



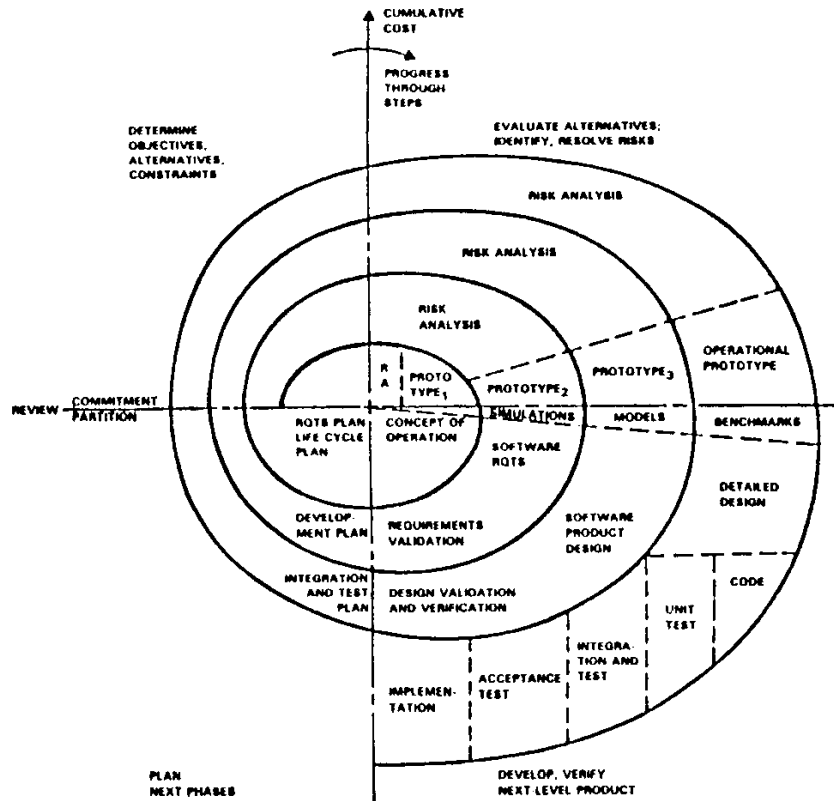
Spiral Development: Experience, Principles, and Refinements

**Barry Boehm, USC
LA SPIN Presentation
June 28, 2000**

**boehm@sunset.usc.edu
<http://sunset.usc.edu/MBASE>**

Spiral Model and MBASE

- Spiral experience
 - Critical success factors
 - Invariants and variants
 - Stud poker analogy
- Spiral refinements
 - WinWin spiral
 - Life cycle anchor points
 - MBASE
- Recent Developments





“Spiral Development Model:” Candidate Definition

The spiral development model is a risk-driven process model generator. It is used to guide multi-stakeholder concurrent engineering of software-intensive systems. It has two main distinguishing features. One is a cyclic approach for incrementally growing a system’s degree of definition and implementation. The other is a set of anchor point milestones for ensuring stakeholder commitment to feasible and mutually satisfactory system solutions.



Spiral Invariants and Variants - 1

- Critical success factor examples

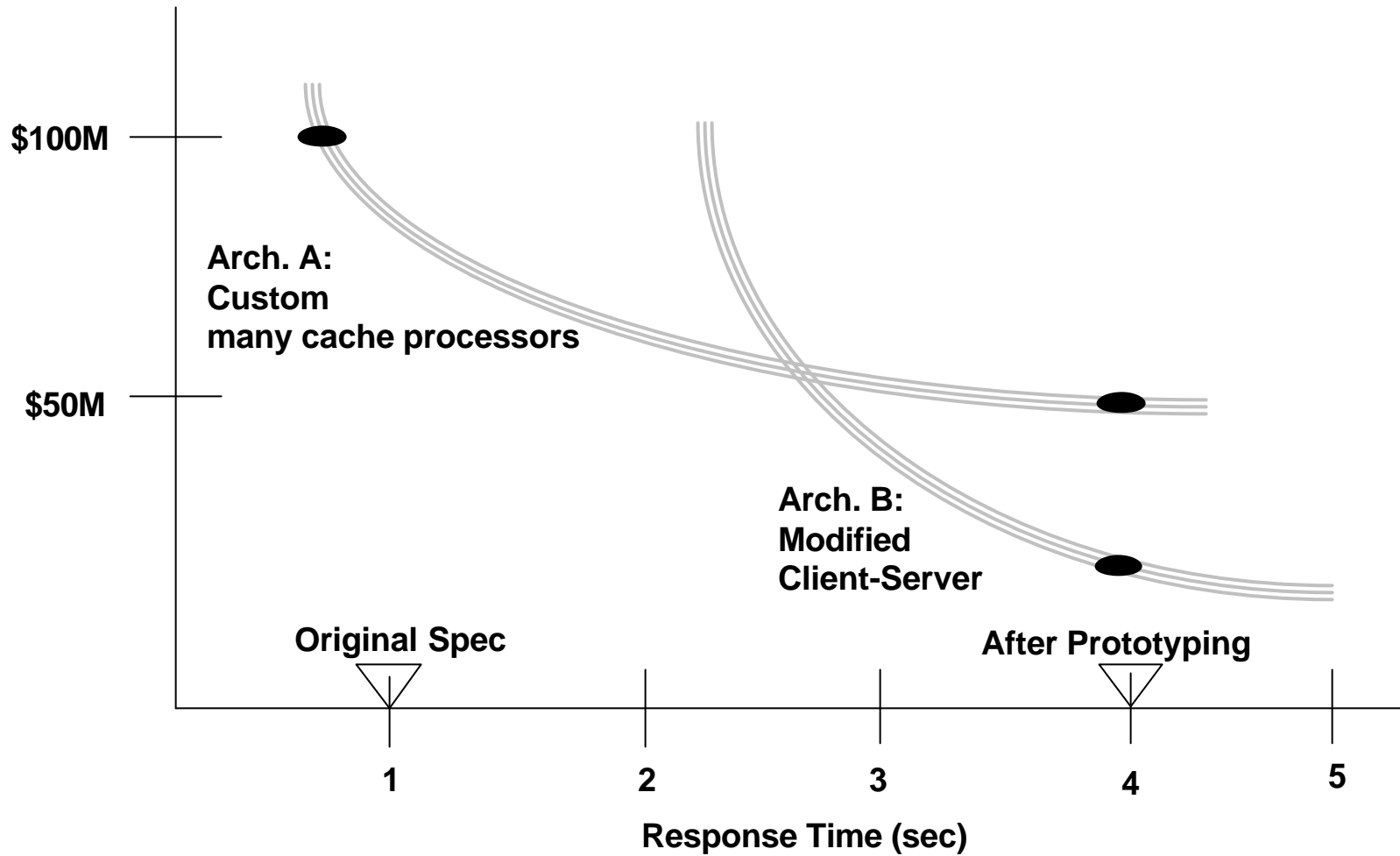
Invariants	Why Invariant	Variants
<p>1. Concurrent rather than sequential determination of artifacts (OCD, Rqts, Design, Code, Plans) in each spiral cycle.</p>	<ul style="list-style-type: none"> Avoids premature sequential commitments to Rqts, Design, COTS, combination of cost/schedule performance - 1 sec. response time 	<p>1a. Relative amount of each artifact developed in each cycle.</p> <p>1b. Number of concurrent mini-cycles in each cycle.</p>
<p>2. Consideration in each cycle of critical-stakeholder objectives and constraints, product and process alternatives, risk identification and resolution, stakeholder review and commitment to proceed.</p> <p>3. Level of effort on each activity within each cycle driven by risk considerations.</p>	<ul style="list-style-type: none"> Avoids commitment to stakeholder-unacceptable or overly risky alternatives. Avoids wasted effort in elaborating unsatisfactory alternatives. - Mac-based COTS Determines “how much is enough” of each activity: domain engr., prototyping, testing, CM, etc. - Pre-ship testing Avoids overkill or belated risk resolution. 	<p>2a. Choice of risk resolution techniques: prototyping, simulation, modeling, benchmarking, reference checking, etc.</p> <p>2b. Level of effort on each activity within each cycle.</p> <p>3a. Choice of methods used to pursue activities: MBASE/ WinWin, Rational USDP, JAD, QFD, ESP, ...</p> <p>3b. Degree of detail of artifacts produced in each cycle.</p>



