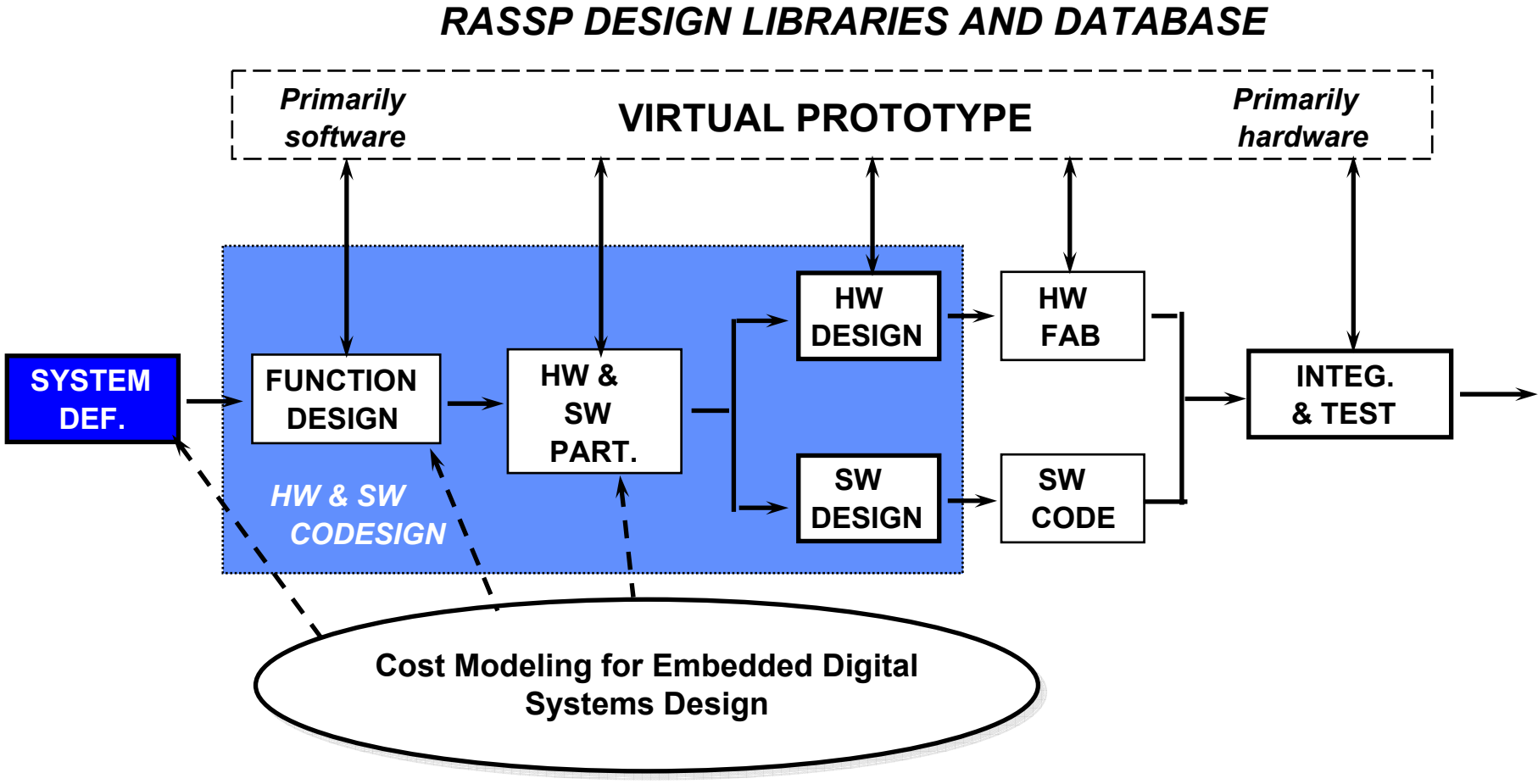
Cost Modeling for Embedded Digital Systems Design

Corso Embedded Systems AA 06-07

Prof. William Fornaciari
Politecnico di Milano – Dipartimento di Elettronica e Informazione
william.fornaciari@polimi.it
www.elet.polimi.it/~fornacia

Rapid Prototyping Design Process



Goals

- Establish the importance of system engineering constraints on embedded system design.
- Survey the methods of cost estimation and modeling for hardware and software design, implementation, and test.
- Establish a cost-modeling based design methodology for embedded systems using cost estimators and performance modeling.
- Understand the benefits of cost modeling through application to an embedded system case study.

Module Outline

- **Introduction to Cost Modeling-Based Embedded Systems Design**
  - **Limitations of Typical Design Process**
  - **Effect of HW Resource Constraints HW/SW Prototyping Costs/Schedule**
  - **Effect of System Development Time on Expected Revenue**
  - **Cost Modeling in the Early Stages of Design**
- **Software Cost Estimation Process**
  - **Software Life Cycle Models**
  - **Basic Steps in the Software Cost Estimating Process**

Module Outline (Cont.)

- Parametric Software Cost Models
  - COCOMO
  - REVIC
  - COCOMO 2.0
- Parametric Hardware Cost Models
  - Full Custom
  - Gate Array
  - FPGA

Module Outline (Cont.)

- Applications of Cost Modeling to the RASSP Design Process
  - Classifications of System-Level Design & Test Methodologies
  - Example Automated Cost-Driven Architecture Selection/Sizing Model
  - Case Study: Synthetic Aperture Radar (SAR) Image Processor
  - Results: Detailed Cost Analysis for SAR Benchmark
- Summary and Major Issues

Module Outline

- **Introduction to Cost Modeling-Based Embedded Systems Design**
- **Software Cost Estimation Process**
- **Parametric Software Cost Models**
- **Parametric Hardware Cost Models**
- **Applications of Cost Modeling to the RASSP Design Process**
- **Summary and Major Issues**

Motivation for Cost Modeling in the Design Process

- **Design Challenge:**
  - **Designers of high-end embedded systems or large volume commercial consumer products are faced with the formidable challenge of rapidly prototyping cost-effective implementations which meet:**
  - **Stringent performance specifications**
    - **Formidable functional and timing requirements**
    - **Tight physical constraints**
- **System *life cycle cost* and *time-to-market cost* are key factors which determine successful products in the competitive electronics marketplace**
  - **These key factors should have a dominant influence on the design of embedded microelectronic systems**

Complexity of the Design Space for Embedded Systems

- A combinatorially significant number of alternatives exist in the implementation of high performance embedded systems

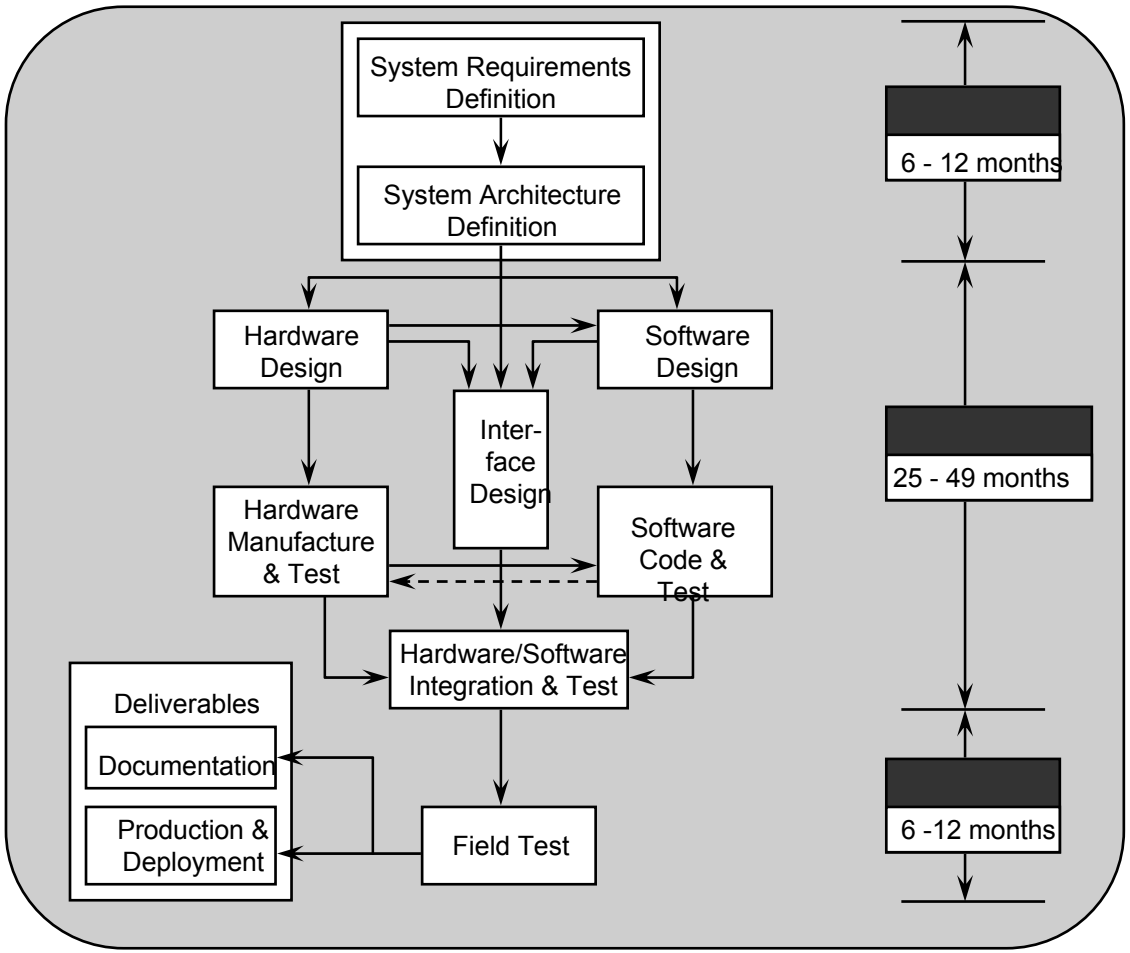
Key Architectural Attributes

- Computational elements
- Communication elements
- Topologies
- Software (Application, Control & Diagnostics)
- Analog Interfaces
- Mechanical Interfaces

Customer Requirements

- Functionality and Performance
- Deployment Schedule
- Interfaces and Packaging
- Size, Volume, and Weight
- Power
- Environment
- Cost
- Software and Hardware Modularity
- Security
- Scalability
- Reliability and Maintainability

Typical Prototyping Process Flow

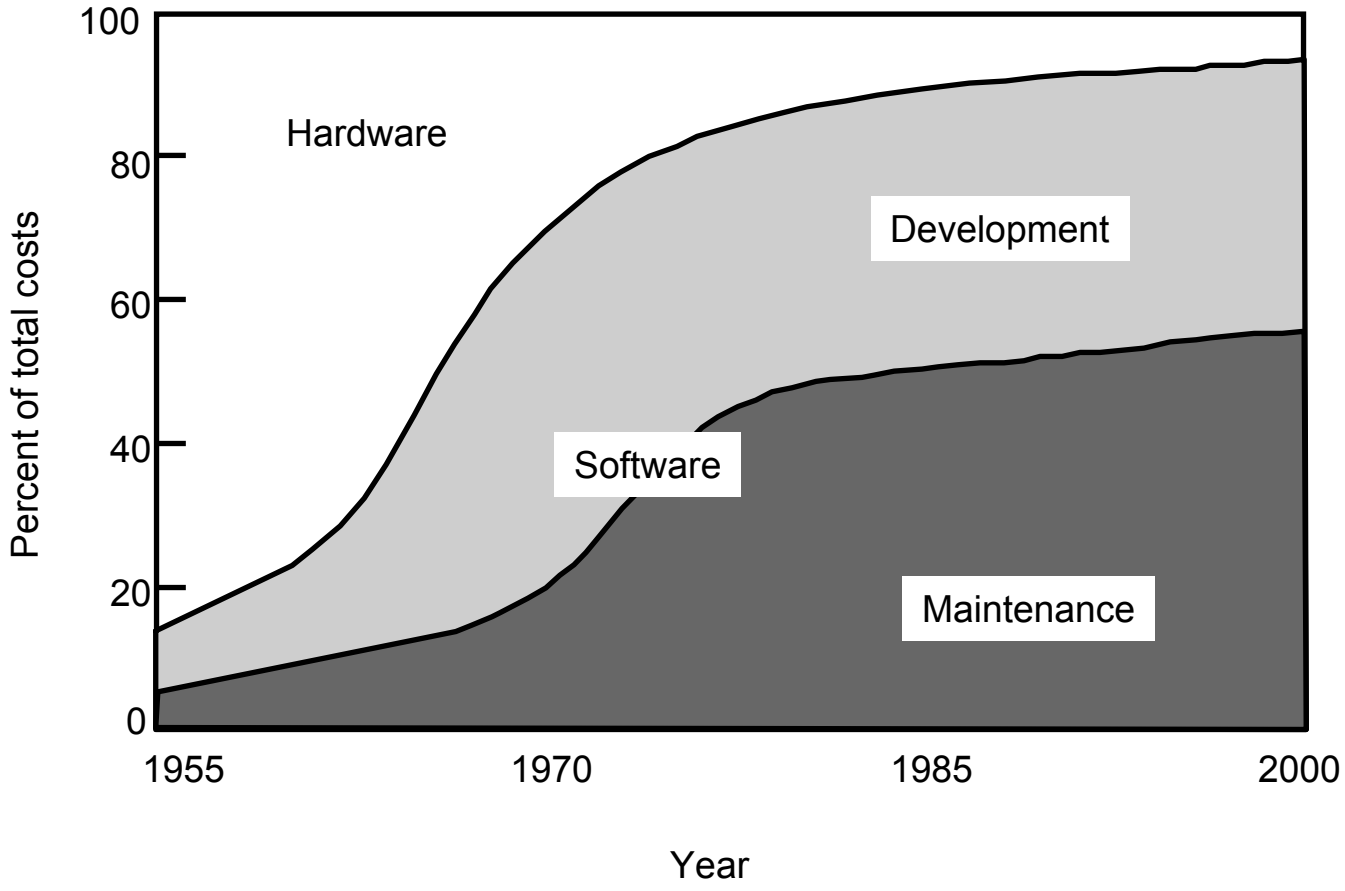


Politecnico di Milano

- **Limitations**
  - **long prototyping times**
  - **high cost of design**
  - **in-cycle silicon fabrication and test**
  - **adhoc techniques used to make architectural and packaging trade-offs**
  - **lack of systematic reuse**
  - **lack of coupling between HW and SW design efforts**

[MAD96]

HW/SW Cost Trends



Example for Modeling Software Development Cost/Time

Politecnico di Milano

Embedded Mode REVIC (REVised Intermediate COCOMO) Model

- Predicts software development life-cycle costs
- Costs including requirements analysis through completion of software acceptance testing
- Maintenance life-cycle costs for fifteen years

Software Development Effort

$$S_E = A \bullet KSLOC^B \bullet F_E \bullet F_M \bullet F_{SCED} \bullet \prod_{i=1}^{16} F_i$$

where

- ***KSLOC*** - thousands of source lines of code
- ***A*** - constant used to capture the multiplicative effects on effort with projects of increasing size (**DEFAULT = 4.44**)
- ***B*** - scale factor which accounts for the relative economies or diseconomies of scale encountered for software projects of different sizes (**DEFAULT = 1.2** -> diseconomy)
- F_i's - cost drivers which model the effect of personnel, computer, product, and project attributes on software cost

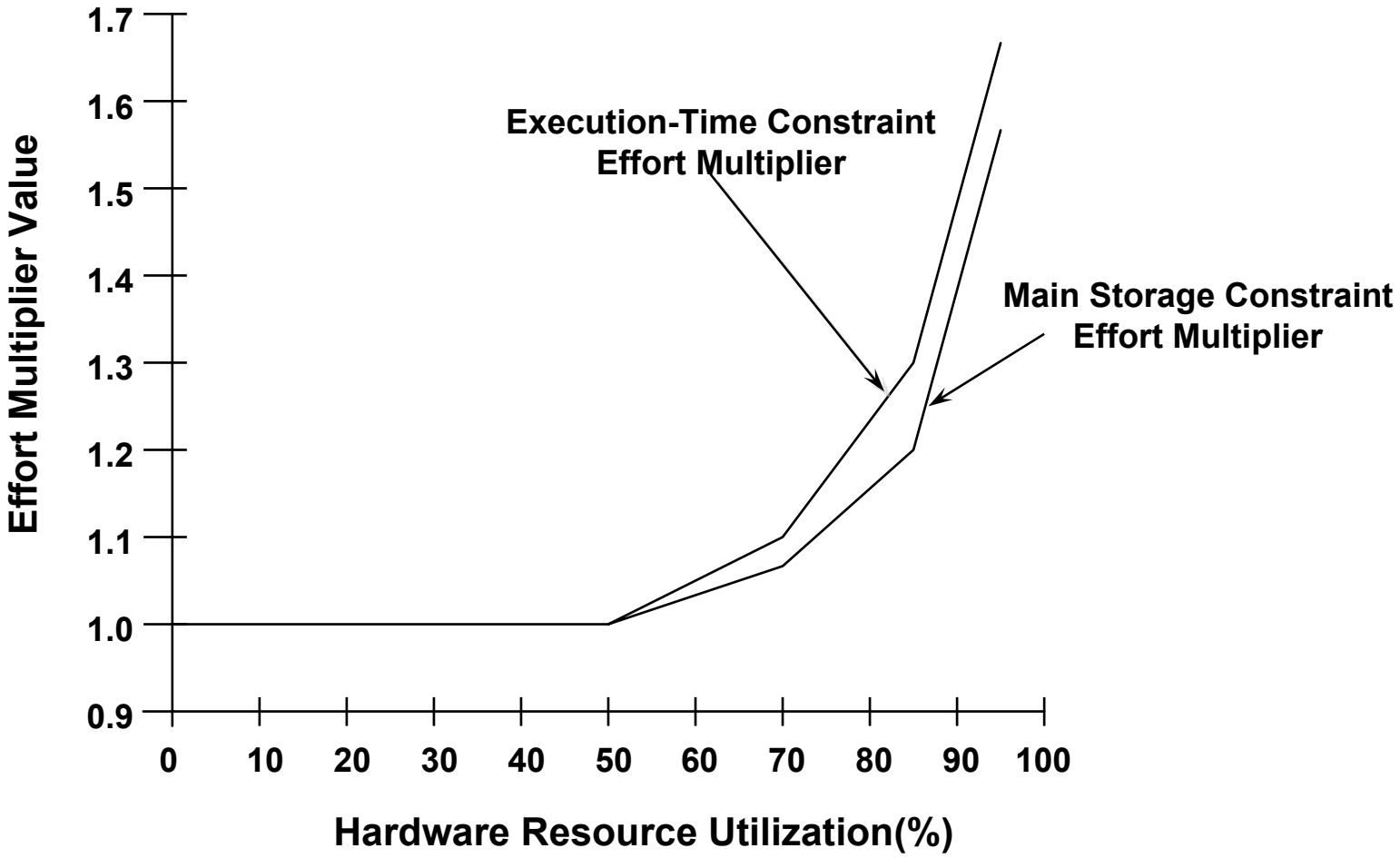
Software Development Time

$$S_T = C \bullet S_{E_{nom}}^D \bullet \frac{PSCED\%}{100}$$

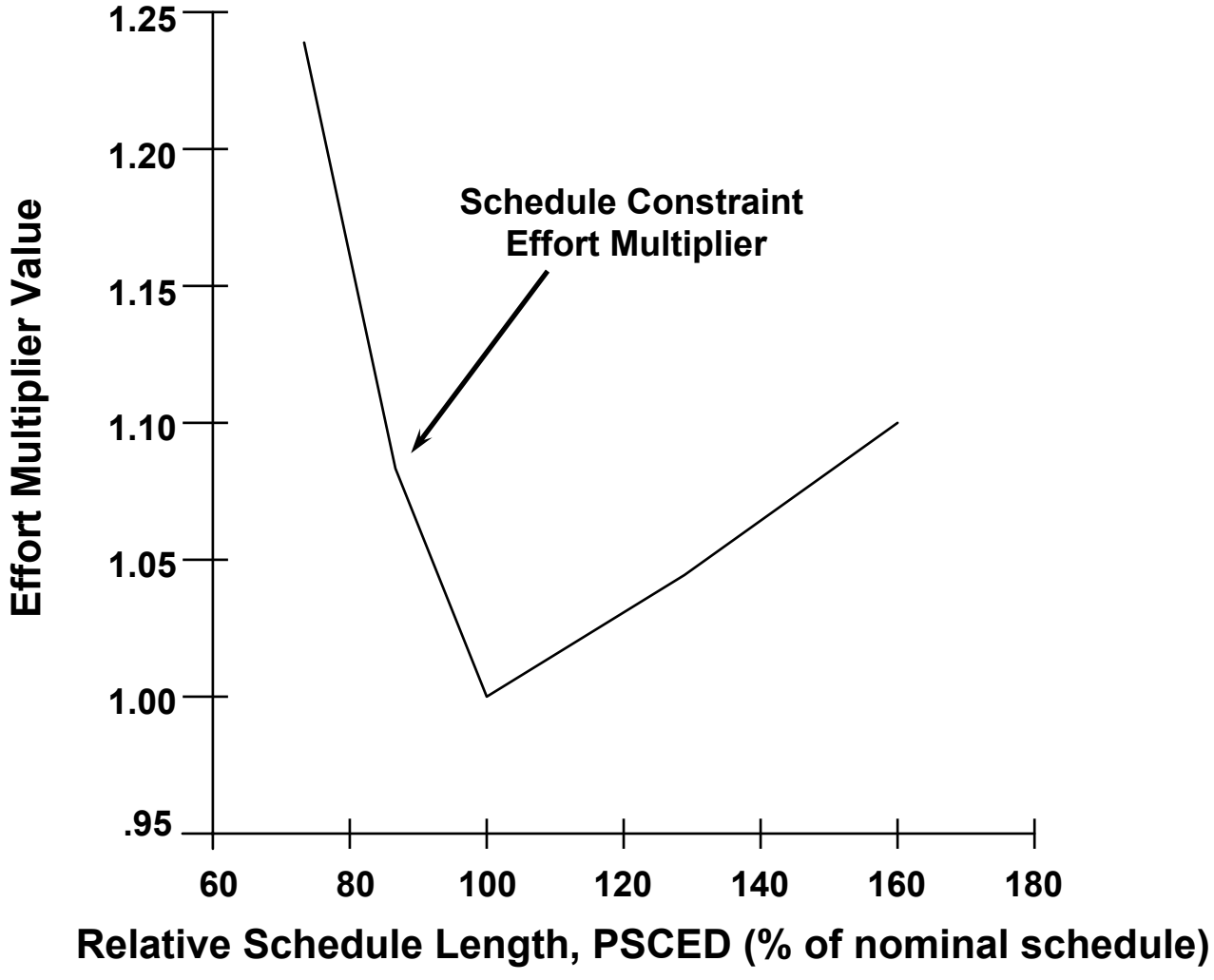
where

- ***C*** - constant used to capture multiplicative effects on time with projects of increasing effort (**DEFAULT = 6.2**)
- ***D*** - scale factor which accounts for the relative economies or diseconomies of scale encountered for projects of different required efforts (**DEFAULT = 0.32**)
- ***PSCED*** - the percent compression/expansion to the **nominal deployment schedule**
- F_E, F_M, F_{SCED} -execution time, main storage, and schedule constraint effort multipliers

