

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/224001127>

# Software Architecture In Practice

Chapter · January 2003

CITATIONS

5,085

READS

97,145

3 authors, including:



[Len Bass](#)

National ICT Australia Ltd

105 PUBLICATIONS 9,263 CITATIONS

[SEE PROFILE](#)



[Rick Kazman](#)

Carnegie Mellon University

349 PUBLICATIONS 18,288 CITATIONS

[SEE PROFILE](#)



Carnegie Mellon University  
**Software Engineering Institute**

# Software Architecture in Practice

**Paul C. Clements**  
**Software Engineering Institute**  
**Carnegie Mellon University**  
**Pittsburgh, PA 15213-3890 USA**

**Sponsored by the U.S. Department of Defense**  
**© 2002 by Carnegie Mellon University**



# Schedule and Outline

**0900 - 0915 Introductions**

**0915 - 0945 The Architecture Business Cycle**

**0945 -1000 What is architecture?**

**1000 - 1030 Why is architecture important?**

**1030 - 1045 Break**

**1045 - 1115 Architectural structures**

**1115 - 1200 New developments in software architecture**



# Schedule and Outline

0900 - 0915 Introductions

**0915 - 0945 The Architecture Business Cycle**

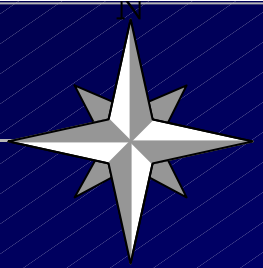
0945 -1000 What is architecture?

1000 - 1030 Why is architecture important?

