SOFTWARE ECONOMICS

UNIT 15

Presented by

Stefano Street

Objectives

- Software economics – its importance and place in software systems
- Provide an empirical view of where money goes
- Why it is important to understand and control software costs
- Identify shortcomings in existing work
- Discuss economic perspective on software design promises
- Review state of the art in software economics
- Provide a roadmap

Introduction

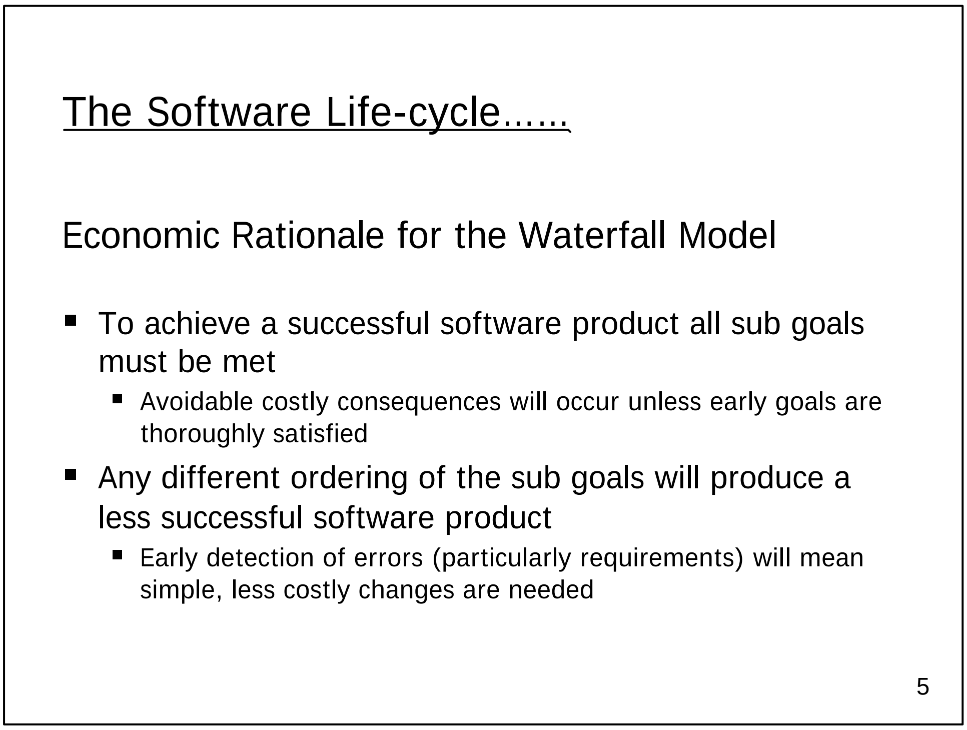
- Rapid, sustained advances in computing & communications
- Hardware is seen as the catalyst, however software embodies new value added functions
- Despite this dependence on software its production is among the most complex and problematical aspects of modern technology development
- Lack of management in risk-return characteristics of software is a serious & difficult problem

The Software Life-cycle

Quick reminder – phases

- Feasibility
- Requirements
- Product Design
- Detailed Design
- Coding
- Integration
- Implementation
- Maintenance

