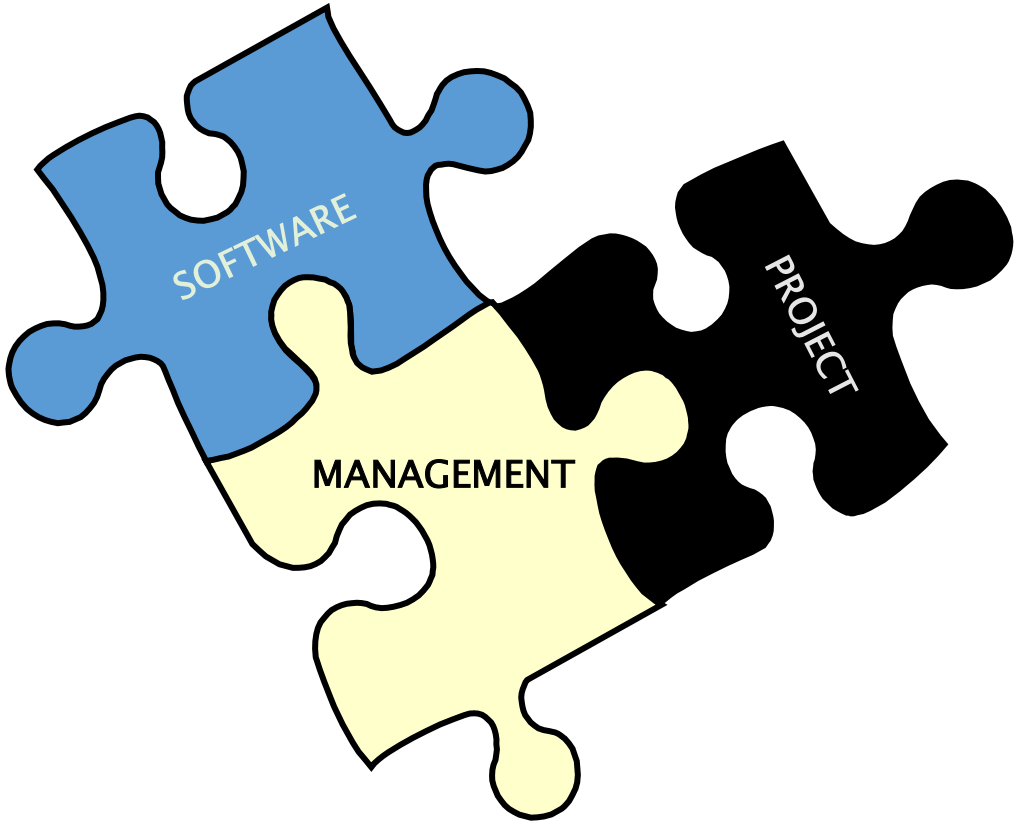
# Software Project Management



# Unit 3. Techniques of Planning, Controlling and Automating Software Process

Prepared By:

**BIJAY MISHRA**

(बिजय  मिश्र)

biizay@gmail.com

 @jijibisha

# UNIT 3. Techniques of Planning, Controlling and Automating Software Process - 15 Hrs.

3.1 Iterative Process Planning (Process Work Breakdown Structure, Planning Guidelines, Cost and Schedule Estimations Process, Iteration Planning Process)

3.2 Project Organization and Responsibilities

3.3 Process Automation – Tools and Environment

3.4 Project Control and Process Instrumentation

3.5 Process Customization

# 3.1 Iterative Process Planning (Process Work Breakdown Structure, Planning Guidelines, Cost and Schedule Estimations Process, Iteration Planning Process)

# Iterative Process Planning

## Work Breakdown Structures

- The development of a work breakdown structure is dependent on the project management style, organizational culture, customer preference, financial constraints and several other hard-to-define parameters.

- A WBS is simply a hierarchy of elements that decomposes the project plan into the discrete work tasks.

- A WBS provides the following information structure:

  - ✓ A delineation of all significant work
  - ✓ A clear task decomposition for assignment of responsibilities
  - ✓ A framework for scheduling, budgeting, and expenditure tracking.

# Iterative Process Planning

## Work Breakdown Structures

- The basic recommendation for the WBS is to organize the hierarchy as follows:
  - ✓**First-level WBS elements** are the workflows (management, environment, requirements, design, implementation, assessment, and deployment).
  - ✓**Second-level WBS elements** are defined for each phase of the life cycle (inception, elaboration, construction, and transition).
  - ✓**Third-level WBS elements** are defined for the focus of activities that produce the artifacts of each phase.

# Iterative Process Planning

## Planning Guidelines

- Software projects span a broad range of application domains.
- It is valuable but risky to make specific planning recommendations independent of project context.
- Project-independent planning advice is also risky.
- There is the risk that the guidelines maybe adopted blindly without being adapted to specific project circumstances.
- Two simple planning guidelines should be considered when a project plan is being initiated or assessed.

# Iterative Process Planning

## Planning Guidelines

The first guideline, detailed in Table 1, prescribes a default allocation of costs among the first-level WBS elements.

| First Level WBS Element | Default Budget |
|---|---|
| Management | 10% |
| Environment | 10% |
| Requirement | 10% |
| Design | 15% |
| Implementation | 25% |
| Assessment | 25% |
| Deployment | 5% |
| Total | 100% |

**Table 1: Web budgeting defaults**

# Iterative Process Planning

## Planning Guidelines

The second guideline, detailed in Table 2, prescribes the allocation of effort and schedule across the lifecycle phases.

| Domain | Inception | Elaboration | Construction | Transition |
|---|---|---|---|---|
| Effort | 5% | 20% | 65% | 10% |
| Schedule | 10% | 30% | 50% | 10% |

**Table 2: Default distributions of effort and schedule by phase**

# Iterative Process Planning

## The Cost and Schedule Estimating Process

- Project plans need to be derived from two perspectives:
  - ✓ The first is a forward-looking, top-down approach.
  - ✓ The second perspective is a backward-looking, bottom-up approach.

- These two planning approaches should be used together, in balance, throughout the life cycle of the project.

# Iterative Process Planning

## The Cost and Schedule Estimating Process

### Forward-looking:

1. The software project manager develops a characterization of the overall size, process, environment, people, and quality required for the project
2. A macro-level estimate of the total effort and schedule is developed using a software cost estimation model
3. The software project manager partitions the estimate for the effort into a top-level WBS, also partitions the schedule into major milestone dates and partitions the effort into a staffing profile
4. At this point, subproject managers are given the responsibility for decomposing each of the WBS elements into lower levels using their top-level allocation, staffing profile, and major milestone dates as constraints.

# Iterative Process Planning

## The Cost and Schedule Estimating Process

### Backward-looking:

1. The lowest level WBS elements are elaborated into detailed tasks, for which budgets and schedules are estimated by the responsible WBS element manager.
2. Estimates are combined and integrated into higher level budgets and milestones.
3. Comparisons are made with the top-down budgets and schedule milestones. Gross differences are assessed and adjustments are made in order to converge on agreement between the top-down and the bottom-up estimates.

# Iterative Process Planning

## The Cost and Schedule Estimating Process

- During the engineering stage, the top-down perspective will dominate because there is usually not enough depth of understanding nor stability in the detailed task sequences to perform credible bottom-up planning.
- During the production stage, there should be enough precedent experience and planning fidelity that the bottom-up planning perspective will dominate.
- Top-down approach should be well tuned to the project-specific parameters, so it should be used more as a global assessment technique.
- Figure 1 illustrates this life-cycle planning balance.

# Iterative Process Planning

## The Cost and Schedule Estimating Process

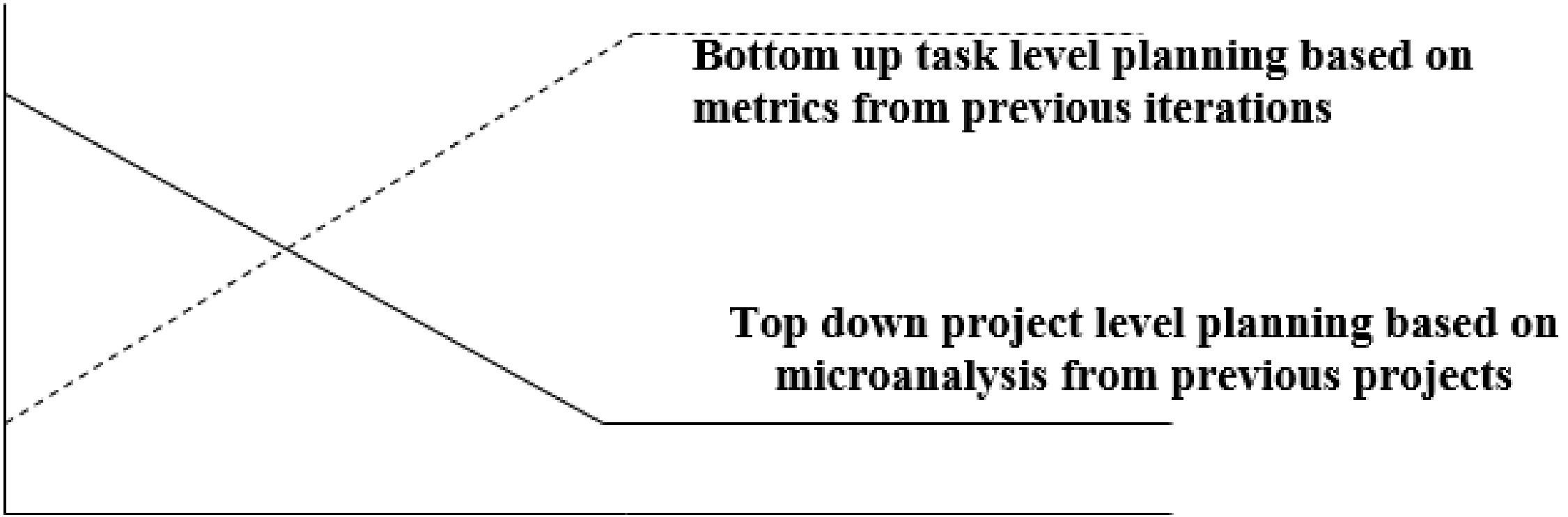
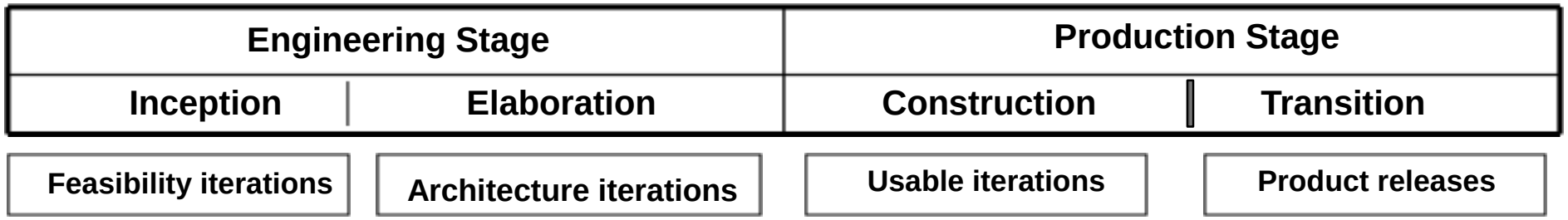



**Figure 1 Planning balance throughout the life cycle**

# Iterative Process Planning

## The Iteration Planning Process




### Engineering stage

Planning emphasis:

- Macro-level task estimation for production-stage artifacts
- Micro-level task estimation for engineering artifacts
- Stakeholder concurrence
- Coarse-grained variance analysis of actual vs. planned expenditures
- Tuning the top-down project-independent planning guidelines into project-specific planning guidelines.