Spiral Invariant 1: Concurrent Determination of Key Artifacts (Ops Concept, Rqts, Design, Code, Plans)

- **Why invariant**
 - Avoids premature sequential commitments to system requirements, design, COTS, combination of cost/schedule/ performance
 - 1 sec response time
- **Variants**
 - 1a. Relative amount of each artifact developed in each cycle.
 - 1b. Number of concurrent mini-cycles in each cycle.
- **Models excluded**
 - Incremental sequential waterfalls with high risk of violating waterfall model assumptions

Sequential Engineering Buries Risk





Waterfall Model Assumptions

- 1. The requirements are knowable in advance of implementation.**
- 2. The requirements have no unresolved, high-risk implications**
 - e.g., risks due to COTS choices, cost, schedule, performance, safety, security, user interfaces, organizational impacts
- 3. The nature of the requirements will not change very much**
 - During development; during evolution
- 4. The requirements are compatible with all the key system stakeholders' expectations**
 - e.g., users, customer, developers, maintainers, investors
- 5. The right architecture for implementing the requirements is well understood.**
- 6. There is enough calendar time to proceed sequentially.**



Spiral Invariant 2: Each cycle does objectives, constraints, alternatives, risks, review, commitment to proceed

- **Why invariant**
 - **Avoids commitment to stakeholder-unacceptable or overly risky alternatives.**
 - **Avoids wasted effort in elaborating unsatisfactory alternatives.**
 - **Windows-only COTS**
- **Variants**
 - 2a. Choice of risk resolution techniques: prototyping, simulation, modeling, benchmarking, reference checking, etc.**
 - 2b. Level of effort on each activity within each cycle.**
- **Models excluded**
 - **Sequential phases with key stakeholders excluded**