Each phase is culminated by verification & validation

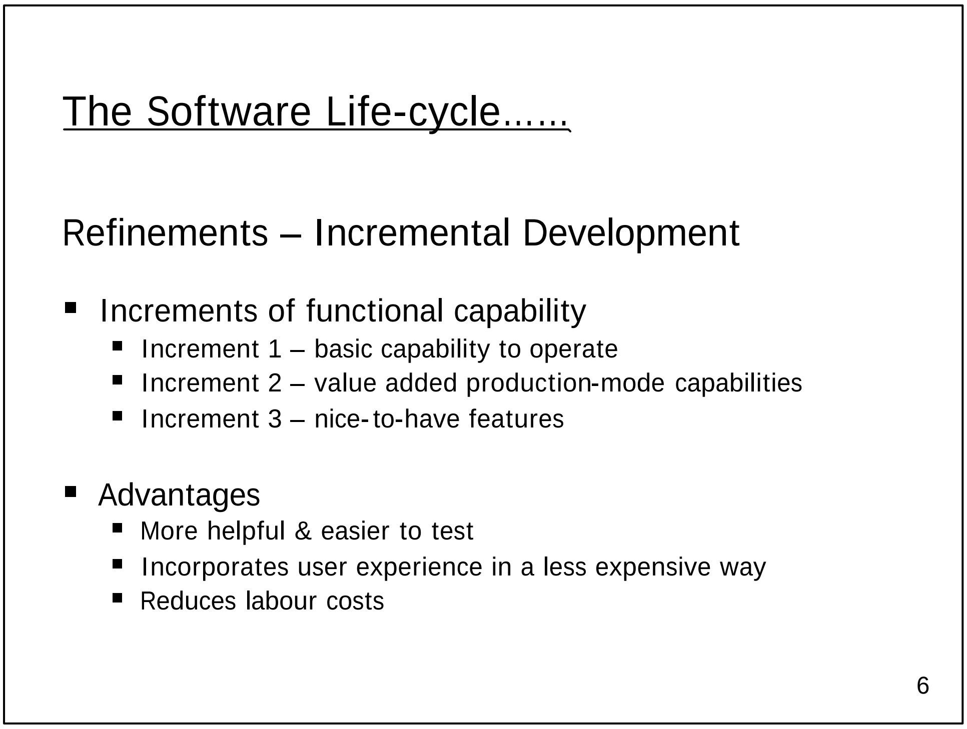


The Software Life-cycle……

Economic Rationale for the Waterfall Model

- To achieve a successful software product all sub goals must be met
  - Avoidable costly consequences will occur unless early goals are thoroughly satisfied
- Any different ordering of the sub goals will produce a less successful software product
  - Early detection of errors (particularly requirements) will mean simple, less costly changes are needed

5



The Software Life-cycle……

Refinements – Incremental Development

- Increments of functional capability
  - Increment 1 – basic capability to operate
  - Increment 2 – value added production-mode capabilities
  - Increment 3 – nice-to-have features

- Advantages
  - More helpful & easier to test
  - Incorporates user experience in a less expensive way
  - Reduces labour costs

6

The Software Life-cycle……

Refinements – advancemanship

- Anticipatory documentation
  - Define detailed objectives & plans for future software development activities
  - Produce early versions of user documentation
- Software scaffolding
  - Extra products that need to be developed to ensure smooth & efficient build of main software
- Advantages
  - Reduce overall costs by limiting the time & energy spent in none productive activities
  - Redistribute costs – greater early investment reduces late investment costs

COCOMO Model

COnstructive COst MOdel

- Basic COCOMO
  - Small-to-medium size products developed
- Good for quick, early, rough order of magnitude estimates of software costs
- Other levels
  - Intermediate / detailed (not covered in this lecture)