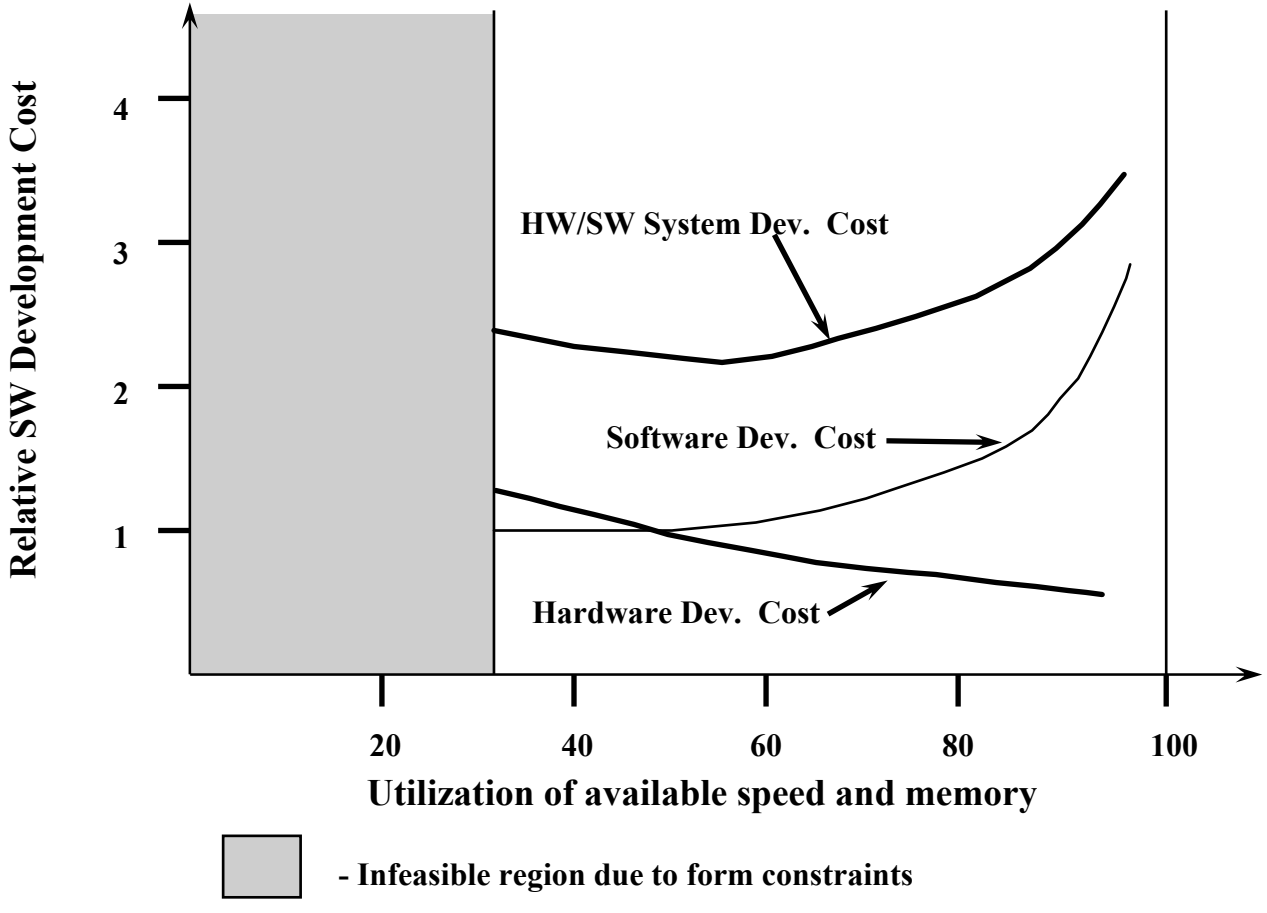
Execution Time, F_E, and Main Storage Constraint, F_M, Effort Multipliers



Cost Impact of Schedule Compression/Extension



Effect of HW Resource Constraints on HW/SW System Prototyping Costs

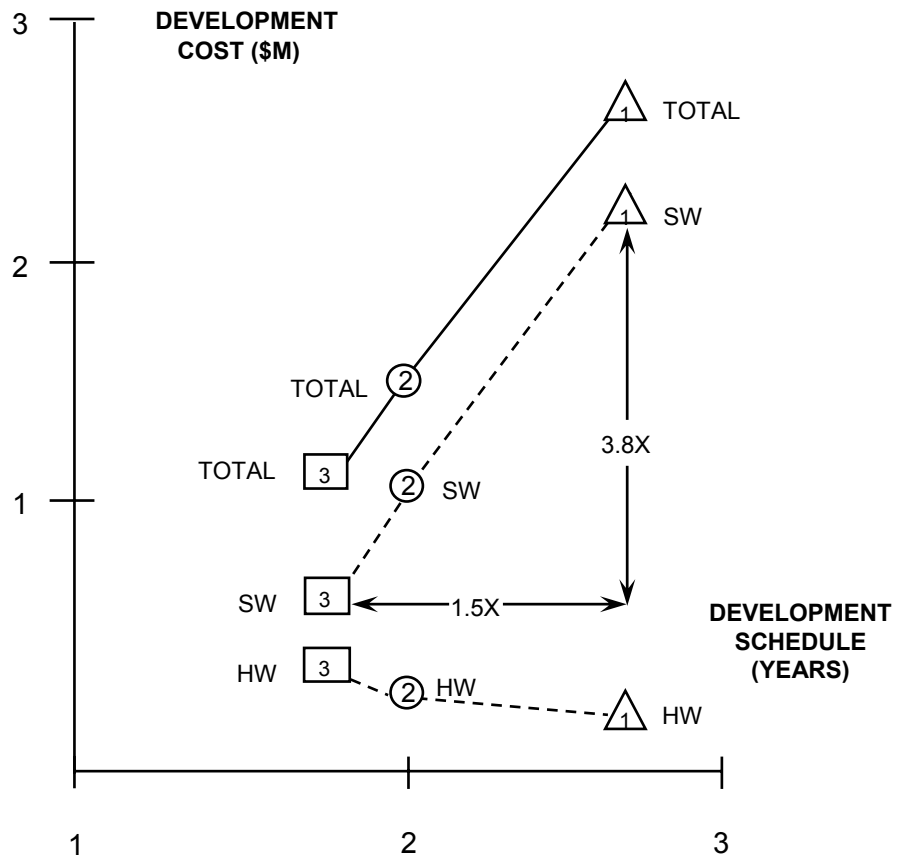


Design Trade-off Example: *SAR Processor with COTS Multiprocessor Cards*

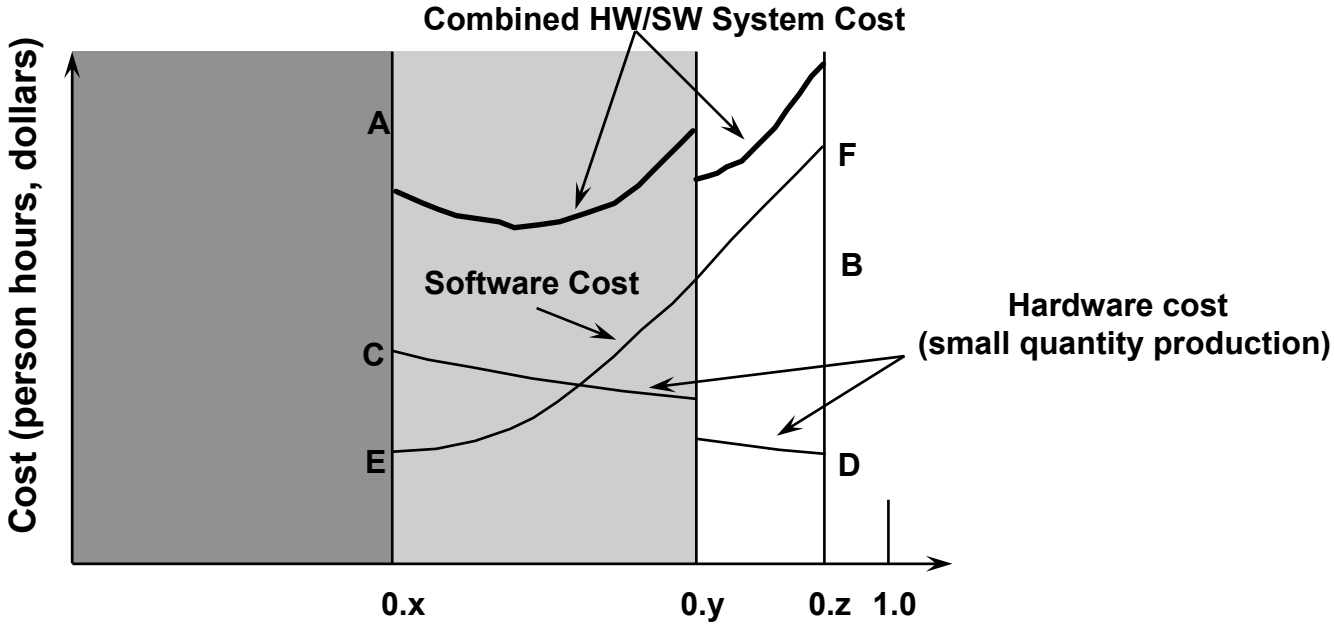
(1) MINIMUM HARDWARE COST
- LOWEST PRODUCTION COST
- MINIMUM SIZE, WEIGHT, & POWER
- NOMINAL MEMORY & COMPUTATION MARGINS

(2) REDUCED DEVELOPMENT COST & TIME
- SAME NUMBER OF CARDS AND LIFE CYCLE COST AS (1)
- RAM ADDED TO IMPROVE MEMORY MARGIN, ALLOW USE OF ADVANCED SOFTWARE DEVELOPMENT TOOLS AND METHODOLOGY

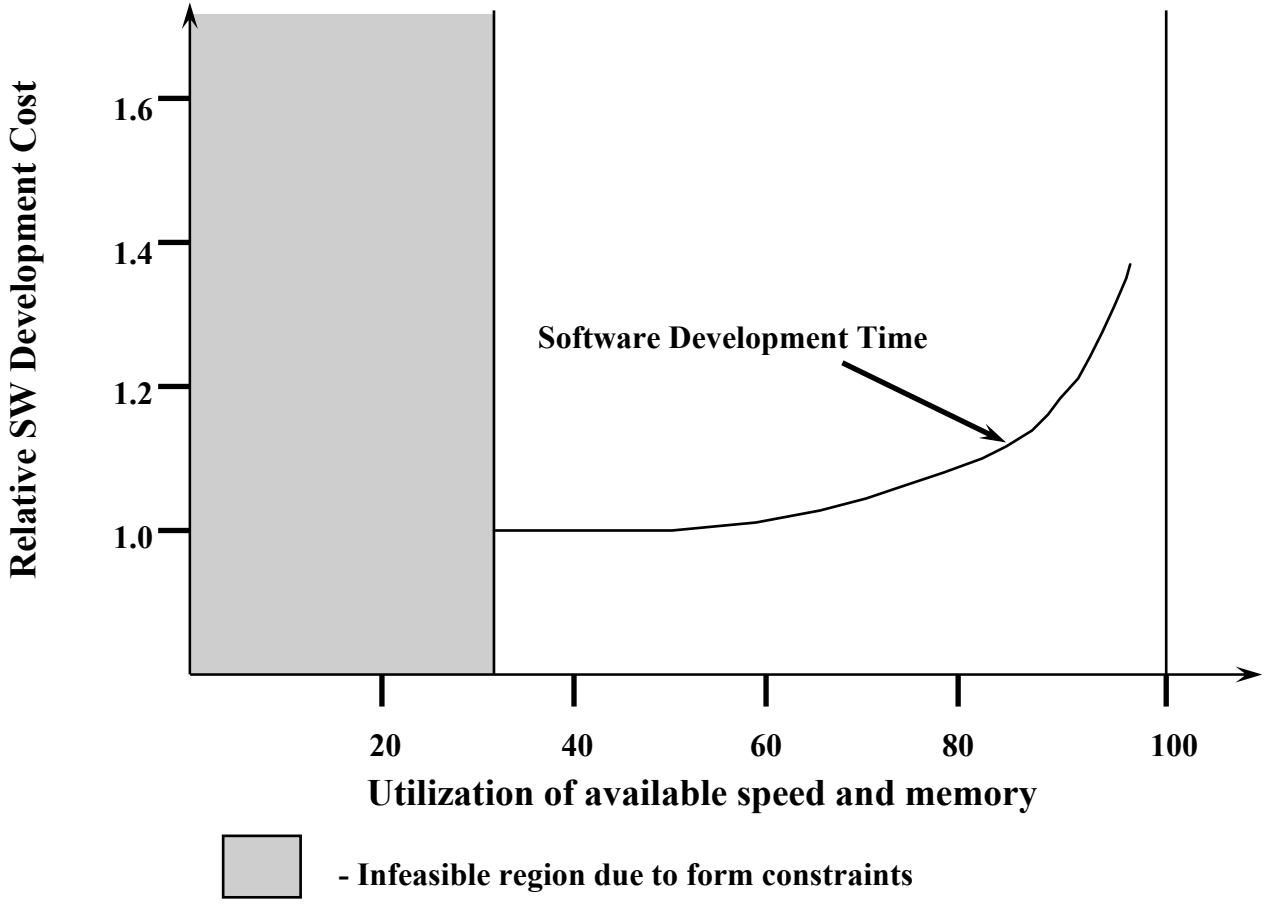
[3] MINIMUM DEVELOPMENT COST & TIME
- PROCESSORS ADDED TO (2)
- IMPROVED COMPUTATION MARGIN FURTHER EASES SOFTWARE DEVELOPMENT



Potential Effect of Packaging on Prototyping Costs



Effect of HW Resource Constraints on Schedule



Effect of Schedule Delays on Time-to-Market Cost

- Time-to-market costs can often outweigh design, prototyping, and production costs
- Reasons:
  - When competitors beat you to market with a comparable product, they gain considerable market share and brand name recognition
  - An earlier market entry has more opportunity to improve yield, thereby improving profits
  - Design and prototyping costs are an up front investment that must produce a return as soon as possible
    - The longer investors must wait for a return, the higher the return must be

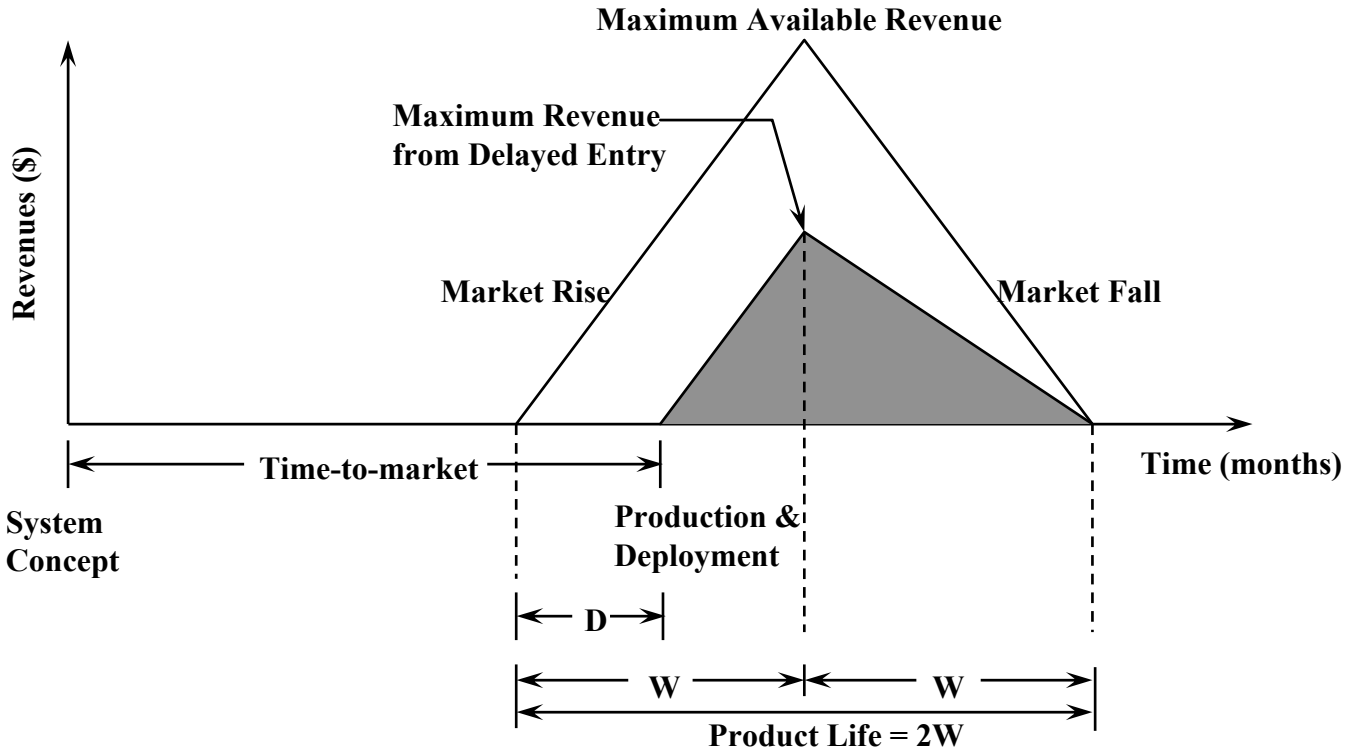
Time-to-Market Cost (Cont.)