### Production stage

Planning emphasis:

- Micro-level task estimation for production-stage artifacts
- Macro-level task estimation for engineering artifacts
- Stakeholder concurrence
- Fine-grained variance analysis of actual vs. planned expenditures

# Iterative Process Planning

## The Iteration Planning Process

Planning is concerned with defining the actual sequence of intermediate results. An evolutionary build plan is important because there are always adjustments in build content and schedule as early conjecture evolves into well-understood project circumstances. Iteration is used to mean a complete synchronization across the project, with a well-orchestrated global assessment of the entire project baseline.

**Inception iterations.** The early prototyping activities integrate the foundation components of a candidate architecture and provide an executable framework for elaborating the critical use cases of the system. This framework includes existing components, commercial components, and custom prototypes sufficient to demonstrate a candidate architecture and sufficient requirements understanding to establish a credible business case, vision, and software development plan.

# Iterative Process Planning

## The Iteration Planning Process

**Elaboration iterations.** These iterations result in architecture, including a complete framework and infrastructure for execution. Upon completion of the architecture iteration, a few critical use cases should be demonstrable: (1) initializing the architecture, (2) injecting a scenario to drive the worst-case data processing flow through the system (for example, the peak transaction throughput or peak load scenario), and (3) injecting a scenario to drive the worst-case control flow through the system (for example, orchestrating the fault-tolerance use cases).

**Construction iterations.** Most projects require at least two major construction iterations: an alpha release and a beta release.

**Transition iterations.** Most projects use a single iteration to transition a beta release into the final product.

# Iterative Process Planning

## The Iteration Planning Process

- The general guideline is that most projects will use between four and nine iterations.
- The typical project would have the following six-iteration profile:
  - ✓ One iteration in inception: an architecture prototype
  - ✓ Two iterations in elaboration: architecture prototype and architecture baseline
  - ✓ Two iterations in construction: alpha and beta releases
  - ✓ One iteration in transition: product release
- A very large or unprecedented project with many stakeholders may require additional inception iteration and two additional iterations in construction, for a total of nine iterations.

# 3.2 Project Organization and Responsibilities

# Project Organizations and Responsibilities

- Organizations engaged in software Line-of-Business need to support projects with the infrastructure necessary to use a common process.
- Project organizations need to allocate artifacts & responsibilities across project team to ensure a balance of global (architecture) & local (component) concerns.
- The organization must evolve with the WBS & Life cycle concerns.
- Software lines of business & product teams have different motivation.
- Software lines of business are motivated by return of investment (ROI), new business discriminators, market diversification & profitability.
- Project teams are motivated by the cost, Schedule & quality of specific deliverables

# Project Organizations and Responsibilities

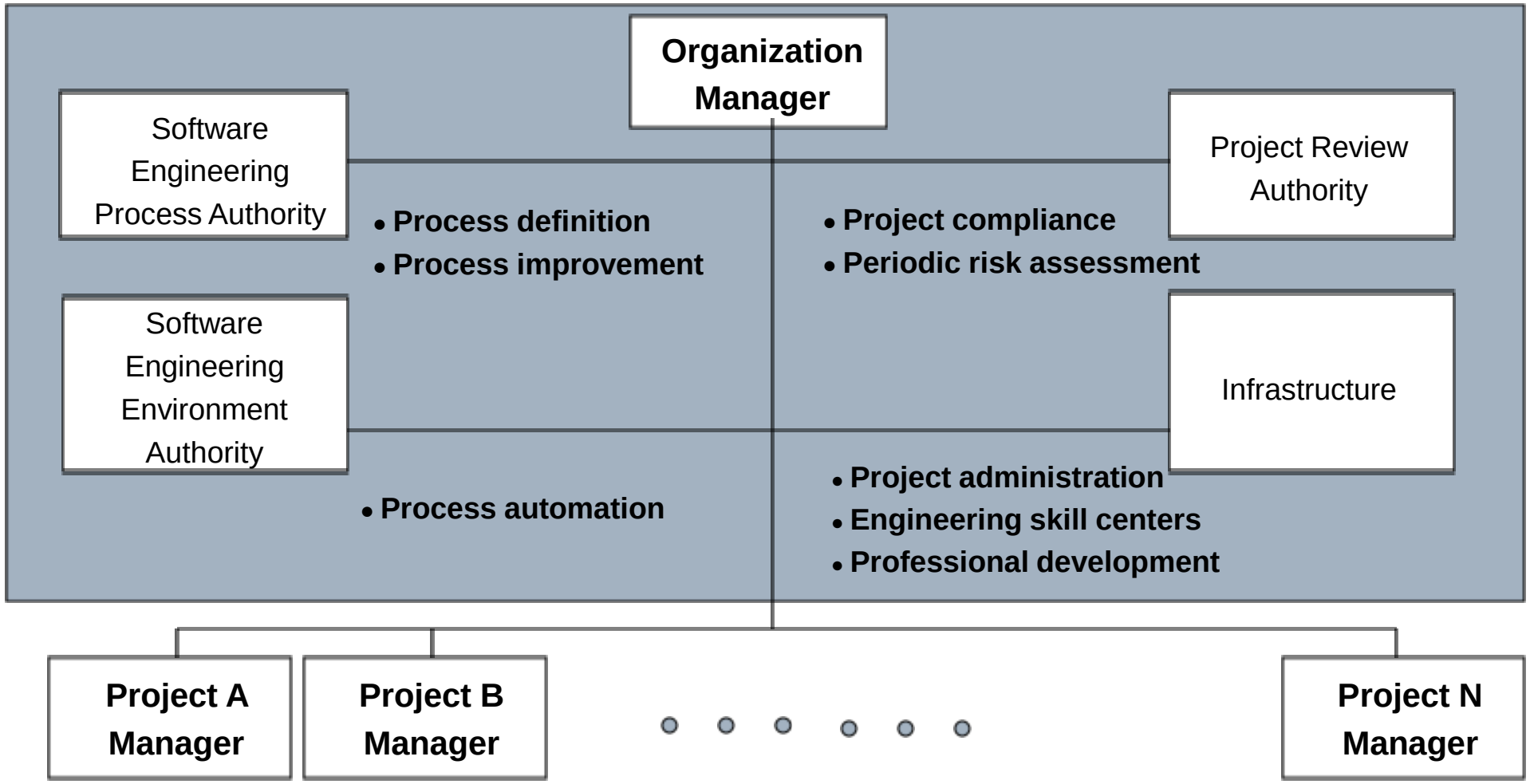
## Line-of-Business Organizations

- The main features of default organization are as follows:
  - ✓ Responsibility for process definition & maintenance is specific to a cohesive line of business.
  - ✓ Responsibility for process automation is an organizational role & is equal in importance to the process definition role.
  - ✓ Organizational role may be fulfilled by a single individual or several different teams.

# Project Organizations and Responsibilities

## Line-of-Business Organizations

**Default roles in a software line-of-business organizations**



# Project Organizations and Responsibilities

## Line-of-Business Organizations

### Software Engineering Process Authority (SEPA)

- ✓ The SEPA facilities the exchange of information & process guidance both to & from project practitioners
- ✓ This role is accountable to General Manager for maintaining a current assessment of the organization's process maturity & its plan for future improvement

### Project Review Authority (PRA)

- ✓ The PRA is the single individual responsible for ensuring that a software project complies with all organizational & business unit software policies, practices & standards
- ✓ A software Project Manager is responsible for meeting the requirements of a contract or some other project compliance standard

# Project Organizations and Responsibilities

## Line-of-Business Organizations

### Software Engineering Environment Authority (SEEA)

- ✓ The SEEA is responsible for automating the organization's process, maintaining the organization's standard environment, training projects to use the environment & maintaining organization-wide reusable assets.
- ✓ The SEEA role is necessary to achieve a significant ROI for common process.

### Infrastructure

- ✓ An organization's infrastructure provides human resources support, project-independent research & development, & other capital software engineering assets.

# Project Organizations and Responsibilities

## Project Organizations

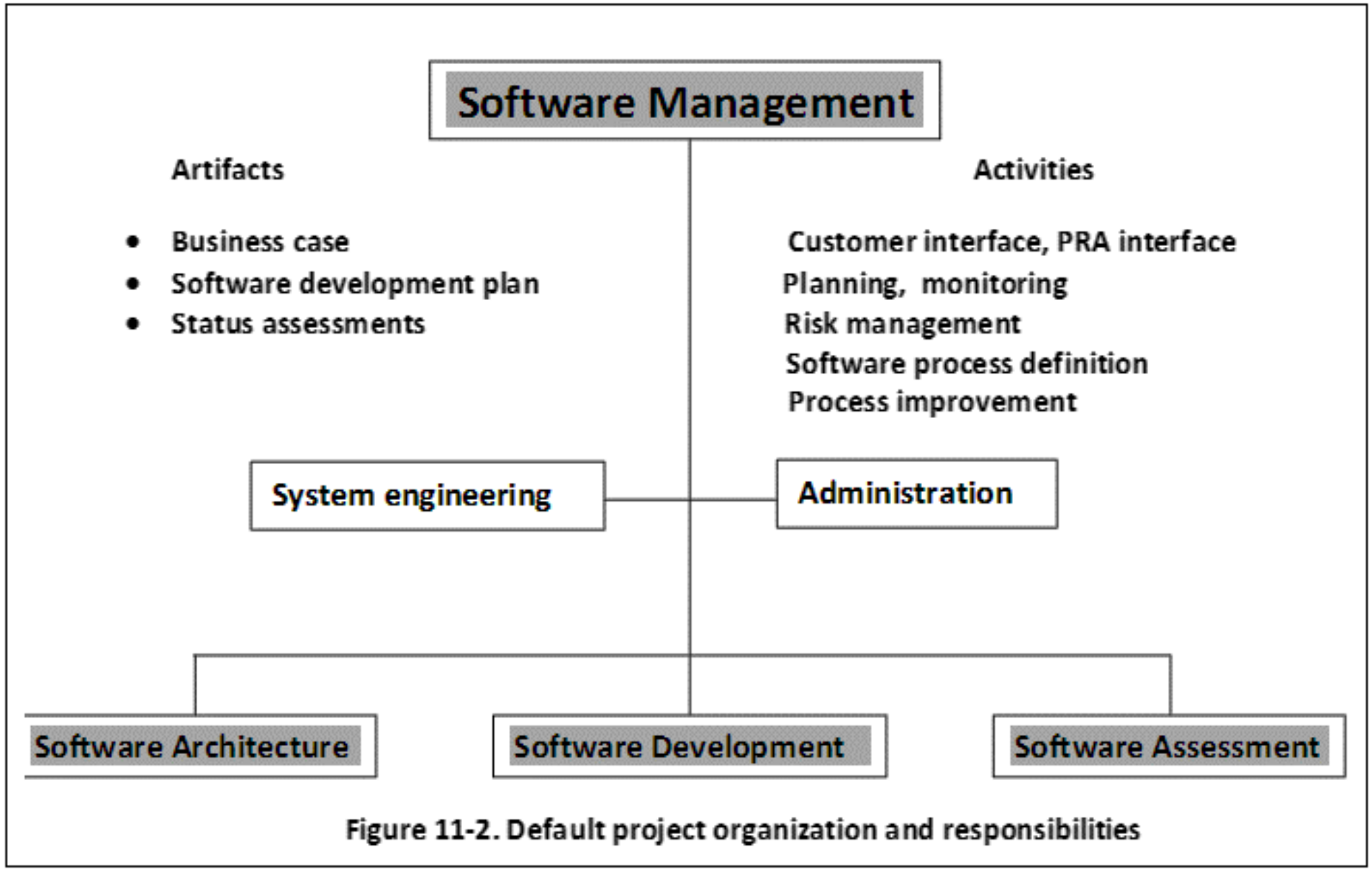



Figure 11-2. Default project organization and responsibilities

# Project Organizations and Responsibilities

## Project Organizations

❖ The main features of the default organization are as follows:

- ✓ The project management team is an active participant, responsible for producing as well as managing.
- ✓ The architecture team is responsible for real artifacts and for the integration of components, not just for staff functions.
- ✓ The development team owns the component construction and maintenance activities.
- ✓ The assessment team is separate from development.
- ✓ Quality is everyone's into all activities and checkpoints.
- ✓ Each team takes responsibility for a different quality perspective.