COCOMO Model

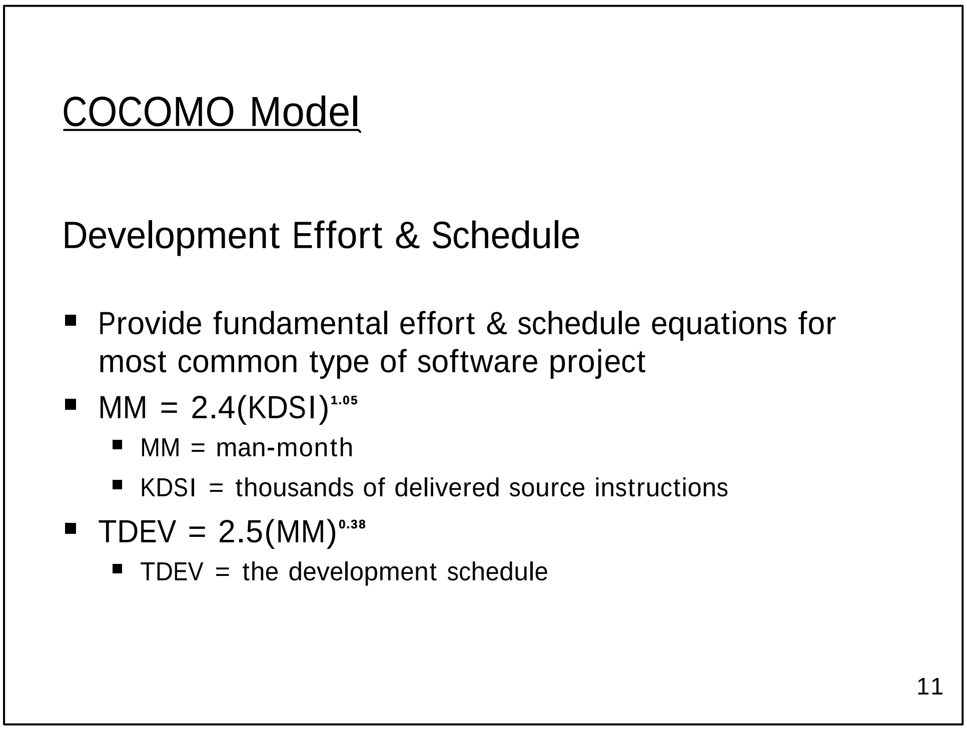
Definitions & Assumptions

- Primary cost driver is the number of Delivered Source Instructions (DSI) developed by the project
- Development period begins at design phase & ends at integration and test phase
- Covers those activities indicated on the software Work Breakdown Structure (WBS)
- Covers all direct-charged labour on project
- A man-month consists of 152 hrs of working time
- Non-productive time kept to a minimum
- Assumes that requirements spec isn't substantially changed after the plans & requirements phase

COCOMO Model

Man-Month versus Dollar Estimates

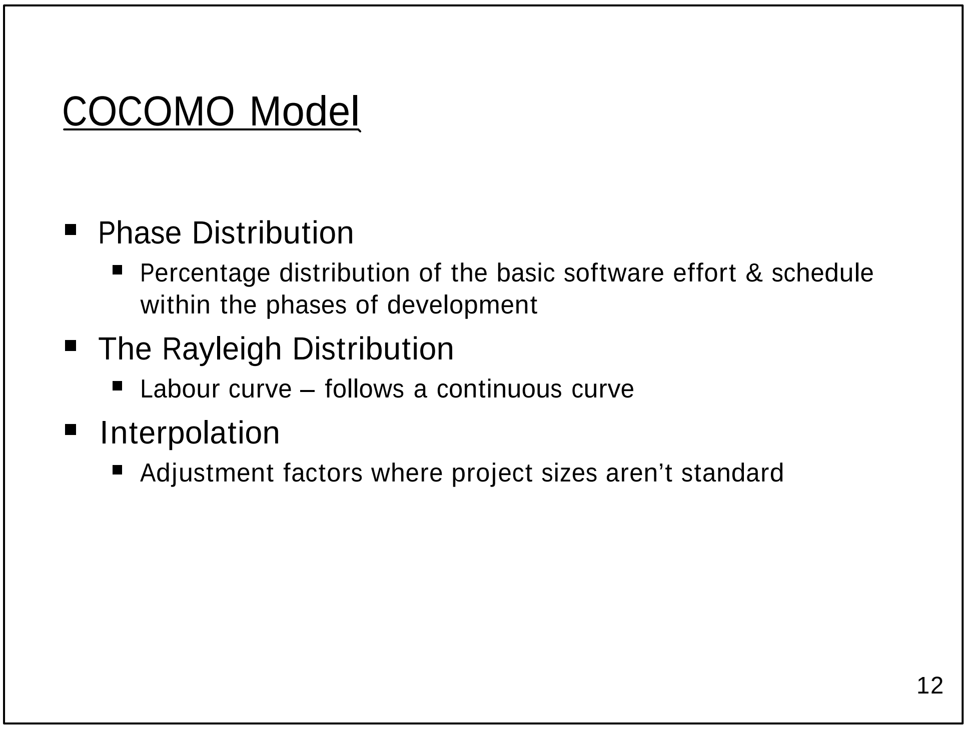
- Labour costs in dollars is avoided
- Man-Months are a stable quantity
- To convert man-months into dollar estimates apply average dollar per man-month figures



COCOMO Model

Development Effort & Schedule

- Provide fundamental effort & schedule equations for most common type of software project
- $MM = 2.4(KDSI)^{1.05}$
  - MM = man-month
  - KDSI = thousands of delivered source instructions
- $TDEV = 2.5(MM)^{0.38}$
  - TDEV = the development schedule



COCOMO Model

- Phase Distribution
  - Percentage distribution of the basic software effort & schedule within the phases of development
- The Rayleigh Distribution
  - Labour curve – follows a continuous curve
- Interpolation
  - Adjustment factors where project sizes aren’t standard

