Dr. Abdallah Al-Sukairi

sukairi@kfupm.edu.sa

Information & Computer Science Department

Software Project Management

Conventional Software Management

- The success rate for software projects is very low. Only 10% of projects are delivered successfully within initial budget and schedule
- Waterfall model
- Finding and fixing a software problem after delivery costs 100 time more
- For every $1 we spend on development, we need $2 on maintenance
- Variations among people account for the biggest difference in software productivity
- Only about 15% of software development is devoted to programming
- Walkthrough catch 60% of the errors
- 80% of contribution comes from 20% of the contributors

Evolution of Software Economics

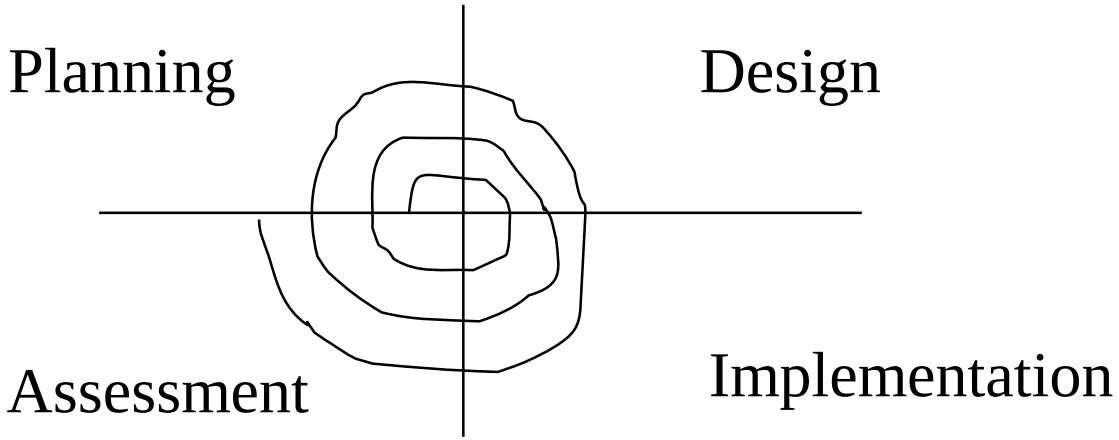
- **Effort**
  - **Personnel**
  - **Environment**
  - **Quality**
  - **Size-Process**
- **Cost estimation models**
  - **COCOMO**

Improving Software Economics

- **Size**
  - **Higher order languages & objected-oriented**
  - **Reuse and commercial components**
- **Process**
  - **Iterative development**
- **Personnel**
  - **Skill development**
  - **Teamwork**
  - **Win-win cultures**
- **Environment**
  - **Integrated tools**
  - **Open systems**
  - **Automation**
- **Quality**
  - **Demo-based assessment**

The Old Way and the New

- **Architecture-first approach**
- **Iterative life-cycle process**
- **Component-based development**
- **Change management environment**
- **Round-trip engineering**



Life-Cycle Phases

- **Engineering stage**
  - **Inspection phase**
  - **Elaboration phase**
- **Construction phase**
- **Transition phase**

Artifacts of the Process

- Requirements set
  - Vision document
  - Regiments model
- Design set
  - Design model
  - Test model
  - Software architecture
- Implementation set
  - Source code
  - Component executable
- Deployment set
  - Product executable
  - User manual

Management Set

- **Planning artifacts**
  - Work breakdown structure
  - Business case
  - Release specifications
  - Software development plan
- **Operational artifacts**
  - Release descriptions
  - Status assessments
  - Software change order database
  - Deployment documents
  - Environment

Software Architecture

- The central design problem of a complex software system
- A significant project milestone
- It describes
  - the structure of software systems
  - their behavior
  - their collaboration
  - their composition
- Architecture should include
  - Requirements: use cases, quality objectives, priority relationship
  - Design: names, attributes, structures, behaviors, groupings, and relationships of significant classes and components
  - Implementation: source component inventory and bill of material
  - Deployment: executable components sufficient to demonstrate the critical use cases

Software Process Workflows

- **Management:**
  - **Business case, Software development plan, Status assessment, Vision, Work breakdown structure**
- **Environment**
  - **Environment, Software change order database**
- **Requirements**
  - **Requirement set, Release specifications, Vision**
- **Design**
  - **Design set, Architecture description**
- **Implementation**
  - **Implementation set, Deployment set**
- **Assessment**
  - **Release specifications, Release descriptions, User manual**
- **Deployment**

Checkpoints of the Process

- **Major milestones**
  - system wide events
  - synchronize the management and engineering perspectives
  - verify that the aims of the phase have been achieved
- **Minor milestones**
  - iteration-focused events
  - review the content of an iteration and to authorized continued work
- **Status assessment**
  - provide management with frequent and regular insight into the progress being made

Work Breakdown Structures (WBS)

- **A hierarchy of elements that decompose the project plan**
  - **Clear task decomposition for assignment of responsibilities**
  - **A framework fro scheduling, budgeting, and expenditure tracking**

Conventional WBS

- **Structured around the product design**

Management
System requirements and design
Subsystem 1
    Component 11
        Requirements
        Design
        Code
        Test
        Documentation
    . . . (similar structures for other components)
    Component 1N
        Requirements
        Design
        Code
        Test
        Documentation
. . . (similar structures for other subsystems)
Subsystem M
    Component M1
        Requirements
        Design
        Code
        Test
        Documentation
    . . . (similar structures for other components)
    Component MN
        Requirements
        Design
        Code
        Test
        Documentation
Integration and test
    Test planning
    Test procedure preparation
    Testing
    Test reports
Other support areas
    Configuration control
    Quality assurance
    System administration