# Project Organizations and Responsibilities

## Project Organizations

### Software Management Team

**Artifacts**

- Business case
- Vision
- Software development plan
- Work breakdown structure
- Status assessments
- Requirements set

- Systems Engineering
- Financial Administration
- Quality Assurance

**Responsibilities**

- Resource commitments
- Personnel assignments
- Plans, priorities,
- Stakeholder satisfaction
- Scope definition
- Risk management
- Project control

### Life-Cycle Focus

| Inception | Elaboration | Construction | Transition |
|---|---|---|---|
| Elaboration phase planning<br>Team formulating<br>Contract base lining<br>Architecture costs | Construction phase planning<br>Full staff recruitment<br>Risk resolution<br>Product acceptance criteria<br>Construction costs | Transition phase planning<br>Construction plan optimization<br>Risk management | Customer satisfaction<br>Contract closure<br>Sales support<br>Next-generation planning |

# Project Organizations and Responsibilities

## Project Organizations

### Software Architecture Team

- Demonstrations
- Use-case modelers
- Design modelers
- Performance analysts

### Artifacts

- Architecture description
- Requirements set
- Design set
- Release specifications

### Responsibilities

- Requirements trade-offs
- Design trade-offs
- Component selection
- Initial integration
- Technical risk solution

### Life-Cycle Focus

| Inception | Elaboration | Construction | Transition |
|---|---|---|---|
| Architecture prototyping<br>Make/buy trade-offs<br>Primary scenario definition<br>Architecture evaluation criteria definition | Architecture base lining<br>Primary scenario demonstration<br>Make/buy trade-offs base lining | Architecture maintenance<br>Multiple-component issue resolution<br>Performance tuning<br>Quality improvements | Architecture maintenance<br>Multiple-component issue resolution<br>Performance tuning<br>Quality improvements |

# Project Organizations and Responsibilities

## Project Organizations

### Software Development Team

- Component teams

**Artifacts**

- Design set
- Implementation set
- Deployment set

Responsibilities

- Component design
- Component implementation
- Component stand-alone test
- Component maintenance
- Component documentation

### Life-Cycle Focus

| Inception | Elaboration | Construction | Transition |
|---|---|---|---|
| Prototyping support<br>Make/buy trade-offs | Critical component design<br>Critical component implementation and test<br>Critical component base line | Component design<br>Component implementation<br>Component stand-alone test<br>Component maintenance | Component maintenance<br>Component documentation |

# Project Organizations and Responsibilities

## Project Organizations

### Software Assessment Team

**Artifacts**

- Deployment set
- SCO database
- User manual
- Environment
- Release specifications
- Release descriptions
- Deployment documents

- Release testing
- Change management
- Deployment
- Environment support

**Responsibilities**

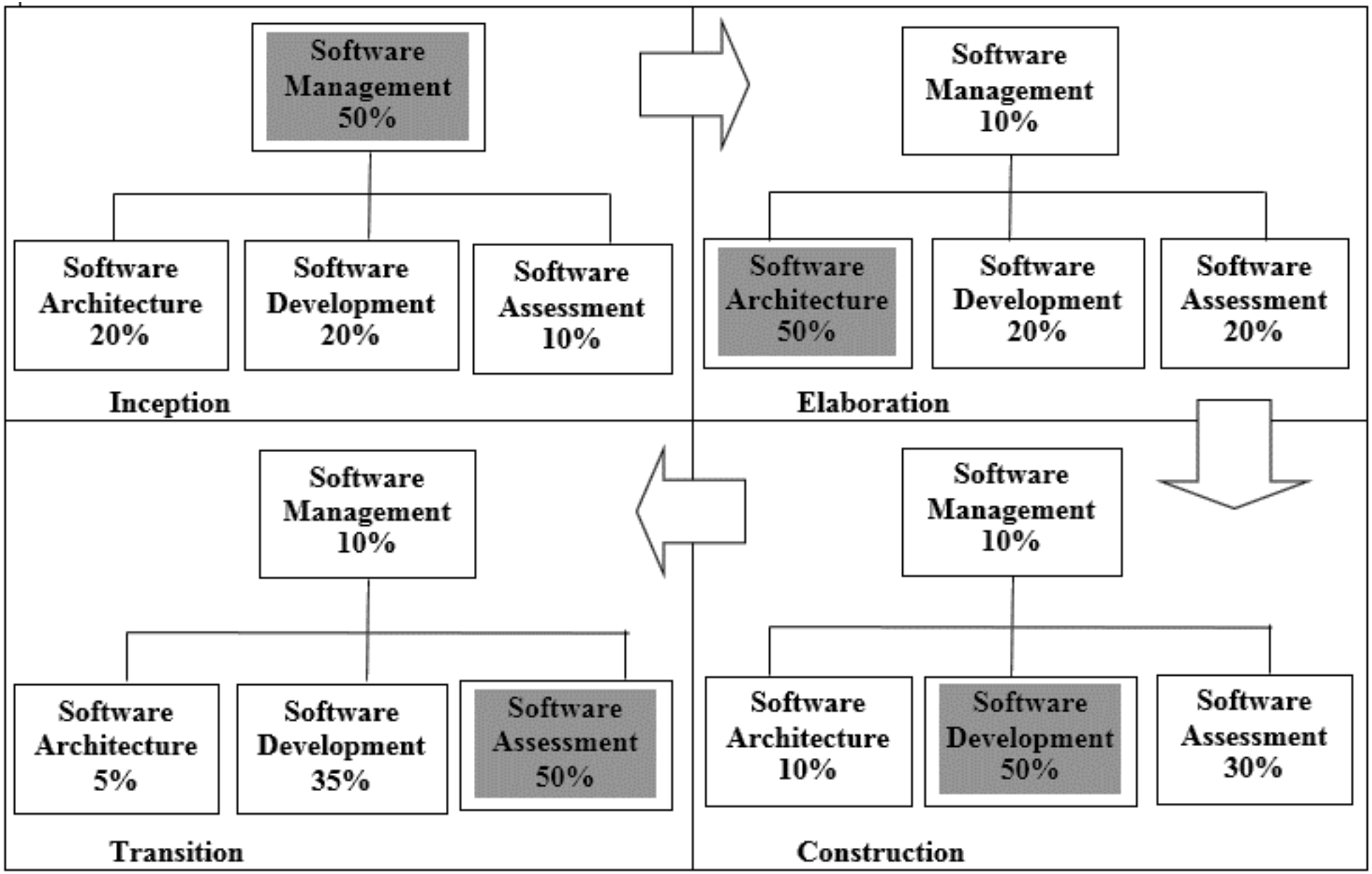
- Project infrastructure
- Independent testing
- Requirements verification
- Metrics analysis
- Configuration control
- Change management
- User deployment

### Life-Cycle Focus

| Inception | Elaboration | Construction | Transition |
|---|---|---|---|
| Infrastructure planning<br>Primary scenario prototyping | Infrastructure base lining<br>Architecture release testing<br>Change management<br>Initial user manual | Infrastructure upgrades<br>Release testing<br>Change management<br>User manual base line<br>Requirements verification | Infrastructure maintenance<br>Release base lining<br>Change management<br>Deployment to users<br>Requirements verification |

# Project Organizations and Responsibilities

## Evolution of Organizations



# Project Organizations and Responsibilities

## Evolution of Organizations

| | |
|---|---|
| **Inception:**<br>Software management: 50%<br>Software Architecture: 20%<br>Software development: 20%<br>Software Assessment (measurement/evaluation):10% | **Elaboration:**<br>Software management: 10%<br>Software Architecture: 50%<br>Software development: 20%<br>Software Assessment (measurement/evaluation):20% |
| **Construction:**<br>Software management: 10%<br>Software Architecture: 10%<br>Software development: 50%<br>Software Assessment (measurement/evaluation):30% | **Transition:**<br>Software management: 10%<br>Software Architecture: 5%<br>Software development: 35%<br>Software Assessment (measurement/evaluation):50% |

# 3.3 Process Automation – Tools and Environment

# Process Automation

- The environment must be the first-class artifact of the process.
- Process automation & change management is critical to an iterative process. If the change is expensive then the development organization will resist it.
- Round-trip engineering & integrated environments promote change freedom & effective evolution of technical artifacts.
- Metric automation is crucial to effective project control.
- External stakeholders need access to environment resources to improve interaction with the development team & add value to the process.
- The three levels of process which requires a certain degree of process automation for the corresponding process to be carried out efficiently.
  - ✓ **Metaprocess (Line of business):** The automation support for this level is called an infrastructure.
  - ✓ **Macroproces (project):** The automation support for a project's process is called an **environment.**
  - ✓ **Microprocess (iteration):** The automation support for generating artifacts is generally called a tool.

