40	30	30	0.04
1982	30	40	30	0.02

# A comparative look

Properties	Design review	Inspection	Walkthrough
Overview meeting	No	Yes	No
Participant's preparations	Yes - thorough	Yes - thorough	Yes - brief
Review session	Yes	Yes	Yes
Follow-up of corrections	Yes	Yes	No
Formal training of participants	No	Yes	No
Participant's use of checklists	No	Yes	No
Error-related data collection	Not formally required	Formally required	Not formally required
Review documentation	Formal design review report	1) Inspection session findings report 2) Inspection session summary report	

# Expert review

- By outside experts
- Reinforcing internal SQA
  - Giving expert judgement about document / code
  - Joining the internal review team
- Application context
  - ✓ Insufficient in-house professional capabilities in a specialized area.
  - ✓ Temporary lack of in-house professionals for review team.
  - ✓ Indecisiveness caused by major disagreements among the organization's senior professionals.
  - ✓ In small organizations, where the number of suitable candidates for a review team is insufficient.

# Code reviews

## Motivations

- How to ensure...
  - Maintainable code?
  - **DRY** code?
  - Readable code?
  - Bug-free code?
- Average defect detection rate for various testing
  - Unit testing: 25%
  - Function testing: 35%
  - Integration testing: 45%

How can this be improved?



# Code reviews

- **Code Review**
- A constructive review of a fellow developer's **code**.
- A required **sign-off** from another team member before a developer is permitted to check in changes or new code.
- **Analogy: writing articles for a newspaper**  
What is the effectiveness of...
  - Spell-check/grammar check
  - Author editing own article
  - Others editing others' articles

# Mechanics of code reviews

- **Who** : Original developer and reviewer, sometimes together in person, sometimes offline.
- **What** : Reviewer gives suggestions for improvement on a logical and/or structural level, to conform to previously agreed upon set of quality standards.
  - Feedback leads to refactoring, followed by a 2nd code review.
  - Eventually reviewer approves code.
- **When** : When code author has finished a coherent system change that is otherwise ready for checkin
  - change shouldn't be too large or too small
  - *before* committing the code to the repository or incorporating it into the new build



# Code review variations

- **Inspection:** A more formalized code review with:
  - roles (moderator, author, reviewer, scribe, etc.)
  - several reviewers looking at the same piece of code
  - a specific checklist of kinds of flaws to look for
    - possibly focusing on flaws that have been seen previously
    - possibly focusing on high-risk areas such as security
  - specific expected outcomes (e.g. report, list of defects)
- **Walkthrough:** informal discussion of code between author and a single reviewer
- **code reading:** Reviewers look at code by themselves (possibly with no actual meeting)