1030 - 1045 Break

1045 - 1115 Architectural structures

1115 - 1200 New developments in software architecture



# Factors Influencing Architectures

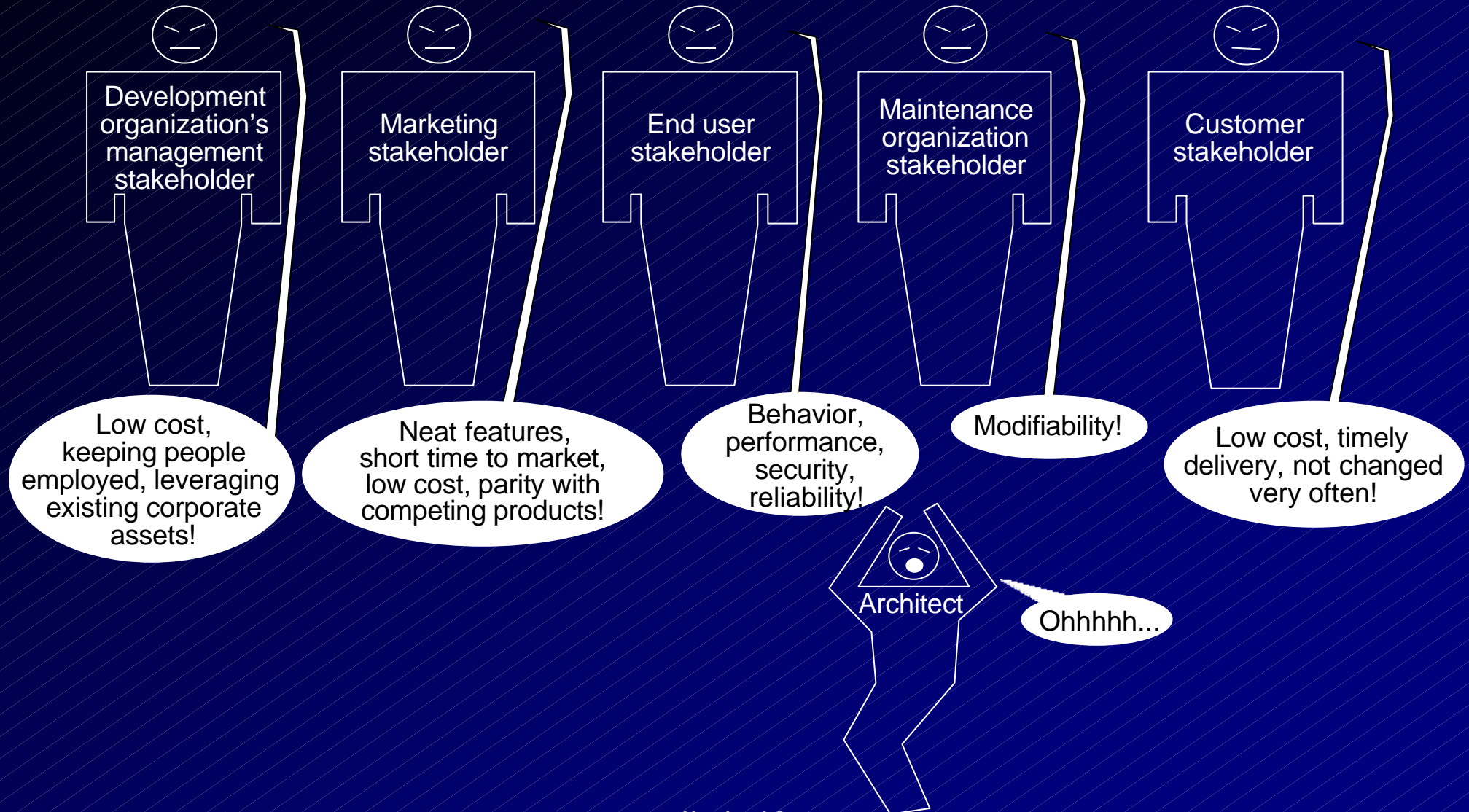
**Architectures are influenced by**

- **stakeholders of a system**
- **technical and organizational factors**
- **architect's background**





# Stakeholders of a System





# Development Organization Concerns

## Business issues

- investing in, and then amortizing the infrastructure
- keeping cost of installation low
- investing in, and then utilizing personnel

## Organizational structure issues

- furthering vested interests, e.g.,
  - maintaining an existing database organization
  - supporting specialized expertise
- maintaining the standard method of doing business



# Technical Environment

**Current trends: today's information system will likely employ a**

- **database management system**
- **Web browser for delivery and distribution across platforms**

**This was not true 10 years ago.**

**Available technology: decisions on using a centralized or decentralized system depend on processor cost and communication speed; both are changing quantities.**