# Process Automation

## Tools: Automation Building blocks:

Many tools are available to automate the software development process. Most of the core software development tools map closely to one of the process workflows

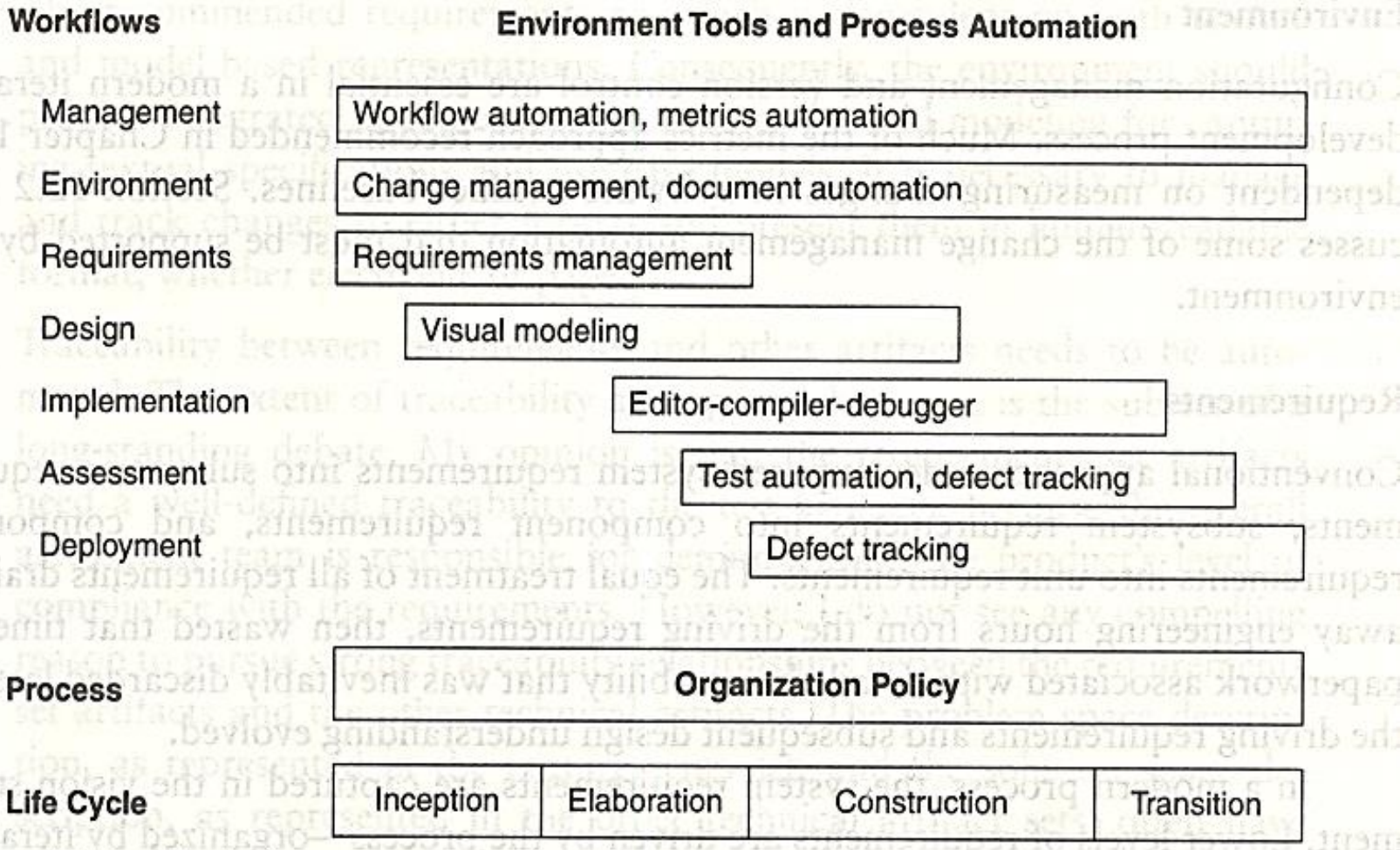



FIGURE 12-1. *Typical automation and tool components that support the process workflows*

# Process Automation

## The Project Environment

The project environment artifacts evolve through three discrete states:
(1) Prototyping Environment.
(2) Development Environment.
(3) Maintenance Environment.

- **The Prototype Environment** includes an architecture test bed for prototyping project architecture to evaluate trade-offs during inception & elaboration phase of the life cycle.

- **The Development environment** should include a full suite of development tools needed to support various process workflows & round-trip engineering to the maximum extent possible.

- **The Maintenance Environment** should typically coincide with the mature version of the development.

# Process Automation

## Round-Trip engineering

### Round Trip Environment

- Tools must be integrated to maintain consistency & traceability.
- Round-Trip engineering is the term used to describe this key requirement for environment that support iterative development.
- As the software industry moves into maintaining different information sets for the engineering artifacts, more automation support is needed to ensure efficient & error free transition of data from one artifacts to another.
- Round-trip engineering is the environment support necessary to maintain Consistency among the engineering artifacts.

# Process Automation

## Round-Trip engineering

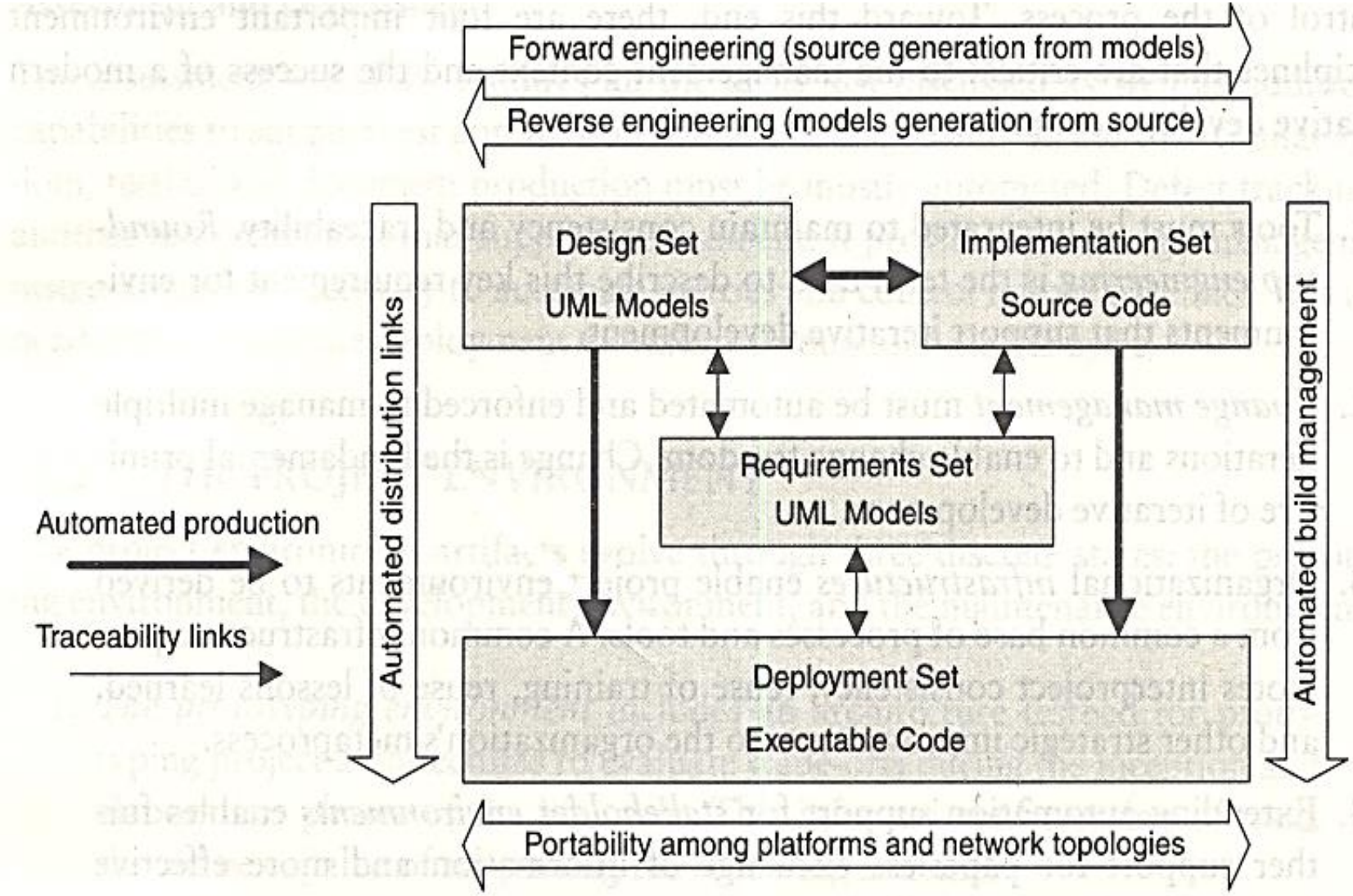



FIGURE 12-2. *Round-trip engineering*

# Process Automation

## Change Management

**Change Management**

- Change management must be automated & enforced to manage multiple iterations & to enable change freedom.
- Change is the fundamental primitive of iterative Development.

***a. Software Change Orders***

- The atomic unit of software work that is authorized to create, modify or obsolesce components within a configuration baseline is called a software change orders ( SCO )
- The basic fields of the SCO are Title, description, metrics, resolution, assessment & disposition

# Process Automation

## Change Management

**Title:** _____

**Description**

Name: _____ Date: _____

Project: _____

**Metrics**

Category: _____ (0/1 error, 2 enhancement, 3 new feature, 4 other)

| Initial Estimate | | Actual Rework Expended | | | |
|---|---|---|---|---|---|
| Breakage: | | Analysis: | | Test: | |
| Rework: | | Implement: | | Document: | |

**Resolution**

Analyst: _____

Software Component: _____

**Assessment**

Method: _____ (inspection, analysis, demonstration, test)

Tester: _____ Platforms: _____ Date: _____

**Disposition**

State: _____ Release: _____ Priority _____

Acceptance: _____ Date: _____

Closure: _____ Date: _____

FIGURE 12-3. *The primitive components of a software change order*

# Process Automation

## Change Management

***b. Configuration Baseline***

- A configuration baseline is a named collection of software components &Supporting documentation that is subjected to change management & is upgraded, maintained, tested, statuses & obsolesced a unit

- There are generally two classes of baselines:
  - ✓ External Product Release
  - ✓ Internal testing Release

- Three levels of baseline releases are required for most Systems
  1. Major release (N)
  2. Minor Release (M)
  3. Interim (temporary) Release (X)

# Process Automation

## Change Management

***c. Configuration Baseline***

- Major release represents a new generation of the product or project
- A minor release represents the same basic product but with enhanced features, performance or quality.
- Major & Minor releases are intended to be external product releases that are persistent & supported for a period of time.
- An interim release corresponds to a developmental configuration that is intended to be transient.
- Once software is placed in a controlled baseline all changes are tracked such that a distinction must be made for the cause of the change.
- Change categories are:

  **Type 0:** Critical Failures (must be fixed before release)

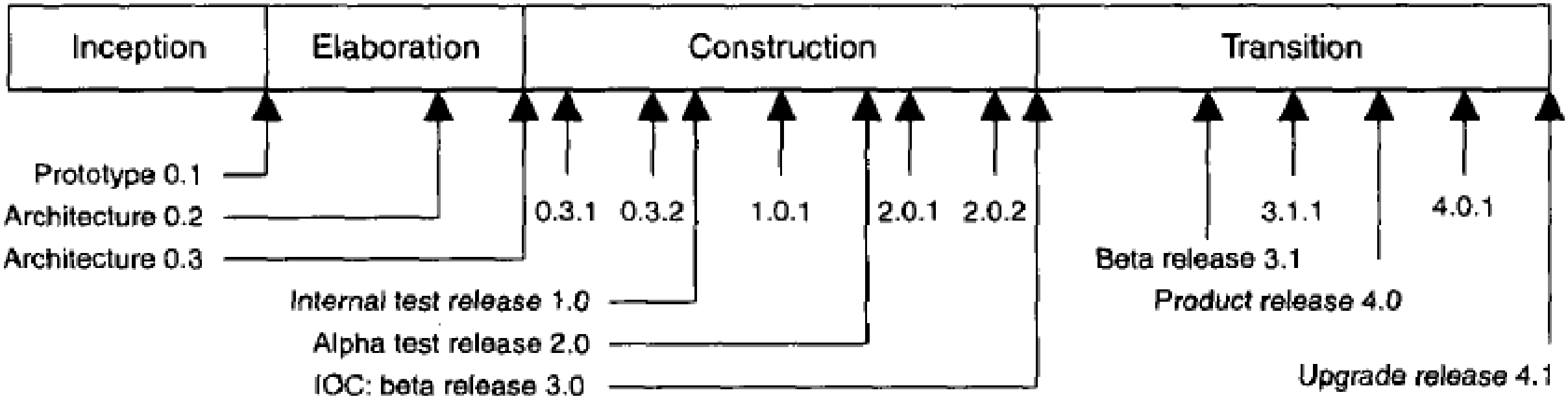
  **Type 1:** A bug or defect either does not impair (Harm) the usefulness of the system or can be worked around