# Benefits of code reviews

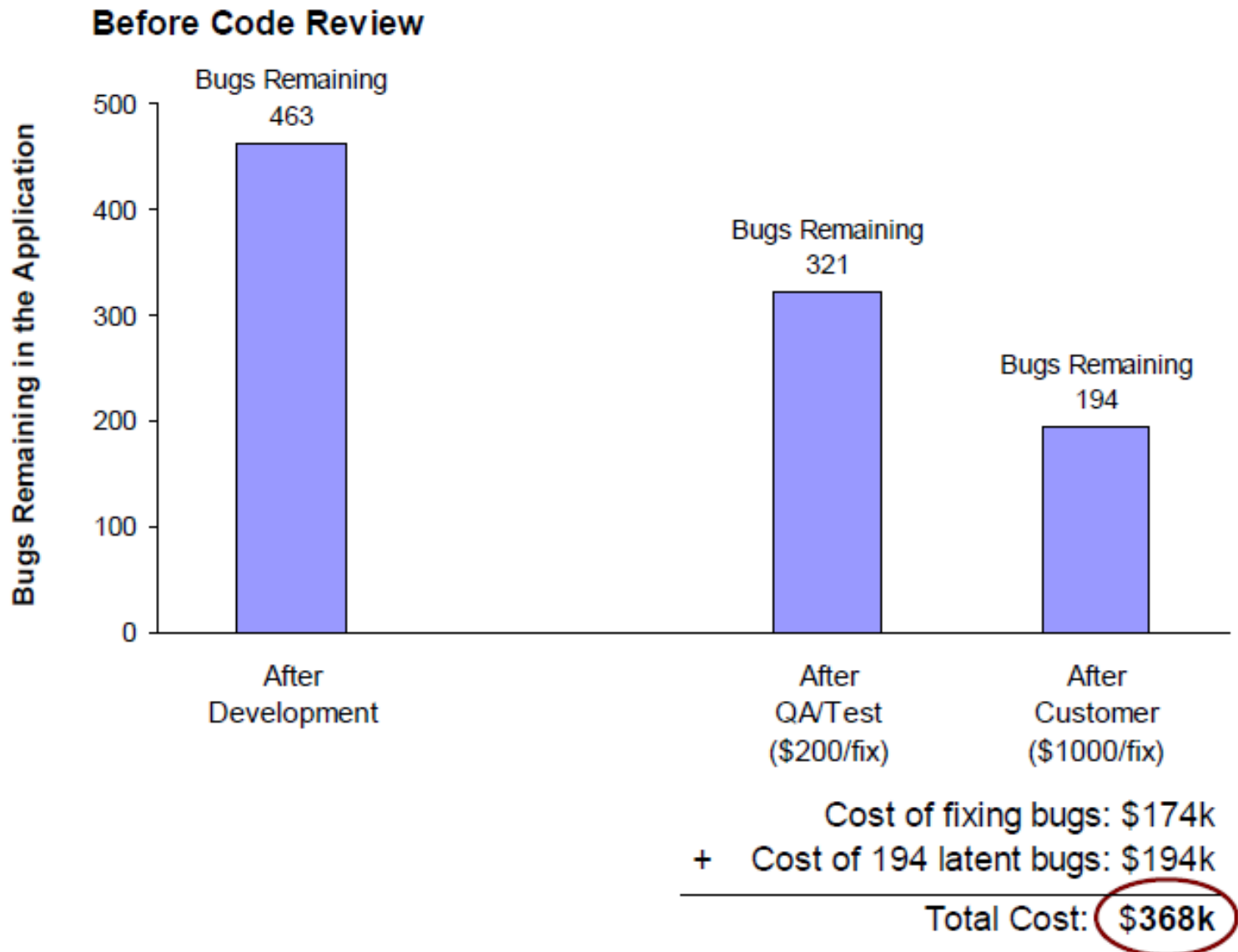
- > 1 person has seen every piece of code
  - Prospect of someone reviewing your code **raises quality threshold**.
- Forces code **authors to articulate** their decisions
- Hands-on **learning experience for rookies** without hurting code quality
  - Pairing them up with experienced developers
- Team **members involved in different parts** of the system
  - Reduces redundancy, enhances overall understanding
- **Author and reviewer both accountable** for committing code