Evolutionary WBS

- **Structured around the process framework**
  - **First level elements are the workflow**
  - **Second level elements are defined for each phase of the lifecycle**
  - **Third level elements are defined for the focus of activities that produce the artifacts of each phase**

Default Work Breakdown Structure

- A *Management*
  - AA Inception phase management
    - AAA Business case development
    - AAB Elaboration phase release specifications
    - AAC Elaboration phase WBS baselining
    - AAD Software development plan
    - AAE Inception phase project control and status assessments
  - AB Elaboration phase management
    - ABA Construction phase release specifications
    - ABB Construction phase WBS baselining
    - ABC Elaboration phase project control and status assessments
  - AC Construction phase management
    - ACA Deployment phase planning
    - ACB Deployment phase WBS baselining
    - ACC Construction phase project control and status assessments
  - AD Transition phase management
    - ADA Next generation planning
    - ADB Transition phase project control and status assessments
- B Environment
  - BA Inception phase environment specification
  - BB Elaboration phase environment baselining
    - BBA Development environment installation and administration
    - BBB Development environment integration and custom toolsmithing
    - BBC SCO database formulation
  - BC Construction phase environment maintenance
    - BCA Development environment installation and administration
    - BCB SCO database maintenance
  - BD Transition phase environment maintenance
    - BDA Development environment maintenance and administration
    - BDB SCO database maintenance
    - BDC Maintenance environment packaging and transition
- C Requirements
  - CA Inception phase requirements development
    - CAA Vision specification
    - CAB Use case modeling
  - CB Elaboration phase requirements baselining
    - CBA Vision baselining
    - CBB Use case model baselining
  - CC Construction phase requirements maintenance
  - CD Transition phase requirements maintenance
- D Design
  - DA Inception phase architecture prototyping
  - DB Elaboration phase architecture baselining
    - DBA Architecture design modeling
    - DBB Design demonstration planning and conduct
    - DBC Software architecture description
  - DC Construction phase design modeling
    - DCA Architecture design model maintenance
    - DCB Component design modeling
  - DD Transition phase design maintenance
- E Implementation
  - EA Inception phase component prototyping
  - EB Elaboration phase component implementation
    - EBA Critical component coding demonstration integration
  - EC Construction phase component implementation
    - ECA Initial release(s) component coding and stand-alone testing
    - ECB Alpha release component coding and stand-alone testing
    - ECC Beta release component coding and stand-alone testing
    - ECD Component maintenance
  - ED Transition phase component maintenance
- F Assessment
  - FA Inception phase assessment planning
  - FB Elaboration phase assessment
    - FBA Test modeling
    - FBB Architecture test scenario implementation
    - FBC Demonstration assessment and release descriptions
  - FC Construction phase assessment
    - FCA Initial release assessment and release description
    - FCB Alpha release assessment and release description
    - FCC Beta release assessment and release description
  - FD Transition phase assessment
    - FDA Product release assessment and release descriptions
- G Deployment
  - GA Inception phase deployment planning
  - GB Elaboration phase deployment planning
  - GC Construction phase deployment
    - GCA User manual baselining
  - GD Transition phase deployment
    - GDA Product transition to user