- Surveys have shown that being six months late to market resulted in an average of 33% profit loss for that product
  - A 9% production cost overrun resulted in a 21% loss
  - A 50% design development cost overrun resulted in only a 3% profit loss
- Engineering managers stated that they would rather have a 100% overrun in design and prototyping costs than be three months late to market with a product

Time-to-Market Cost Models

- Simple quantitative time-to-market models can be used to estimate the cost of delivering a product to market late
- Examples:
  - Simplified triangular time-to-market model
  - Growth-Stagnation-Decline (GSD) time-to-market model

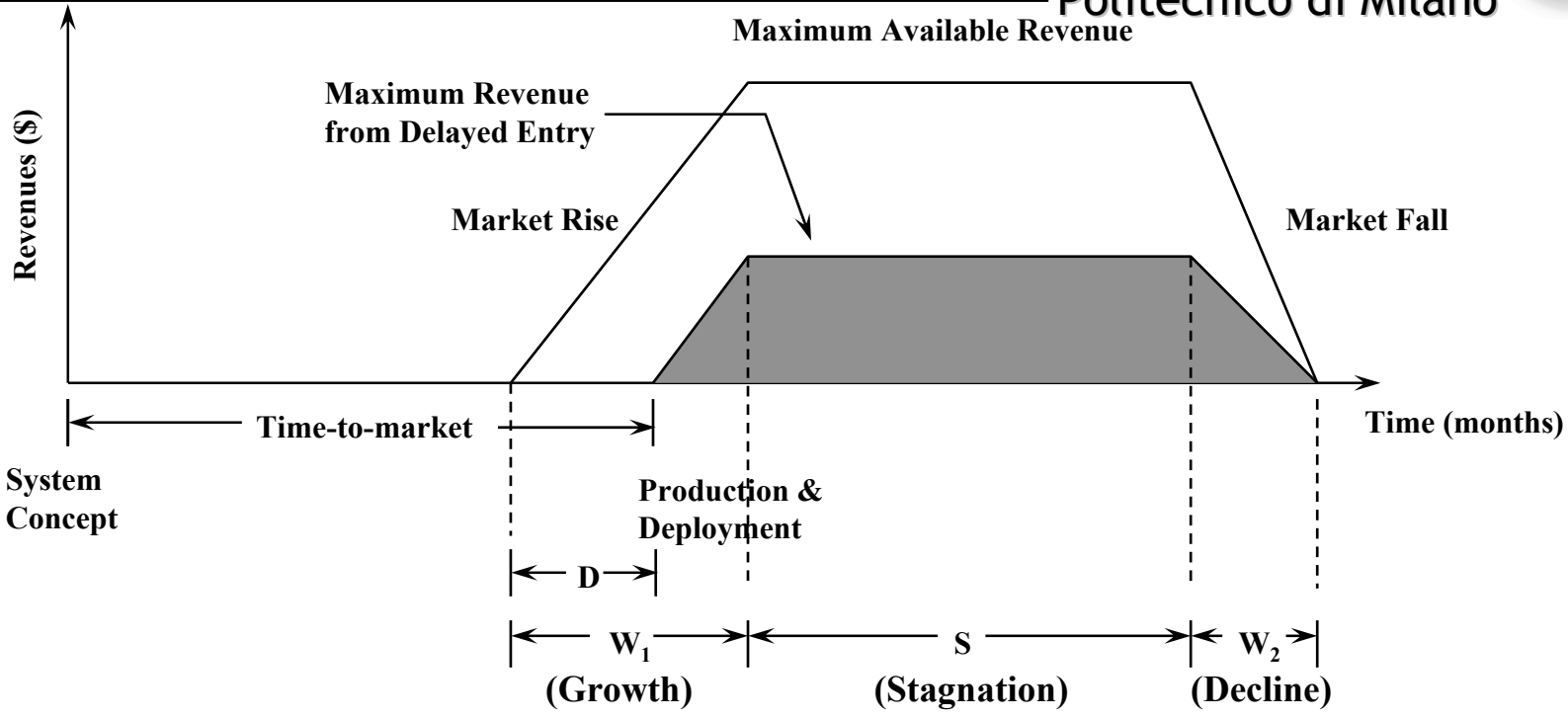
Triangular Time-to-Market Cost Model



Revenue Loss Equation

$$R_L = R_0 \bullet D(3W - D) \Big/ \left(2W^2\right)$$

GSD Time-to-Market Model



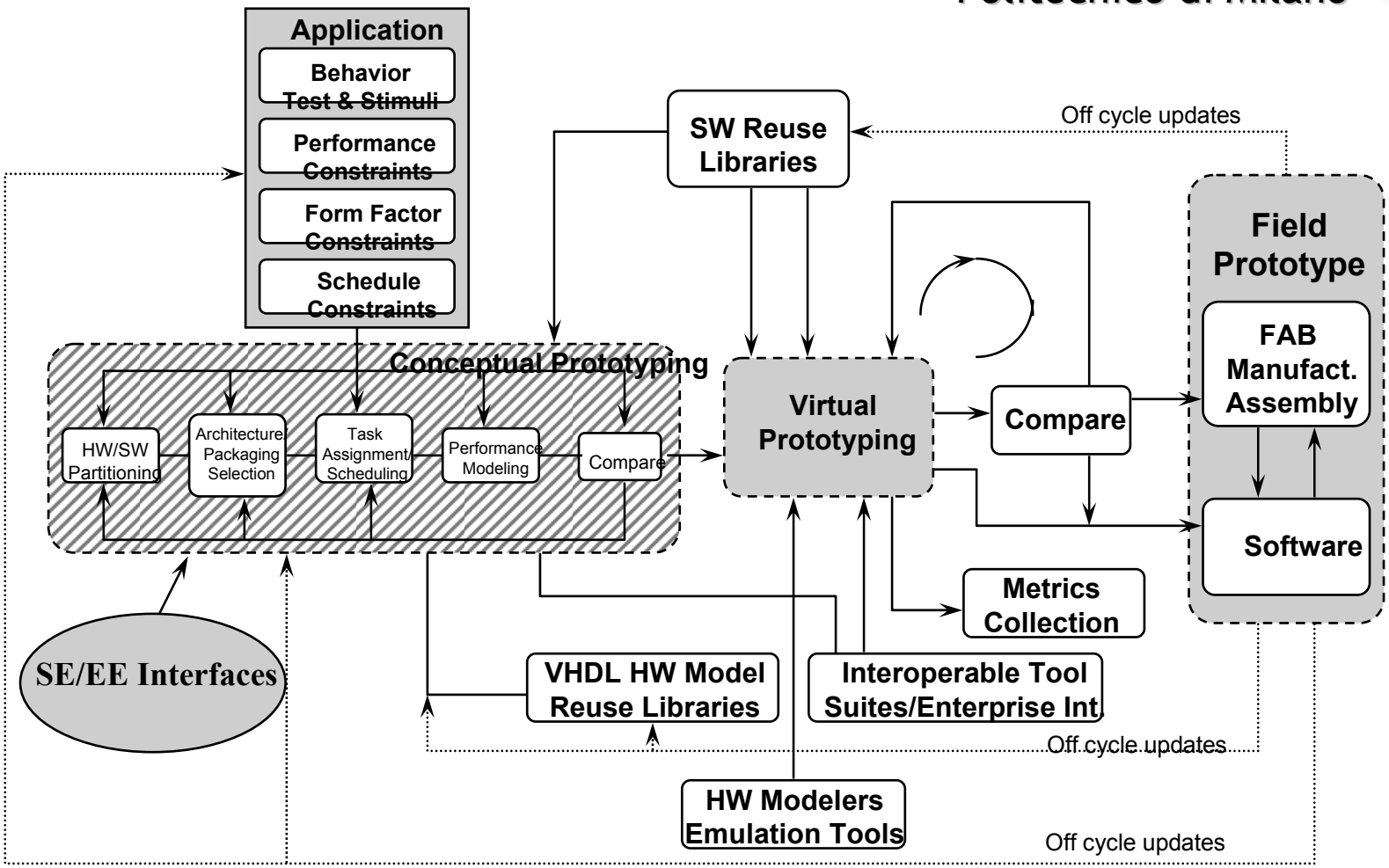
Revenue Loss Equation

$$R_L = R_0 \left[\frac{D\left(2W_1 - D + 2S + W_2\right)}{W_1\left(W_1 + W_2 + 2S\right)} \right]$$

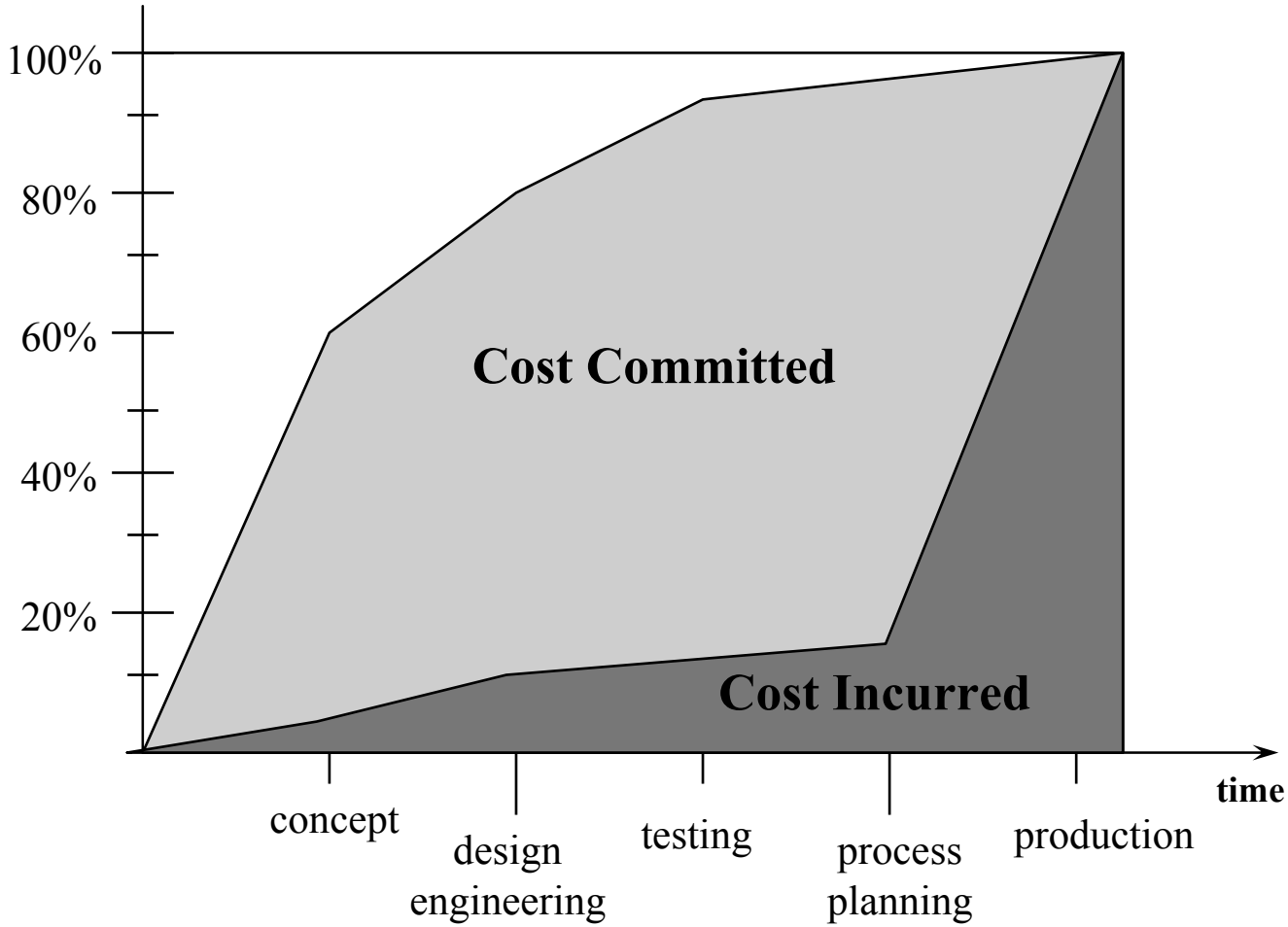
Economic Cost-Driven System-Level Design

- To effectively address these system-level design challenges, a unified approach that considers the cost attributes of software options and hardware options is required.
- Solution: *Economic Cost-Driven System-Level Design*
  - This concept attempts to converge the hardware and software design efforts into a combined methodology that improves cost, cycle time, and quality, which enhancing the exploration of the HW/SW design space.
  - Parametric cost and development time estimation models are used to drive the design process
  - A cost estimation-driven architecture design engine is seamlessly integrated within a hardware-less, library-based cosimulation and coverification environment for rapid prototyping

Overall Library-Based System Design Methodology

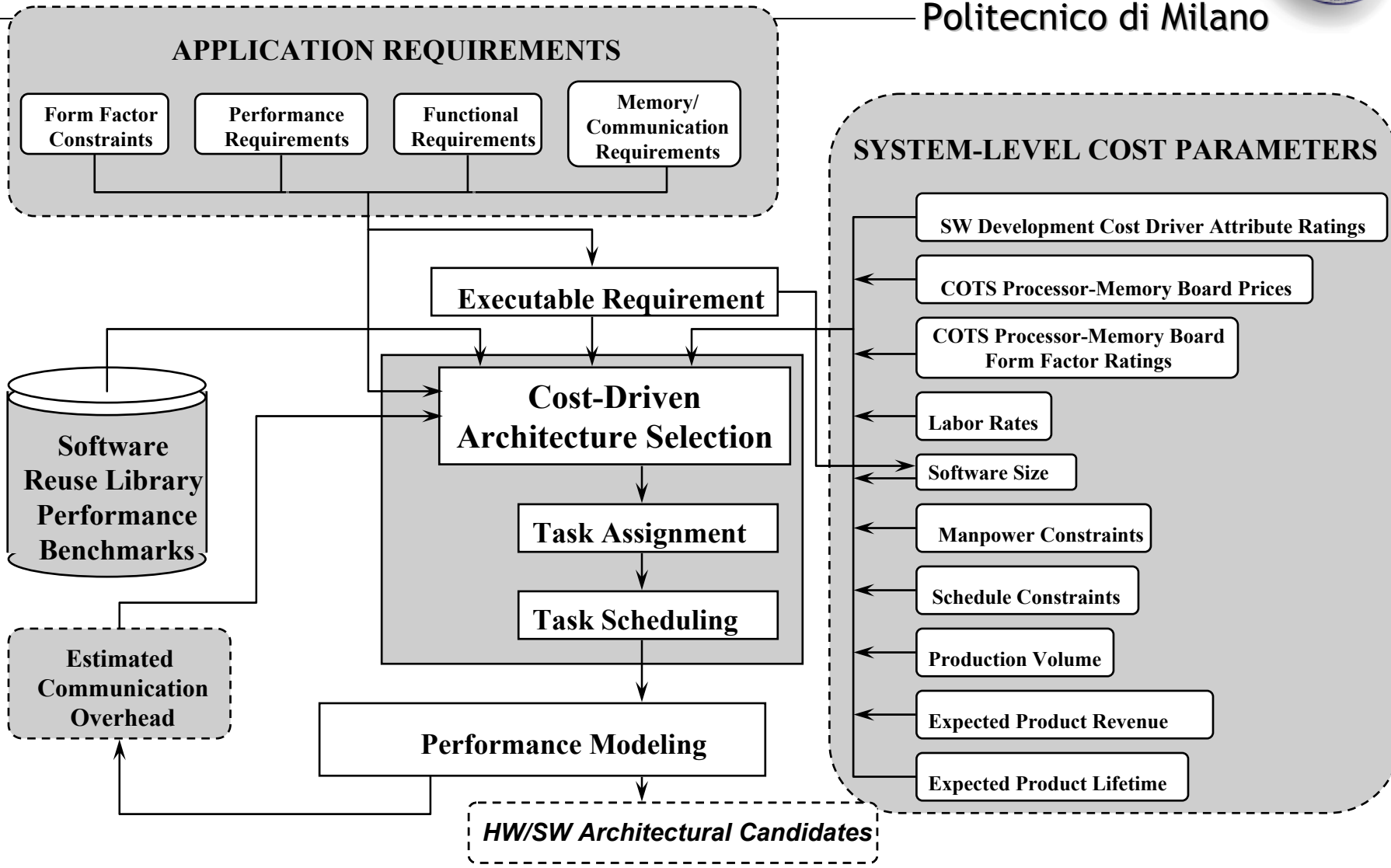


Cost Committed Over the Product Life Cycle



Cost-Driven Front-End Design Process

Politecnico di Milano



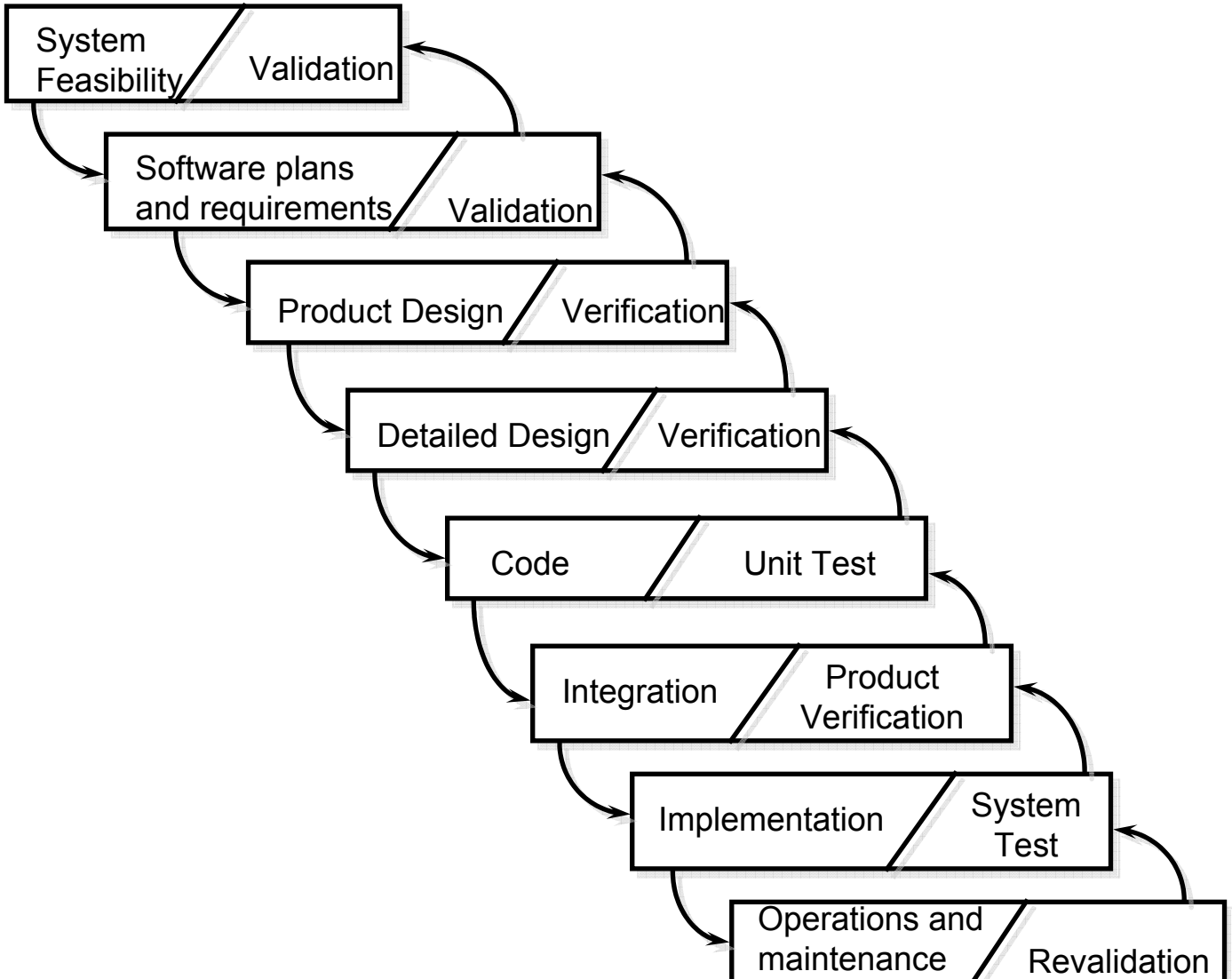
Module Outline

- Introduction to Cost Modeling-Based Embedded Systems Design
- **Software Cost Estimation Process**
- Parametric Software Cost Models
- Parametric Hardware Cost Models
- Applications of Cost Modeling to the RASSP Design Process
- Summary and Major Issues

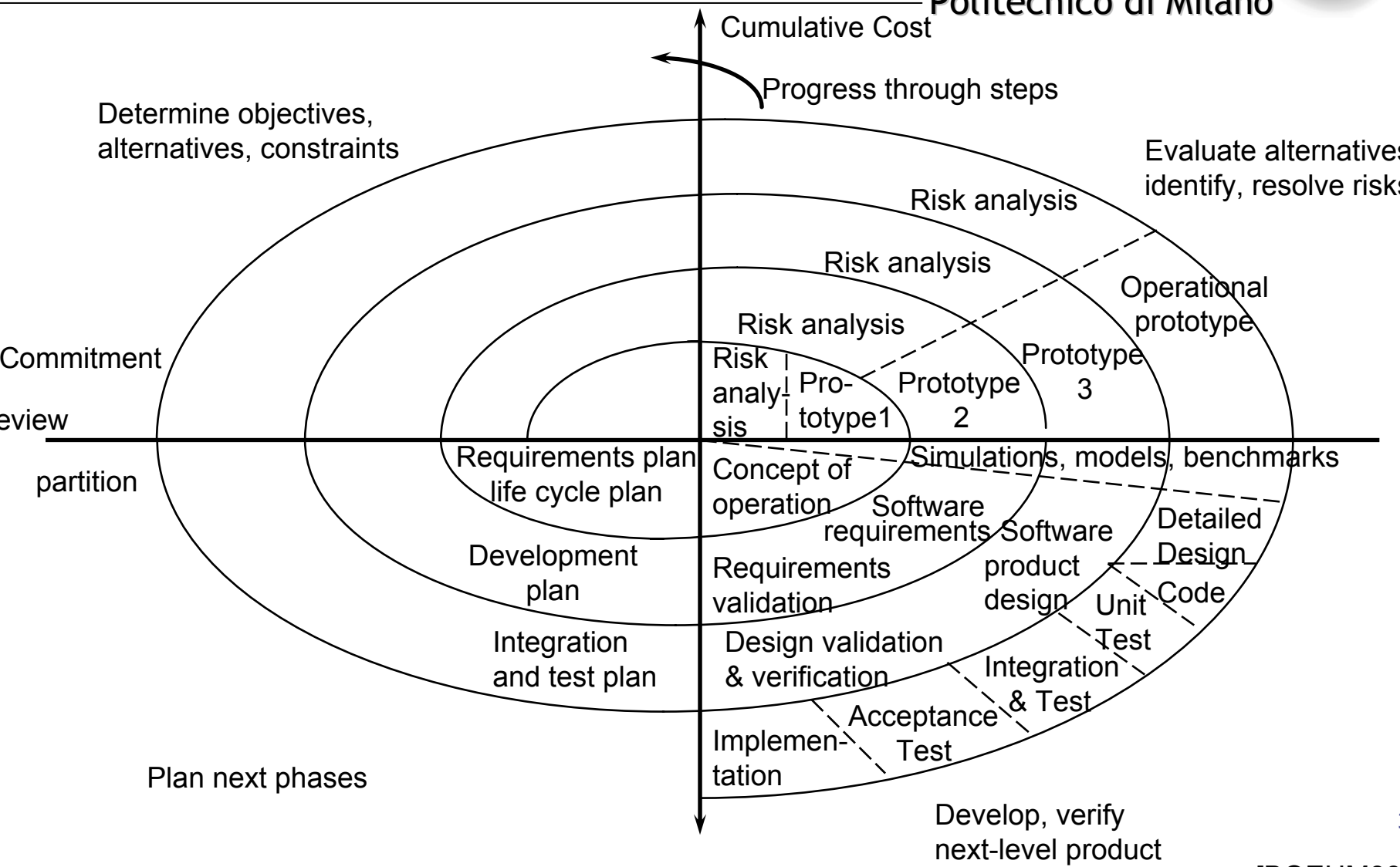
Software Life Cycle Models

- Software life cycle models identify various phases and associated activities required to develop and maintain software
- Some common life cycle models include:
  - Waterfall model
  - Spiral development model
  - Reusable software model
- Models form a baseline from which to begin the software estimation process