  **Type 2:** A change that is an enhancement rather than a response to a defect

  **Type 3:** A change that is necessitated by the update to the environment

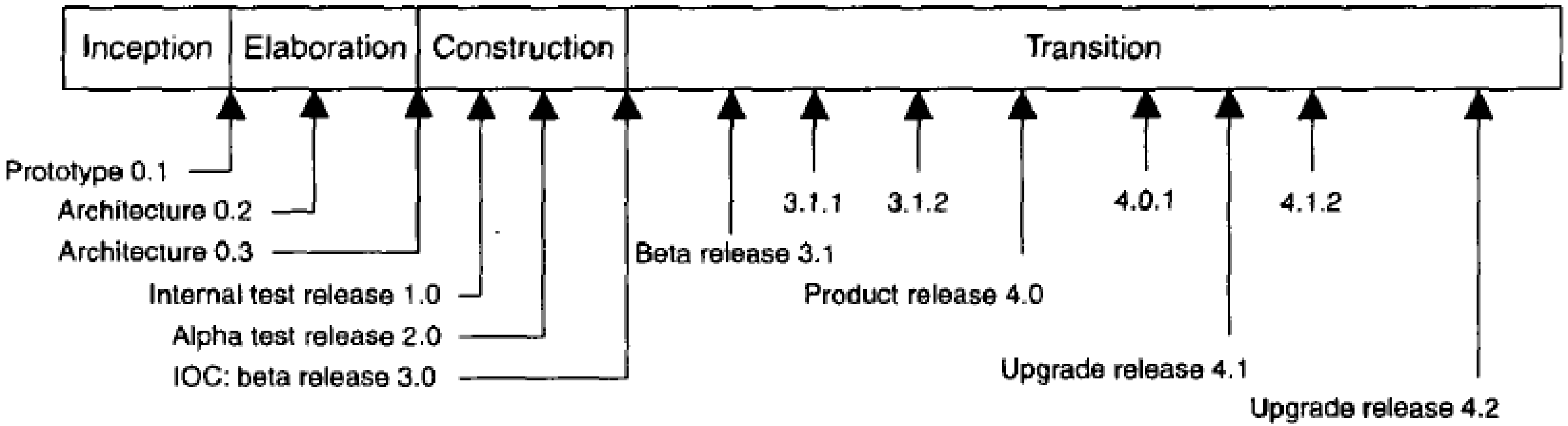
  **Type 4:** Changes that are not accommodated by the other categories.

## Typical project release sequence for a large-scale, one-of-a-kind project




*Example release histories for a typical project and a typical product*

## Typical project release sequence for a small commercial product




*Example release histories for a typical project and a typical product*

# Process Automation

## Change Management

***d. Configuration Control Board (CCB)***

- A CCB is a team of people that functions as the decision authority on the content of configuration baselines
- A CCB includes:
    1. Software managers
    2. Software Architecture managers
    3. Software Development managers
    4. Software Assessment managers
    5. Other Stakeholders who are integral to the maintenance of the controlled software delivery system?

# Process Automation

## Infrastructure

The organization infrastructure provides the organization's capital assets including two key artifacts - Policy & Environment

## 1. Organization Policy:

- A Policy captures the standards for project software development processes
- The organization policy is usually packaged as a handbook that defines the life cycles & the process primitives such as:
  - ✓ Major milestones
  - ✓ Intermediate Artifacts
  - ✓ Engineering repositories
  - ✓ Metrics
  - ✓ Roles & Responsibilities

# Process Automation

## Infrastructure

**I. Process-primitive definitions**
- A. Life-cycle phases (inception, elaboration, construction, transition)
- B. Checkpoints (major milestones, minor milestones, status assessments)
- C. Artifacts (requirements, design, implementation, deployment, management sets)
- D. Roles and responsibilities (PRA, SEPA, SEEA, project teams)

**II. Organizational software policies**
- A. Work breakdown structure
- B. Software development plan
- C. Baseline change management
- D. Software metrics
- E. Development environment
- F. Evaluation criteria and acceptance criteria
- G. Risk management
- H. Testing and assessment

**III. Waiver policy**

**IV. Appendixes**
- A. Current process assessment
- B. Software process improvement plan

FIGURE 12-5. *Organization policy outline*

# Process Automation

## Infrastructure

### 2. Organization Environment

The Environment that captures an inventory of tools which are building blocks from which project environments can be configured efficiently & economically

***Stakeholder Environment***

Many large scale projects include people in external organizations that represent other stakeholders participating in the development process they might include:

- ✓ Procurement agency contract monitors
- ✓ End-user engineering support personnel
- ✓ Third party maintenance contractors
- ✓ Independent verification & validation contractors
- ✓ Representatives of regulatory agencies & others.

# Process Automation

## Infrastructure

- These stakeholder representatives also need to access to development resources so that they can contribute value to overall effort.
- These stakeholders will be access through on-line
- An on-line environment accessible by the external stakeholders allow them to participate in the process
- Accept & use executable increments for the hands-on evaluation.
- Use the same on-line tools, data & reports that the development organization uses to manage & monitor the project
- Avoid excessive travel, paper interchange delays, format translations, paper * shipping costs & other overhead cost

# Process Automation

## Infrastructure

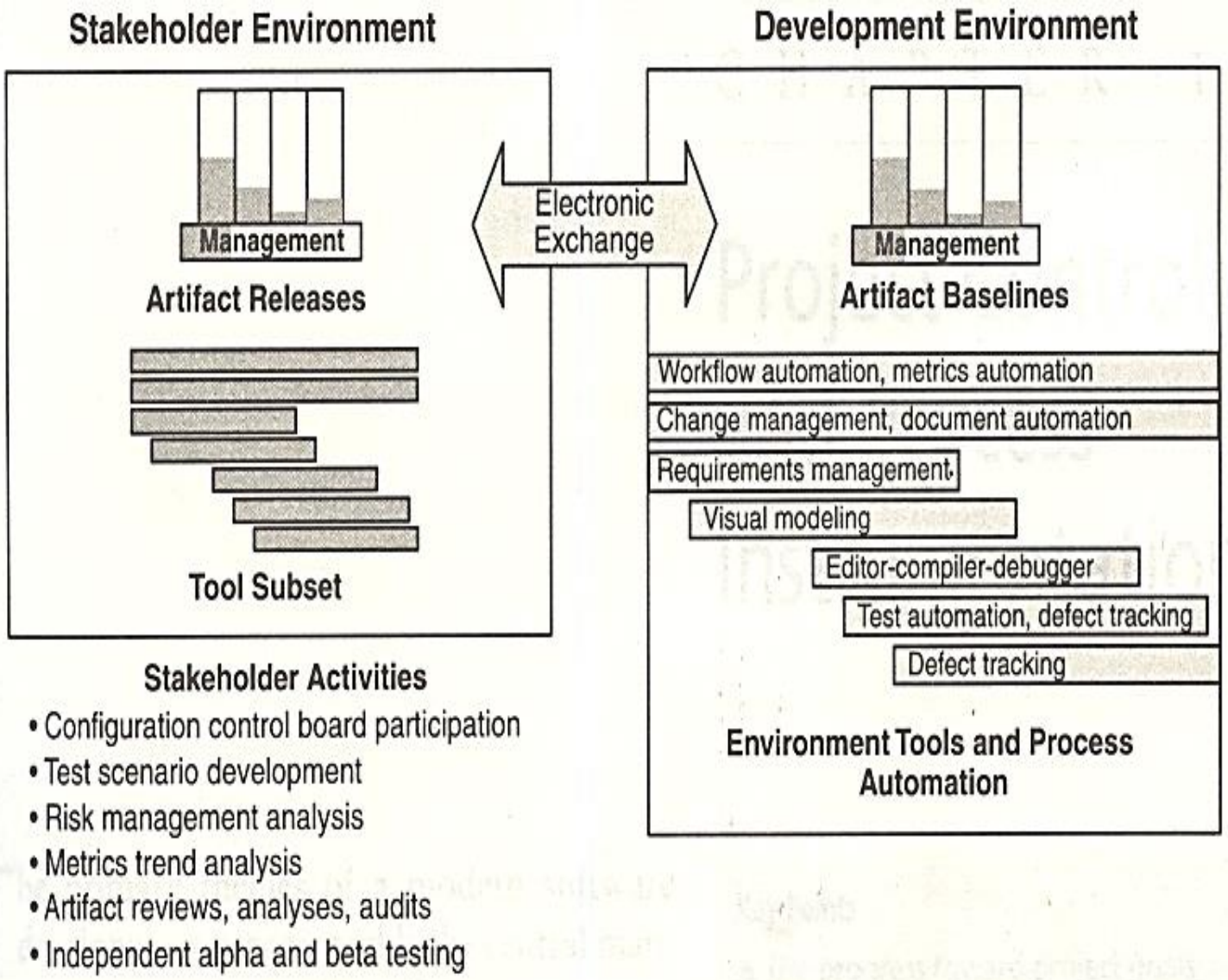



FIGURE 12-6. *Extending environments into stakeholder domains*

# 3.4 Project Control and Process Instrumentation

# Project Control and Process Instrumentation

## The Seven Core Metrics

### Management Indicators

- Work and progress (work performed over time)
- Budgeted cost and expenditures (cost incurred over time)
- Staffing and team dynamics (personnel changes over time)

### Quality Indicators

- Change traffic and stability (change traffic over time)
- Breakage and modularity (average breakage per change over time)
- Rework and adaptability (average rework per change over time)
- Mean time between failures (MTBF) and maturity (defect rate over time)

# Project Control and Process Instrumentation

## The Seven Core Metrics

| METRIC | PURPOSE | PERSPECTIVES |
|---|---|---|
| Work and progress | Iteration planning, plan vs. actuals, management indicator | SLOC, function points, object points, scenarios, test cases, SCOs |
| Budgeted cost and expenditures | Financial insight, plan vs. actuals, management indicator | Cost per month, full-time staff per month, percentage of budget expended |
| Staffing and team dynamics | Resource plan vs. actuals, hiring rate, attrition rate | People per month added, people per month leaving |
| Change traffic and stability | Iteration planning, management indicator of schedule convergence | SCOs opened vs. SCOs closed, by type (0,1,2,3,4), by release/component/subsystem |
| Breakage and modularity | Convergence, software scrap, quality indicator | Reworked SLOC per change, by type (0,1,2,3,4), by release/component/subsystem |
| Rework and adaptability | Convergence, software rework, quality indicator | Average hours per change, by type (0,1,2,3,4), by release/component/subsystem |
| MTBF and maturity | Test coverage/adequacy, robustness for use, quality indicator | Failure counts, test hours until failure, by release/component/subsystem |