Planning Guidelines

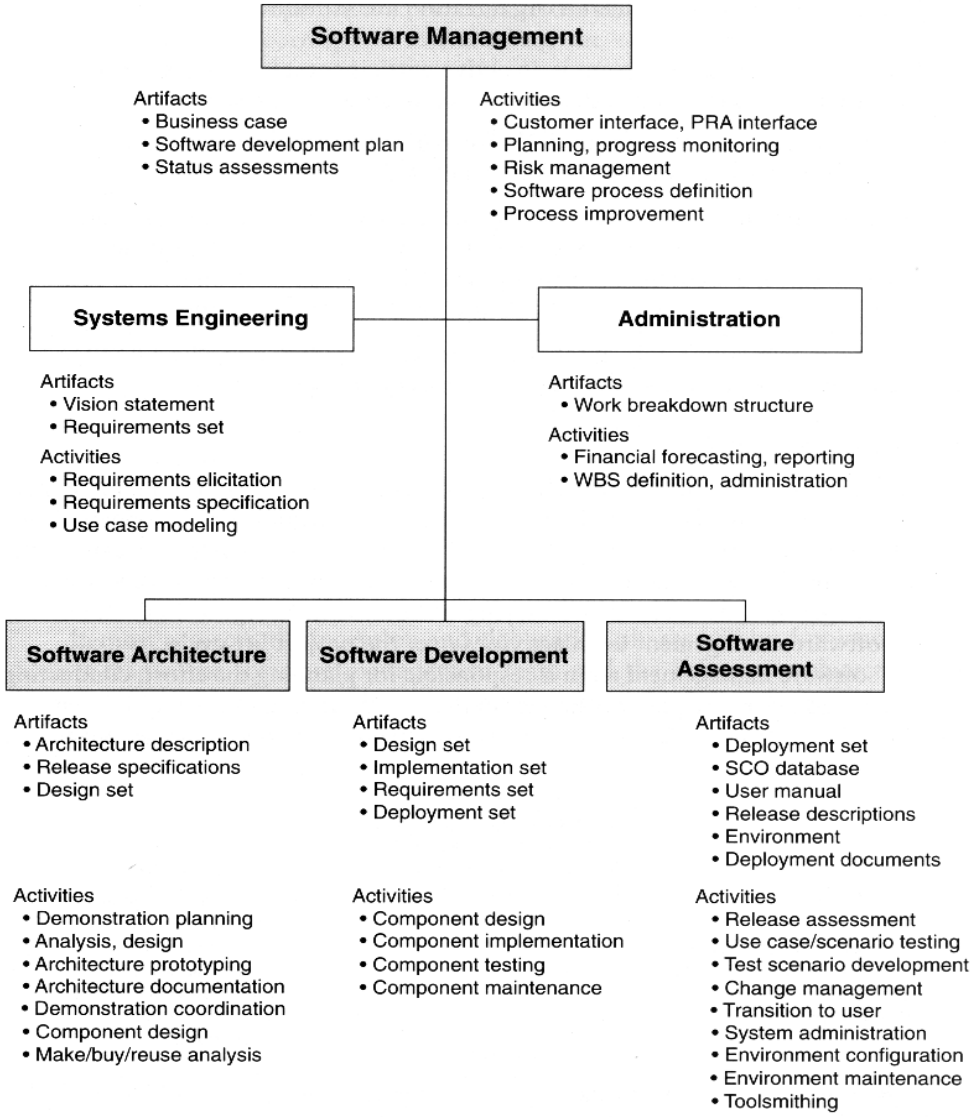
TABLE 10-1. *WBS budgeting defaults*

FIRST-LEVEL WBS ELEMENT	DEFAULT BUDGET
Management	10%
Environment	10%
Requirements	10%
Design	15%
Implementation	25%
Assessment	25%
Deployment	5%
Total	100%

TABLE 10-2. *Default distributions of effort and schedule by phase*

DOMAIN	INCEPTION	ELABORATION	CONSTRUCTION	TRANSITION
Effort	5%	20%	65%	10%
Schedule	10%	30%	50%	10%

Project Organizations & Responsibilities



Software Management Team Activities

Software Management

- Systems engineering
- Financial administration
- Quality assurance

Artifacts

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

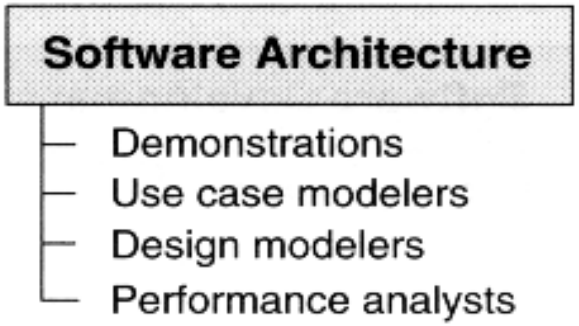
Responsibilities

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

Life-Cycle Focus

Inception	Elaboration	Construction	Transition
Elaboration phase planning Team formulation Contract baselining Architecture costs	Construction phase planning Full staff recruitment Risk resolution Product acceptance criteria Construction costs	Transition phase planning Construction plan optimization Risk management	Customer satisfaction Contract closure Sales support Next-generation planning

Software Architecture Team Activities



Artifacts

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

Responsibilities

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

Life-Cycle Focus

Inception	Elaboration	Construction	Transition
Architecture prototyping Make/buy trade-offs Primary scenario definition Architecture evaluation criteria definition	Architecture baselining Primary scenario demonstration Make/buy trade-off baselining	Architecture maintenance Multiple-component issue resolution Performance tuning Quality improvements	Architecture maintenance Multiple-component issue resolution Performance tuning Quality improvements

Software Development Team Activities

Software Development

└ 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 Make/buy trade-offs	Critical component design Critical component implementation and test Critical component baseline	Component design Component implementation Component stand-alone test Component maintenance	Component maintenance Component documentation