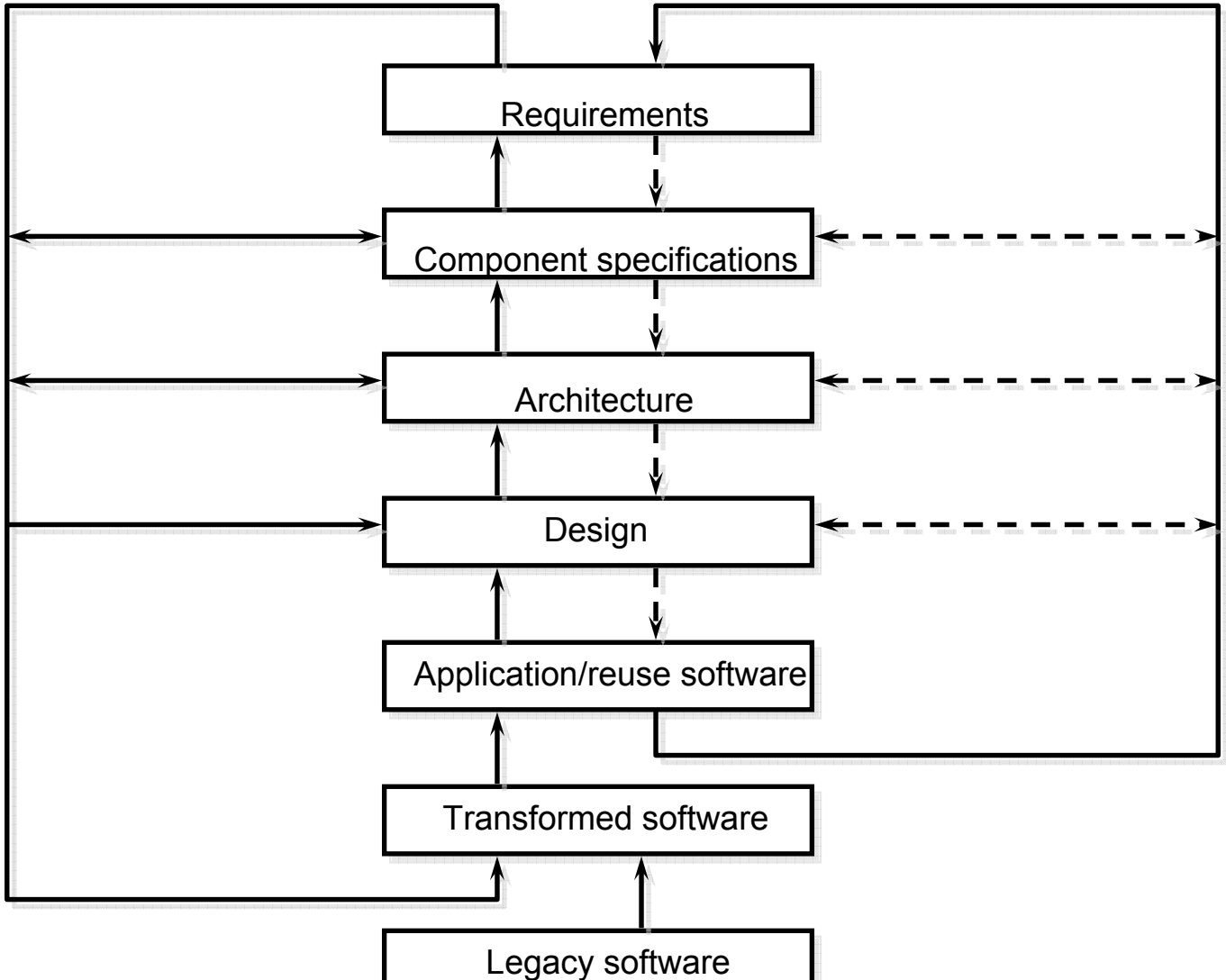
Waterfall Model



Spiral Model



Legacy and Reuse Software Life Cycle



[AHRENS9

Basic Software Cost Estimation Process

- The software cost estimation activity is a miniproject
- Basic Steps
  - Define project objectives and requirements
  - Plan the software estimation activities
    - Develop detailed work breakdown structure (WBS)
  - Estimate software size
  - Define and weigh potential software estimation risks
  - Use several independent cost estimation methodologies

Establish Objectives

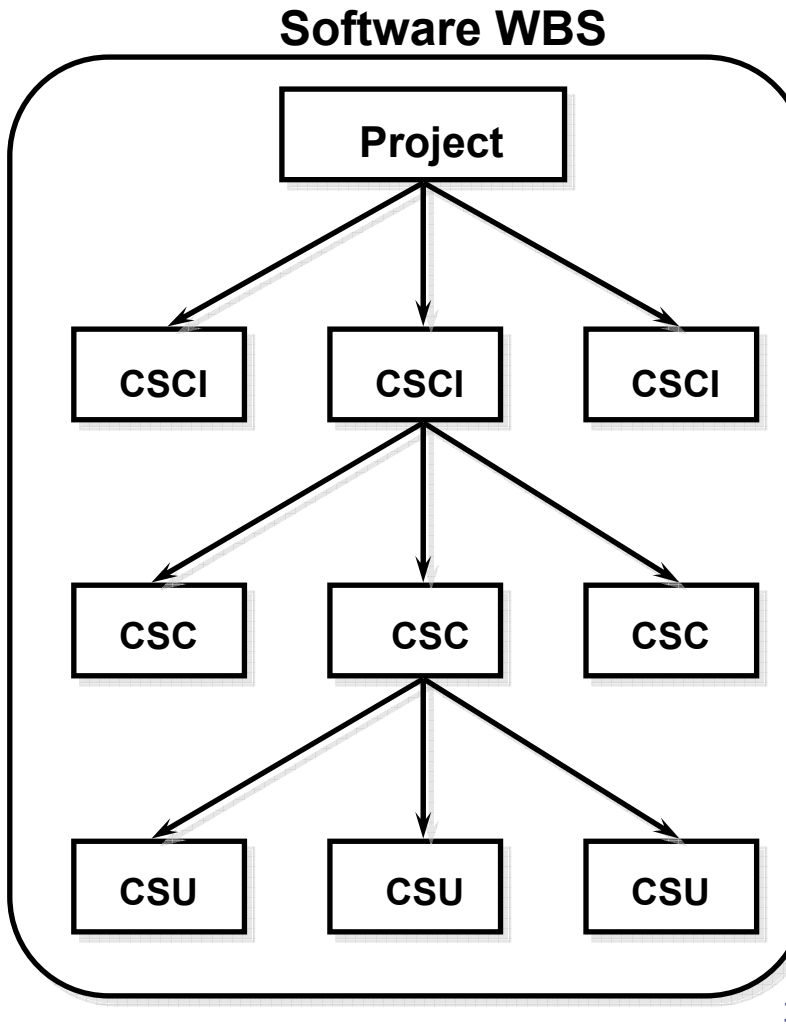
- **Guidelines for establishing the objectives of a cost estimation activity**
  1. **Key the estimating objectives to the needs for decision making information**
  2. **Balance the estimating accuracy objectives for the various system components of the cost estimates**
  3. **Re-examine estimating objectives as the process proceeds, and modify them where appropriate**

Pin Down Software Requirements

- The best way to determine to what extent a software specification is costable is to determine to what extent it is testable.
- A specification is testable to the extent that one can define a clear pass/fail test for determining whether or not the developed software will satisfy the specification.
- In order to be testable, specifications must be specific, unambiguous, and quantitative wherever possible.

Develop a Software Work Breakdown Structure (WBS)

- The WBS helps to establish a hierarchical view and organization of the project
- The WBS should depict only:
  - major software functions
    - computer software configuration items (CSCI)
  - major subdivisions
    - computer software components (CSC)
    - computer software units (CSU)
- The WBS should include all software associated with the project regardless of whether it developed, furnished, or published



Software Size Estimation

- **Sizing By Analogy**
  - Involves relating the proposed project to previously completed projects of similar application, environment and complexity
  - Basic Steps
    - Develop a list of functions and the number of lines of code to implement each function.
    - Identify similarities and differences between previously developed data base items and those data base items to be developed.
    - From the data developed in the previous steps, select those items which are applicable to serve as a basis for the estimate.
    - Generate a size estimate

Software Size Estimation (Cont.)

- **Sizing by Program Evaluation and Review Technique (PERT)**
  - **Based on beta distribution and on separate estimation of individual software components**

$$E_i = \frac{a_i + 4m_i + b_i}{6} \qquad \sigma_i = \frac{b_i - a_i}{6}$$

$$E = \sum_{i=1}^{n} E_i \qquad \sigma E = \left(\sum_{i=1}^{n} \sigma_i^2 \right)^{1/2}$$

  - **Calculation of standard deviation assumes that the estimates are unbiased toward either underestimation or overestimation**
  - **Experience shows that "most likely" estimates tend to cluster toward the lower limit, while actual software product sizes tend to cluster more toward the upper limit**

Software Size Estimation (Cont.)

- **Function Points**
  - **Method of estimating size during the requirements phase based on the functionality to be built into the system**
  - **General Approach**
    - **Count the number of inputs, outputs, inquiries, master files, and interfaces required.**
    - **Multiply these counts by the following factors:**
      - **Inputs - 4**
      - **Outputs - 5**
      - **Inquires - 4**
      - **Master files - 10**
      - **Interfaces - 10**
    - **Adjust the total of these products +25%, 0, or -25% based on the program's complexity**

Software Reuse

- Many software products consists of newly developed software and previously developed software
  - The previously developed software is adapted for use in the new product
- Some effort is required to adapt existing software
  - Redesigning the adapted software to meet the objectives of the new product
  - Reworking portions of the code to accommodate redesigned features or changes in the new product's environment
  - Integrating the adapted code into the new product environment and testing the resulting software product

Effects of Software Reuse on Size Estimate

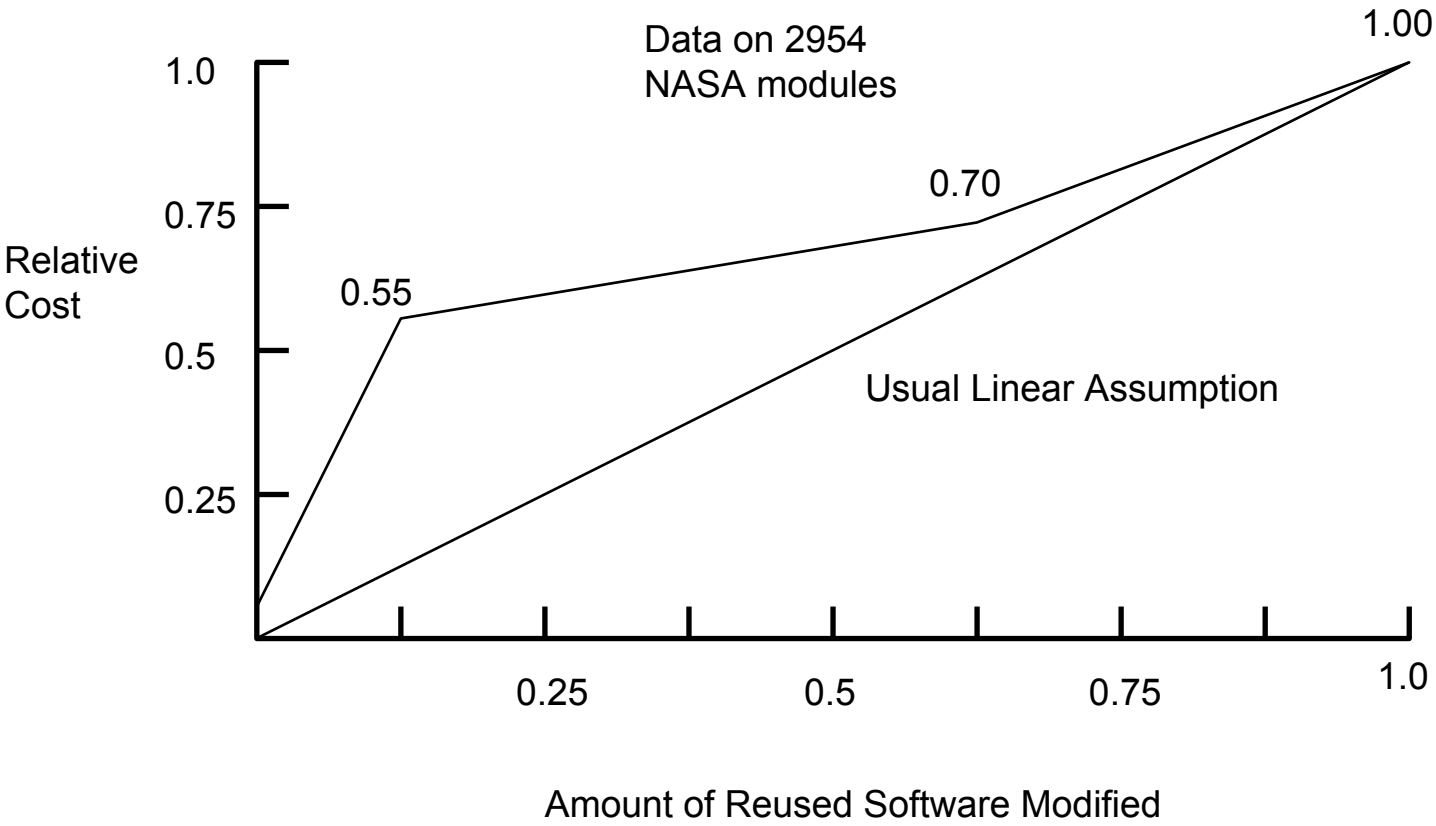
- **The effects of reuse are estimated by calculating the Equivalent Number of Source Lines of Code (*ESLOC*)**
- **Basic Reuse Model**

$$ESLOC = NSLOC + ASLOC \bullet \frac{0.40(DM) + 0.30(CM) + 0.30(IM)}{100}$$

where

- ***ASLOC* is the size of the adapted COTS software component expressed in thousands of adapted source lines of code**
- ***NSLOC* is the size of new software component expressed in thousands of new source lines of code**
- ***DM* is the percent design modified**
- ***CM* is the percent code modified**
- ***IM* is the percent of integration required for modified software**

Nonlinear Reuse Effects



[BOEHM95

Nonlinear Reuse Model

- **Nonlinear reuse estimation model**

$$ESLOC = NSLOC + ASLOC \times \frac{\left(AA + SU + 0.4 \times DM + 0.3 \times CM + 0.3 \times IM\right)}{100}$$

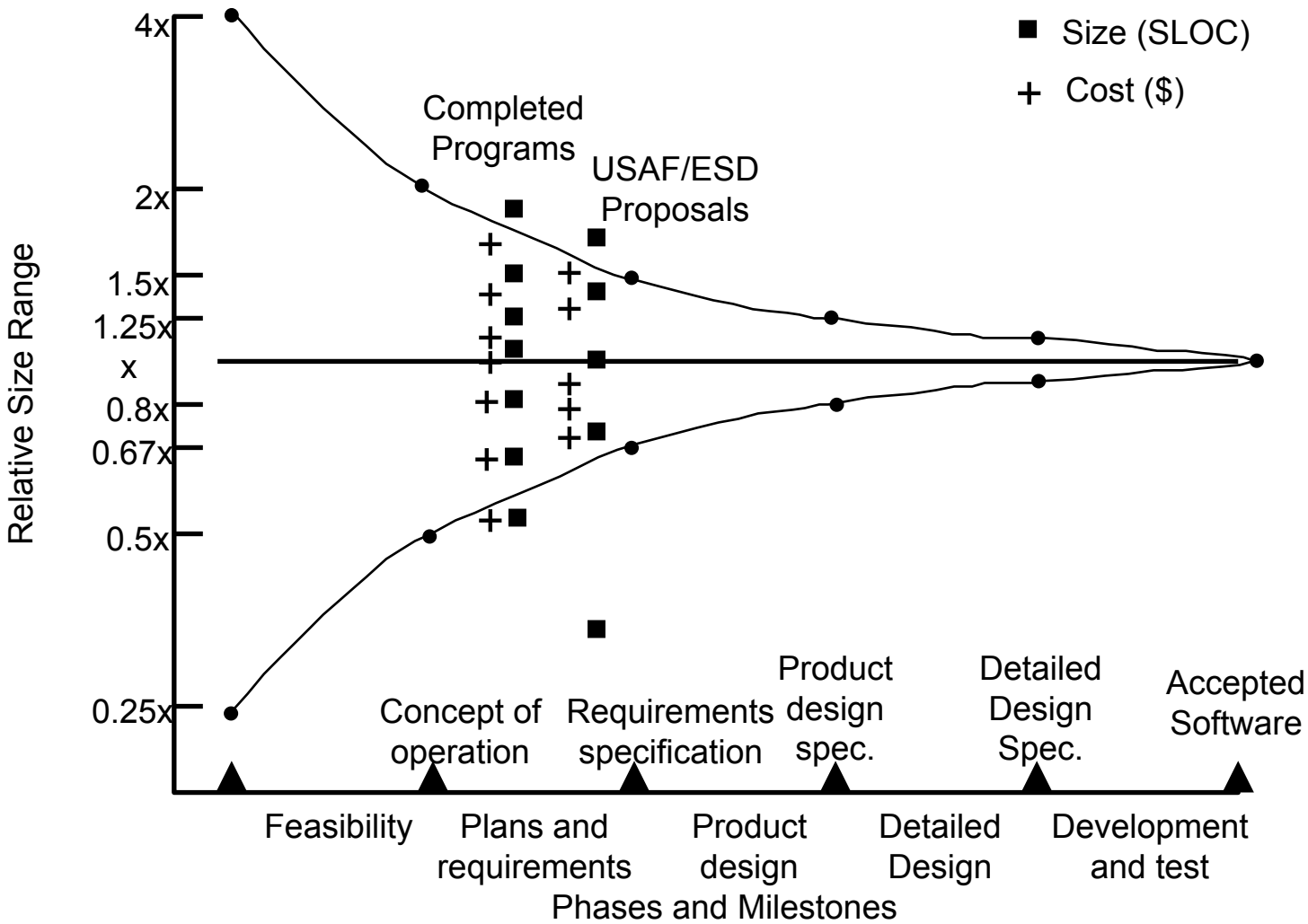
  - **where**
    - ***SU*** **is the software understanding increment which is expressed as a percentage (ranges from 10-50%).**
    - ***AA*** **deals with the degree of assessment and assimilation needed to determine whether a fully-reused software module is appropriate to the application and to integrate its description into the overall product description (ranges from 0-8%).**
    - **The remaining variables in the equation are the same as those used in the original COCOMO.**

Software Estimation Risks

- Adverse effects due to inaccurate software estimation stems from an inability to accurately assess risks associated with various software development projects
- Four major risk areas
  - Software project sizing
  - Specification of development environment
  - Assessment of staff skills
  - Definition of objectives, requirements, and specifications
- All potential risks should be defined and weighed, and impacts to project cost should be determined

Software Risk Example: Software Sizing Accuracy Versus Phase

Politecnico di Milano



[BOEHM9

Use Several Independent Estimation Techniques

- Techniques available for cost estimation
  - Expert Judgment
  - Analogy
  - Top-Down
  - Bottom-Up
  - Parametric or Algorithmic Models
- Important to use a combination of techniques
  - None of the techniques are better than the others from all aspects
  - Their strengths and weaknesses are complementary

Expert Judgment

- These techniques involve consulting one or more experts, who use their experience and understanding of proposed project to arrive at and estimate
- May utilize group consensus techniques
  - Compute median or mean of individual expert estimates
  - Group meeting
  - Delphi (iterative process)
    - Anonymous estimation
    - No group discussion
  - Wideband Delphi (iterative process)
    - Combines the free discussion advantages of the group meeting technique and the advantages of anonymous estimation

Expert Judgment (Cont.)

- Strengths
  - Able to factor in the differences between past project experiences and the new techniques, architectures, or applications of the future project
  - Can also factor in exceptional personnel characteristics and interactions, or other unique project considerations
- Weaknesses
  - The estimate is no better than the expertise and objectivity of the estimator (biases, incomplete recall)
  - Difficult to strike balance between:
    - Quick response expert estimate
      - timely, efficient, but hard to calibrate and rationalize
    - Group consensus estimate
      - Soundly based but highly time consuming and not always exactly repeatable

Estimation By Analogy

- Reasoning by analogy with one or more completed projects to relate their actual costs to an estimate of the cost of a similar new project
- Strengths
  - Useful when developing a new product when a systematic historical cost database does not exist
  - Based on actual experience on a project
- Weaknesses
  - A high degree of judgment is required when making adjustments for differences in analogous products
  - Not clear to what degree the previous project is actually representative of the constraints, techniques, personnel, and functions to performed on the new project

Top-Down Estimating

- Overall cost estimate for the project is derived from the global properties of the software project
- Cost is then split up among the various components
- Can be done in conjunction with any of the before-mentioned methods
- Strengths
  - System level focus
  - Efficient, easier to implement, requires minimal detail
- Weaknesses
  - Can be less accurate due to lack of detail
  - Tends to overlook lower level components and possible technical problems

Bottom-Up Estimating

- The cost of each software component is estimated by an individual and these costs are then summed to arrive at an estimated cost for the overall product
- Strengths
  - More detailed basis
  - More stable
  - Fosters individual commitment
- Weaknesses
  - May overlook system level costs
  - Requires more effort

Parametric Models

- **Mathematical expressions**
  - Derived from the statistical correlation of historical system costs with performance and/or physical attributes of the system
  - Can produce high quality, consistent estimates if derived from a sound representative database
- **Focus on major cost drivers (not minute details)**
  - Predominant effect on system costs
  - Independence (cost drivers are uncorrelated)
- **Strengths**
  - Objective
  - Repeatable
  - Efficient
  - Facilitates sensitivity analysis

Parametric Models (Cont.)