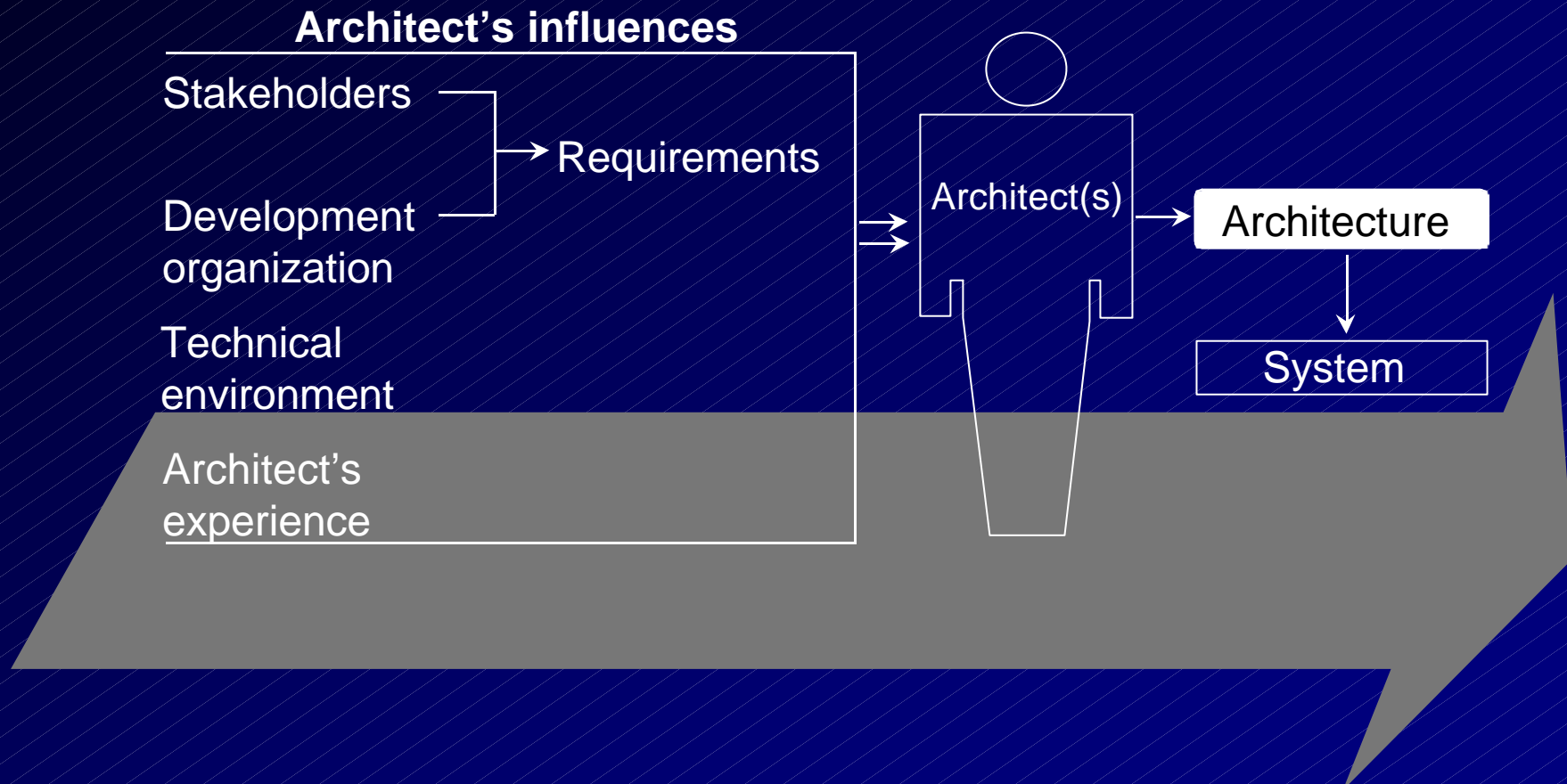
# Architect's Background

**Architects develop their mindset from their past experiences.**

- **Prior good experiences will lead to replication of prior designs.**
- **Prior bad experiences will be avoided in the new design.**



# Summary: Influences on the Architect





# What Makes a Good Architect?

**People skills: must be able to**

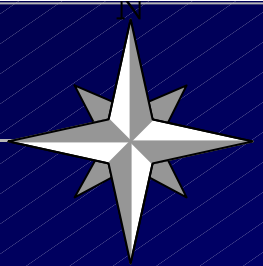
- negotiate competing interests of stakeholders
- promote inter-team collaboration

**Technical skills: must understand**

- the relationships between qualities and structures
- current technology
- that most requirements for an architecture are not written down in any requirements document

**Communication skills: must be able to**

- clearly convey the architecture to teams (both verbally and in writing)
- listen to and understand multiple viewpoints



# Factors Influenced by Architectures

**Structure of the development organization**

**Enterprise goals of the development organization**

**Customer requirements**

**Architect's experience**

**Technical environment**

**The architecture itself**



# Architecture Influences the Development Organization Structure

**Short term: work units are organized around architectural units for a particular system under construction.**

**Long term: when company constructs a collection of similar systems, organizational units reflect common components (e.g., operating system unit or database unit).**