Software Assessment Team Activities

Artifacts

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

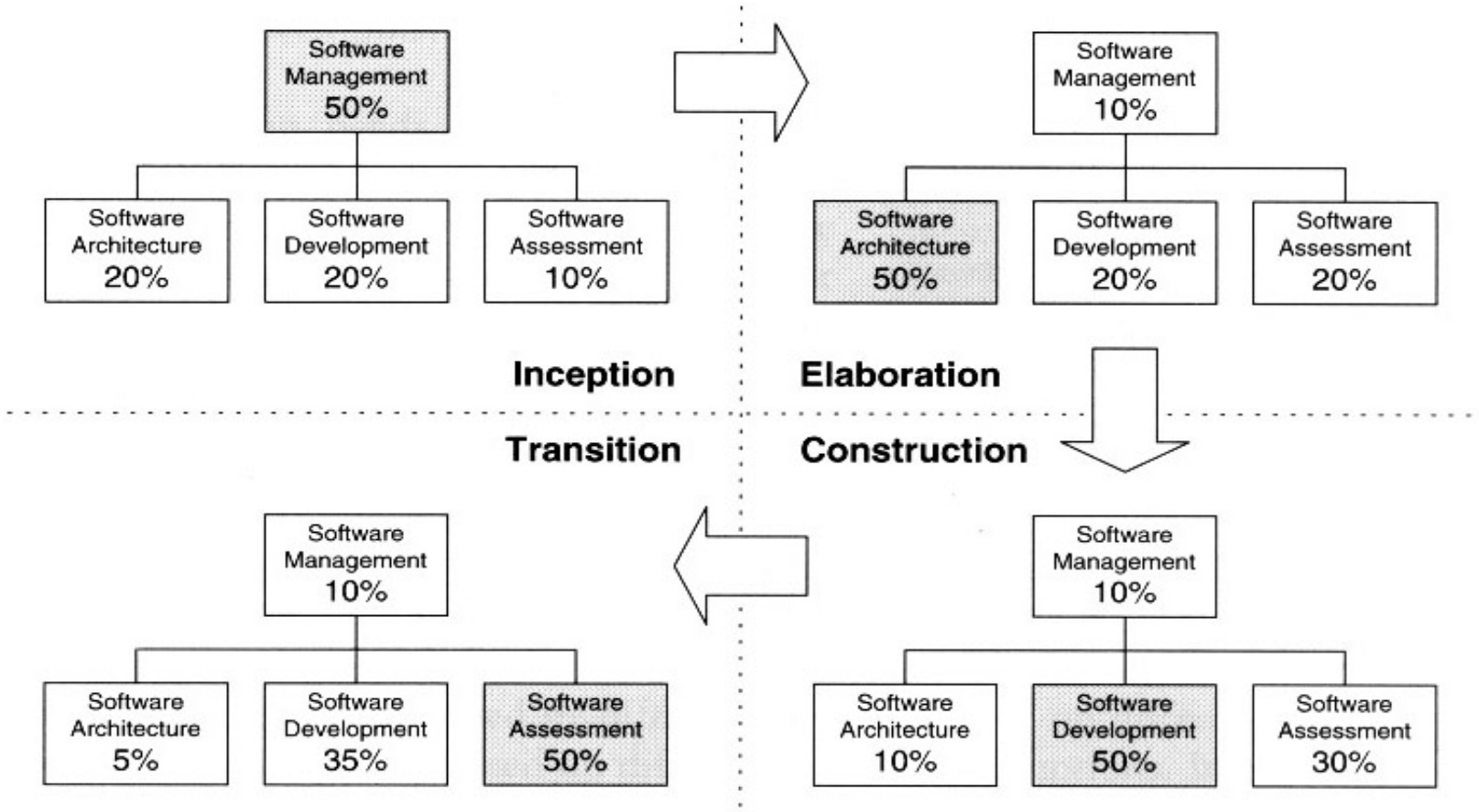
Responsibilities

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

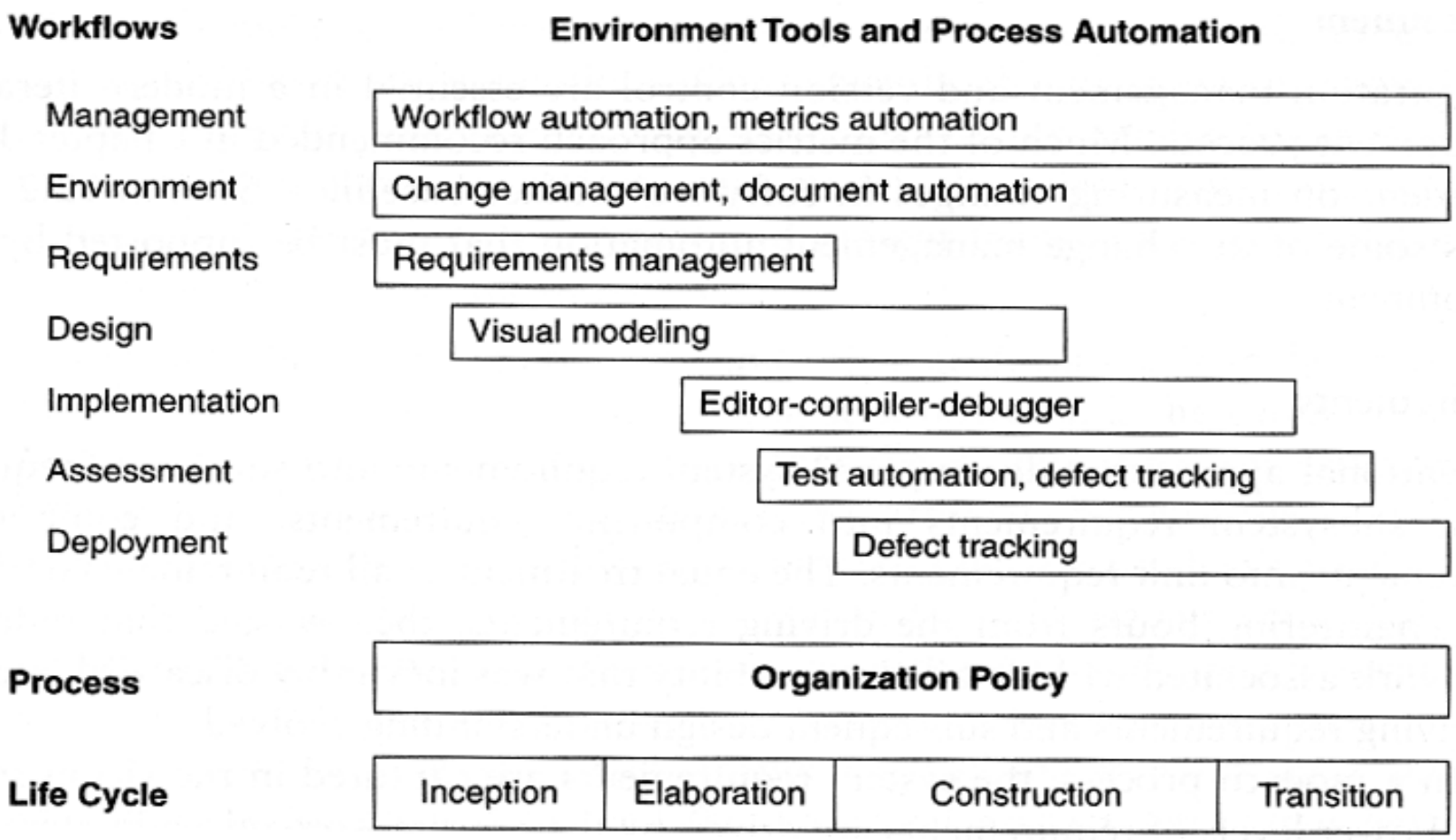
Life-Cycle Focus

Inception	Elaboration	Construction	Transition
Infrastructure planning Primary scenario prototyping	Infrastructure baseline Architecture release testing Change management Initial user manual	Infrastructure upgrades Release testing Change management User manual baseline Requirements verification	Infrastructure maintenance Release baselining Change management Deployment to users Requirements verification