- **Linear models**

$$Effort = a_0 + \sum_{i=1}^{n} a_i x_i$$

- **Multiplicative models**

$$Effort = a_0 \prod_{i=1}^{n} a_i^{x_i}$$

- **Analytic models**

$$Effort = f(x_1, \ldots, x_n)$$

- **Tabular models**
  - **Contains tables which relate the values of cost driver variables to multipliers used to adjust the effort estimate**
- **Composite models**
  - **Incorporate a combination of linear, multiplicative, analytic, and tabular functions to estimate software effort as a function of cost driver variables**

Information Needed to Use a Parametric Model

- Well documented parameters identifying:
  - source of data used to derive the parametric model
  - size of database
  - time frame of database
  - range of database
  - how parameters were derived
  - limitations spelled out
  - how well the parametric model estimates its own database
  - consistent and well defined WBS dictionary

- Realistic estimates of most-likely range for independent variable values
- Top functional experts knowledgeable about the project being estimated
  - to identify most-likely range for cost drivers
  - to confirm applicability of parametric from technical perspective

Module Outline

- Introduction to Cost Modeling-Based Embedded Systems Design
- Software Cost Estimation Process
- **Parametric Software Cost Models**
- Parametric Hardware Cost Models
- Applications of Cost Modeling to the RASSP Design Process
- Summary and Major Issues

Example Parametric Software Cost Estimators

- A number of automated software estimation tools are available which allow for quick effort and schedule estimates based on size estimates and cost driver attributes
- 15 to 20 cost driver attributes that reflect the development environment depending on the model used
- Examples
  - COCOMO
  - COCOMO 2.0
  - REVIC (REVised Intermediate COCOMO)
  - PRICE S Model

Constructive Cost Model (COCOMO)

- Hierarchy of software cost estimation models
  - Basic COCOMO
    - Estimates software development effort and cost solely as a function of the size of the software product in source instructions
  - Intermediate COCOMO
    - Estimates software development effort as a function of the most significant software cost drivers as well as size
  - Detailed COCOMO
    - Represents the effects of the cost drivers on each individual development cycle phase
    - Used during detailed phases of design to refine cost estimates

COCOMO (Cont.)

- Models three modes of software development
  - Organic
    - Small-to-medium size product developed in a familiar, in-house environment
  - Embedded
    - Product must operate within tight constraints
    - Product is embedded in a strongly coupled complex of hardware, software, regulations, and operational procedures
  - Semidetached
    - Intermediate stage between organic and embedded mode
    - Team members have varied experience with related systems

Summary of COCOMO Hierarchy of Models

	COCOMO Level		
Estimate	**Basic**	**Intermediate**	**Detailed**
Development Effort, MM_{DEV}	*f*(mode, KDSI)	*f*(mode, KDSI, 15 cost drivers)	*f*(mode, KDSI, 15 cost drivers)
Development schedule	*f*(mode, MM_{DEV})	Same as for Basic level	Same as for Basic level
Maintenance effort	*f*(MM_{DEV}, ACT)	*f*(MM_{DEV}, ACT, 15 cost drivers)	Same as for Intermediate level
Product hierarchy	Entire System	System/components	System/subsystem/ module
Phase distribution of effort	*f*(mode, KDSI)	Same as for Basic level	*f*(mode, KDSI, 15 cost drivers)
Phase distribution of schedule	*f*(mode, KDSI)	Same as for Basic level	*f*(Basic schedule distr., Detailed schedule distr.)

Intermediate COCOMO Model

Embedded Mode Intermediate COCOMO Model

- Costs including product design through the completion of the integration and test phase

Software Development Effort

$$S_E = A \bullet KSLOC^B \bullet \prod_{i=1}^{15} F_i$$

where

- ***KSLOC*** - software size in thousands of source lines of code
- ***A*** - constant used to capture the multiplicative effects on effort with projects of increasing size (**DEFAULT = 2.8**)
- ***B*** - scale factor which accounts for the relative economies or diseconomies of scale encountered for software projects of different sizes (**DEFAULT = 1.2** -> diseconomy)
- F_i's - cost drivers which model the effect of personnel, computer, product, and project attributes on software cost

Software Development Time

$$S_T = C \bullet S_{E_{nom}}^D \bullet \frac{PSCED\%}{100}$$

where

- ***C*** - constant used to capture multiplicative effects on time with projects of increasing effort (**DEFAULT = 2.5**)
- ***D*** - scale factor which accounts for the relative economies or diseconomies of scale encountered for projects of different required efforts (**DEFAULT = 0.32**)
- ***PSCED*** - the percent compression/expansion to the **nominal deployment schedule**

Software Cost Attributes (F_i) in Intermediate COCOMO

- **Computer Attributes**
  - TIME - Execution Time Constraint
  - STOR - Main Storage Constraint
  - VIRT - Virtual Machine Volatility
  - TURN - Computer Turnaround Time
- **Product Attributes**
  - RELY - Required Software Reliability
  - DATA - Database Size
  - CPLX - Product Complexity

- **Project Attributes**
  - MODP - Modern Programming Practices
  - TOOL - Use of Software Tools
  - SCED - Required Development Schedule
- **Personnel Attributes**
  - ACAP - Analyst Capability
  - AEXP - Applications Experience
  - PCAP - Programmer Capability
  - VEXP - Virtual Machine Experience
  - LEXP - Programming Language Experience

Intermediate COCOMO (Cont.)

Software Development Effort Multipliers

Cost Drivers	Ratings					
	Very Low	Low	Nominal	High	Very High	Extra High
Product Attributes						
RELY	.75	.88	1.00	1.15	1.40	
DATA		.94	1.00	1.08	1.16	
CPLX	.70	.85	1.00	1.15	1.30	1.65
Computer Attributes						
TIME			1.00	1.11	1.30	1.66
STOR			1.00	1.06	1.21	1.56
VIRT		.87	1.00	1.15	1.30	
TURN		.87	1.00	1.07	1.15	

[BOEHM81

Intermediate COCOMO (Cont.)

Software Development Effort Multipliers (Cont.)

Cost Drivers	Ratings					
	Very Low	Low	Nominal	High	Very High	Extra High
Personnel Attributes						
ACAP	1.46	1.19	1.00	.86	.71	
AEXP	1.29	1.13	1.00	.91	.82	
PCAP	1.42	1.17	1.00	.86	.70	
VEXP	1.21	1.10	1.00	.90		
LEXP	1.14	1.07	1.00	.95		
Project Attributes						
MODP	1.24	1.10	1.00	.91	.82	
TOOL	1.24	1.10	1.00	.91	.83	
SCED	1.23	1.08	1.00	1.04	1.10	

[BOEHM8

Calibrating Nominal Effort Equations (Cont.)

- **Calibrating the Software Development Mode**
  - **Use a similar least squares technique to calibrate the coefficient term *c* and the scale factor *b* in the effort equation**

$$MM = c(KSLOC)^b \prod_{i=1}^{15} F_i$$

  - **Given completed projects $p_1, \ldots, p_n$, solve for *c* and *b* which satisfy the following equations**

$$\log \bar{c} = \frac{a_2 d_0 - a_1 d_1}{a_0 a_2 - a_1^2}, \qquad b = \frac{a_0 d_1 - a_1 d_0}{a_0 a_2 - a_1^2}$$

    - **where**

$$a_0 = n, \qquad a_1 = \sum_{i=1}^{n} \log(KSLOC)_i, \qquad a_2 = \sum_{i=1}^{n} \left[\log(KSLOC)_i\right]^2$$

$$d_0 = \sum_{i=1}^{n} \log\left(MM / \prod_j F_j\right)_i, \qquad d_1 = \sum_{i=1}^{n} \log\left(MM / \prod_j F_j\right)_i \log(KSLOC)_i$$

Intermediate COCOMO - Maintenance Estimate

- **Basic model assumption**
  - **Software maintenance costs are determined by substantially the same cost driver attributes that determine software development costs.**
  - **Annual maintenance effort equation**

$$(MM)_{AM} = (1.0)(ACT)(MM)_{NOM} \prod_{i=1}^{15} F_i$$

where

- ***ACT* is the annual change traffic**
- ***(MM)*$_{NOM}$ is the nominal software development effort which is only a function of the software size**
- ***F*$_i$ is the maintenance effort adjustment factor for cost driver attribute *i***

Intermediate COCOMO - Maintenance Estimate (Cont.)

- **Two cost driver attributes have different effort multipliers for maintenance**
  - **Required Reliability (RELY)**

Very Low	Low	Nominal	High	Very High
1.35	1.15	1.00	0.98	1.10

  - **Use of Modern Programming Practices (MODP)**

	Rating				
Product Size (KDSI)	Very Low	Low	Nominal	High	Very High
2	1.25	1.12	1.00	0.90	0.81
8	1.30	1.14	1.00	0.88	0.77
32	1.35	1.16	1.00	0.86	0.74
128	1.40	1.18	1.00	0.85	0.72
512	1.45	1.20	1.00	0.84	0.70

[BOEHM81]

Calibrating a Cost Model to a Particular Organization

- **Example: COCOMO calibration**
  - **Calibrating the COCOMO nominal effort equations to an organization's design experience**
    - **Calibrating the constant term**
    - **Calibrating the software development mode**
  - **Other calibration methods**
    - **Consolidating or eliminating redundant cost driver attributes within the model**
    - **Adding further cost driver attributes which may be significant at this organization, but not in general**

Calibrating Nominal Effort Equations

- **Calibrating the constant term *c***
  - **Ex. Embedded mode**

$$MM = c(KSLOC)^{1.2}\prod_{i=1}^{15} F_i$$

  - **To calibrate to an organization's completed projects $p_1, \ldots, p_n$ with**
    - **sizes: $KSLOC_1, KSLOC_2, \ldots, KSLOC_n$**
    - **overall effort adjustment factors: $F_1, F_2, \ldots, F_n$**
    - **actual development efforts: $MM_1, MM_2, \ldots, MM_n$**
  - **Solve for the value of *a* which minimizes the sum of squares of residual errors**

$$S = \sum_{i=1}^{n}\left[c(KSLOC)^{1.2}F_i - MM_i\right]^2 \Rightarrow \qquad \bar{c} = \frac{\sum_{i=1}^{n} MM_i (KSLOC)^{1.2} F_i}{\sum_{i=1}^{n}\left[(KSLOC)^{1.2} F_i\right]^2}$$

REVIC

- **REVIC (REVised Intermediate COCOMO) predicts software development life-cycle costs**
  - **Costs including requirements analysis through completion of software acceptance testing**
  - **Maintenance life-cycle costs for fifteen years**
- **REVIC Software Development Modes**

Mode	Description
Organic	Stand alone program with few interfaces, a stable development environment, no new algorithms, and few constraints- Usually very small programs
Embedded	Programs with considerable interfaces, new algorithms, or extremely tight constraints. Usually very large or complicated programs.
Semidetached	A combination of organic and embedded features.
ADA	Programs developed using an object-oriented analysis methodology or use of the Ada language, with emphasis on the separately compilable specs and body parts of the code

REVIC Development Phase Descriptions

Phase	Start Milestone	End Milestone
SW Requirements Engineering	General Contract Award	Completion of Software Specification Review (SRR)
Preliminary Design	Completion of SSR	Completion of Preliminary Design Review (PDR) or equivalent
Critical Design	Completion of PDR or equivalent	Completion of Critical Design Review (CDR) or equivalent
Code & Unit Test	Completion of CDR or equivalent	Completion of CSC Testing by the programmers (CUT)
Integration & Test	Completion of CUT	Completion of Formal Qualification Test (FQT) at the CSCI level
Development Test & Evaluation	Completion of FQT at the CSCI level	Completion of SW-to-SW and SW-to-HW integration and Functional & Physical Configuration Audits

REVIC Development Phase Descriptions (Cont.)

- REVIC development equations predict effort and schedule from Preliminary Design through Integration & Test
- REVIC predicts the effort and schedule in the *Software Requirements Engineering* and *Development Test & Evaluation* phases by taking a percentage of the development phases
  - Software Requirements Engineering (default values)
    - 12% of development effort
    - 30% of development schedule
  - Development Test & Evaluation (default values)
    - 22% of development effort
    - 26% of development schedule

REVIC Software Development Cost/Schedule Models

Politecnico di Milano

Embedded Mode REVIC (REVised Intermediate COCOMO) Model

- Predicts software development costs
- Costs including requirements analysis through completion of software acceptance testing
- Maintenance life-cycle costs for fifteen years

Software Development Effort

$$S_E = A \bullet KSLOC^B \bullet \prod_{i=1}^{19} F_i$$

where

- ***KSLOC*** - thousands of source lines of code
- ***A*** - constant used to capture the multiplicative effects on effort with projects of increasing size (**DEFAULT = 4.44**)
- ***B*** - scale factor which accounts for the relative economies or diseconomies of scale encountered for software projects of different sizes (**DEFAULT = 1.2** -> diseconomy)
- F_i's - cost drivers which model the effect of personnel, computer, product, and project

Software Development Time

$$S_T = C \bullet S_{E_{nom}}^D \bullet \frac{PSCED\%}{100}$$

where

- ***C*** - constant used to capture multiplicative effects on time with projects of increasing effort (**DEFAULT = 6.2**)
- ***D*** - scale factor which accounts for the relative economies or diseconomies of scale encountered for projects of different required efforts (**DEFAULT = 0.32**)
- ***PSCED*** - the percent compression/expansion to the **nominal deployment schedule**

Additional Project Attributes for REVIC

Politecnico di Milano

Cost Drivers	Ratings					
	Very Low	Low	Nominal	High	Very High	Extra High
Project Attributes						
MODP	1.24	1.10	1.00	.91	.82	
TOOL	1.24	1.10	1.00	.91	.83	0.73 0.62 -XXH
REQV		0.91	1.00	1.19	1.38	1.62
REUSE			1.00	1.10	1.30	1.50
SECU			UNCL-1.00	CLASS-1.10		
PLAT	1.00	1.20	1.4	1.6	1.8	2.0 XXH 2.5
SCED	1.23	1.08	1.00	1.04	1.10	

REVIC Maintenance Phase Description

- REVIC estimates the maintenance of a software product over a fifteen year period
- REVIC assumes the presence of a transition period after the delivery of the software
  - Residual errors are found before reaching a steady state condition providing a declining, positive delta to the ACT during the first three years
  - Maintenance Equation

$$MM_{am} = (MM_{nom})\Pi(MF_i)ACT$$

where

  - MM_{nom} is the nominal development effort
  - ACT is the annual change traffic (default value = 15%)
  - MF_i is the set of maintenance adjustment factors

Differences Between REVIC and the Original COCOMO

- REVIC utilizes an updated set of coefficients used in the basic effort and schedule equations
  - Calibrated using recently completed DOD projects
- REVIC provides a single weighted average distribution for effort and schedule over the development phases
  - COCOMO provides a table for distributing the effort and schedule over the development phases
- REVIC allows the user to vary the percentages in the system requirements engineering and DT&E phases
- REVIC automatically calculates the standard deviation for each Computer Software Component (CSC) for risk assessment

COCOMO 2.0 Model

- The original COCOMO and its successor, ADA COCOMO were well-matched to their target software projects
  - Largely custom, build-to-specification software
- COCOMO has been experiencing increasing difficulty in estimating costs and schedules for software developed using new approaches, e.g.
- Object-oriented software
  - Spiral or evolutionary development approaches
  - COTS and reuse-driven approaches
- COCOMO 2.0 is was developed to address these limitations

COCOMO 2.0 Three-Stage Model Series

- **Application Composition Model**
  - Supports prototyping efforts to resolve potential high-risk issues such as user interfaces, software/system interaction, performance, or technology maturity
  - Uses the number of *object points* as the sizing input
- **Early Design Model**
  - Supports exploration of alternative software system architectures and concepts of operation
  - Not enough information is generally available for fine-grain cost estimation during the early stages of the software project
  - Uses the number of *function points* as the sizing input
  - Utilizes 7 cost driver attributes
- **Post-Architecture Model**
  - Same granularity as previous COCOMO and ADA COCOMO
  - Utilizes 17 effort multipliers

COCOMO 2.0
Development Effort Estimates

- **Nominal Person Months**

$$PM_{no\,\min al} = 3.0 \times (Size)^{B}$$

- **COCOMO 2.0 Scaling Approach**

$$B = 1.01 + 0.01\left(\sum W_i\right)$$

  - **Scale factors W_i represent the rating for the following scale drivers:**
    - **Precedentedness**
    - **Development Flexibility**
    - **Architecture/Risk Resolution**
    - **Team Cohesion**
    - **Process Maturity**

Breakage Percentage

- COCOMO 2.0 uses a breakage percentage to adjust the effective size of the product based on the requirements volatility in a project.
- Breakage Percentage
  - The percentage of code thrown away due to requirements volatility
- COCOMO 2.0 adjustment (not used in Application Composition)

$$PM_{no\,\min al} = 3.0\left[\left(\frac{1+BRAK}{100}\right)Size\right]^{B}$$

  - where
    - ***BRAK*** is the breakage percentage

COCOMO 2.0 Re-engineering and Conversion Estimation

- Additional refinement is needed to estimate the costs of software re-engineering or conversion
- COCOMO 2.0 re-engineering and conversion approach

$$PM = 3.0 \times (Size)^{B} + \left[\frac{ASLOC\left(\frac{AT}{100}\right)}{ATPROD} \right]$$

  - where
    - ***AT*** is the percentage of the code that is re-engineered by automatic translation.
    - ***ATPROD*** is the productivity for automatic translation in source statements/person month.

Post-Architecture Model Effort and Schedule Estimates

- **Adjusting Development Effort (Person-Months)**

$$PM_{adjusted} = PM_{no\,\min al} \times \left(\prod_{i=1}^{17} EM_i \right)$$

- **Development Schedule Estimates**

$$TDEV = \left[3.0 \times \left(PM^{(0.33+0.2\times(B-1.01))} \right) \right] \times \frac{SCED\%}{100}$$

- **Output Ranges**
  - **Optimistic Estimate: 0.80($PM_{adjusted}$)**
  - **Pessimistic Estimate: 1.25($PM_{adjusted}$)**

Comparison of COCOMO and COCOMO 2.0

	COCOMO	COCOMO 2.0 A. C. Model	COCOMO 2.0 E. D. Model	COCOMO 2.0 P. A. Model
Size	Delivered Source Instructions (DSI) or Source lines of code (SLOC)	Object Points	Function Points (FP) and Language	FP and Language or SLOC
Reuse	Equivalent SLOC = Linear f(DM, CM, IM)	Implicit in model	% unmodified reuse: SR % modified reuse: nonlinear f(AA, SU, DM, CM, IM)	Equivalent SLOC = nonlinear f(AA, SU, DM, CM, IM)
Breakage	Requirements Volatility rating: (RVOL)	Implicit in model	Breakage %: BRAK	BRAK
Maintenance	Annual Change Traffic (ACT) = %added + %modified	Object Point Reuse Model	Reuse Model	Reuse Model
Scale (b) in $MM_{NOM} = a(Size)^b$	Organic: 1.05 Semidetached: 1.12 Embedded: 1.20	1.0	1.01 - 1.26 depending on the degree of: • precedentedness • conformity • early architecture, risk resolution • team cohesion • process maturity	1.01 - 1.26 depending on the degree of: • precedentedness • conformity • early architecture, risk resolution • team cohesion • process maturity
Product Cost Drivers	RELY, DATA, CPLX	None	RCPX, RUSE	RELY, DATA, DOCU, CPLX, RUSE
Platform Cost Drivers	TIME, STOR, VIRT, TURN	None	Platform difficulty: PDIF	TIME, STOR, PVOL (=VIRT)
Personnel Cost Drivers	ACAP, AEXP, PCAP, VEXP, LEXP	None	Personnel capability and experience: PERS, PREX	ACAP, AEXP, PCAP, PEXP, LTEX, PCON
Project Cost Drivers	MODP, TOOL, SCED	None	SCED, FCIL	TOOL, SCED, SITE

Summary of SW Cost Estimators

	Intermediate COCOMO	COCOMO 2.0 P.A. Model	REVIC
Size	Delivered Source Instructions (DSI) or Source lines of code (SLOC)	FP and Language or SLOC	Delivered Source Instructions (DSI) or Source lines of code (SLOC)
Reuse	Equivalent SLOC = Linear *f*(DM, CM, IM)	Equivalent SLOC = nonlinear *f*(AA, SU, DM, CM, IM)	Equivalent SLOC = Linear *f*(DM, CM, IM)
SW Dev. Effort (MM_{DEV})	*f*(mode, SLOC, 15 cost drivers)	*f*(SLOC, 18 cost drivers)	*f*(mode, SLOC, 19 cost drivers)
Maintenance	*f*(MM_{DEV}, ACT)	*f* (MM_{DEV}, ACT)	*f* (MM_{DEV}, ACT)
SW Dev. Schedule	*f*(mode, MM_{DEV}, SCED%)	*f*(MM_{DEV}, SCED%)	*f*(mode, MM_{DEV}, SCED%)

Module Outline

- Introduction to Cost Modeling-Based Embedded Systems Design
- Software Cost Estimation Process
- Parametric Software Cost Models
- **Parametric Hardware Cost Models**
- Applications of Cost Modeling to the RASSP Design Process
- Summary and Major Issues

ASIC Cost/Schedule Estimation

- Paraskevopoulos and Fey showed that VLSI design schedules are a function of only one parameter:
  - ASIC development effort (person-months)
- Development effort is a function of organizational productivity and design complexity (gates per IC)
- VLSI schedule models are similar to the software schedule models developed by Boehm
- VLSI design classifications
  - full custom
  - cell based
  - standard cells
  - gate arrays

Full Custom ASIC Cost Equation

- **Basic ASIC effort equation**

$$M = (1+D)^{YR}\left[A + B(Size)^{H}\right]$$

  - **where**
    - ***M*** **is the ASIC development effort in person-months**
    - ***D*** **is the average annual improvement factor**
    - ***A*** **is the startup manpower**
    - ***B*** **is a measure of productivity**
    - ***YR*** **is (1984 - current year)**
    - ***H*** **is the economy or diseconomy of scale**
    - **Size is the equivalent number of transistors**

Full Custom Design Complexity Calculation

- **Computation of equivalent number of transistors**

Type of Transistor	# of Equivalent Transistors
Unique Random Logic (UNQ)	1*UNQ Transistors
Repeated Random Logic (RPT)	C*RPT Transistors
Programmed Logic Array (PLA)	E*PLA Transistors
RAM	F*(RAM Transistors)**0.5
ROM	G*(ROM Transistors)**0.5