# Benefits of code reviews

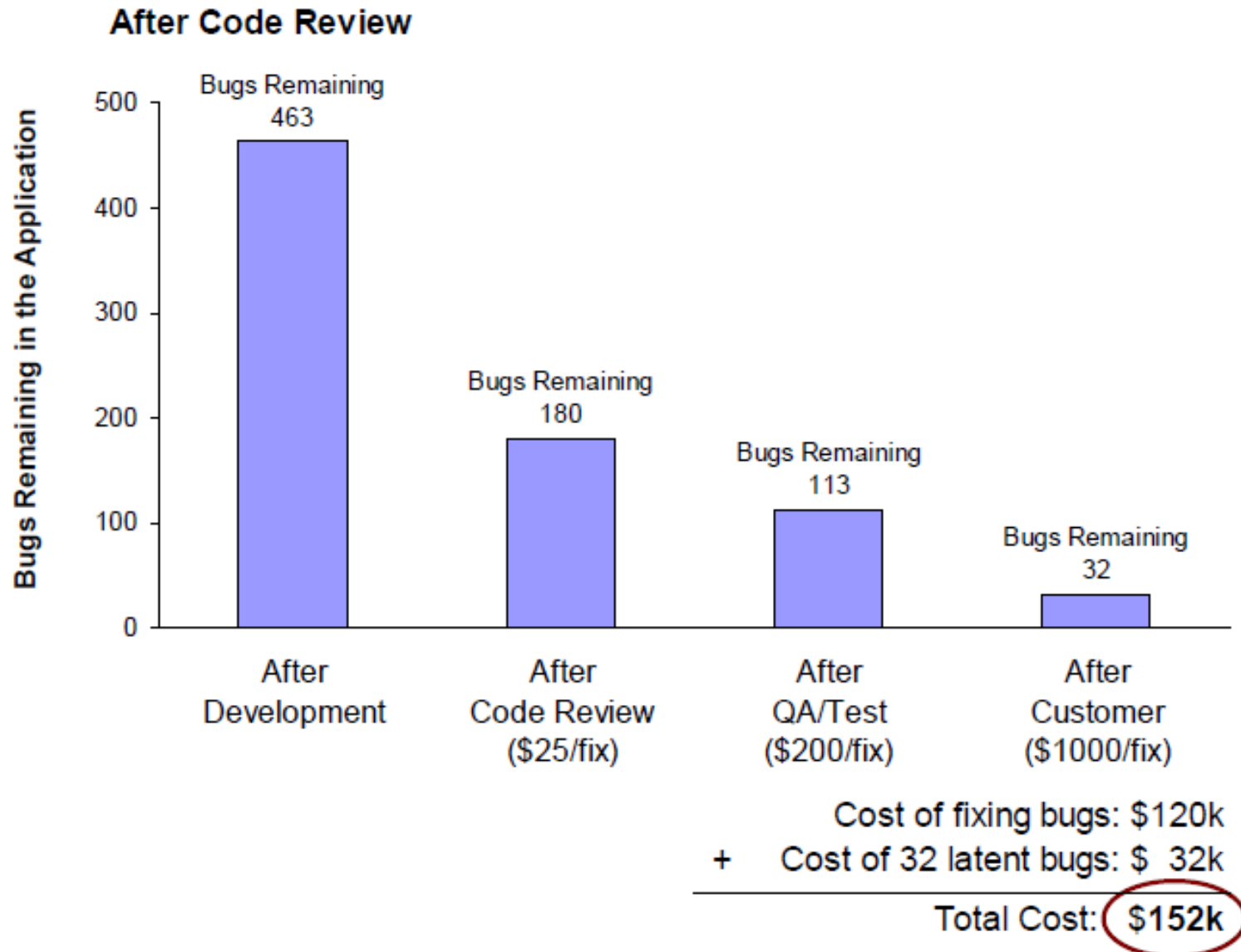
- Average defect detection rate
  - Unit testing: 25%
  - Function testing: 35%
  - Integration testing: 45%
  - Design review: 55%
  - Code inspections (formal code reviews): **60%**
- 11 programs developed by the same group of people
  - First 5 without reviews: average **4.5 errors per 100 lines** of code
  - Remaining 6 with reviews: average **0.82 errors per 100 lines** of code
    - Errors reduced by > **80 percent**