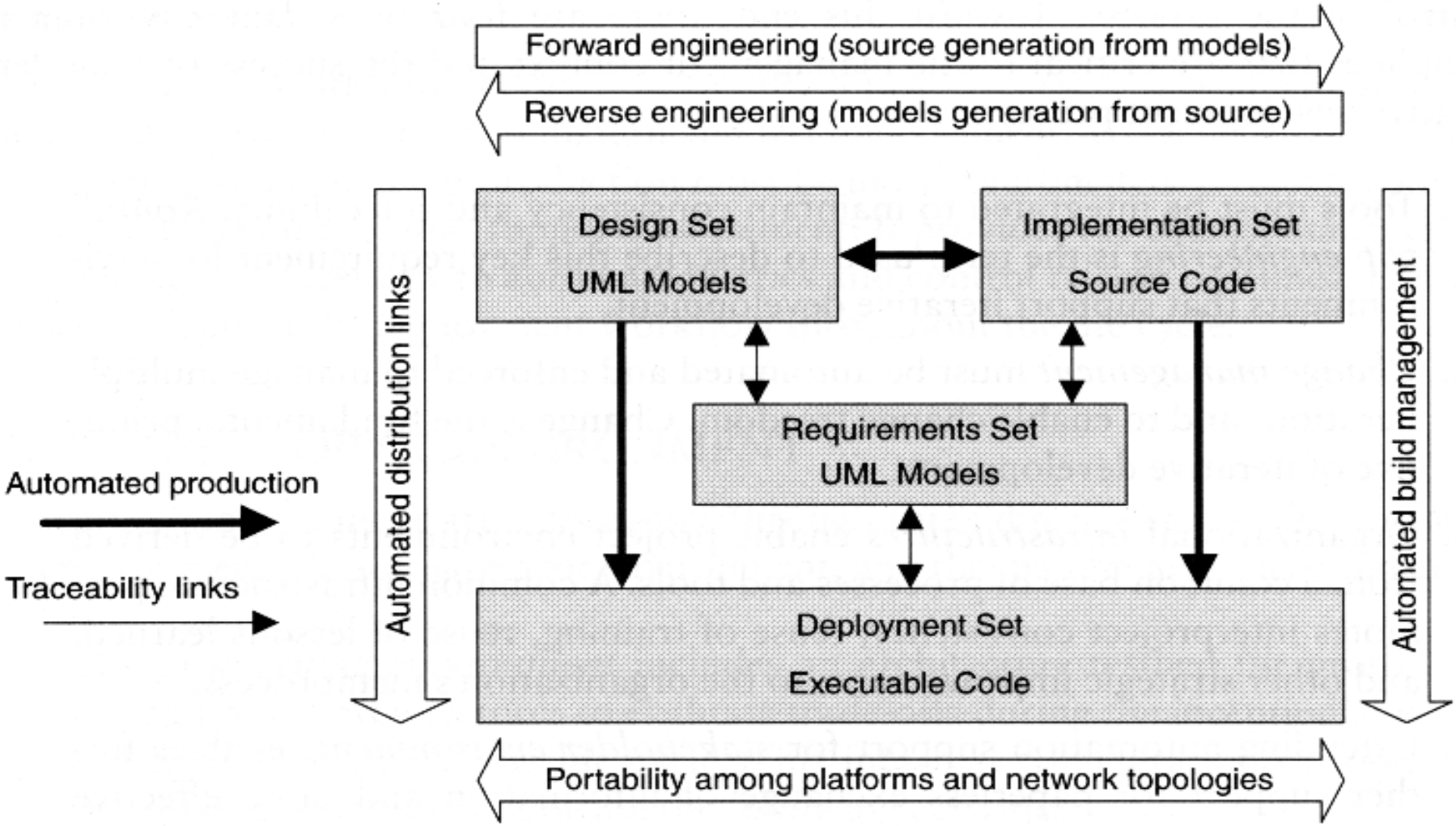
Evolution of Organization



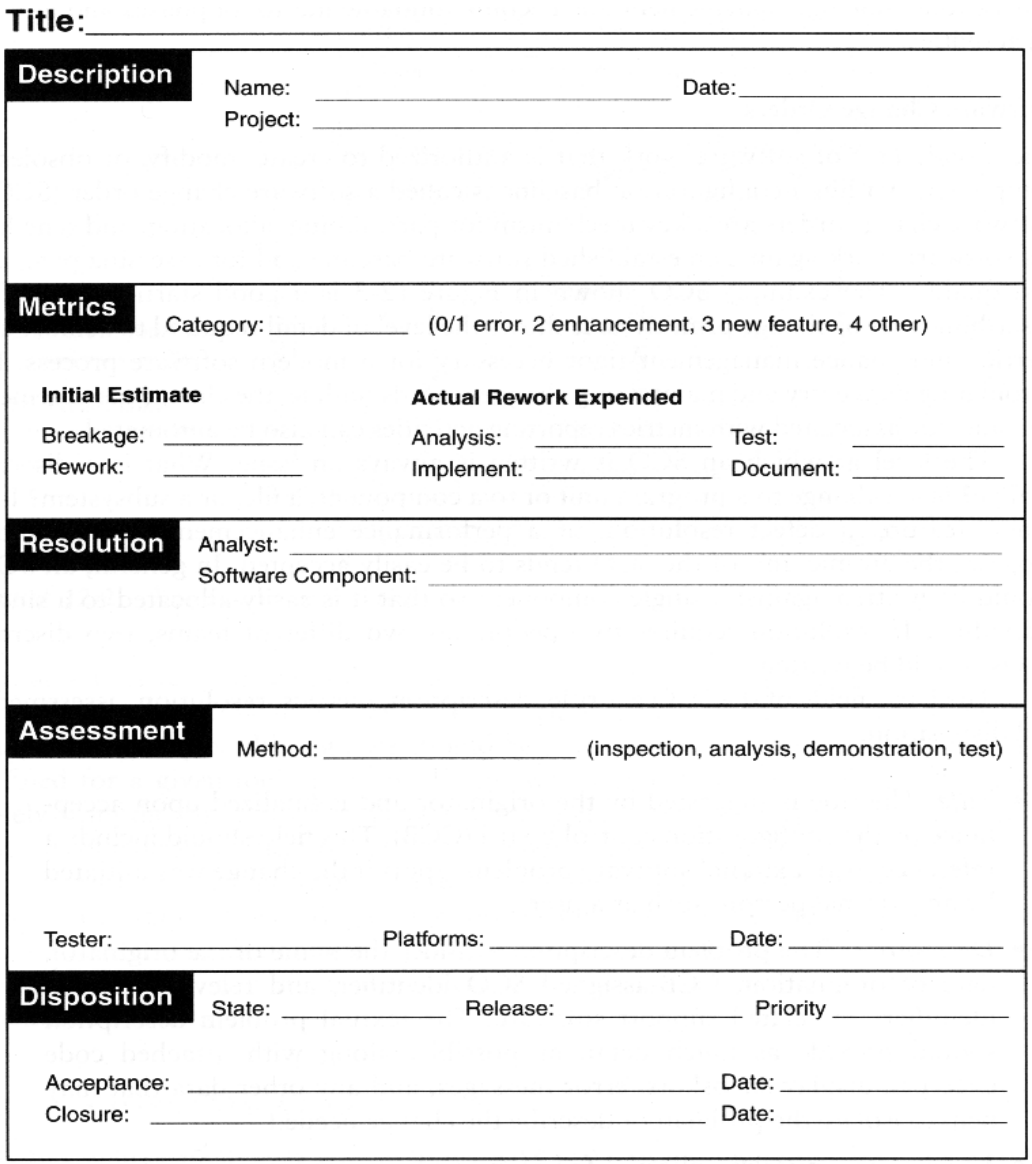
Process Automation



Round-Trip Engineering



Software Change Orders



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: _____

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

Indicators

- **Management**
  - Work and Progress
  - Budgeted Cost and Expenditure