- **Size equation**

$$Size = UNQ + C \bullet RPT + E \bullet PLA + F \bullet RAM^{0.5} + G \bullet ROM^{0.5}$$

Full Custom Design Parameter Values

- Design effort model parameter values

Parameters	Low Value	Estimate	High Value
A, Constant	0	0	3
B, Productivity	6	12	20
C, Repeated Logic	0.05	0.13	0.25
D, Improvement	-0.05	0.02	0.10
E, PLA	0.1	0.37	0.7
F, RAM	0.1	0.65	1.3
G, ROM	0.05	0.08	0.15
H, Complexity	1.05	1.13	1.40

Gate Array ASIC Design Productivity

- **Basic gate array productivity model**

$$P_B = 17.1 \bullet G^{0.61} \qquad 0.5 \le G \le 25$$

  - **where**
    - ***G*** **is the number of gates used in thousands**
- **More detailed gate array productivity model**

$$P = 16.2\left(0.61^{I}\right)\left(0.86^{U}\right)\left(0.64^{R}\right)\left(1.17^{D}\right)G^{0.6} \qquad 0.5 \le G \le 25$$

  - **where**
    - ***I*** **is the adjusted number of I/Os [(I/O)$^{0.5}$/*K* gates**
    - ***U*** **is the maximum of percentage of gates used minus 90%and 0**
    - ***R*** **is the complexity rating on a scale of 1 to 5**
    - ***D*** **is the number of previous designs completed by the designer**

Gate Array Design Cost

- **Gate array development effort (person-weeks)**

$$M = \frac{200 \bullet G}{P} \qquad 0.5 \le G \le 25$$

  - **where**
    - ***M*** **is the development effort in person-months**
    - ***P*** **can be the detailed or basic productivity values**
- **Example:**
  - **Using basic productivity model**

$$M = 11.7 \bullet G^{0.39}$$

    - **Development effort is proportional to an exponent of the number of gates**

ASIC Schedule Equations

- **Basic ASIC schedule equation**

  $T = h \bullet M^{g}$

  - where
    - ***M*** **is the development effort in person-months**
    - ***h*** **is the model coefficient**
    - ***g*** **is the economy/diseconomy of scale**
- **ASIC schedule models**

Model Name	Functional Form	
VLSI	$T = M$	$M < 6.7$
	$T = 3.5*M^{0.34}$	$M \geq 6.7$
Full Custom	$T = 3.3*M^{0.35}$	$M \geq 6.7$

FPGA vs. ASIC Design Costs

- **Typical cost values for a 10,000 gate ASIC (gate array) or FPGA design**

Cost Attributes	ASIC	FPGA
Engineering Costs	$79,000	$25,000
NRE	$25,000	0
Tools	$10,000	$10,000
Average Price	$13	$39

FPGA vs. ASIC Development Time

- **Typical schedule for a 10,000 gate design**

Time Attributes	ASIC (weeks)	FPGA (weeks)
Engineering Labor	-	4
Training	2	1
Design Capture	3	2
Simulation	2	2
Test-vector Development	6	0
Place & Route	1	1
Back Annotation	1	0
Final Annotation	1	0
Prototype Cycle	2	0
Qualification	5	3
Production Lead time	9	2
Total Time	32	11

Commercial ASIC Hardware Models

- **PRICE-M**
  - Produces estimates of development and production costs/schedule for ASICs and electronic modules (MCMs)
  - Uses parametric cost estimates based:
    - number of transistors/gates
    - percentage of new circuit cells and design repeat
    - specification level
    - degree of computer-aided design
    - product familiarity and engineering experience
- **SEER-IC**
  - Estimates integrated circuit & multi-chip module development and manufacturing costs
  - Utilizes industry wide knowledge bases

Module Outline

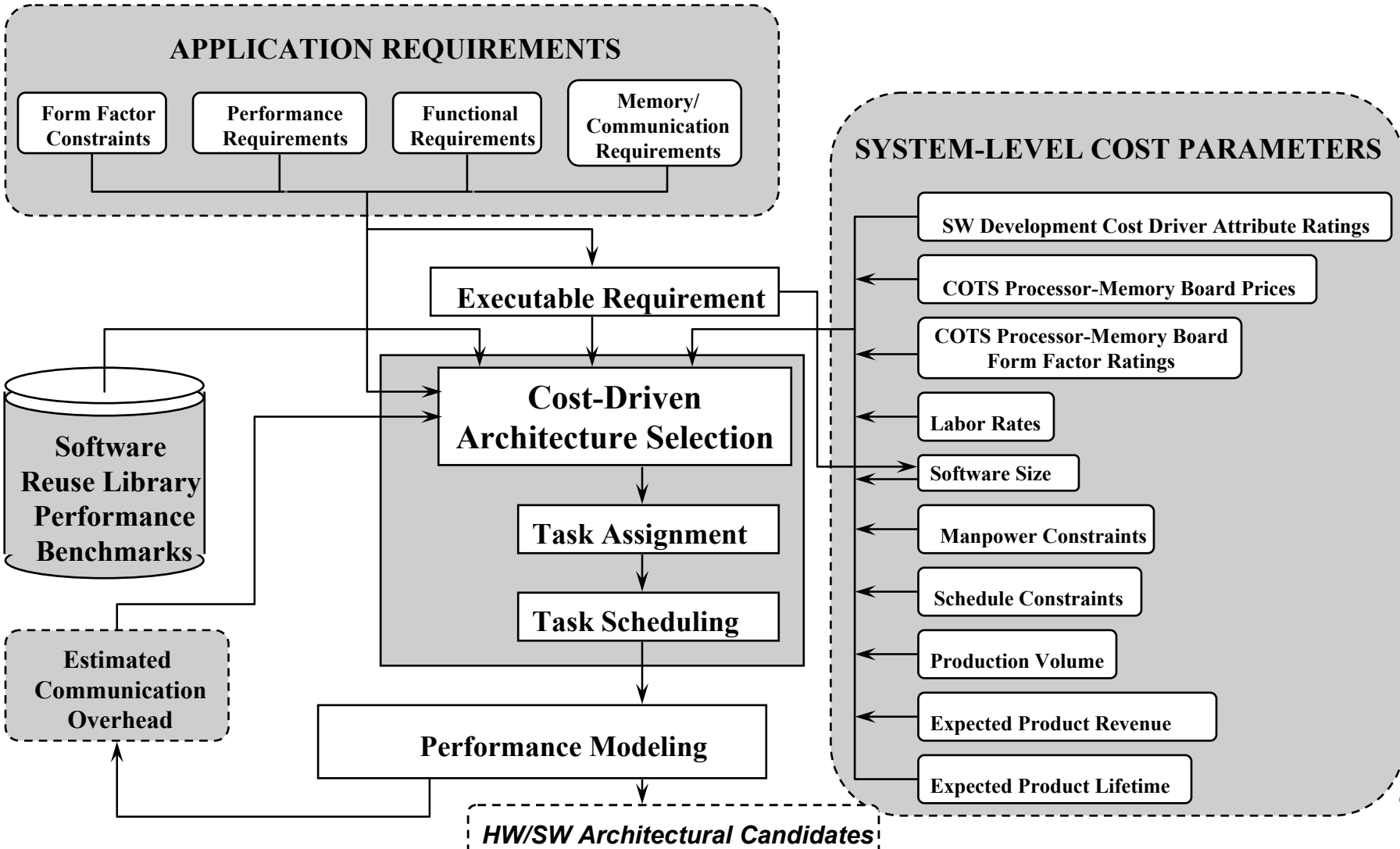
- Introduction to Cost Modeling-Based Embedded Systems Design
- Software Cost Estimation Process
- Parametric Software Cost Models
- Parametric Hardware Cost Models
- **Applications of Cost Modeling to the RASSP Design Process**
- Summary and Major Issues

Classifications of Design and Test Methodologies

- **Methodology I:**
  - Standard COTS - Minimum Hardware Cost
- **Methodology II:**
  - COTS with Cost Modeling - Minimum System Cost
- **Methodology III:**
  - Full Custom - Custom Hardware plus Software
- **Methodology IV:**
  - COTS with Cost Modeling and Virtual Prototyping
- **Methodology V:**
  - COTS with Cost Modeling, Virtual Prototyping, and Use of Advanced Top-Down Programming Methodologies and Reuse

Cost Modeling-Based Front-End Design Approach

Politecnico di Milano



Example: Cost-Driven Architecture Selection Model

Politecnico di Milano

- **Minimize LCC/Maximize Profits**
  - **Objective function:**
    - **SW development cost**
      $C^S s_E +$
    - **SW maintenance cost**
      $C^{MAINT} T^{MAINT} m_E +$
    - **Multiprocessor board cost**
      $V\left[\sum_i \sum_j \left(C_{ij}^P x_{ij} + C^M y\right)\right] +$
    - **Time-to-market cost**
      $R_0 \bullet d(3W - d) / \left(2W^2\right)$
- **Definitions**
  - **SW development effort (REVIC):**

$$s_E = 4.44(KSLOC)^{1.2} \bullet f_E \bullet f_M \bullet f_{SCED} \bullet \prod_{i=1}^{16} F_i$$

  - **SW maintenance effort (REVIC):**

$$m_E = ACT \bullet 4.44(KSLOC)^{1.2} \bullet f_E \bullet f_M \bullet f_{SCED} \prod_{i=1}^{16} MF_i$$

  - **Execution time/main storage effort multipliers:**

$$f_E = 1 + 0.55u_2^p + 1.267u_3^p + 3.6u_4^p$$

$$f_M = 1 + 0.3u_2^m + u_3^m + 3.5u_4^m$$

  - **Processor/memory utilization:**

$$u_m = \frac{M_R}{y} \qquad u_m = \sum_{i=1}^{4} u_i^m \qquad u_p = \sum_{i=1}^{4} u_i^p$$

$$u_p = \frac{T_R}{LCM(RT1, RT2, \ldots) \sum_i \sum_j N_j^P x_{ij}}$$

  - **Time processor utilized per period:**

$$T_R = \sum_j \left(\sum_{k \in T1} W_1 T_{jk}^{exe} q_j + \sum_{k \in T2} W_2 T_{jk}^{exe} q_j + \ldots \right) + COMM$$

  - **Schedule constraint effort mult.:**

$$f_{SCED} = 1.23 - 1.5t_1^{sced} - \left(\frac{8}{15}\right) t_2^{sced} + \left(\frac{2}{15}\right) t_3^{sced} + 0.2t_4^{sced}$$

  - **Percent schedule compression**

$$SCHED = \sum_{i=1}^{4} t_i^{sced}$$

Cost-Driven Architecture Selection Model (Cont.)

- **Time-to-market delay:**
  $$d = s_T - T_{SCHED} + d^-$$
- **SW development time (REVIC):**
  $$s_T = 6.2\left(s_{E_{nom}}\right)^{0.32} \bullet SCHED$$

subject to:

- **Manpower constraint:**
  $$F_{sp} \geq \frac{s_E}{s_T}$$
- **Schedule constraint:**
  $$s_T \leq W$$

- **Design constraints**
  - **Performance**
    $$u_m \leq 1.0 \qquad u_p \leq 1.0$$
  - **Size**
    $$U_A \geq \sum_i \left(A_{ij}^p x_{ij} + A^m y\right)$$
  - **Power**
    $$U_P \geq \sum_i \left(P_{ij}^p x_{ij} + P^m y\right)$$

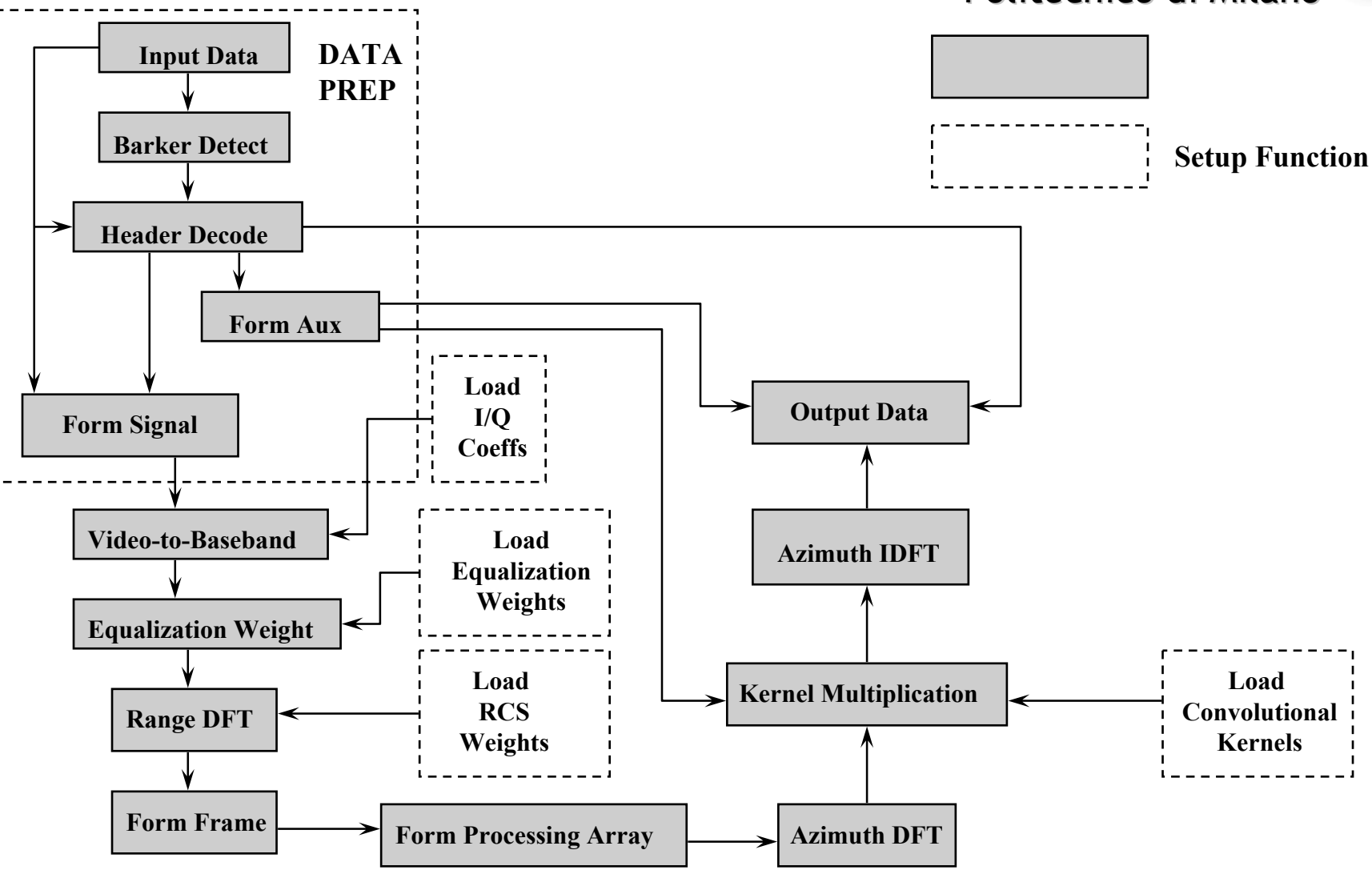
- **Primary decision variables**
  - **total number of processor boards/ packaging type:** x_{ij}
  - **processor type:** q_i
  - **memory size:** y

- The architecture selection problem is formulated as a mixed-integer non-linear programming (MINLP) model.
- GAMS/DICOPT MINLP package can be employed to solve models of this form.
- Linearization techniques are used to transform non-convex constraints in the model to convex constraints.

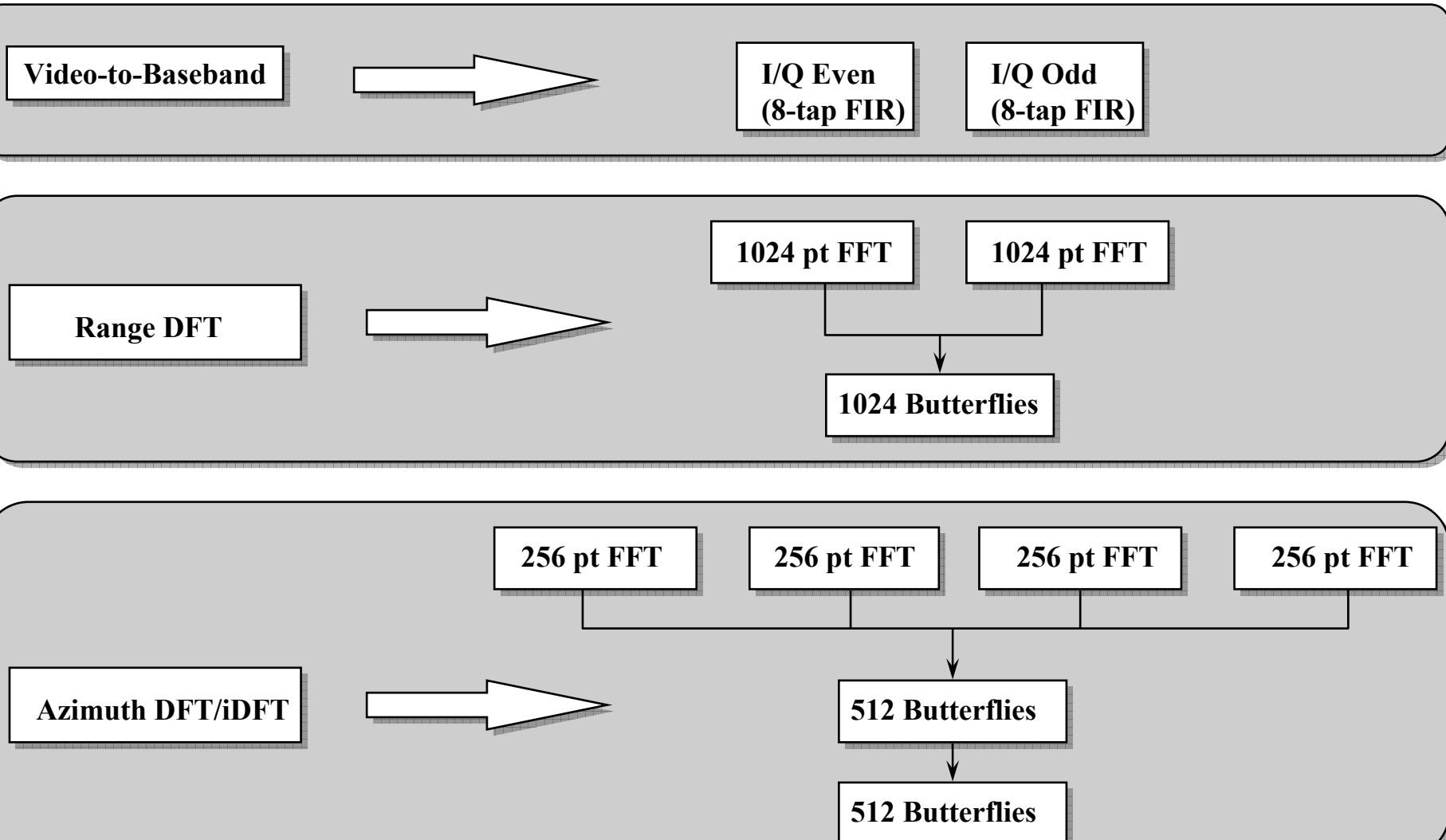
Case Study: SAR Multiprocessor

- SAR Processor is required to form images in real-time on board an F-22 or an unmanned air vehicle (UAV)
- Pulse Repetition Frequency - 556 Hz
  - Delivers 512 pulses in 0.92 seconds
  - Each pulse contains 4064 data samples for each of four polarizations
- Processor must be able to form a 512-pulse image for each of three polarizations in real-time
  - Maximum latency is 3 seconds
- Computational Requirement - 1.1 Gflop/sec
- Memory Requirement - 77 MB

SAR Algorithm: Single Polarization



Functional Decomposition



SAR System-Level Cost Parameters

Parameter	Value
SHARC 2-Processor Board Price (C_{11}^{P})	$8K
SHARC 4-Processor Board Price (C_{12}^{P})	$15K
SHARC 6-Processor Board Price (C_{13}^{P})	$25K
SHARC 8-Processor Board Price (C_{14}^{P})	$40K
SHARC 2-Processor Board Power (P_{11}^{P})	12W
SHARC 4-Processor Board Power (P_{12}^{P})	20W
SHARC 6-Processor Board Power (P_{13}^{P})	25W
SHARC 8-Processor Board Power (P_{14}^{P})	28W
DRAM Price per 4 MB (C^{M})	$1400
Estimated KSLOC	8.75
Maintenance Period (T^{MAINT})	15 years
Software Labor Cost per Person-Month (C^{S})	$15K
Software Maintenance Cost/person-month (C^{MAINT})	$15K
Max. Number of Full-Time SW Personnel (F_{SP})	3
Annual Change Traffic (ACT)	15%
Product Deployment Deadlines (T_{SCHED})	24 months (tight) / 32 months (loose)
Product Life Cycle ($2W$)	36 months
SAR Processor Unit Price	$1M
Expected Product Revenue (R_0)	$1M * Production Volume

SAR Software Cost Driver Ratings

Cost Driver	Situation	Rating	Effort Multiplier
RELY	Local use of system. No serious recovery problems	Nominal	1.00
DATA	256 KBytes	Nominal	1.00
CPLX	real-time signal processing routines	Very high	1.30
VIRT	Based on COTS microprocessor hardware/software (one change every 3 months)	Nominal	1.00
TURN	Two-hour average turnaround time	Nominal	1.00
ACAP	Average senior analysts	Nominal	1.00
AEXP	Three years	Nominal	1.00
PCAP	Average senior programmers	Nominal	1.00
VEXP	Ten months	Nominal	1.00
LEXP	Eighteen months	Nominal	1.00
MODP	Some techniques in use over one year	Nominal	1.00
TOOL	At basic minicomputer tool level	Nominal	1.00
REQV	requirements have a very small amount of ambiguity	Nominal	1.00
REUSE	No reuse is required	Nominal	1.00
SECU	commercial product (unclassified)	Nominal	1.00
PLAT	Unmanned airborne	Nominal	1.40
	Effort adjustment factor (product of effort multipliers)		1.82