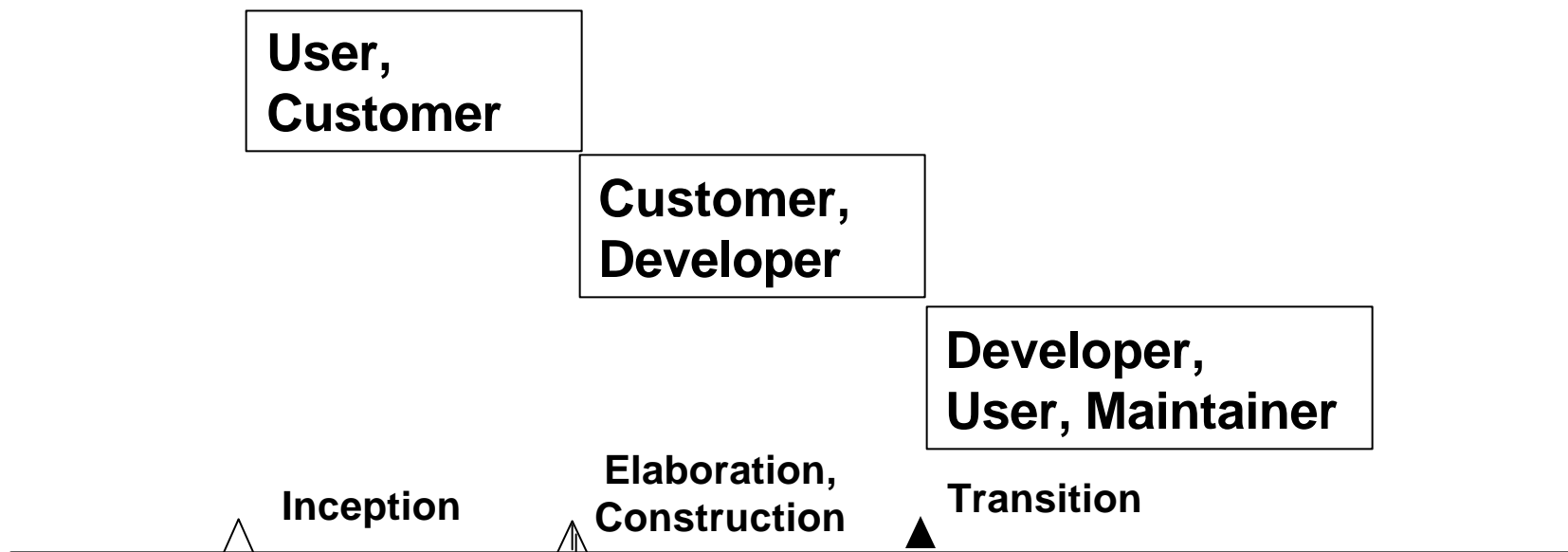


Windows-Only COTS Example: Digital Library Artifact Viewer

- **Great prototype using ER Mapper**
 - Tremendous resolution
 - Incremental-resolution artifact display
 - Powerful zoom and navigation features
- **Only runs well on Windows**
 - Mac, Unix user communities forced to wait
 - Overoptimistic assumptions on length of wait
- **Eventual decision to drop ER Mapper**



Models Excluded: Sequential Phases Without Key Stakeholders



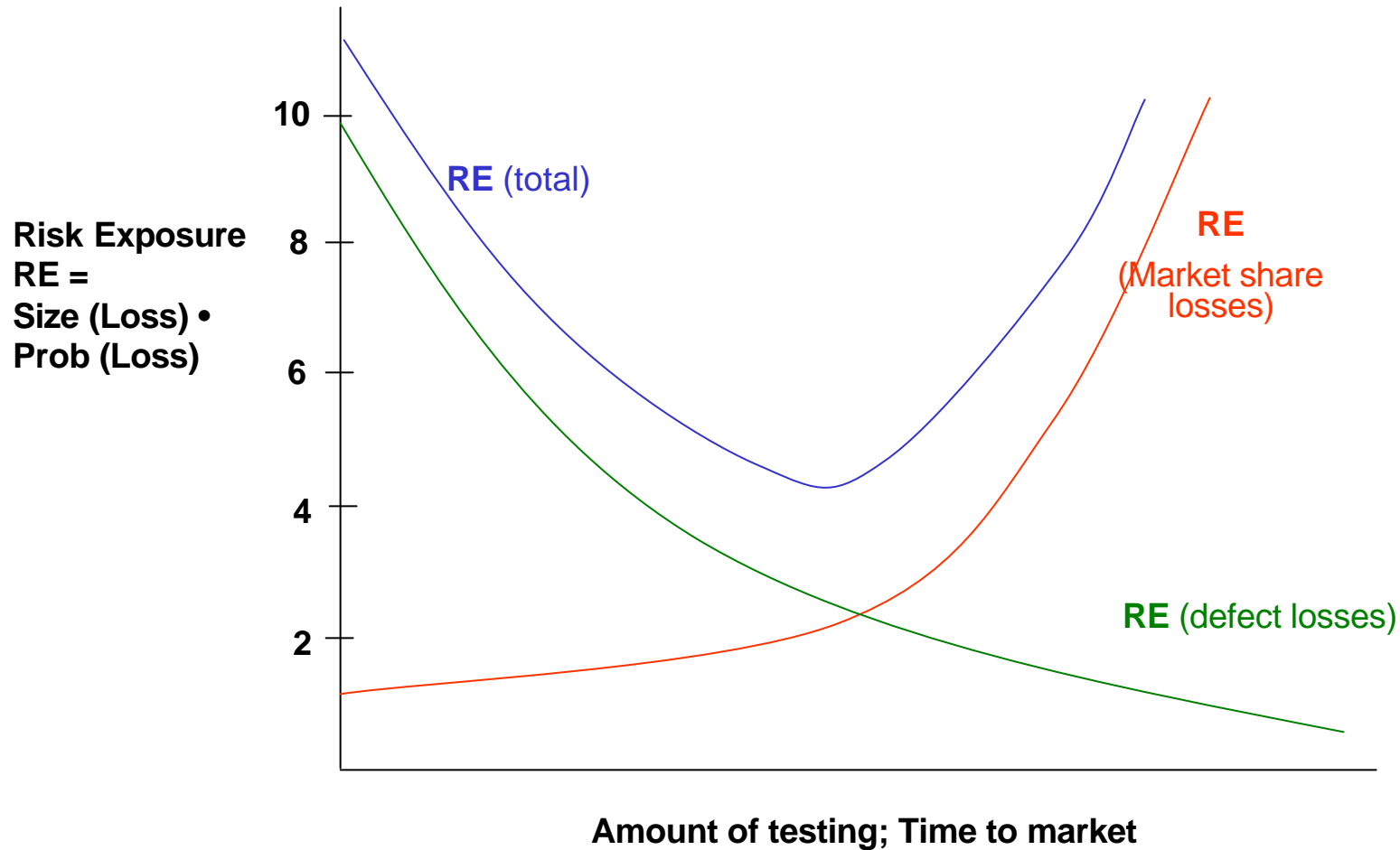
- High risk of win-lose even with spiral phases
 - Win-lose evolves into lose-lose
- Key criteria for IPT members (AFI 63-123)
 - Representative, empowered, knowledgeable, collaborative, committed



Spiral Invariant 3: Level of Effort Driven by Risk Considerations

- **Why invariant**
 - Determines ‘how much is enough’ of each activity: domain engr., prototyping, testing, CM, etc.
 - Pre-ship testing
 - Avoids overkill or belated risk resolution.
- **Variants**
 - 3a. Choice of methods used to pursue activities: MBASE/WinWin, Rational RUP, JAD, QFD, ESP, . . .
 - 3b. Degree of detail of artifacts produced in each cycle.
- **Models excluded**
 - Risk-insensitive evolutionary or incremental development