## WORK AND PROGRESS

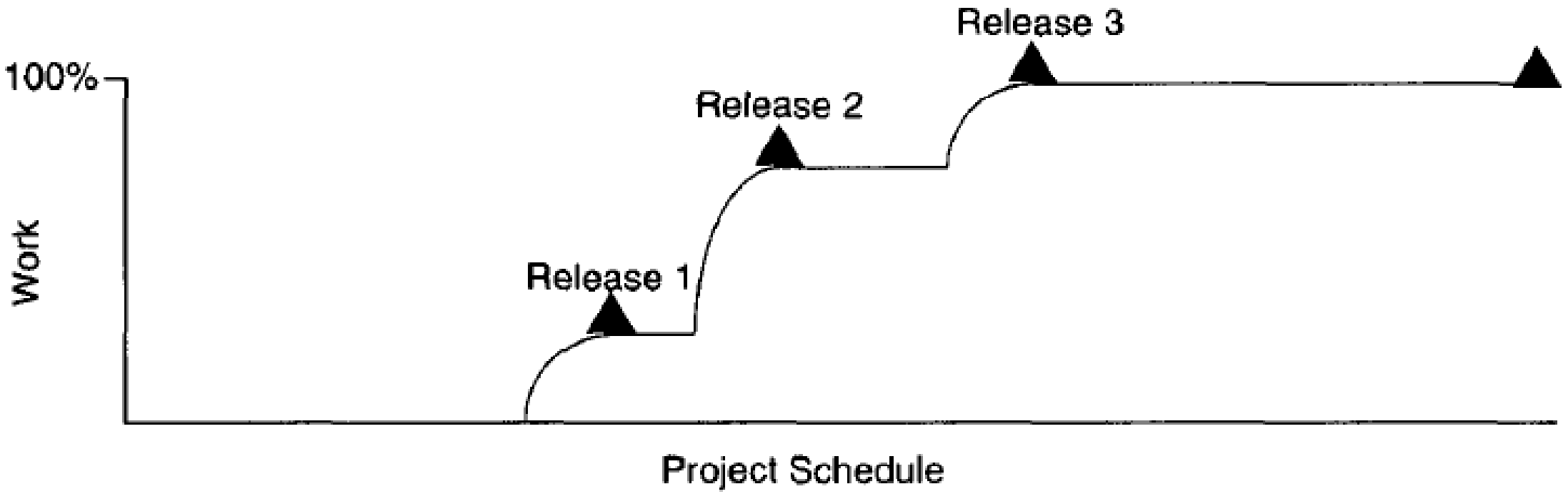



***Figure:*** *Expected progress for a typical project with three major releases*

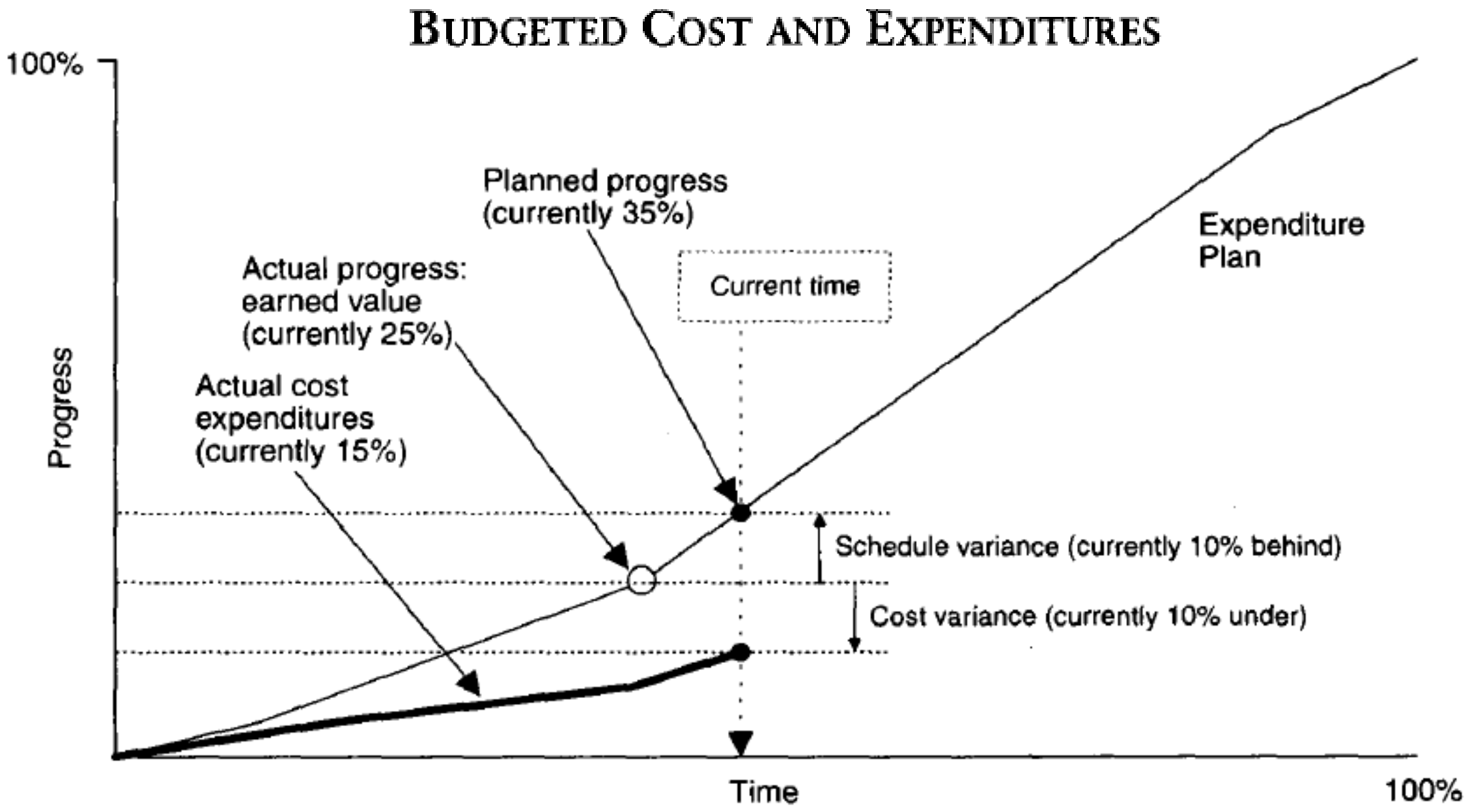



***Figure:*** *The basic parameters of an earned value system*

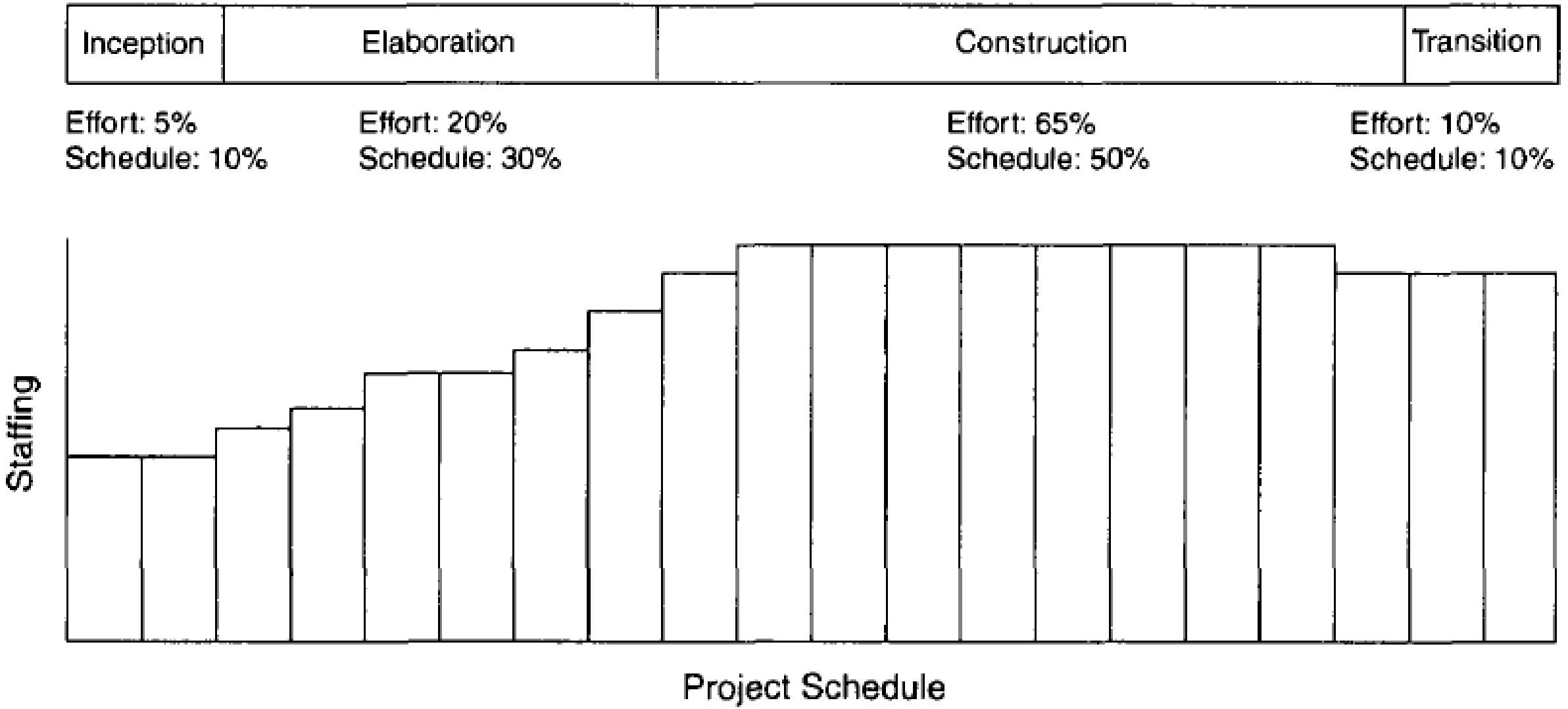



***Figure:*** *Typical staffing profile*

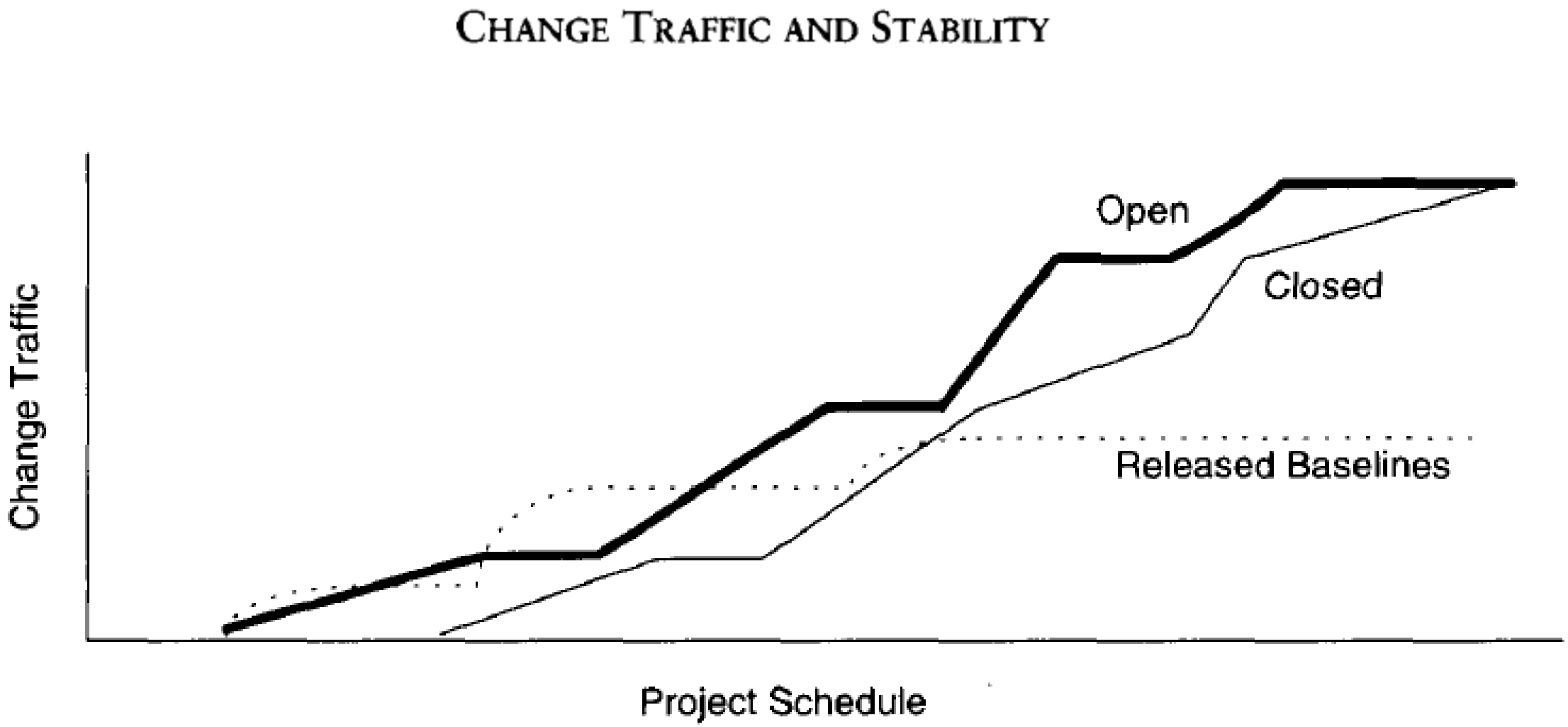



***Figure:*** *Stability expectation over a healthy project's life cycle*

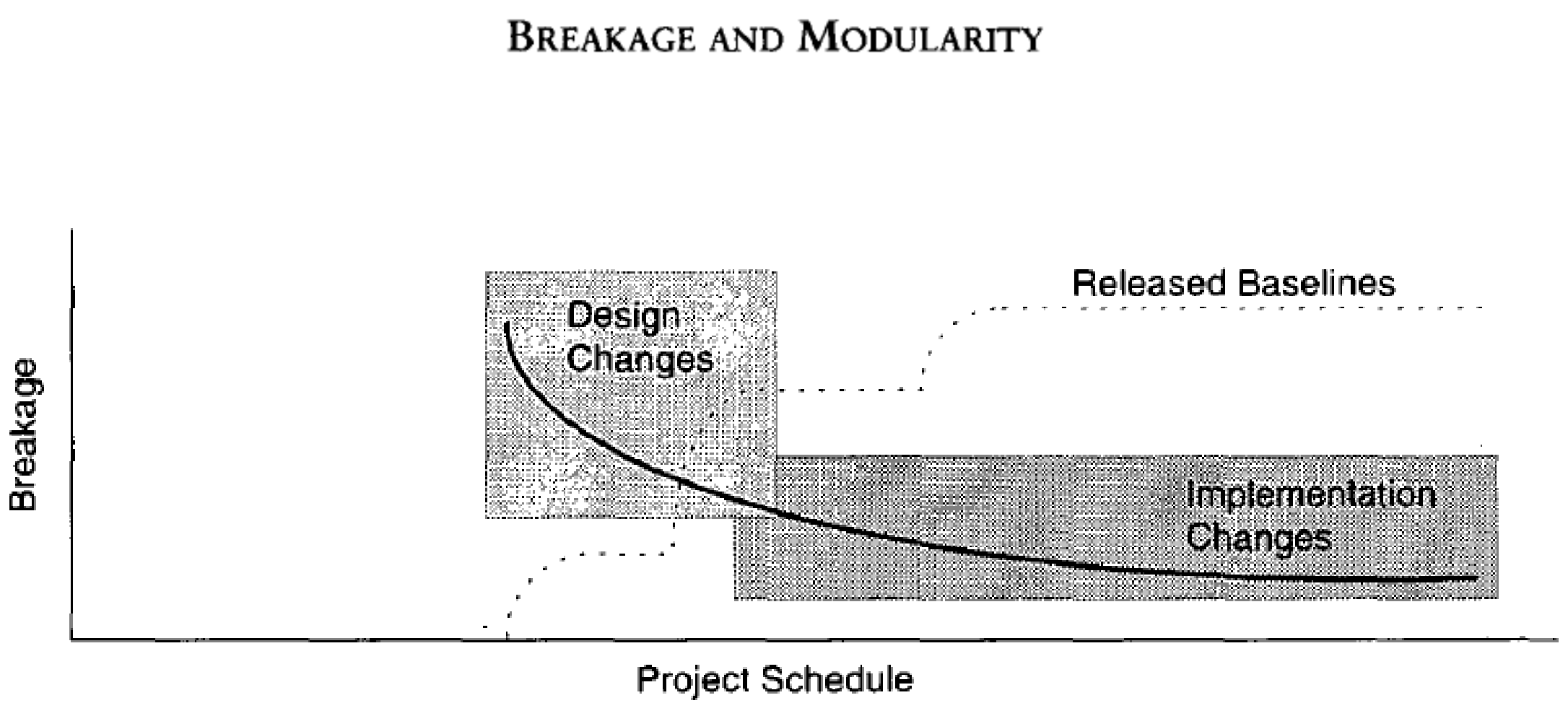



***Figure:*** *Modularity expectation over a healthy project's life cycle*

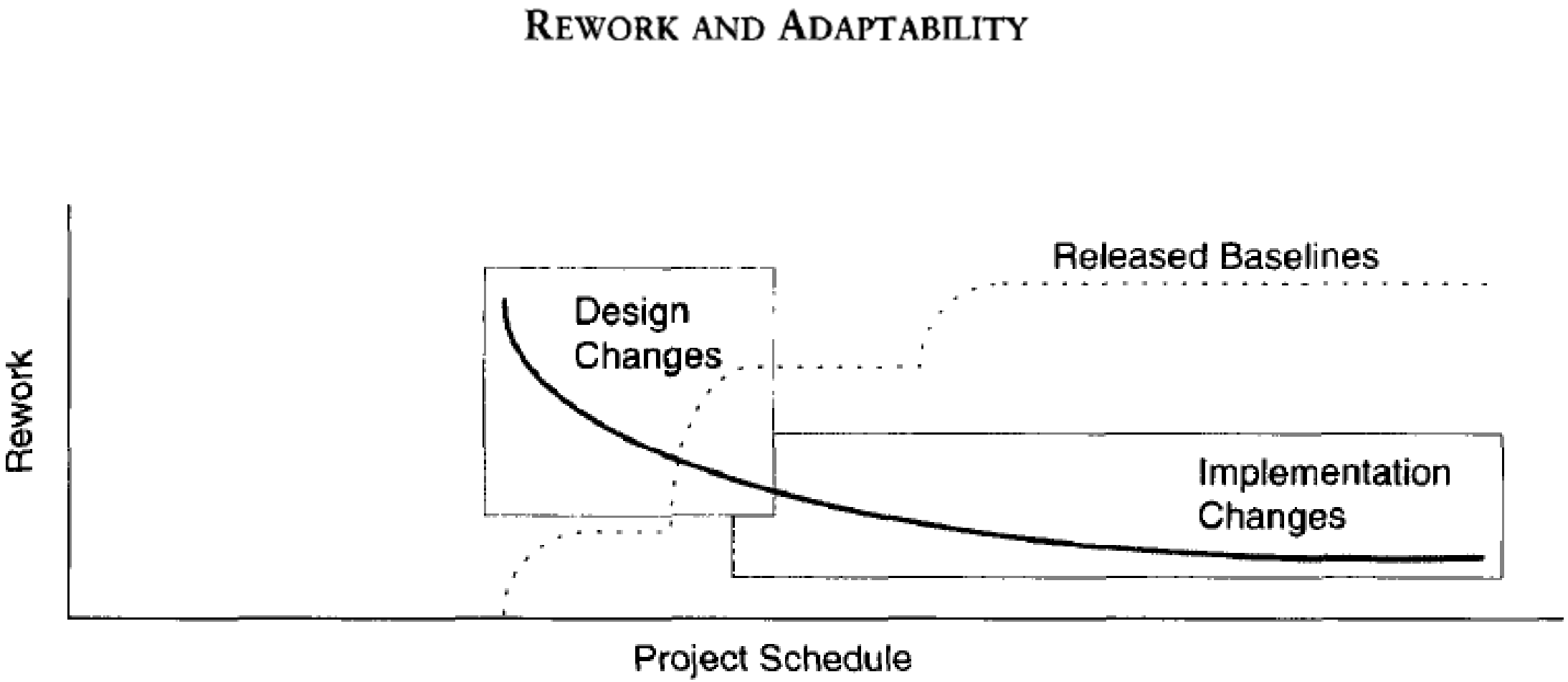



***Figure:*** *Adaptability expectation over a healthy project's life cycle*

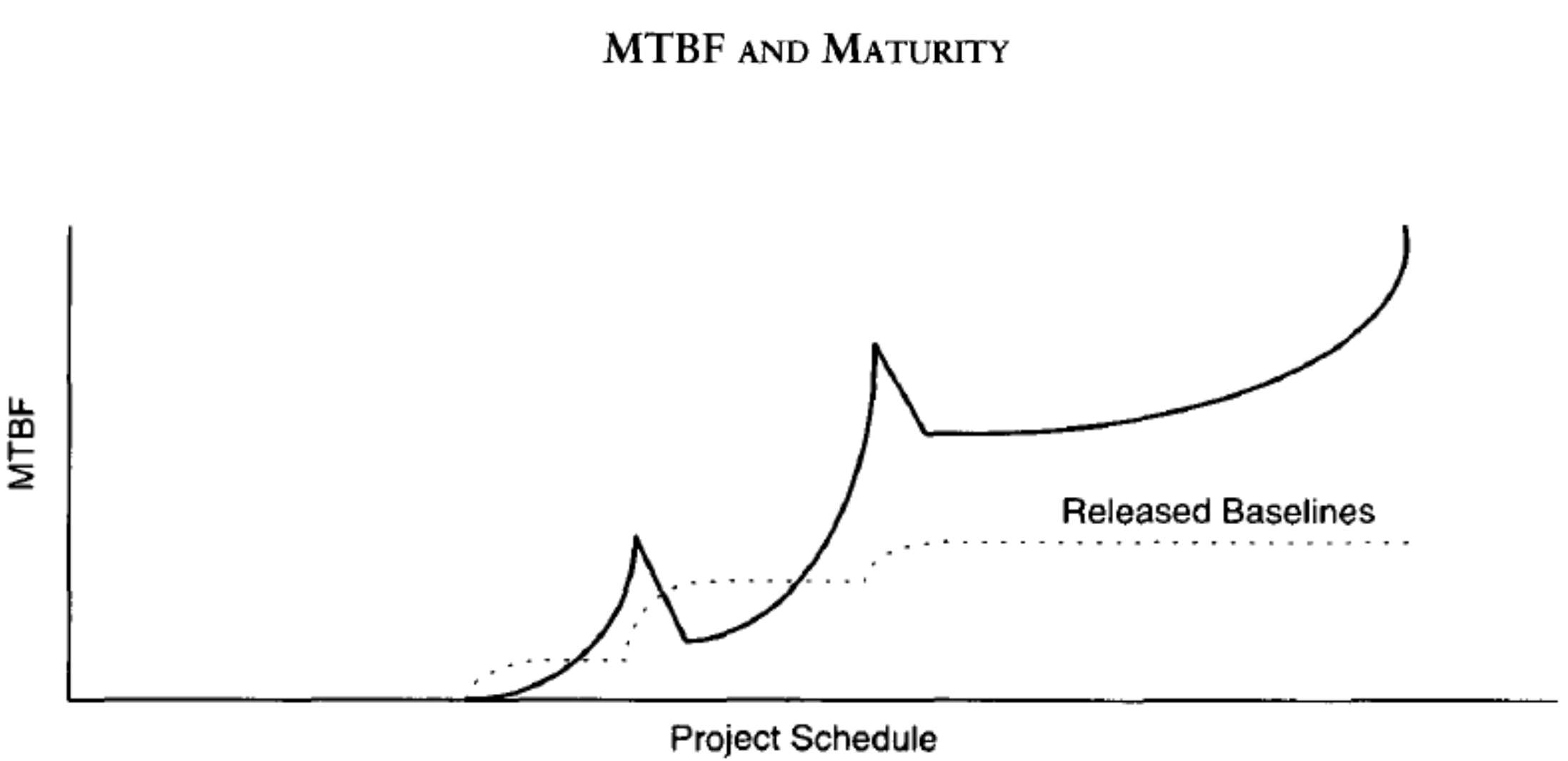