COCOMO Model

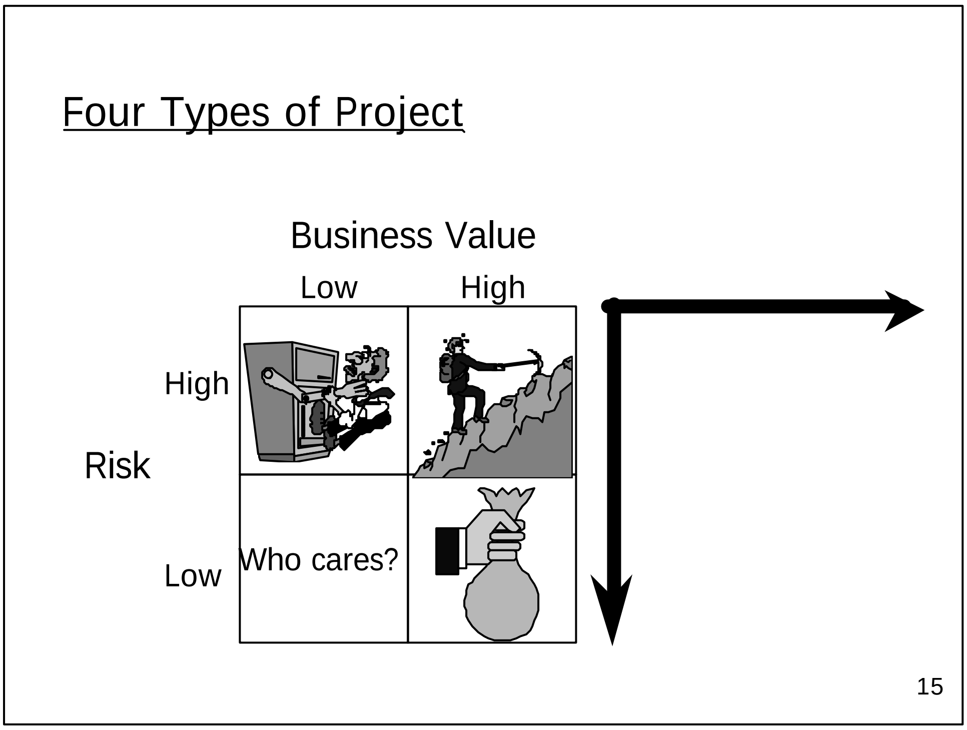
Three COCOMO Modes of Software Development

- The organic mode
  - Small software teams
  - Stable development environment
- The semidetached mode
  - Represents an intermediate stage – project characteristics or mix of organic & embedded modes
- The embedded mode
  - Operate within tight constraints
  - Early completion is beneficial