(Data from Steve McConnell's [Code Complete](#))

# Benefits of code reviews

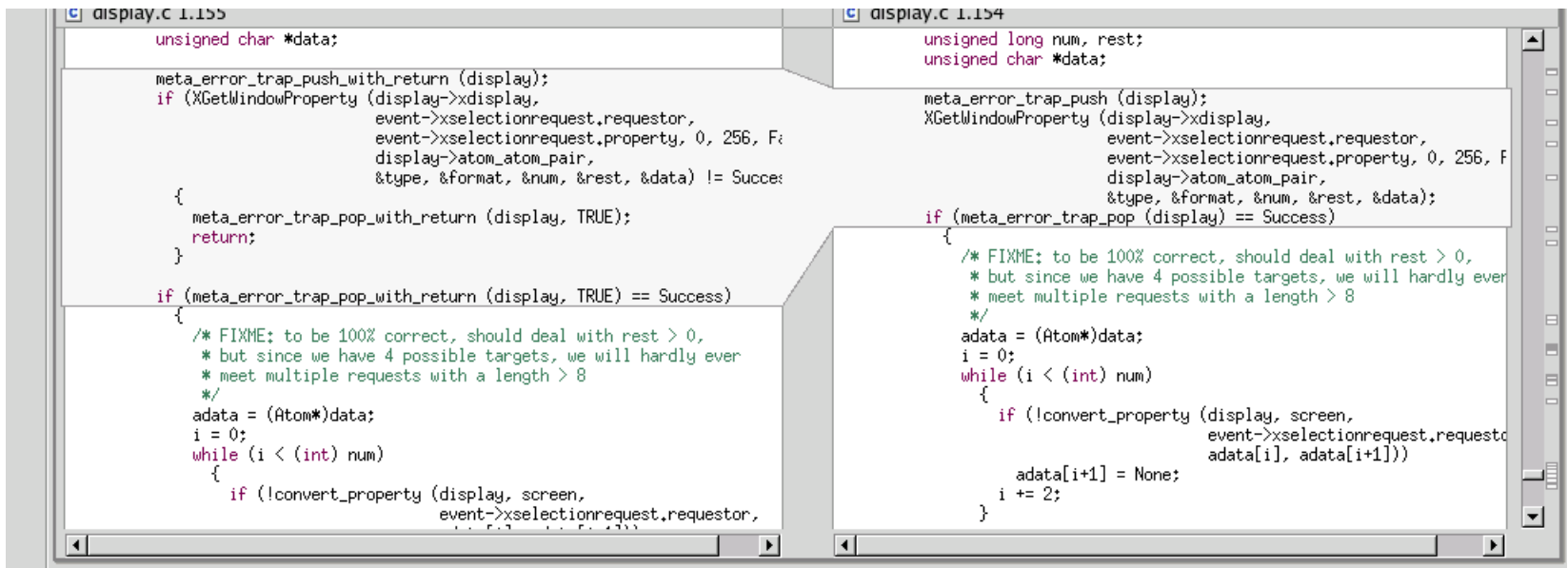


# Benefits of code reviews



# Code reviews in industry

- Code reviews are a **very** common industry practice.
- Made easier by advanced tools that:
  - integrate with configuration management systems
  - highlight changes (i.e., diff function)
  - allow traversing back into history
  - E.g.: Eclipse, SVN tools



```
display.c 1.155
unsigned char *data;

meta_error_trap_push_with_return (display);
if (XGetWindowProperty (display->xdisplay,
    event->xselectionrequest.requestor,
    event->xselectionrequest.property, 0, 256, False,
    display->atom_atom_pair,
    &type, &format, &num, &rest, &data) != Success)
{
    meta_error_trap_pop_with_return (display, TRUE);
    return;
}

if (meta_error_trap_pop_with_return (display, TRUE) == Success)
{
    /* FIXME: to be 100% correct, should deal with rest > 0,
     * but since we have 4 possible targets, we will hardly ever
     * meet multiple requests with a length > 8
     */
    adata = (Atom*)data;
    i = 0;
    while (i < (int) num)
    {
        if (!convert_property (display, screen,
            event->xselectionrequest.requestor,
            event->xselectionrequest.property, &adata[i], &adata[i+1]))
        {
            adata[i+1] = None;
            i += 2;
        }
    }
}
```

```
display.c 1.154
unsigned long num, rest;
unsigned char *data;

meta_error_trap_push (display);
XGetWindowProperty (display->xdisplay,
    event->xselectionrequest.requestor,
    event->xselectionrequest.property, 0, 256, False,
    display->atom_atom_pair,
    &type, &format, &num, &rest, &data);
if (meta_error_trap_pop (display) == Success)
{
    /* FIXME: to be 100% correct, should deal with rest > 0,
     * but since we have 4 possible targets, we will hardly ever
     * meet multiple requests with a length > 8
     */
    adata = (Atom*)data;
    i = 0;
    while (i < (int) num)
    {
        if (!convert_property (display, screen,
            event->xselectionrequest.requestor,
            event->xselectionrequest.property, &adata[i], &adata[i+1]))
        {
            adata[i+1] = None;
            i += 2;
        }
    }
}
```

# Code Reviews at Google

- "All code that gets submitted **needs to be reviewed** by at least one other person, and either the code writer or the reviewer needs to have readability in that language. Most people use [Mondrian](#) to do code reviews, and obviously, we spend a good chunk of our time reviewing code."

-- Amanda Camp, Software Engineer, Google



# Code reviews at Facebook

- "At Facebook, we have an internally-developed web-based tool to aid the code review process. Once an engineer has prepared a change, she submits it to this tool, which will notify the person or people she has asked to **review the change**, along with others that may be interested in the change -- such as people who have worked on a function that got changed.

At this point, **the reviewers can make comments, ask questions, request changes, or accept the changes**. If changes are requested, the submitter must submit a new version of the change to be reviewed. All versions submitted are retained, so reviewers can compare the change to the original, or just changes from the last version they reviewed. **Once a change has been submitted, the engineer can merge her change into the main source tree for deployment** to the site during the next weekly push, or earlier if the change warrants quicker release."



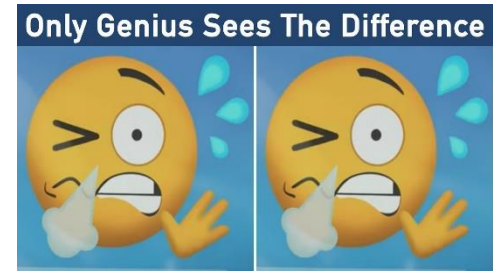
# Software Audits

- A major SQA **technique**
- A special kind of **review**
- Perform **product evaluation and process monitoring**, routinely throughout the software development process
- Look at a **process and/or product** in depth and compare to established standards and procedures
- **Purposes:** to ensure
  - Proper control procedures followed
  - Required documentation maintained
  - Developer's status reports consistent



# Software Audits

- Versus other kinds of review
  - Conducted by **external, independent** personnel
  - Concerned with product/process **compliance** w.r.t standards
  - Not concerned with technical content / quality



- Who
  - **Initiator** - decide the needs, organize, schedule
  - **Lead auditor** - plan, manage
  - **Recorder** - document findings, decisions, recommendations
  - **Auditors** - examine product/process
  - Audited organization **liaison** - provide information needed