Pre-Ship Test Risk Exposure





Spiral Invariants and Variants - 2

Invariants	Why Invariant	Variants
<p>4. Degree of detail of artifacts produced in each cycle driven by risk considerations.</p>	<ul style="list-style-type: none"> Determines “how much is enough” of each artifact (OCD, Rqts, Design, Code, Plans) in each cycle. Avoids overkill or belated risk resolution 	<p>4a. Choice of artifact representations (SA/SD, UML, MBASE, formal specs, programming languages, etc.)</p>
<p>5. Managing stakeholder life-cycle commitments via the LCO, LCA, and IOC Anchor Point milestones (getting engaged, getting married, having your first child),</p>	<ul style="list-style-type: none"> Avoids analysis paralysis, unrealistic expectations, requirements creep, architectural drift, COTS shortfalls and incompatibilities, unsustainable architectures, traumatic cutovers, useless systems. 	<p>5a. Number of spiral cycles or increments between anchor points.</p> <p>5b. Situation-specific merging of anchor point milestones.</p>
<p>6. Emphasis on system and life cycle activities and artifacts rather than software and initial development activities and artifacts.</p>	<ul style="list-style-type: none"> Avoids premature suboptimization on hardware, software, or initial development considerations. 	<p>6a. Relative amount of hardware and software determined in each cycle.</p> <p>6b. Relative amount of capability in each life cycle increment.</p> <p>6c. Degree of productization (alpha, beta, shrink-wrap, etc.) of each life cycle increment.</p>



Spiral Invariant 4: Degree of Detail Driven by Risk Considerations

- **Why invariant**
 - Determines “how much is enough” of each artifact (OCD, Rqts, Design, Code, Plans) in each cycle.
 - Screen layout rqts.
 - Avoids overkill or belated risk resolution.
- **Variants**
 - 4a. Choice of artifact representations (SA/SD, UML, MBASE, formal specs, programming languages, etc.)
- **Models excluded**
 - Complete, consistent, traceable, testable requirements specification for systems involving significant levels of GUI, COTS, or deferred decisions



Risk-Driven Specifications

- If it's risky not to specify precisely, Do
 - Hardware-software interface
 - Prime-subcontractor interface
- If it's risky to specify precisely, Don't
 - GUI layout
 - COTS behavior



Spiral Invariant 5: Use of LCO, LCA, IOC, Anchor Point Milestones

- **Why invariant**
 - Avoids analysis paralysis, unrealistic expectations, requirements creep, architectural drift, COTS shortfalls and incompatibilities, unsustainable architectures, traumatic cutovers, useless systems.
- **Variants**
 - 5a. Number of spiral cycles or increments between anchor points.
 - 5b. Situation-specific merging of anchor point milestones
 - Can merge LCO and LCA when adopting an architecture from mature 4GL, product line
- **Models excluded**
 - Evolutionary or incremental development with no life cycle architecture



Life Cycle Anchor Points

- **Common System/Software stakeholder commitment points**
 - Defined in concert with Government, industry affiliates
 - Coordinated with the Rational Unified Process
- **Life Cycle Objectives (LCO)**
 - Stakeholders' commitment to support architecting
 - Like getting engaged
- **Life Cycle Architecture (LCA)**
 - Stakeholders' commitment to support full life cycle
 - Like getting married
- **Initial Operational Capability (IOC)**
 - Stakeholders' commitment to support operations
 - Like having first child



Win Win Spiral Anchor Points

(Risk-driven level of detail for each element)

Milestone Element	Life Cycle Objectives (LCO)	Life Cycle Architecture (LCA)
Definition of Operational Concept	<ul style="list-style-type: none"> • Top-level system objectives and scope <ul style="list-style-type: none"> - System boundary - Environment parameters and assumptions - Evolution parameters • Operational concept <ul style="list-style-type: none"> - Operations and maintenance scenarios and parameters - Organizational life-cycle responsibilities (stakeholders) 	<ul style="list-style-type: none"> • Elaboration of system objectives and scope of increments • Elaboration of operational concept by increment
System Prototype(s)	<ul style="list-style-type: none"> • Exercise key usage scenarios • Resolve critical risks 	<ul style="list-style-type: none"> • Exercise range of usage scenarios • Resolve major outstanding risks
Definition of System Requirements	<ul style="list-style-type: none"> • Top-level functions, interfaces, quality attribute levels, including: <ul style="list-style-type: none"> - Growth vectors and priorities - Prototypes • Stakeholders' concurrence on essentials 	<ul style="list-style-type: none"> • Elaboration of functions, interfaces, quality attributes, and prototypes by increment <ul style="list-style-type: none"> - Identification of TBD's (to-be-determined items) • Stakeholders' concurrence on their priority concerns
Definition of System and Software Architecture	<ul style="list-style-type: none"> • Top-level definition of at least one feasible architecture <ul style="list-style-type: none"> - Physical and logical elements and relationships - Choices of COTS and reusable software elements • Identification of infeasible architecture options 	<ul style="list-style-type: none"> • Choice of architecture and elaboration by increment <ul style="list-style-type: none"> - Physical and logical components, connectors, configurations, constraints - COTS, reuse choices - Domain-architecture and architectural style choices • Architecture evolution parameters
Definition of Life-Cycle Plan	<ul style="list-style-type: none"> • Identification of life-cycle stakeholders <ul style="list-style-type: none"> - Users, customers, developers, maintainers, interoperations, general public, others • Identification of life-cycle process model <ul style="list-style-type: none"> - Top-level stages, increments • Top-level WWWWWHH* by stage 	<ul style="list-style-type: none"> • Elaboration of WWWWWHH* for Initial Operational Capability (IOC) <ul style="list-style-type: none"> - Partial elaboration, identification of key TBD's for later increments
Feasibility Rationale	<ul style="list-style-type: none"> • Assurance of consistency among elements above <ul style="list-style-type: none"> - via analysis, measurement, prototyping, simulation, etc. • Business case analysis for requirements, feasible architectures 	<ul style="list-style-type: none"> • Assurance of consistency among elements above <ul style="list-style-type: none"> - All major risks resolved or covered by risk management