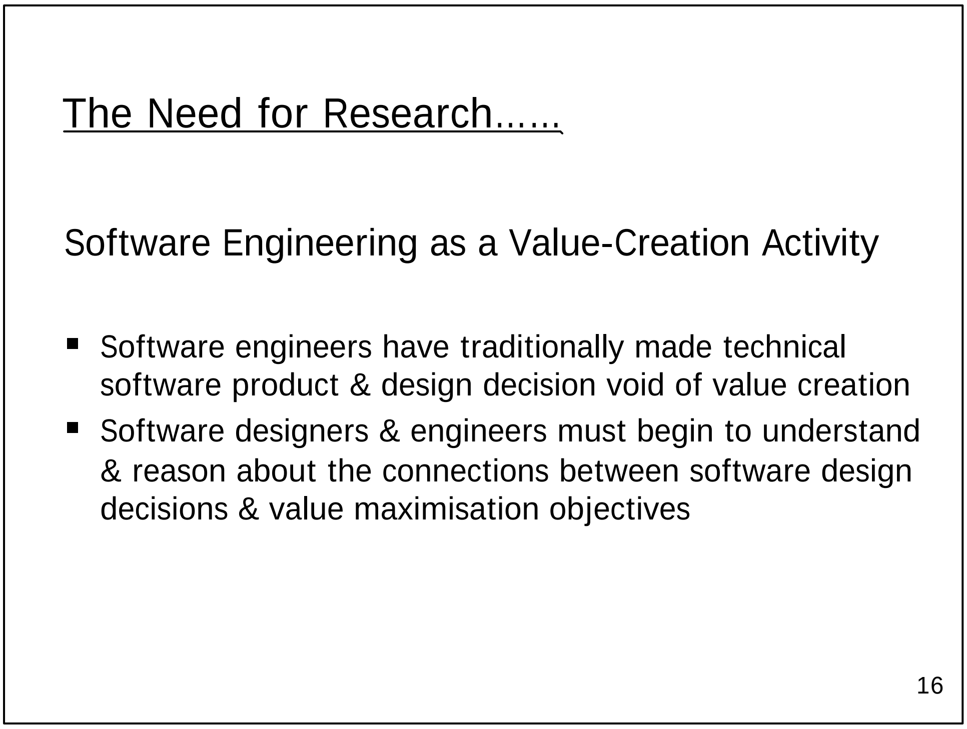
The Need for Research

Software Engineering Decision-Making Today

- Principal of separation of concerns
- Design decisions are now intimately coupled with fundamental business decisions in every endeavour
- Software development involves the investment of valuable resources