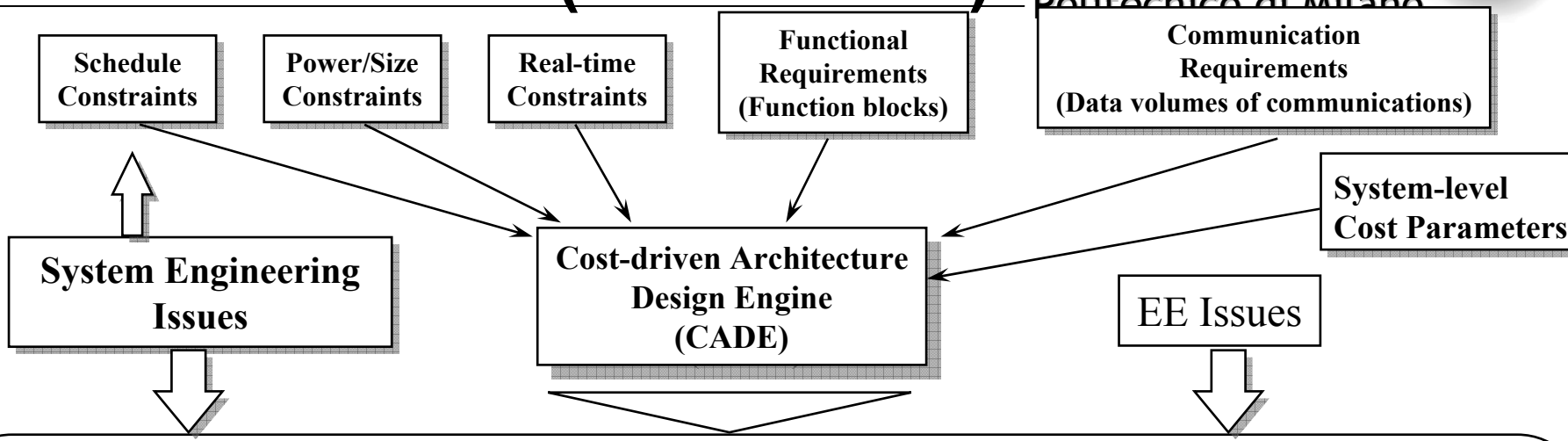
STEP 1

STEP 2

$$\left(\prod_{i=1}^{16} F_i\right)$$

Automated Cost-Driven Architecture Design (RACEWAY)

Politecnico di Milano



Results from SAR Benchmark Conceptual Design

Processor Type	Board Type	Number of Processor Boards
SHARC	2_PROC	1
SHARC	4_PROC	8

Total Number of Processors: 34.00
Total Memory Capacity: 132.00 Mbytes

Processor Utilization: 0.71
Memory Utilization: 0.58
Time-to-market: 23.93 months
Total Life Cycle Cost($M): 4.31
Architecture Sizing Model Solution Time: 7.81 sec

Assignment of Tasks to Processor Architecture

Processor Index	Tasks Assigned
1	HH-DATA_PREP HH-IQ_ODD
2	HH-1024_FFT1 HH-1024_FFT2 HH-1024_BUTT HH-RCS
3	HH-IQ_EVEN HH-EQUALIZE
4	VH-1024_FFT1 VH-1024_FFT2 VH-1024_BUTT VH-RCS
5	VH-IQ_ODD VH-EQUALIZE
6	VH-DATA_PREP VH-IQ_EVEN
7	VV-1024_FFT1 VV-1024_BUTT VV-RCS
8	VV-DATA_PREP VV-1024_FFT2
9	VV-IQ_EVEN VV-EQUALIZE
10	VV-IQ_ODD
11	HH-I256_FFT4 HH-I512_BUTT1 HH-I512_BUTT2
12	HH-KERN_MULT HH-I256_FFT2
13	HH-256_FFT4
14	HH-I256_FFT3
15	HH-256_FFT2
16	HH-I256_FFT1
17	HH-FORM_FRAME HH-PROC_ARRAY HH-256_FFT3
18	HH-256_FFT1 HH-512_BUTT1 HH-512_BUTT2
19	VH-256_FFT2
20	VH-I256_FFT4 VH-I512_BUTT1 VH-I512_BUTT2
21	VH-KERN_MULT VH-I256_FFT2
22	VH-256_FFT4 VH-512_BUTT1 VH-512_BUTT2
23	VH-I256_FFT1
24	VH-PROC_ARRAY VH-256_FFT1
25	VH-I256_FFT3
26	VH-256_FFT3
27	VH-FORM_FRAME VV-I256_FFT2
28	VV-256_FFT3 VV-512_BUTT1 VV-I512_BUTT2
29	VV-256_FFT4
30	VV-I256_FFT3 VV-I512_BUTT1
31	VV-FORM_FRAME VV-PROC_ARRAY VV-256_FFT2
32	VV-512_BUTT2 VV-I256_FFT1
33	VV-256_FFT1
34	VV-KERN_MULT VV-I256_FFT4

SAR Architectural Profiles

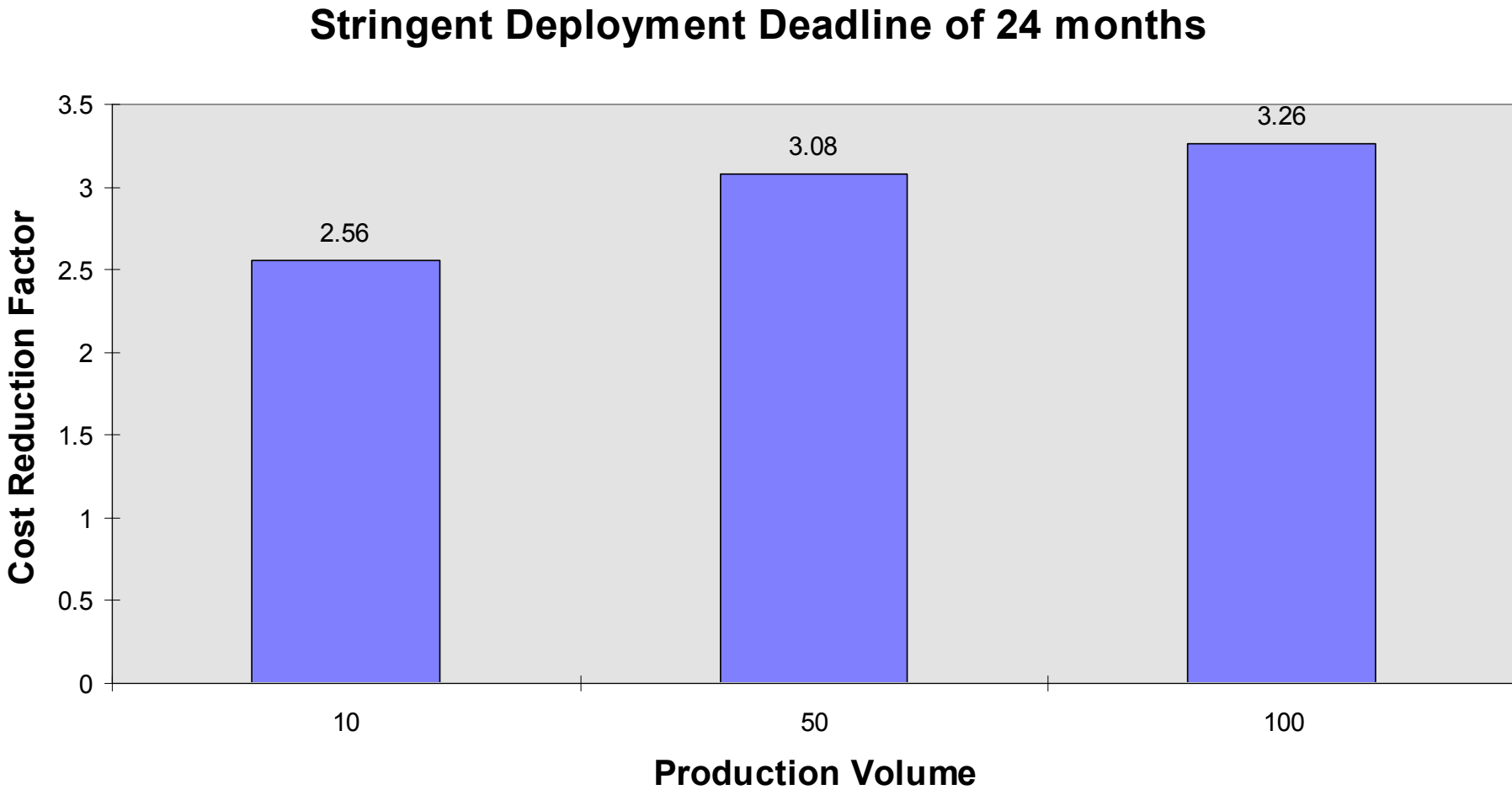
Design Methodology	Volume	Processor Architecture (number of boards per config.)					Number of Proc.	Mem. Alloc.	Utilization (%)	
		Proc. Type	2-Proc.	4-Proc.	6-Proc.	8-Proc.			Proc.	Mem.
I (Min HW)	*	SHARC	1	6	0	0	26	84	95	91
II (CM) Tight Schedule Constraint	10	SHARC	0	9	0	0	36	144	69	53
	50	SHARC	1	8	0	0	34	124	73	62
	100	SHARC	1	8	0	0	34	124	73	62
II (CM) Relaxed Schedule Constraint	10	SHARC	0	9	0	0	36	144	69	53
	50	SHARC	1	7	0	0	30	108	83	71
	100	SHARC	0	7	0	0	28	108	89	71
II (CM) Relaxed Schedule and Power Constraints	10	SHARC	0	0	0	4	32	152	78	50
	50	SHARC	0	0	0	4	32	152	78	50
	100	SHARC	0	0	0	4	32	152	78	50
IV(CM +VP) Tight Schedule Constraint	10	SHARC	0	9	0	0	36	112	69	68
	50	SHARC	0	7	0	0	28	100	89	77
	100	SHARC	0	7	0	0	28	92	89	83
V(IV +RASSP Meth.) Tight Schedule Constraint	10	SHARC	1	7	0	0	30	108	83	71
	50	SHARC	0	7	0	0	28	92	89	83
	100	SHARC	0	7	0	0	28	88	89	87

SAR Benchmark Cost Analysis

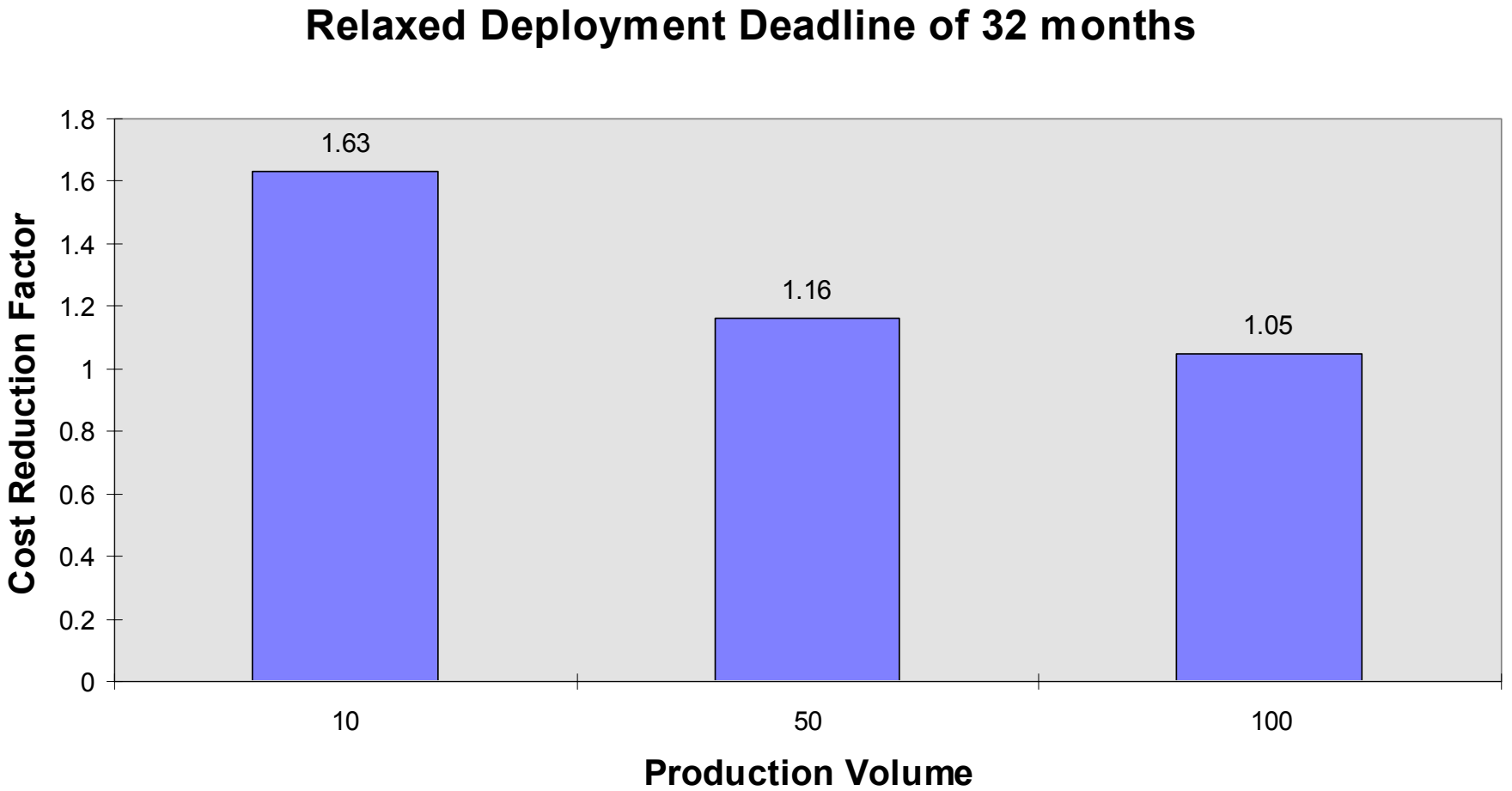
Politecnico di Milano

Design Methodology	Volume	Cost Breakdown (% total cost)				Total Cost ($ millions)	Schedule (months)
		TTM	SW Dev.	SW Maint.	HW Prod.		
I (Min HW) Tight Schedule Constraint	10	36.3	16.6	37.4	9.7	13.1	30.5
	50	63.9	5.9	13.2	17.1	37.2	30.5
	100	70.6	3.2	7.3	18.9	67.3	30.5
II (CM) Tight Schedule Constraint	10	0	19.6	44.2	36.2	5.1	23.8
	50	0	9.0	20.0	71.0	12.1	24.0
	100	0	5.2	11.7	83.1	20.6	24.0
I (Min HW) Relaxed Schedule Constraint	10	0	26.1	58.7	15.3	8.3	30.5
	50	0	16.2	36.4	47.4	13.4	30.5
	100	0	11.0	24.7	64.3	19.8	30.5
II (CM) Relaxed Schedule Constraint	10	0	19.6	44.2	36.2	5.1	23.8
	50	0	10.8	23.9	65.3	11.5	27.6
	100	0	7.6	16.6	75.8	18.8	31.7
II (CM) Tight Schedule and Power Constraints	10	4.0	18.5	41.5	36.1	5.9	24.3
	50	7.7	7.1	15.9	69.3	15.4	24.3
	100	8.7	4.0	9.0	78.3	27.2	24.3
III (Cust.) Tight Schedule Constraint	10	44.1	14.7	33.1	8.2	21.6	40.4
	50	75.9	5.1	11.4	7.7	62.8	40.4
	100	83.4	2.8	6.3	7.6	114.2	40.4
IV (CM +VP) Tight Schedule Constraint	10	0	15.2	34.3	50.5	3.5	19.3
	50	0	7.8	17.6	74.6	9.4	21.5
	100	0	4.8	10.7	84.5	16.2	21.9
V (IV + RASSP Meth.) Tight Schedule Constraint	10	0	12.4	27.8	59.8	2.5	16.4
	50	0	4.8	10.9	84.3	8.1	17.6
	100	0	2.8	6.4	90.8	15.0	18.1

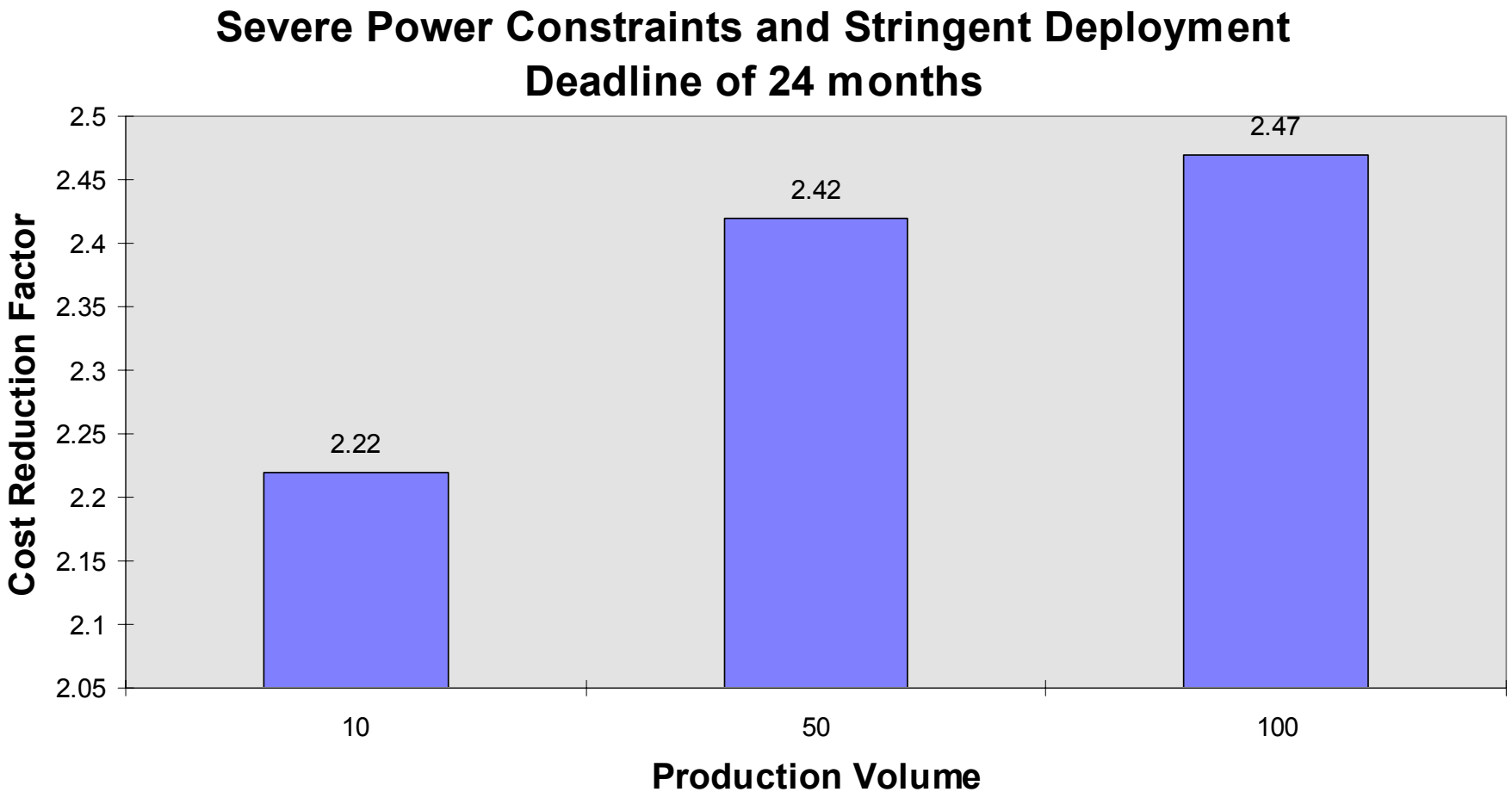
Cost Improvement: Methodology II over I



Cost Improvement: Methodology II over I (Cont.)