*WWWWHH: Why, What, When, Who, Where, How, How Much



Initial Operational Capability (IOC)

- **Software preparation**
 - Operational and support software
 - Data preparation, COTS licenses
 - Operational readiness testing
- **Site preparation**
 - Facilities, equipment, supplies, vendor support
- **User, operator, and maintainer preparation**
 - Selection, teambuilding, training



Evolutionary Development Assumptions

- 1. The initial release is sufficiently satisfactory to key system stakeholders that they will continue to participate in its evolution.**
- 2. The architecture of the initial release is scalable to accommodate the full set of system life cycle requirements (e.g., performance, safety, security, distribution, localization).**
- 3. The operational user organizations are sufficiently flexible to adapt to the pace of system evolution**
- 4. The dimensions of system evolution are compatible with the dimensions of evolving-out the legacy systems it is replacing.**

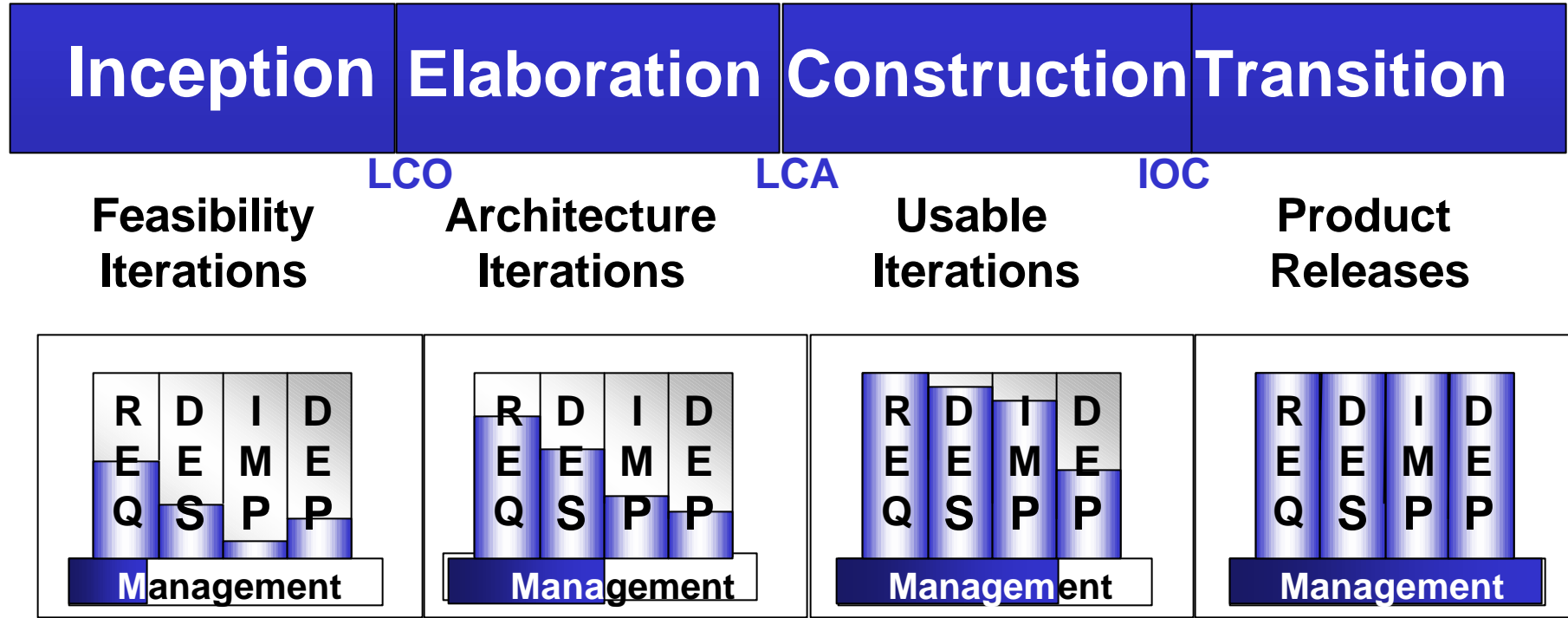
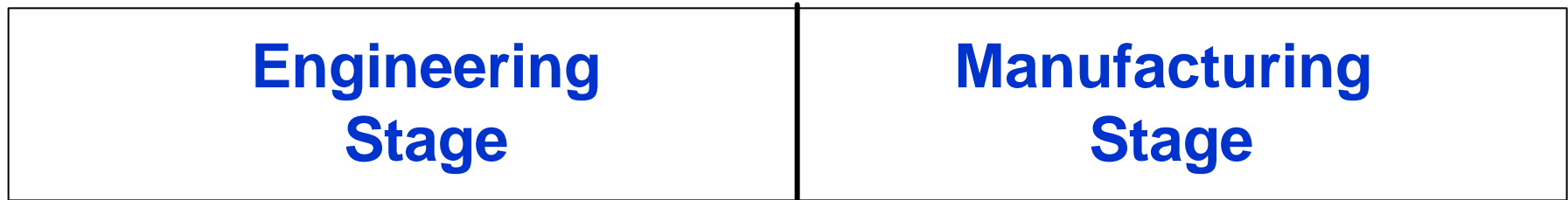


Spiral Model and Incremental Commitment: Stud Poker Analogy

- **Evaluate alternative courses of action**
 - **Fold: save resources for other deals**
 - **Ante: buy at least one more round**
- **Using incomplete information**
 - **Hole cards: competitive situation**
 - **Rest of deck: chance of getting winner**
- **Anticipating future possibilities**
 - **Likelihood that next round will clarify outcome**
- **Commit incrementally rather than all at once**
 - **Challenge: DoD POM process makes this hard to do**



Anchor Points and Rational RUP Phases



Spiral Model Refinements

- Where do objectives, constraints, alternatives come from?