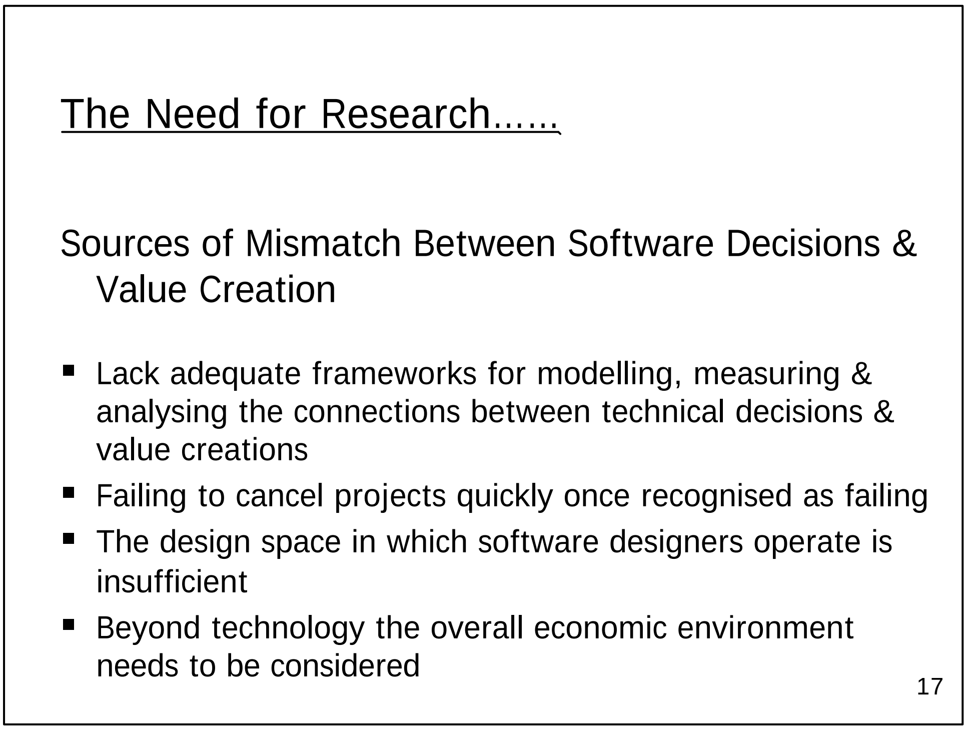
Four Types of Project



The Need for Research……

Software Engineering as a Value-Creation Activity

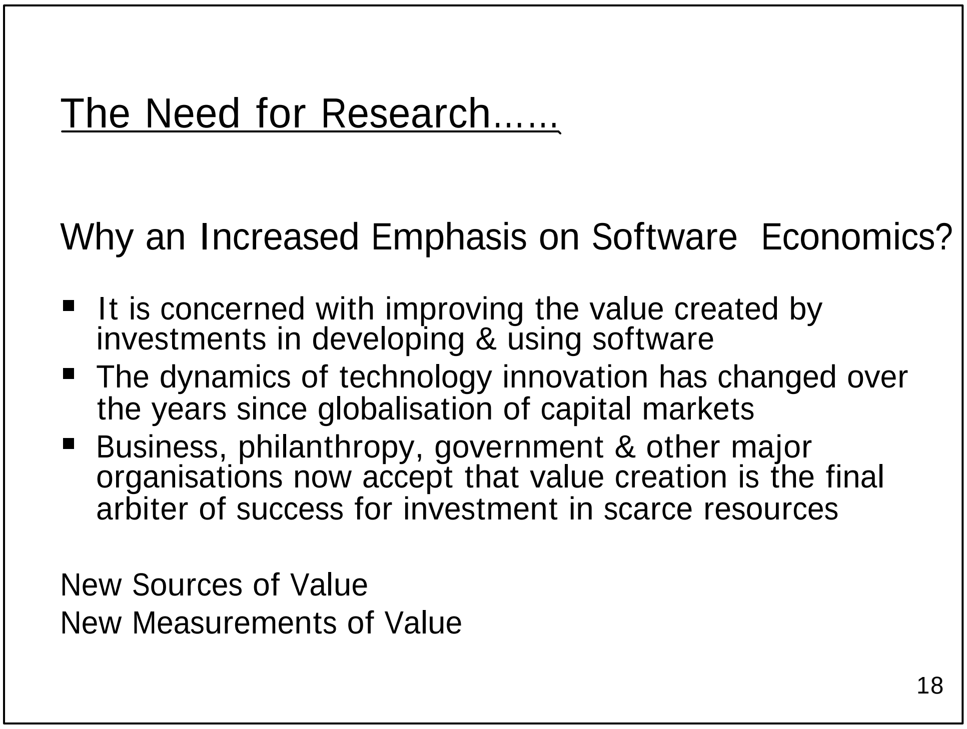
- Software engineers have traditionally made technical software product & design decision void of value creation
- Software designers & engineers must begin to understand & reason about the connections between software design decisions & value maximisation objectives



The Need for Research……

Sources of Mismatch Between Software Decisions & Value Creation

- Lack adequate frameworks for modelling, measuring & analysing the connections between technical decisions & value creations
- Failing to cancel projects quickly once recognised as failing
- The design space in which software designers operate is insufficient
- Beyond technology the overall economic environment needs to be considered



The Need for Research……

Why an Increased Emphasis on Software Economics?

- It is concerned with improving the value created by investments in developing & using software
- The dynamics of technology innovation has changed over the years since globalisation of capital markets
- Business, philanthropy, government & other major organisations now accept that value creation is the final arbiter of success for investment in scarce resources

New Sources of Value
New Measurements of Value

The Need for Research……

Shortcomings that Need to be Addressed

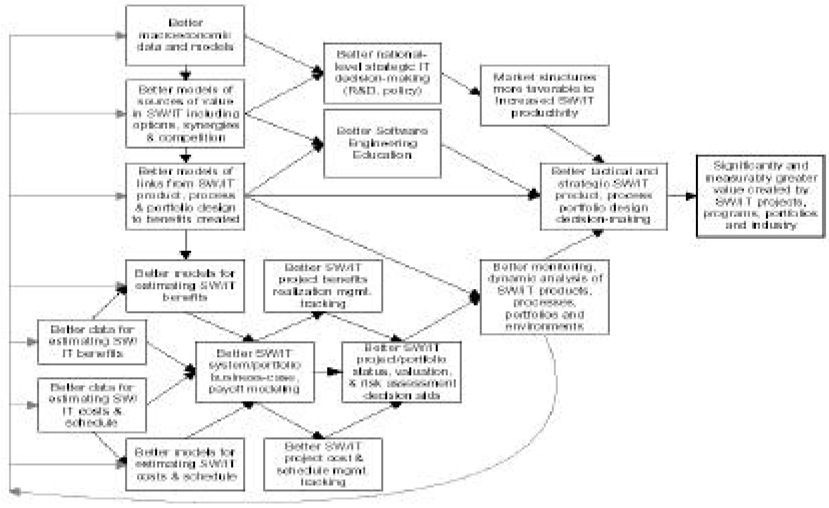
- Currently our ability to reason about software costs is stronger than that about software benefits
  - The ability to reason about both costs & benefits (sometimes in sophisticated terms) is a critical success factor for future enterprises
  - Most software cost & schedule models are calibrated to a minimal cost strategy
  - Better software development estimation models are needed
  - Software economics needs to move from static notions of (usually uncertain) cost & benefit to dynamic & strategic concepts of value creation

Software Economics Roadmap

The Goal of Software Economics

- Develop fundamental knowledge to enable significant, measurable increases in the value created over time
- There are subtleties
- *Value* itself can be a complex & subtle quantity
- We identify a network of important intermediate outcomes