# Approaches to quality control

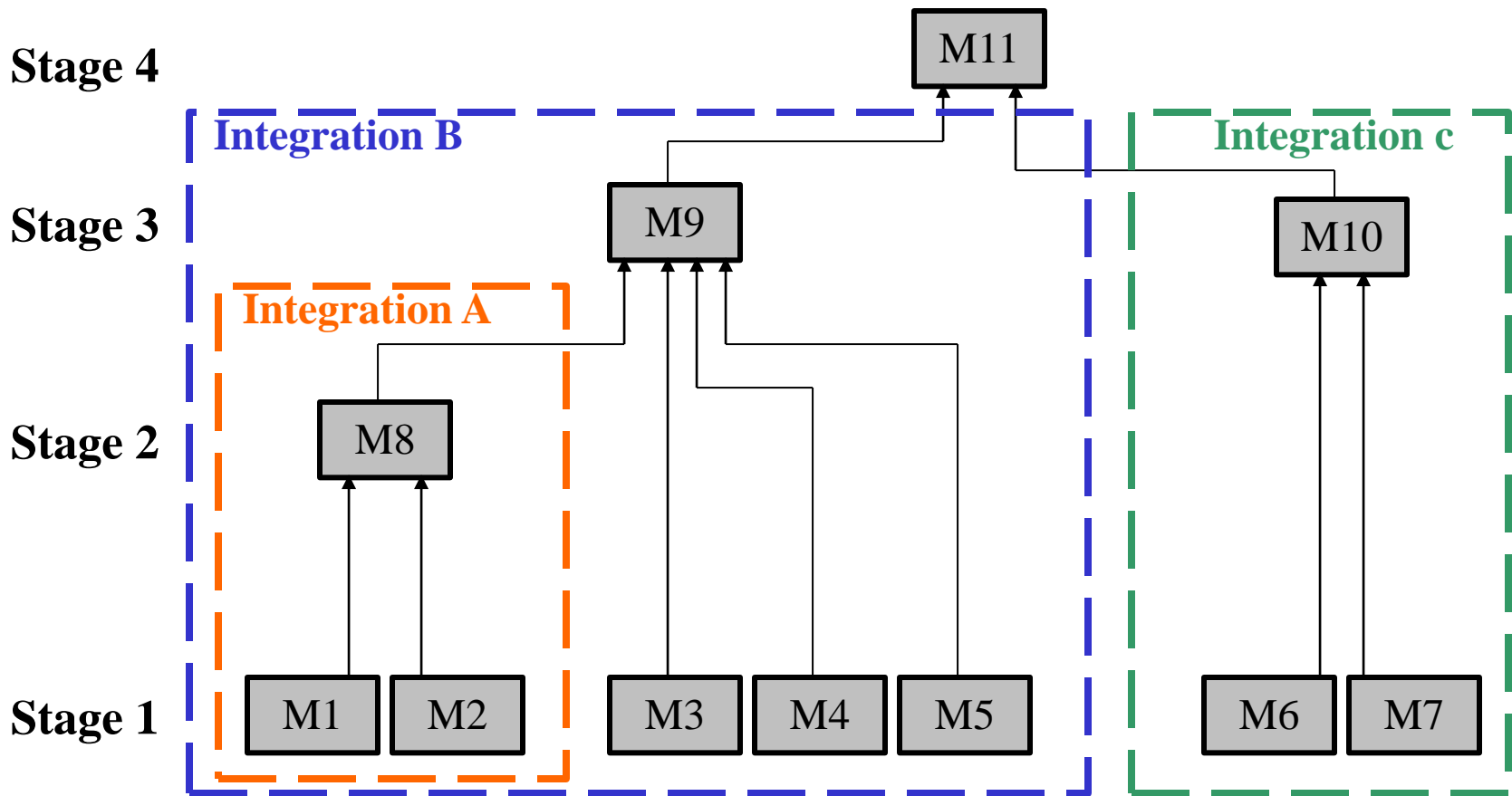
- Quality reviews
  - Reviews of process and products
- Quality audits
  - Independent/external reviews of process
- Testing
  - Verification/validation (V&V) of resulting software
    - Verification versus validation