- Win Win extensions

- Lack of intermediate milestones

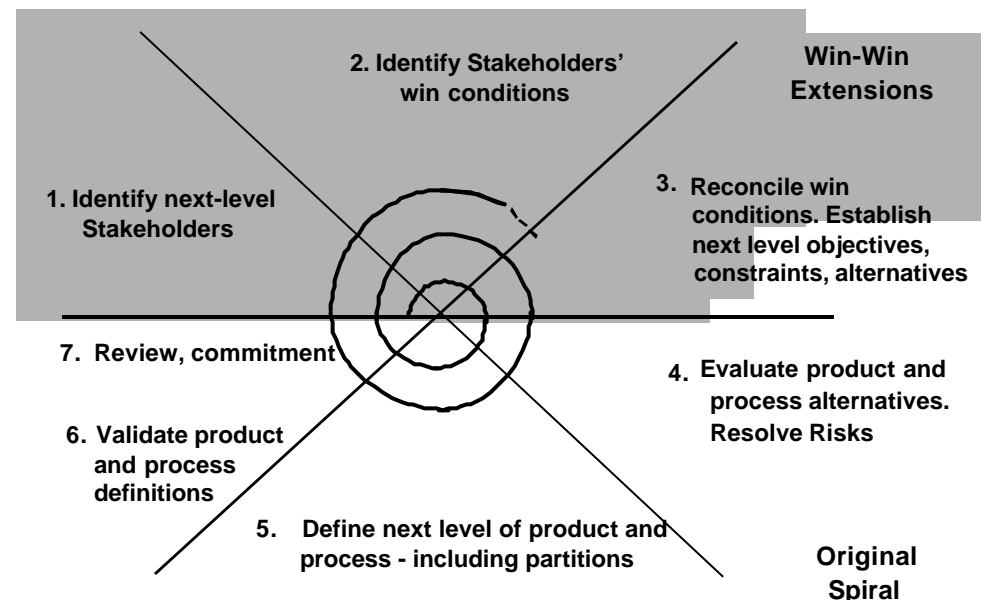
- Anchor Points: LCO, LCA, IOC

- Concurrent-engineering spirals between anchor points

- Need to avoid model clashes, provide more specific guidance

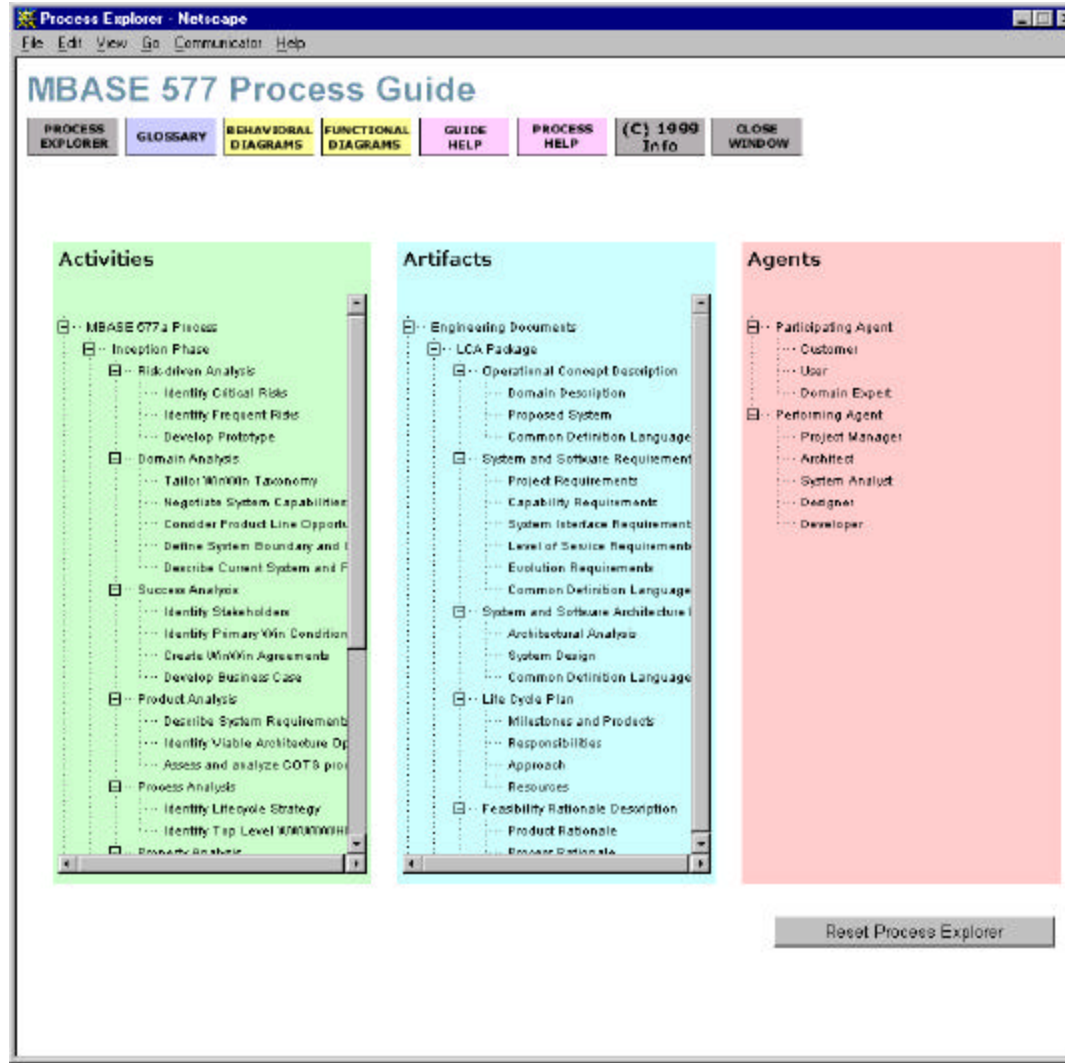
- MBASE

The WinWin Spiral Model





MBASE Electronic Process Guide (1)