Software Economics Roadmap……


Better macroeconomic data and models
Better models of sources of value in SW/IT including options, synergies & competition
Better models of links from SW/IT product, process & portfolio design to benefits created
Better national-level strategic IT decision-making (R&D, policy)
Better Software Engineering Education
Market structures more favorable to Increased SW/IT productivity
Better tactical and strategic SW/IT product, process portfolio design decision-making
Significantly and measurably greater value created by SW/IT projects, programs, portfolios and industry
Better models for estimating SW/IT benefits
Better SW/IT project benefits realization mgmt. tracking
Better monitoring, dynamic analysis of SW/IT products, processes, portfolios and environments
Better data for estimating SW/IT benefits
Better SW/IT system/portfolio business-case, payoff modeling
Better SW/IT project/portfolio status, valuation, & risk assessment decision aids
Better data for estimating SW/IT costs & schedule
Better models for estimating SW/IT costs & schedule
Better SW/IT project cost & schedule mgmt. tracking

Roadmap for Research in Software Engineering Economics

Software Economics Roadmap……

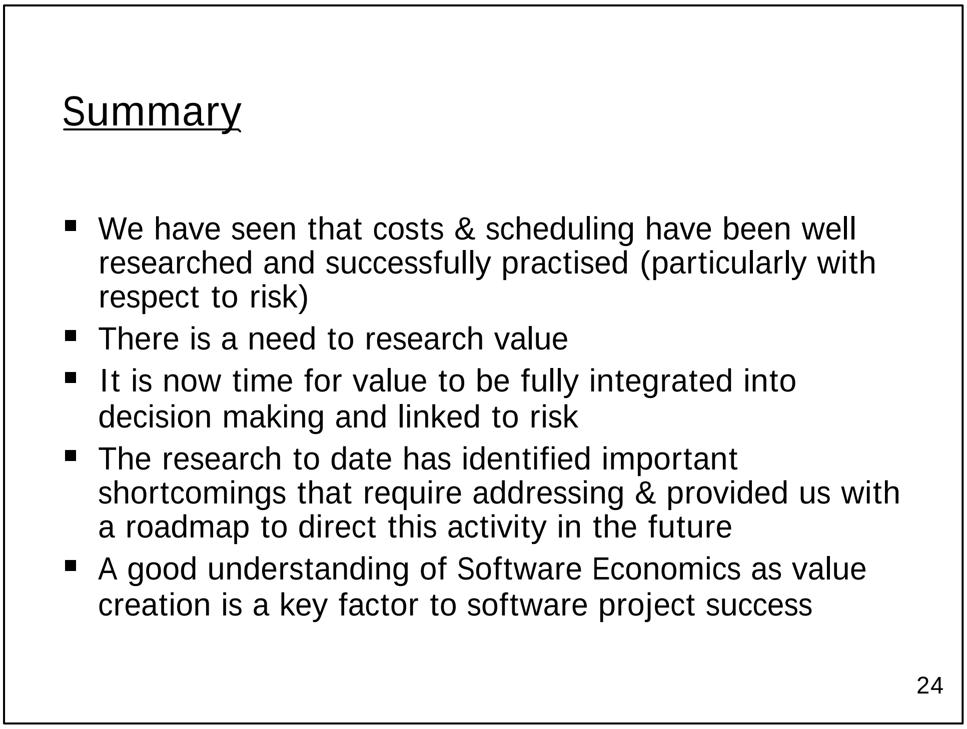
Making Decisions that are Better for Value Creation

- The goal of the roadmap is supported by a key intermediate outcome:
  - designers at all levels must make design decisions that are better for value added than those they make today
- Better decision-making is the key enabler of greater value added
- Design decision making depends in turn on other advances
- Now we look at the roadmap in more detail……

Software Economics Roadmap……

Richer Design Spaces

- The space in which software designers operate today is inadequate
- The market structures within which software development occurs are still primitive in comparison to those supporting other industries
- Less able to build systems from specialized, efficiently produced, volume-priced third-party components
- Less able to use markets to manage risk through warranties, liability insurance, etc.
- Inability to manage risk by the use of market mechanisms is a major hindrance to efficient production



Summary

- We have seen that costs & scheduling have been well researched and successfully practised (particularly with respect to risk)
- There is a need to research value
- It is now time for value to be fully integrated into decision making and linked to risk
- The research to date has identified important shortcomings that require addressing & provided us with a roadmap to direct this activity in the future
- A good understanding of Software Economics as value creation is a key factor to software project success