  - Staffing and Team Dynamics
- **Quality**
  - Change Traffic and Stability
  - Breakage and Modularity
  - Rework and Adaptability
  - MTBF and Maturity

Tailoring the Process

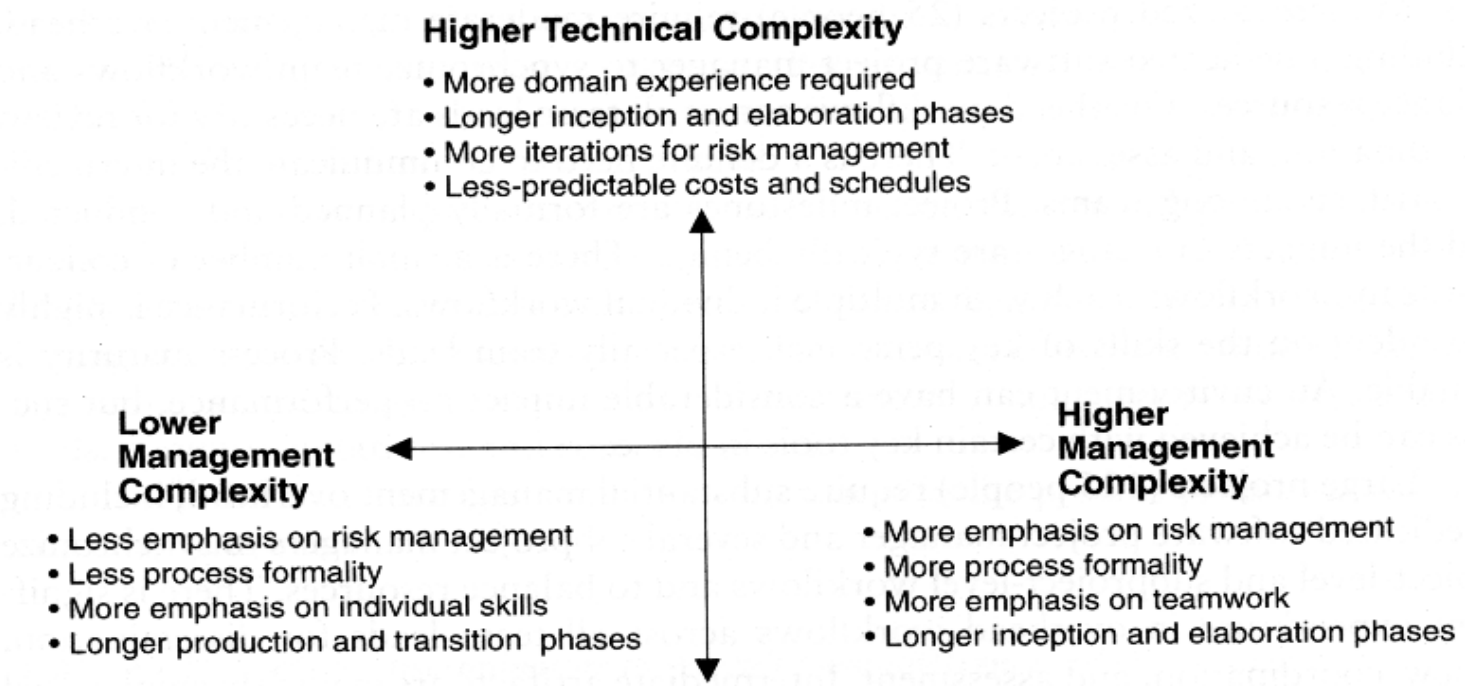


FIGURE 14-2. *Priorities for tailoring the process framework*

Project Scale

TABLE 14-1. *Process discriminators that result from differences in project size*