MBASE Electronic Process Guide (2)

MBASE 577a Process

Overview

The process followed by students of CS 577a for Digital Library projects

Purpose

The objectives for the MBASE 577a process are:

- To define a life cycle process for the students of CS 577a for use in Digital Library and similar projects
- To provide guidance to process enactors about the inter-dependence of MBASE process elements

Decomposition

The activity MBASE 577a Process is decomposed into the following:

- Inception Phase
- Elaboration Phase
- Record Project Effort

Description

Model-Based (System) Architecture and Software Engineering (MBASE) is an approach for developing software intensive systems. The MBASE approach integrates the four common development models (success, product, process and property) around the creation and use of a software architecture package. Using MBASE, model clashes can be recognized and reconciled as a matter of fact rather than after the fact.

The MBASE 577 process is a variant of the MBASE framework used by students in the course CS 577 offered at USC. The process consists of four distinct phases: Inception, Elaboration, Construction and Transition. Each phase is completed with a commitment from the stakeholders during a review.

The tasks to be performed during the MBASE 577 process are:

- Identifying and resolving the critical risks
- Analyzing the problem domain
- Identifying the feasible architecture
- Developing a prototype to test the validity of the architecture and satisfaction of stakeholders' concerns
- Understanding the requirements of the life cycle and obtaining concurrence from the system stakeholders

Tools and Techniques

Electronic Process Guide for MBASE

Guidelines for Model-Based Architecture and Software Engineering (MBASE) deliverables: Inception and Elaboration

Pitfalls

Operational Concept Description

Overview

Provides the overall context of the proposed system and its operational concept

Purpose

- Describe the overall context of the system to be developed, why it's being built, what exists now, and where the project is starting from
- Describe to the stakeholders of the system to be developed ('developed' is meant to include such terms as 'enhanced', 'updated', 're-engineered', 'automated'), how the system will work in practice once it is deployed
- Enable the operational stakeholders to evolve knowledgeably from their current operational concept to the new operational concept, and to collaboratively adapt the operational concept as developments arise, to make clear the value of developing the new system

Owner

The artifact Operational Concept Description is owned by the agent System Analyst...

Decomposition

The artifact Operational Concept Description is decomposed into the following:

- Domain Description
- Proposed System
- Common Definition Language for Domain

Description

OCDA Audience and Participants

- Audience
- Customer for Domain Description Domain
- Domain Expert for System Analysis
- Use language and define CDL appropriately for intended audience
- Participants
- Same stakeholders as WinWin negotiation
- Establish concept of operation agreed on by all stakeholders

OCDA High-Level Dependencies

- WinWin Negotiations Give
- System Responsibilities
- Changes Considered But Not Included
- Domain Description Terms
- Project Goals, Quality Goals
- OCDA Yields
- Project, System and Quality Reqs for SSRD
- Domain Description and Initial Analysis for SSAD
- Stakeholder and Organizational Responsibilities

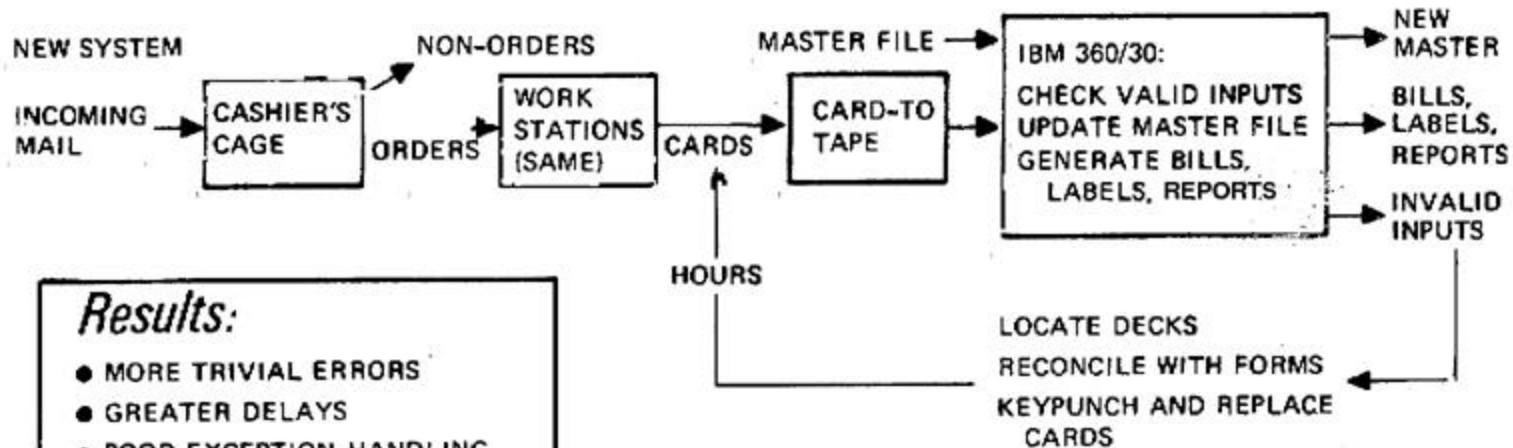
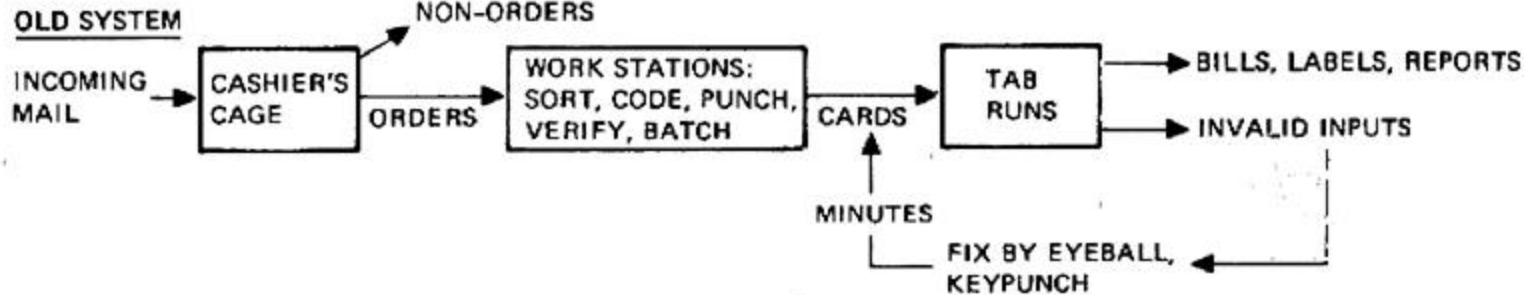