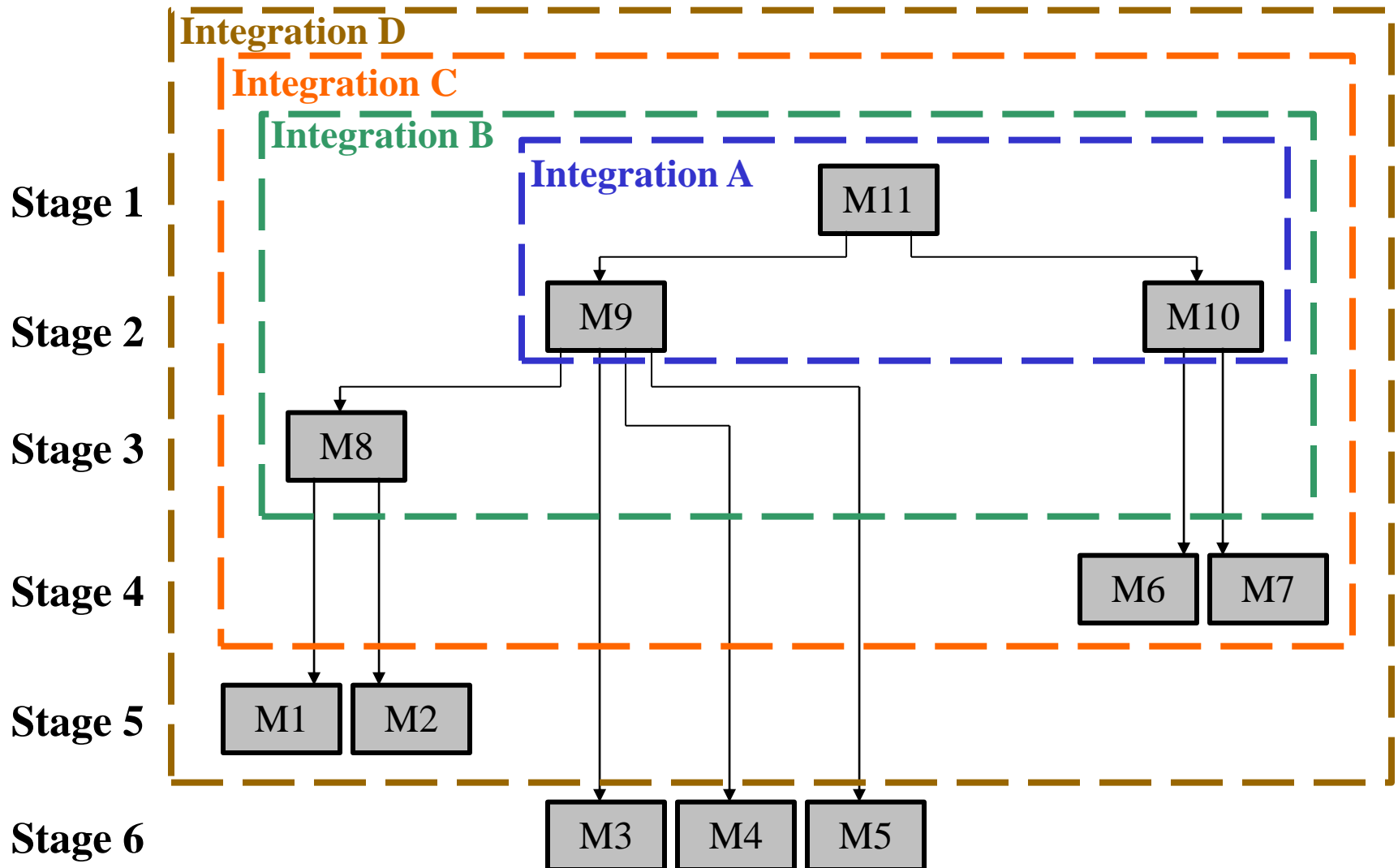
# Testing strategies

- *Incremental testing strategies:*
  - Test incrementally: Unit testing; Integration testing; System testing
    - Bottom-up testing
    - Top-down testing
- *Big bang testing*
  - Test entire software at one time.