PROCESS PRIMITIVE	SMALLER TEAM	LARGER TEAM
Life-cycle phases	Weak boundaries between phases	Well-defined phase transitions to synchronize progress among concurrent activities
Artifacts	Focus on technical artifacts Few discrete baselines Very few management artifacts required	Change management of technical artifacts, which may result in numerous baselines Management artifacts important
Workflow effort allocations	More need for generalists, people who perform roles in multiple workflows	Higher percentage of specialists More people and teams focused on a specific workflow
Checkpoints	Many informal events for maintaining technical consistency No schedule disruption	A few formal events Synchronization among teams, which can take days
Management discipline	Informal planning, project control, and organization	Formal planning, project control, and organization
Automation discipline	More ad hoc environments, managed by individuals	Infrastructure to ensure a consistent, up-to-date environment available across all teams Additional tool integration to support project control and change control

Process Flexibility

TABLE 14-3. *Process discriminators that result from differences in process flexibility*

PROCESS PRIMITIVE	FLEXIBLE PROCESS	INFLEXIBLE PROCESS
Life-cycle phases	Tolerant of cavalier phase commitments	More credible basis required for inception phase commitments
Artifacts	Changeable business case and vision	Carefully controlled changes to business case and vision
Workflow effort allocations	(insignificant)	Increased levels of management and assessment workflows
Checkpoints	Many informal events for maintaining technical consistency	3 or 4 formal events Synchronization among stakeholder teams, which can impede progress for days or weeks
Management discipline	(insignificant)	More fidelity required for planning and project control
Automation discipline	(insignificant)	(insignificant)

Process Maturity

TABLE 14-4. *Process discriminators that result from differences in process maturity*

PROCESS PRIMITIVE	MATURE, LEVEL 3 OR 4 ORGANIZATION	LEVEL 1 ORGANIZATION
Life-cycle phases	Well-established criteria for phase transitions	(insignificant)
Artifacts	Well-established format, content, and production methods	Free-form
Workflow effort allocations	Well-established basis	No basis
Checkpoints	Well-defined combination of formal and informal events	(insignificant)
Management discipline	Predictable planning Objective status assessments	Informal planning and project control
Automation discipline	Requires high levels of automation for round-trip engineering, change management, and process instrumentation	Little automation or disconnected islands of automation

Process Maturity

TABLE 14-5. *Process discriminators that result from differences in architectural risk*

PROCESS PRIMITIVE	COMPLETE ARCHITECTURE FEASIBILITY DEMONSTRATION	NO ARCHITECTURE FEASIBILITY DEMONSTRATION
Life-cycle phases	More inception and elaboration phase iterations	Fewer early iterations More construction iterations
Artifacts	Earlier breadth and depth across technical artifacts	(insignificant)
Workflow effort allocations	Higher level of design effort Lower levels of implementation and assessment	Higher levels of implementation and assessment to deal with increased scrap and rework
Checkpoints	More emphasis on executable demonstrations	More emphasis on briefings, documents, and simulations
Management discipline	(insignificant)	(insignificant)
Automation discipline	More environment resources required earlier in the life cycle	Less environment demand early in the life cycle

Domain Experience

TABLE 14-6. *Process discriminators that result from differences in domain experience*

PROCESS PRIMITIVE	EXPERIENCED TEAM	INEXPERIENCED TEAM
Life-cycle phases	Shorter engineering stage	Longer engineering stage
Artifacts	Less scrap and rework in requirements and design sets	More scrap and rework in requirements and design sets
Workflow effort allocations	Lower levels of requirements and design	Higher levels of requirements and design
Checkpoints	(insignificant)	(insignificant)
Management discipline	Less emphasis on risk management Less-frequent status assessments needed	More-frequent status assessments required
Automation discipline	(insignificant)	(insignificant)

Example: Small vs. Large

TABLE 14-7. *Schedule distribution across phases for small and large projects*

	ENGINEERING		PRODUCTION	
DOMAIN	INCEPTION	ELABORATION	CONSTRUCTION	TRANSITION
Small commercial project	10%	20%	50%	20%
Large, complex project	15%	30%	40%	15%

TABLE 14-8. *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	Assessment	Environment
6	Management	Implementation
7	Environment	Deployment

… Example: Small vs. Large

TABLE 14-9. *Differences in artifacts between small and large projects*

ARTIFACT	SMALL COMMERCIAL PROJECT	LARGE, COMPLEX PROJECT
Work breakdown structure	1-page spreadsheet with 2 levels of WBS elements	Financial management system with 5 or 6 levels of WBS elements
Business case	Spreadsheet and short memo	3-volume proposal including technical volume, cost volume, and related experience
Vision statement	10-page concept paper	200-page subsystem specification
Development plan	10-page plan	200-page development plan
Release specifications and number of releases	3 interim release specifications	8 to 10 interim release specifications
Architecture description	5 critical use cases, 50 UML diagrams, 20 pages of text, other graphics	25 critical use cases, 200 UML diagrams, 100 pages of text, other graphics
Software	50,000 lines of Visual Basic code	300,000 lines of C++ code
Release description	10-page release notes	100-page summary
Deployment	User training course Sales rollout kit	Transition plan Installation plan
User manual	On-line help and 100-page user manual	200-page user manual
Status assessment	Quarterly project reviews	Monthly project management reviews

Modern Project Profiles

TABLE 15-1. *Differences in workflow cost allocations between a conventional process and a modern process*

SOFTWARE ENGINEERING WORKFLOWS	CONVENTIONAL PROCESS EXPENDITURES	MODERN PROCESS EXPENDITURES
Management	5%	10%
Environment	5%	10%
Requirements	5%	10%
Design	10%	15%
Implementation	30%	25%
Assessment	40%	25%
Deployment	5%	5%
Total	100%	100%