Spiral Invariant 6: Emphasis on System and Life Cycle Activities and Artifacts

- **Why invariant**
 - Avoids premature suboptimization on hardware, software, or development considerations.
 - Scientific American
- **Variants**
 - 6a. Relative amount of hardware and software determined in each cycle.
 - 6b. Relative amount of capability in each life cycle increment
 - 6c. Degree of productization (alpha, beta, shrink-wrap, etc.) of each life cycle increment.
- **Models excluded**
 - Purely logical object-oriented methods
 - Insensitive to operational, performance, cost risks

Problems With Programming-Oriented Top-Down Development

"SCIENTIFIC AMERICAN" SUBSCRIPTION PROCESSING



Results:

- MORE TRIVIAL ERRORS
- GREATER DELAYS
- POOR EXCEPTION-HANDLING
- CUMBERSOME INPUT CONTROLS
- MORE LABOR-INTENSIVE

TRW



Summary: Hazardous Spiral Look-Alikes

- **Incremental sequential waterfalls with significant COTS, user interface, or technology risks**
- **Sequential spiral phases with key stakeholders excluded from phases**
- **Risk-insensitive evolutionary or incremental development**
- **Evolutionary development with no life-cycle architecture**
- **Insistence on complete specs for COTS, user interface, or deferred-decision situations**
- **Purely logical object-oriented methods with operational, performance, or cost risks**
- **Impeccable spiral plan with no commitment to managing risks**



Summary: Successful Spiral Examples

- **Rapid commercial: C-Bridge's RAPID process**
- **Large commercial: AT&T/Lucent/Telcordia spiral extensions**
- **Commercial hardware-software: Xerox Time-to-Market process**
- **Large aerospace: TRW CCPDS-R**
- **Variety of projects: Rational Unified Process, SPC Evolutionary Spiral Process, USC MBASE approach**



Recent Developments

- **Workshop contents available on SEI web site**
 - <http://www.sei.cmu.edu/cbs/spiral> 2000
 - SEI Tech Reports forthcoming
- **Followon Workshop September 13-15, 2000**
 - DUSD/S&T sponsor; n Washington DC
 - SEI lead organizer, in collaboration with USC
- **New DoDD 5000-series acquisition directives due July-Aug**
 - Emphasize evolutionary acquisition; spiral
- **ASD/C3I Panel report on evolutionary acquisition**
 - recommends use of spiral development



Hands-on Tutorials at USC, July 25-27

- **Easy WinWin: July 25-26**
 - **Objective:** Learn to initiate and conduct Easy WinWin requirements negotiations in your organization
 - **Mode:** Hands-on at individual workstation
 - **Registration:** Free to USC-CSE Affiliates; \$500. otherwise
- **MBASE* Electronic Process Guide: July 27**
 - **Objective:** Learn to use MBASE and its EPG
 - **Mode:** Lecture in morning; hands-on in afternoon
 - **Registration:** Free to USC-CSE Affiliates: \$250. otherwise
- **Details at <http://sunset.usc.edu/upcoming-events>**

*MBASE: Model-Based (System) Architecting and Software Engineering



References

(MBASE material available at <http://sunset.usc.edu/MBASE>)

[Boehm, 1988]. “A Spiral Model of Software Development and Enhancement,” Computer, May 1988, pp. 61-72.

[Boehm, 1989]. “Software Risk Management”, IEEE Computer Society Press, 1989.

[Boehm-Ross, 1989]. “Theory W Software Project Management: Principles and Examples” IEEE Trans. Software Engr., July 1989.

[Boehm-Bose, 1994]. “A Collaborative Spiral Software Process Model Based on Theory W,” Proceedings, ICSP 3, IEEE, Reston, Va. October 1994.

[Boehm-Port, 1999b]. “When Models Collide: Lessons from Software Systems Analysis,” IEEE IT Professional, January/February 1999, pp. 49-56.



B. Boehm, D. Port, “Escaping the Software Tar Pit: Model Clashes and How to Avoid Them,” ACM Software Engineering Notes, January, 1999, pp. 36-48.

B. Boehm et al., “Using the Win Win Spiral Model: A Case Study,” IEEE Computer, July 1998, pp. 33-44.

B. Boehm et al., “Developing Multimedia Applications with the WinWin Spiral Model,” Proceedings, ESEC/FSE 97, Springer Verlag, 1997.

M.J. Carr, S.L. Konda, I. Monarch, F.C. Ulrich, and C.F. Walker, “Taxonomy-Based Risk Identification,” CMU/SEI-93-TR-06, Software Engineering Institute, 1993

R.N. Charette, Software Engineering Risk Analysis and Management, McGraw Hill, 1989.

M.A. Cusumano and R. W. Selby, Microsoft Secrets, Free Press, 1995

E. Hall, Managing Risk, Addison Wesley, 1998.

W. E. Royce, Software Project Management: A Unified Framework, Addison Wesley, 1998.

J. Thorp and DMR, The Information Paradox, McGraw Hill, 1998.