# Testing strategies – bottom up

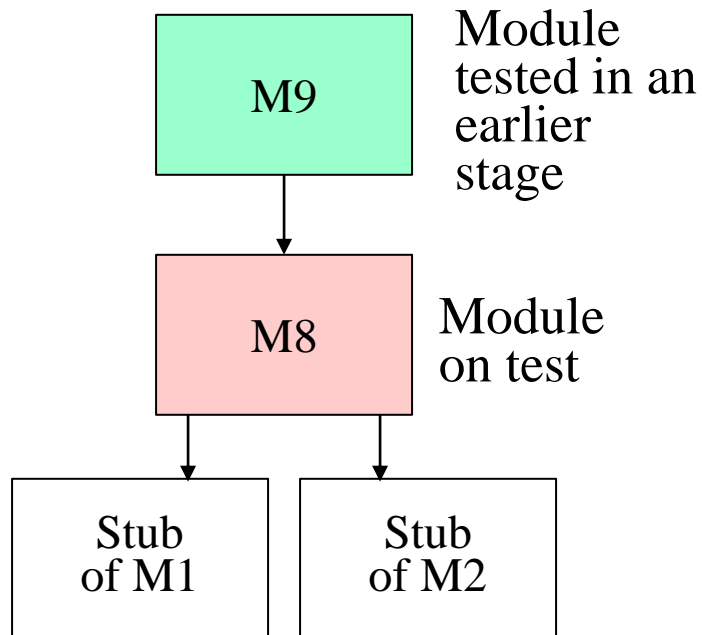


# Testing strategies – top down

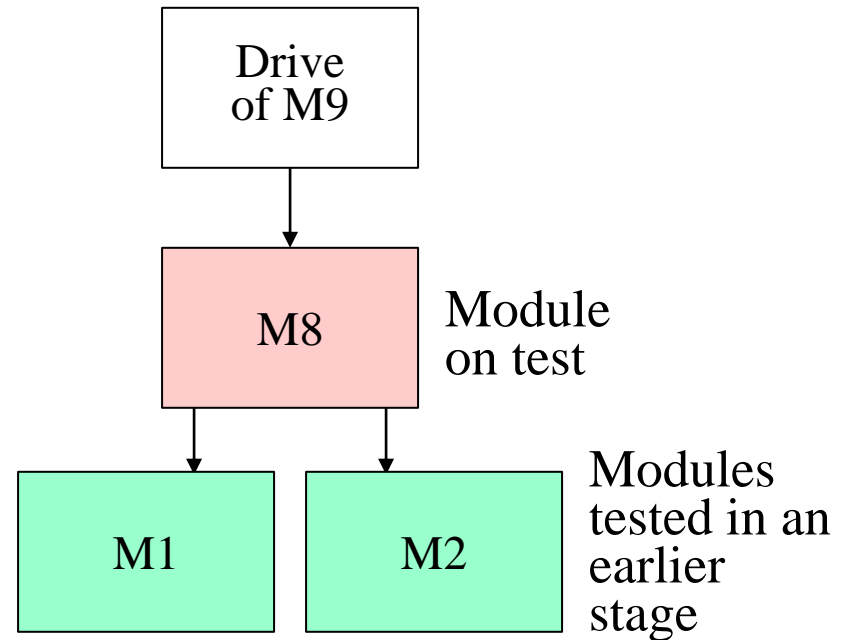


# Stubs & Drivers

## Top-down testing of module M8



## Bottom-up testing of module M8



# Comparison: Bottom-Up versus Top Down

- Bottom Up -

- Main advantage is the relative ease of its performance.
- Main disadvantage is the lateness at which the program as a whole can be observed.
- Maybe difficult to write drivers

- Top Down -

- Main advantage: show entire program functions shortly.
- Main disadvantage: require complicated programming.
- Maybe difficult to write stubs



# Big bang approach (vs incremental)

## Big Bang

- Not recommended in general; for small/simple software.
  - Difficult to identify errors and where they are located.
  - Simply way too much code / functionality to evaluate at one time.

## Incremental testing

- Test on usually small hunks of code, like unit or integration tests.
- Easier to identify errors than with whole project
- Correction is much simpler and requires far fewer resources too.
- Find errors much earlier in process.
- Prevents migration of errors into later, more complex stages.
- But you do have overhead of developing drivers and stubs and drivers for integration testing.
- Also, you carry out many testing operations on the same program vice only a single testing operation.

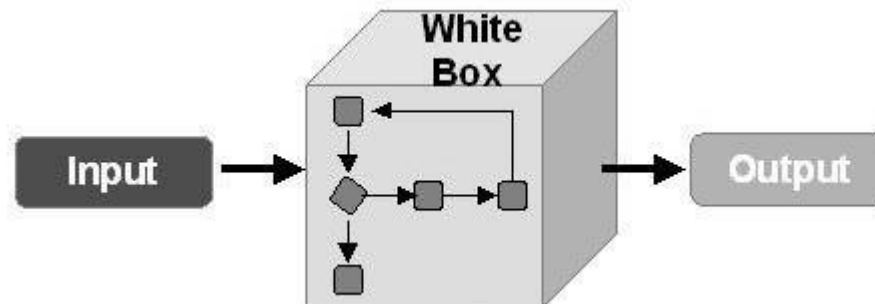
# Black Box versus White Box testing

- Black-Box (BB) testing
  - Identifies bugs only according to software malfunctioning as they are revealed in its erroneous **outputs**.
    - If outputs okay, BB Testing disregards internal paths taken, calculations and any processing.
  - Focuses on the compliance of a system/component with specified **functional requirements**.