Top 10 Management Principles

- Base the process on an architecture-first approach
- Establish an iterative life-cycle process
- Transition design methods to emphasize component-based development
- Establish a change management environment
- Enhance change freedom through tools that support round-trip engineering
- Capture design artifacts in rigorous, model-based notation
- Instrument the process for objective quality control and progress assessment
- Use a demonstration-based approach to assess intermediate artifacts
- Plan intermediate release in groups of usage scenario with evolving levels of details
- Establish a configurable process that is economically scalable

Balanced Principles

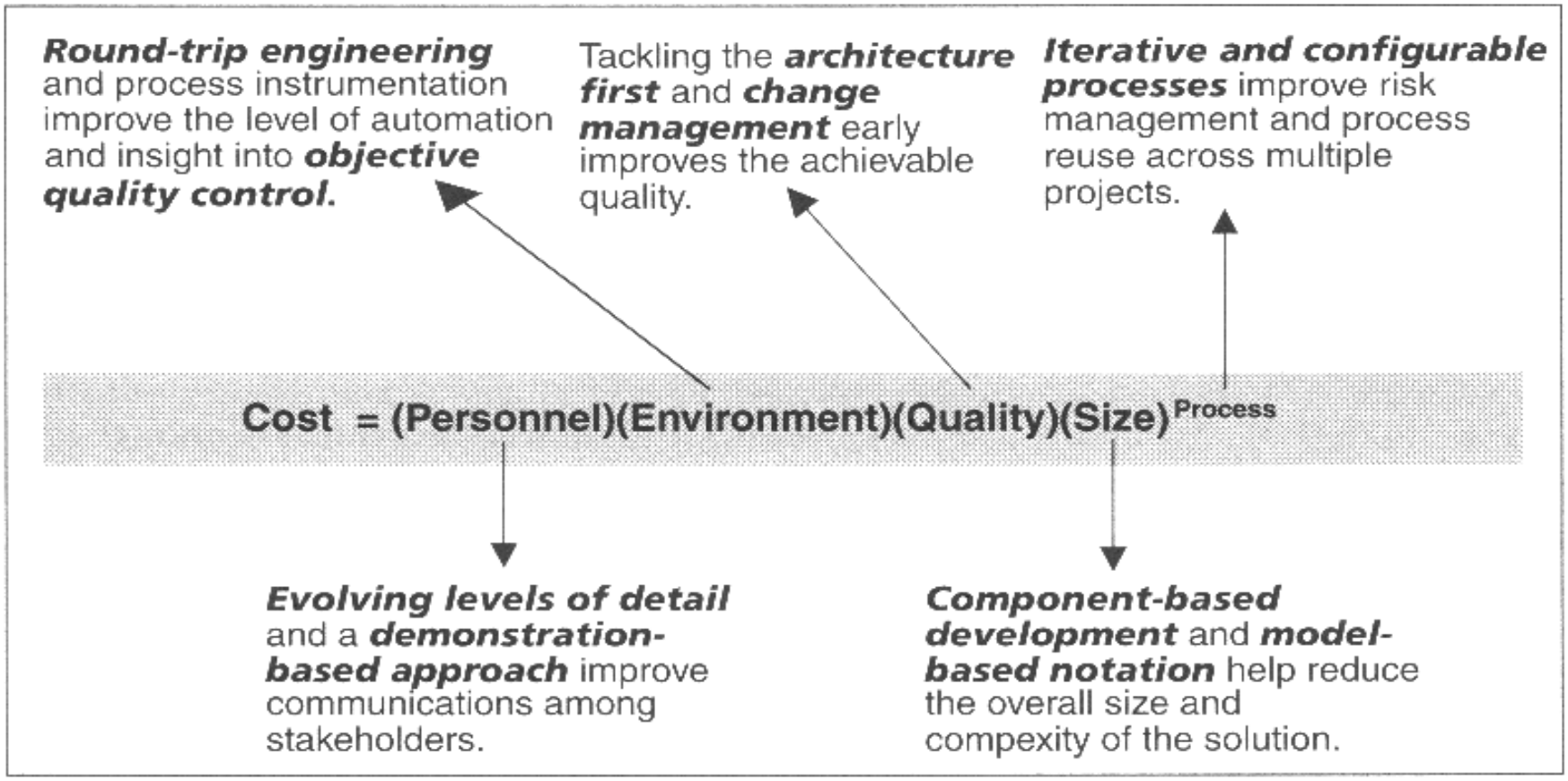


FIGURE 15-4. *Balanced application of modern principles to achieve economic results*

Modern Software Economics

- Finding and fixing a software problem after delivery costs 100 times more
- You can compress software development schedules 25% of nominal, but not more
- For every $1 you spend on development, you will spend $2 on maintenance
- Software development and maintenance costs are primarily a function of the number of source lines of code
- Variations among people account for the biggest differences in software productivity
- The overall ratio of software to hardware costs is still growing
- Only about 15% of software development effort is devoted to programming
- Software systems and products typically costs 3 times as much per SLOC as individual software programs
- Walkthrough catch 60% of errors
- 80% of the contribution comes from 20% of the contributors

Culture Shifts

- Lower level and mid-level managers are performers
- Requirements and designs are fluid and tangible
- Ambitious demonstrations are encouraged
- Good and bad project performance is much more obvious earlier in the life cycle
- Early increments will be immature
- Artifacts are less important early, more important later
- Real issues are surfaced and resolved systematically
- Quality assurance is everyone’s job, not a separate discipline
- Performance issues arise early in the life cycle
- Investments is automation are necessary
- Good software organizations should be more profitable