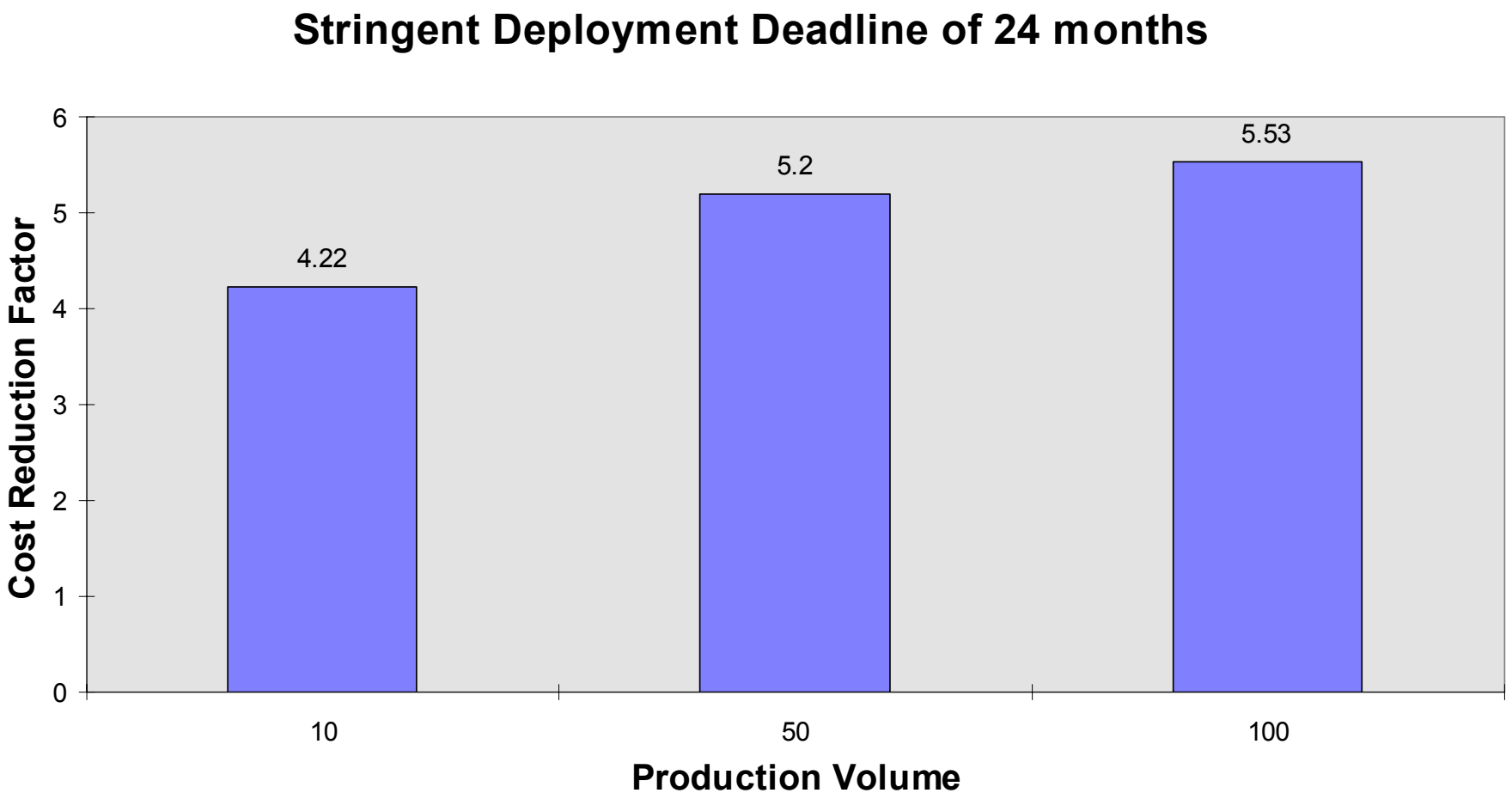


Cost Improvement: Methodology II over I (Cont.)

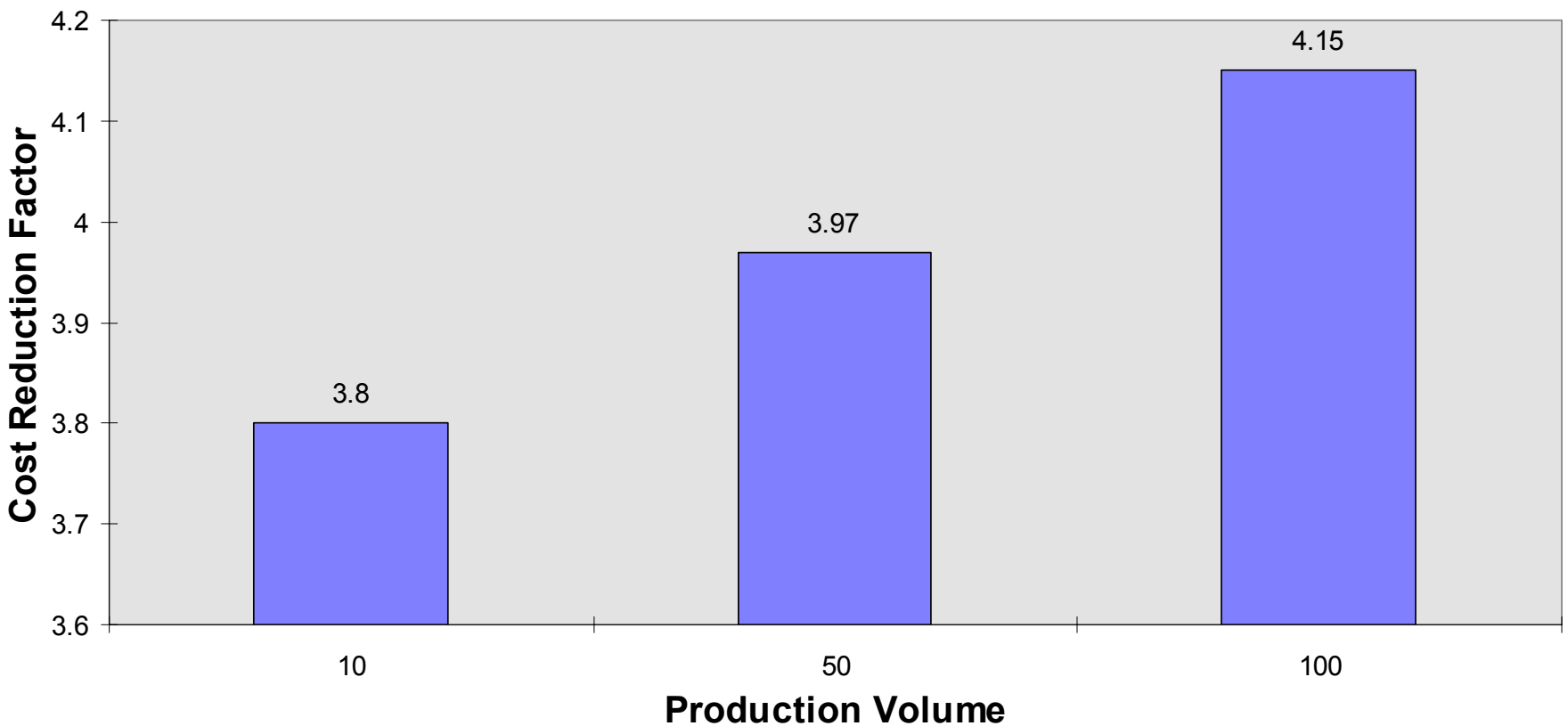


Cost Improvement: Methodology II over III

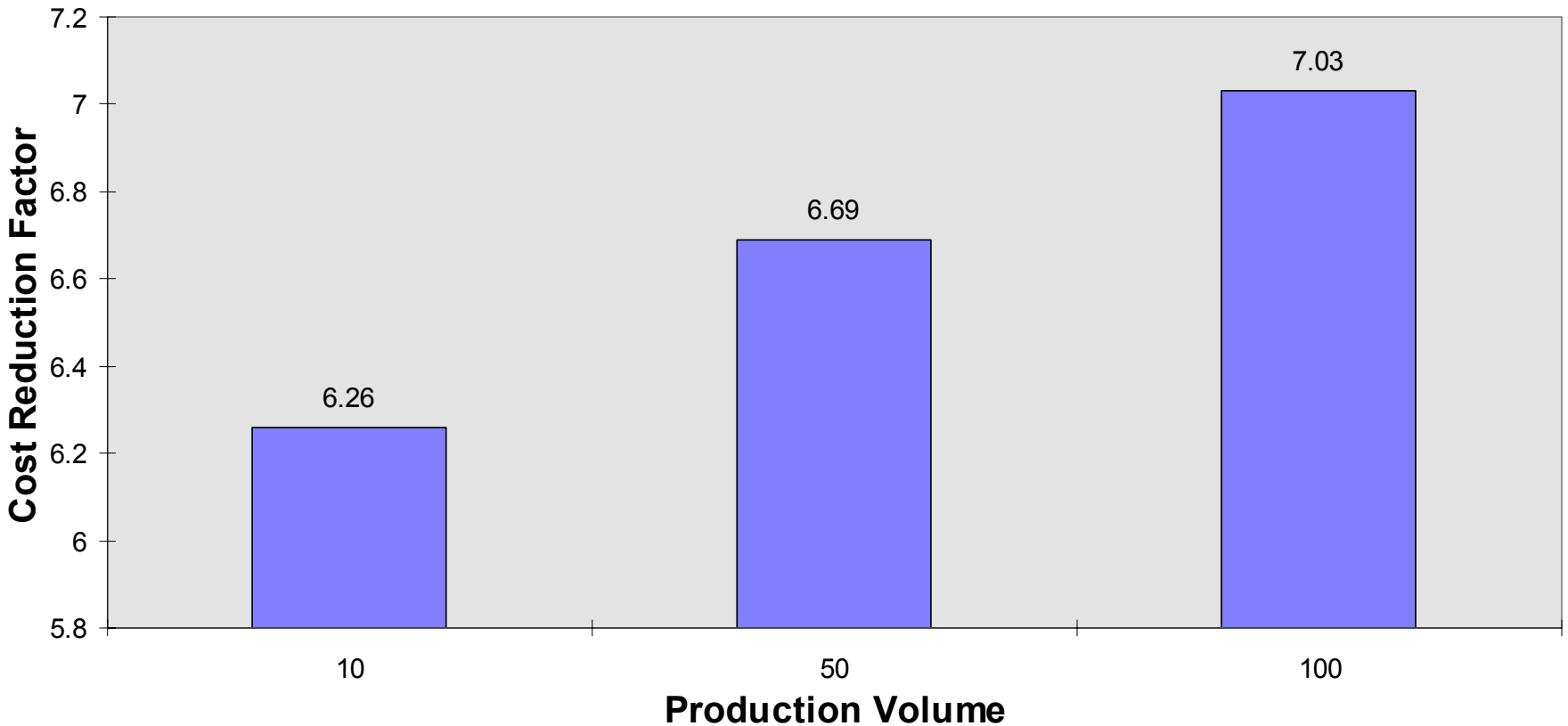
Politecnico di Milano



Cost Improvement: Methodology IV over I

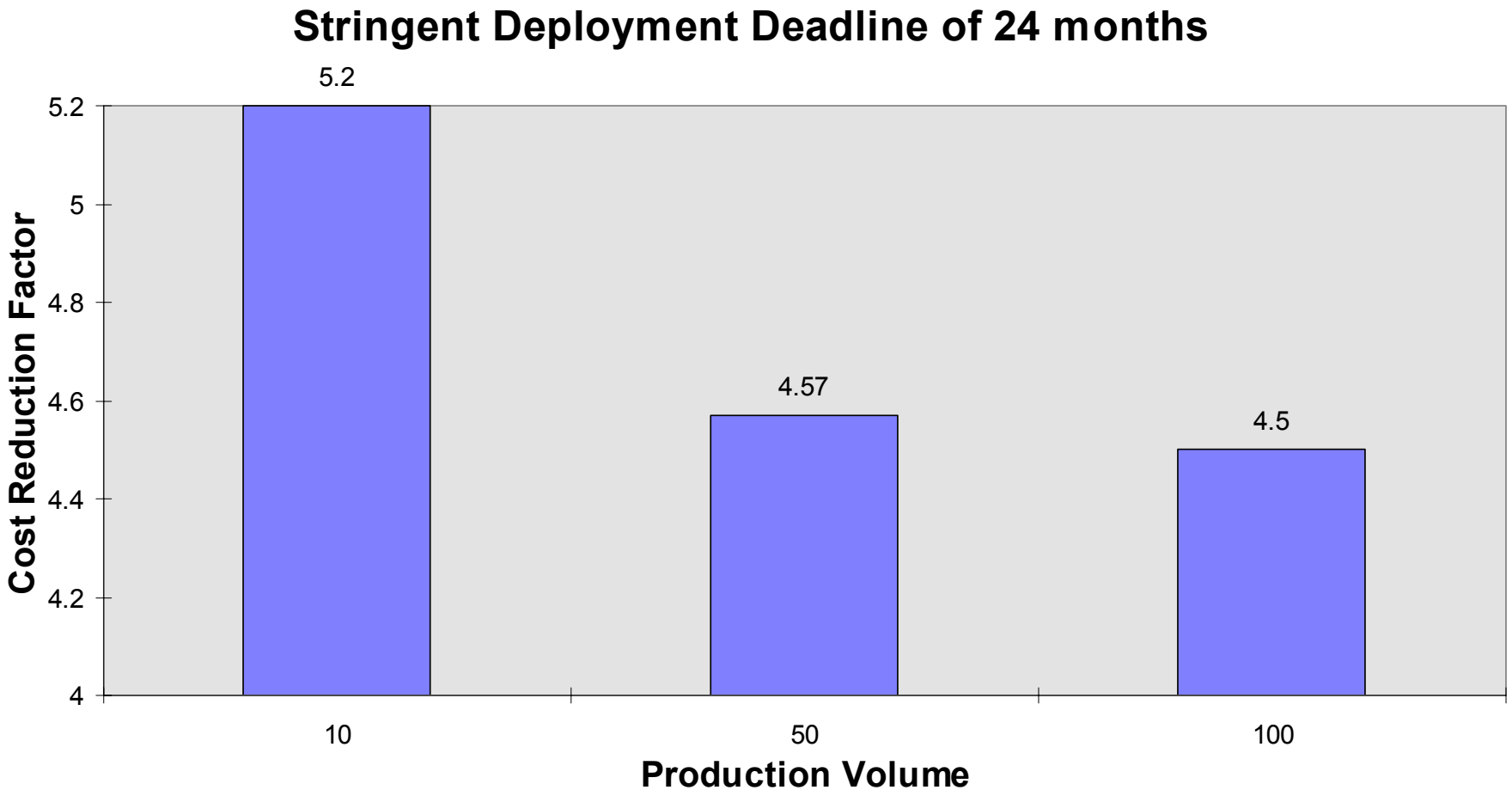


Cost Improvement: Methodology IV over III



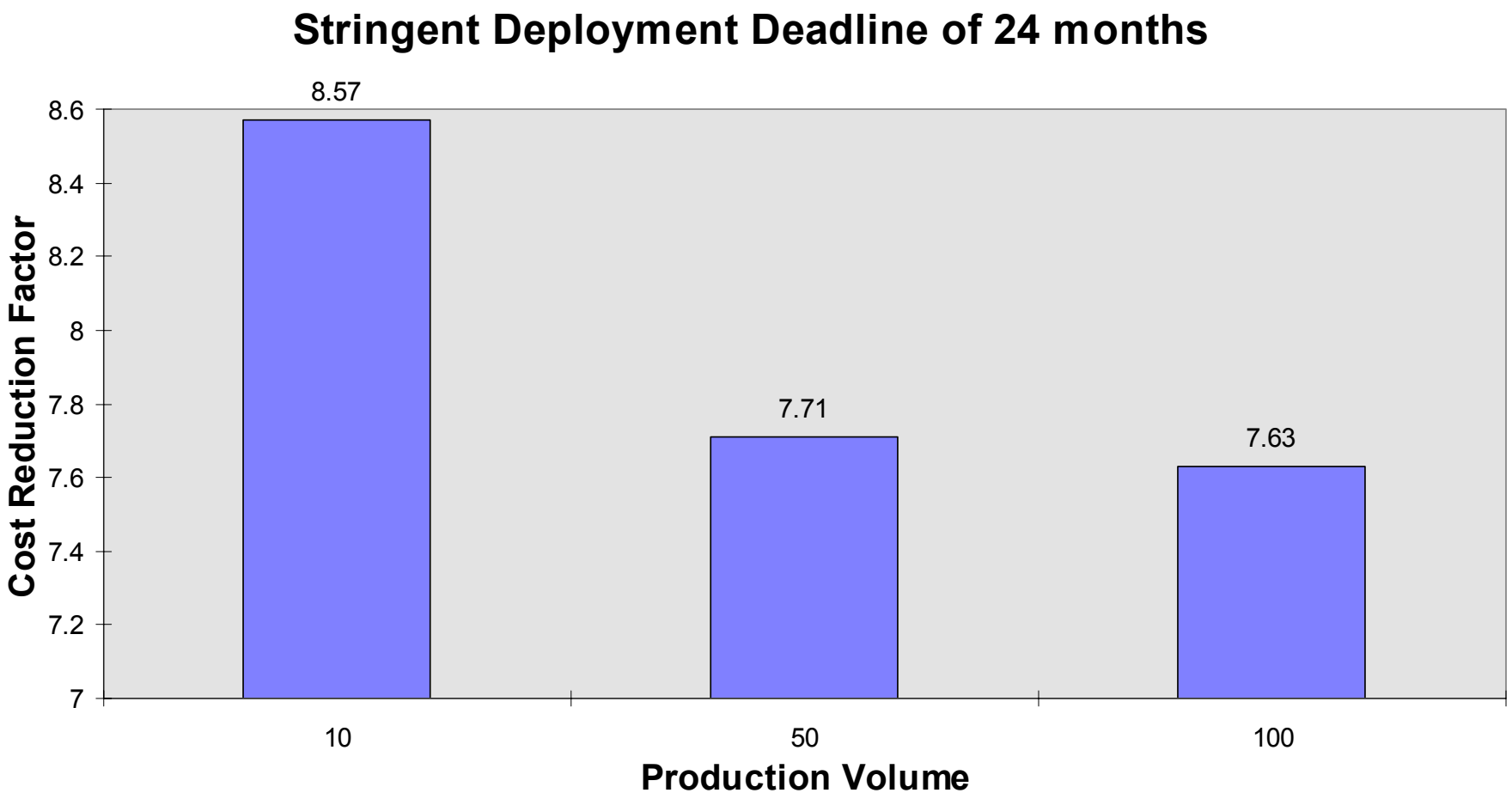
Cost Improvement: Methodology V over I

Politecnico di Milano



Cost Improvement: Methodology V over III

Politecnico di Milano



Module Outline

- Introduction to Cost Modeling-Based Embedded Systems Design
- Software Cost Estimation Process
- Parametric Software Cost Models
- Parametric Hardware Cost Models
- Applications of Cost Modeling to the RASSP Design Process
- Summary and Major Issues

Summary

- This module demonstrated the cost-effectiveness system-level design methodologies that incorporate cost modeling
- In practice, system engineering (SE) drives EE design
  - Front-end SE typically involves less than 4% of the total prototyping time and cost, but accounts for over 70% of a project's life cycle cost
- Cost models, often organization-specific for maximum accuracy, are most effective if used to drive early system design
  - No specific cost model or objective function is recommended for use by all sectors of the electronics industry
  - These cost models can only be developed and refined through historical information collected within the specific industry sectors
  - Cost models can be integrated into the EE aspects of the design process in a seamless manner

Summary (Cont.)

- The focus of this module was on requirements of the SE/EE interface as opposed to the specification
- The SAR case study demonstrated:
  - how modern design and test methodologies can benefit from the automated use of cost models early on in the design process
  - the relationship between time-to-market/cost with the system architecture
  - an order of magnitude improvement in cost-related objective functions
- Parametric cost estimators have recently been used in industry with a great deal of success
  - SEER was used on Motorola's Iridium project, one largest and most complex software projects ever
  - The tool appears to have estimated software development cost within 3% of project actuals

References

- [AHRENS95] J. Ahrens, N. Prywes, "Transition to a Legacy- and Reuse-Based Software Life Cycle," *IEEE Computer*, pp. 27-36, Oct. 1995.
- [AF94] U.S. Air Force Analysis Agency, *REVIC Software Cost Estimating Model User's Manual Version 9.2*, Dec. 1992.
- [AND95] J. Anderson, "Projecting RASSP Benefits," *Proc. Second Annual RASSP Conf.*, ARPA, US Dept. of Defense, Arlington, VA 1995.
- [BOEHM81] B. Boehm, *Software Engineering Economics*, Prentice-Hall, Inc. Englewood Cliffs NJ, 1981.
- [BOEHM88] B. Boehm, "A Spiral Model of Software Development and Enhancement," *IEEE Computer*, pp. 61- 72, May 1988.
- [BOEHM95] B. Boehm, et al., "Cost Models for Future Software Processes: Cocomo 2.0," *Annals of Software Engineering*, 1995.
- [DEB97] J. DeBardelaben, V. Madisetti, A. Gadient, "The Economics of Design & Test of COTS-based Embedded Microelectronics Systems, " to appear in *IEEE Design & Test of Computers*, Fall 1997.
- [DOANE93] D. Doane, P. Franzon, *Multichip Module Technologies and Alternatives - The BASICS*, Van Nostrand Reinhold, 1993.
- [DOD95] Department of Defense, *Parametric Cost Estimating Handbook*, Fall 1995.
- [FEY85] C. Fey, "Custom LSI/VLSI Chip Design Productivity," *IEEE Journal of Solid-State Circuits*, Vol. sc-20, No. 2, April 1985, pp. 555-561.

References (Cont.)

- [FP89] C. Fey, D. Paraskevopoulos, "Studies in VLSI Technology Economics IV: Models for Gate Array Design Productivity," *IEEE Journal of Solid-State Circuits*, Vol. 24, No. 4, August 1989, pp. 1085-1091.
- [LEVITT92] M. Levitt, "Economic and Productivity Considerations in ASIC Test and Design-for-Test," *Digest of Papers: Compcon '92*, pp. 440-445, Spring 1992.[LIU95] J. Liu, "Detailed Model Shows FPGAs' True Costs," *EDN*, pp. 153-158, May 11, 1995.
- [MAD95] V. Madisetti, T. Egolf, "Virtual Prototyping of Embedded Microcontroller-Based DSP Systems," *IEEE Micro*, Oct. 1995.
- [MAD96] V. Madisetti, "Rapid Digital System Prototyping: Current Practice, Future Challenges," *IEEE Design & Test of Computers*, Fall 1996, pp. 12-22.
- [MINK95] A. Minkiewicz, A. DeMarco, " The PRICE Software Model," PRICE Systems Internal Document, June 1995.
- [PF87] D. Paraskevopoulos, C. Fey, "Studies in LSI Technology Economics III: Design Schedules for Application-Specific Integrated Circuits," *IEEE Journal of Solid-State Circuits*, Vol. sc-22, No. 2, April 1987, pp. 223-229.
- [SEPO96] Software Engineering Process Office (SEPO), NCCOSC RDTE DIV, *Software Size, Cost, and Schedule Estimation Process Version 2.0*, March 1996.
- [ZUERN94] B. Zuerndorfer, G. Shaw, "SAR Processing for RASSP Application," *Proc. 1st Annual RASSP Conf.*, pp. 253-268, Arlington, VA, August 1994.