# Black Box versus White Box testing

- White-Box (WB) testing (or glass-box testing)
  - Identifies bugs by examining the **internal structure** of programs and execution paths.
  - Consider the **internal mechanism** of the system
  - Only producing right outputs is not enough
  - Code structure and intermediate steps must be right as well



# Black Box versus White Box testing


## Black-box

- Approaches
  - Input equivalence class, Boundary testing
- Application scenarios
  - **Validation**
  - Output correctness test
  - Availability test
  - Reliability test
  - Security test
  - Flexibility test
  - Testability test
  - Portability test
  - Interoperability test

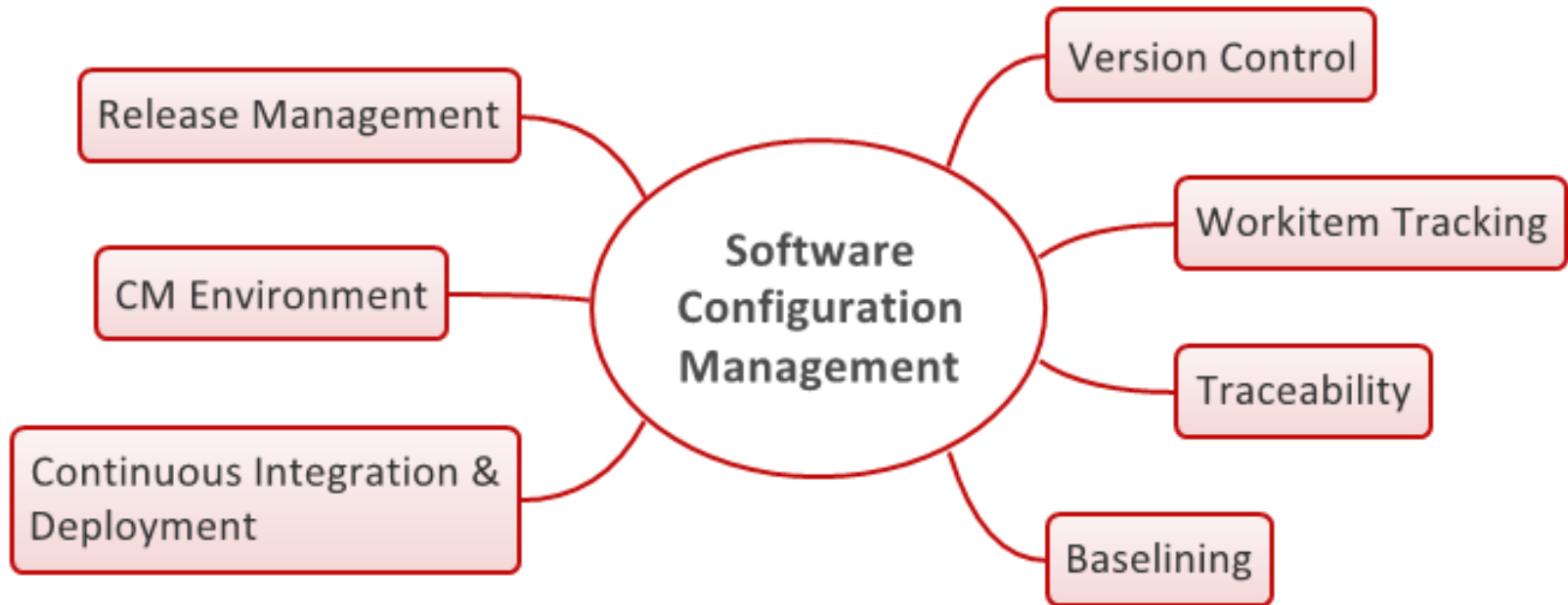
## White-box

- Approaches
  - Path testing, branch testing, static/dynamic code analysis
- Application scenarios
  - **Verification**
  - Cyclomatic complexity
  - Reusability test
  - Maintainability test
  - data correctness test
  - Processing correctness test
  - Software qualification

# Software configuration management

- Active software is constantly in the state of flux
    - **Changes** are the name of the game: constant, and many
    - All changes need to be quality-assured
  - Users might not be using the same version
- 
- The graphic shows a laptop with a blue screen. On the screen, the text 'Software Quality Assurance' is displayed in white. Below this, in smaller white text, are the terms 'source code control', 'code review', 'configuration management', and 'software testing'. To the right of the laptop, several blue 3D cubes are floating, some appearing to be part of a larger structure that is breaking apart or being built.
- Which version to work on?
  - How to get a specific version?
  - Which version of design document matches a specific version of software?
  - What are the changes introduced in a specific version?
  - Does a particular version include undocumented changes?
  - .....

# Software configuration management



Dealing with issues related to control of software changes, proper documentation of changes, registering and storing the approved software versions, tracking registered versions and more **throughout the software system's life cycle.**

# Summary

- **Scope of SQA:** in relation to quality control, in terms of objectives and timing
- **SQA architecture:** hierarchy, pre-project components, project lifecycle components, quality management, standards, and human components
  - SQA team: roles, and tasks
  - Quality management, software standards
- **SQA approaches:** different ways of software reviews, review process, program inspections, V&V, SCM