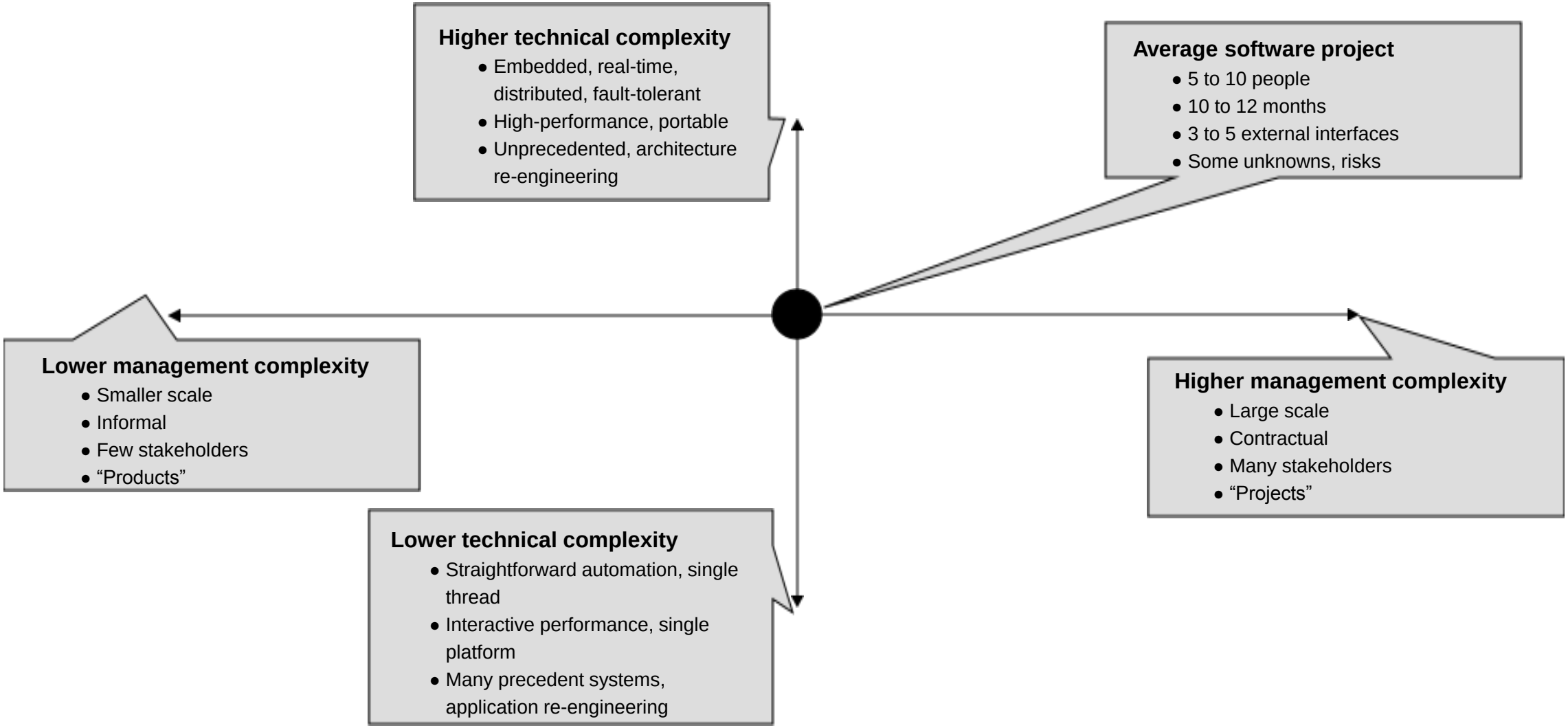

***Figure:*** *Maturity expectation over a healthy project's life cycle*

# 3.5 Process Customization

# Tailoring the Process

## Process Discriminants

The two primary dimensions of process variability



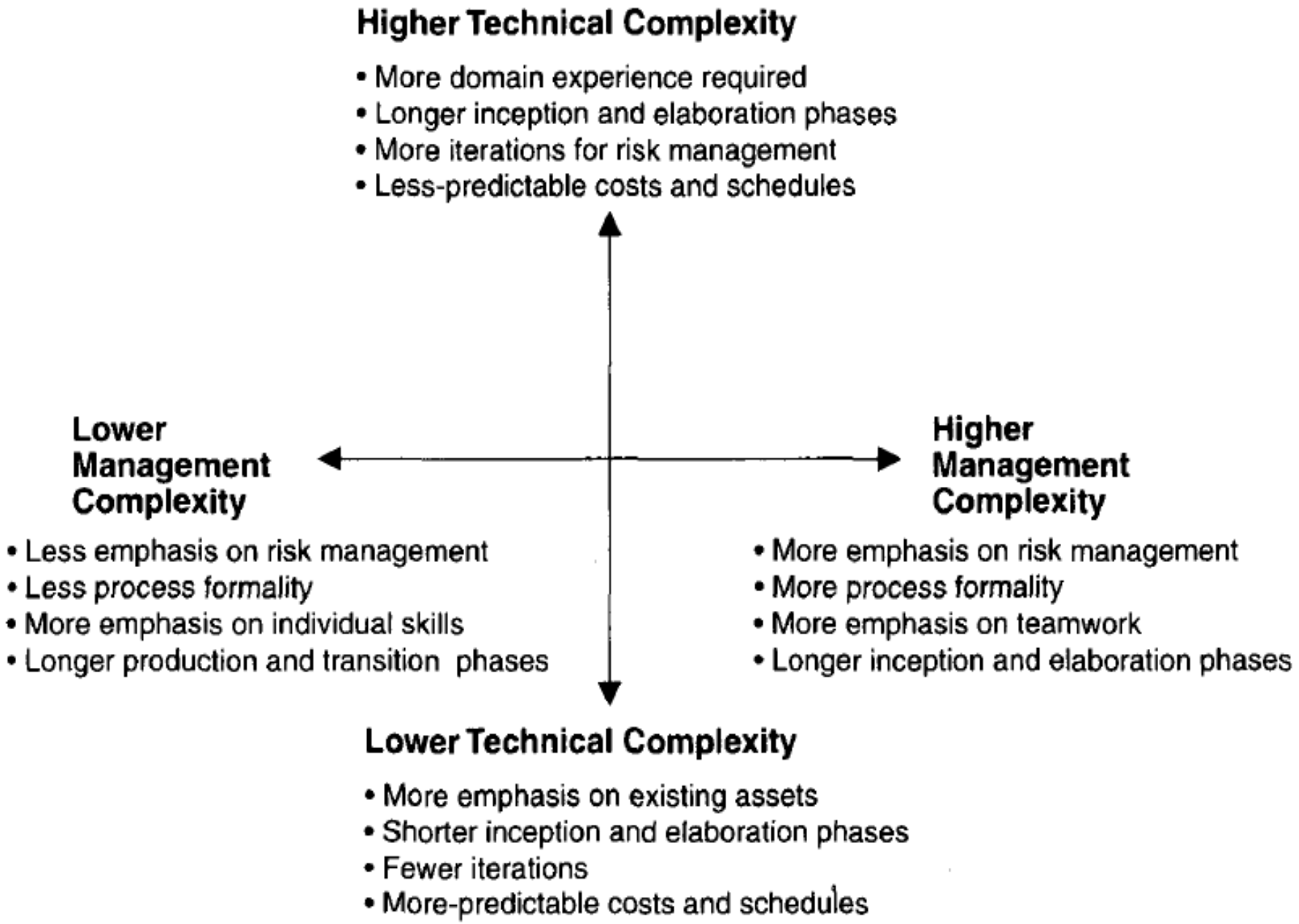



***Figure:*** *Priorities for tailoring the process framework*

# Tailoring the Process

## Example: Small-Scale Project vs. Large-Scale Project

**Differences in workflow priorities between small and large projects**

| Rank | Small Commercial Project | Large Complex Project |
|---|---|---|
| 1 | Design | Management |
| 2 | Implementation | Design |
| 3 | Deployment | Requirements |
| 4 | Requirements | Assessment |
| 5 | Assessments | Environment |
| 6 | Management | Implementation |
| 7 | Environment | Deployment |


Any Questions
?

# Book References:


1. Software Project Management – A Unifies Framework, Walker Royce, 1998, Addison Wesley
2. Software Project Management – From Concept to Deployment, Conway, K., 2001.
3. Software Project Management, Bob Hughes and Mike Cotterell, Latest Publication
4. Software Project Management, Rajeev Chopra, 2009
5. Software Engineering – A Practitioner's approach, Roger S. Pressman Latest Plublication
6. Managing Global Software Projects, Ramesh, 2001, TMH


Thank You