# Architecture Influences the Development Organization Enterprise Goals

**Development of a system may establish a foothold in the market niche.**

**Being known for developing particular kinds of systems becomes a marketing device.**

**Architecture becomes a leveraging point for additional market opportunities and networking.**



# Architecture Influences Customer Requirements

**Knowledge of similar fielded systems leads customers to ask for particular features.**

**Customers will alter their requirements on the basis of the availability of existing systems.**



# Architecture Influences the Architect's Experience and Technical Environment

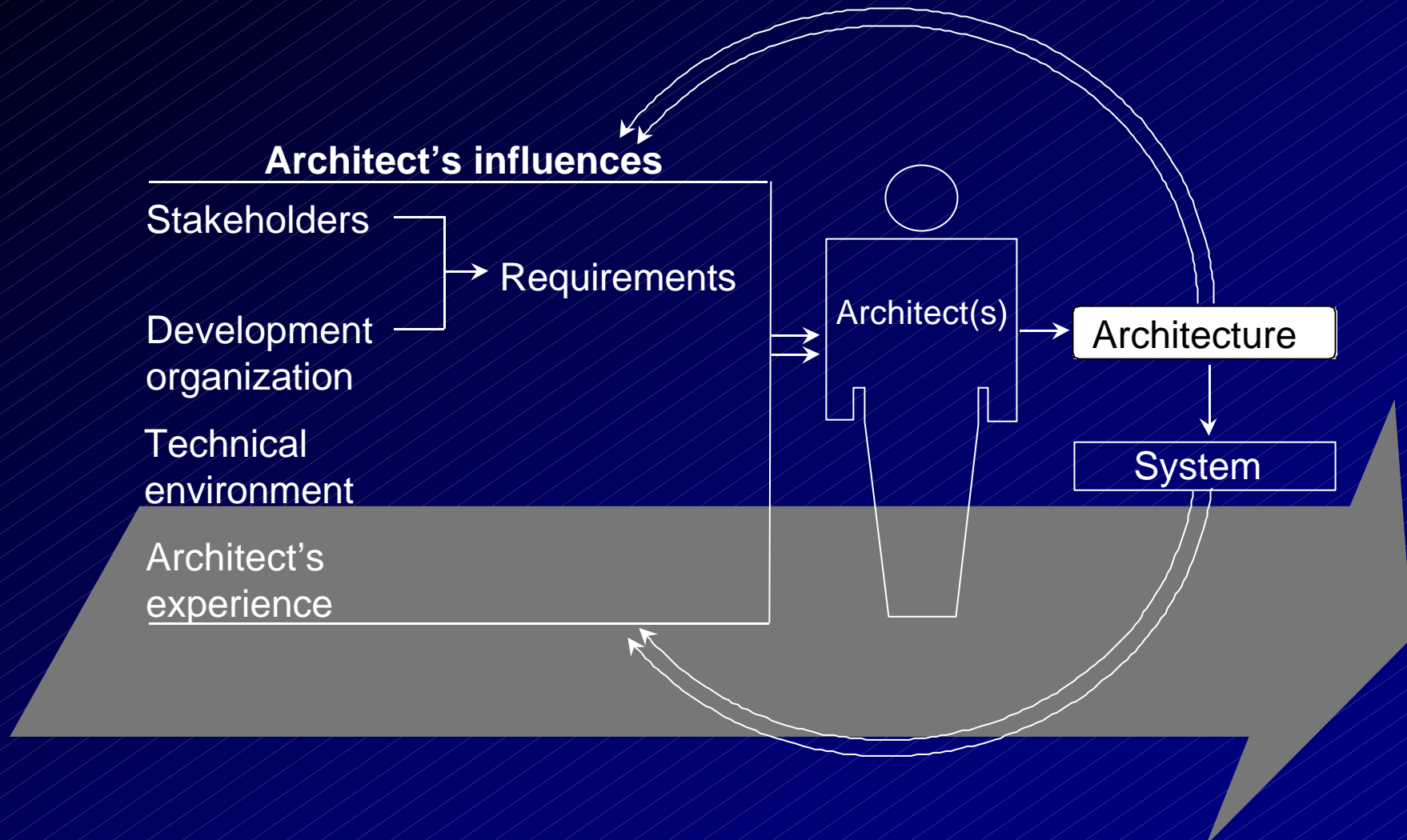
**Creation of a system affects the architect's background.**

**Occasionally, a system or an architecture will affect the technical environment.**

- **the WWW for information systems**
- **the three-tier architecture for database systems**



# Architecture Business Cycle (ABC)





# ABC Summary

**Architecture involves more than just technical requirements for a system. It also involves non-technical factors, such as the**

- **architect's background**
- **development environment**
- **business goals of the sponsoring organization**

**Architecture influences the factors that affect it.**

- **Architects learn from experience.**
- **The development environment is expanded and altered.**
- **Businesses gain new marketing possibilities.**





# Schedule and Outline

**0900 - 0915** Introductions

**0915 - 0945** The Architecture Business Cycle

**0945 -1000** **What is architecture?**

**1000 - 1030** Why is architecture important?

**1030 - 1045** Break

**1045 - 1115** Architectural structures

**1115 - 1200** New developments in software architecture

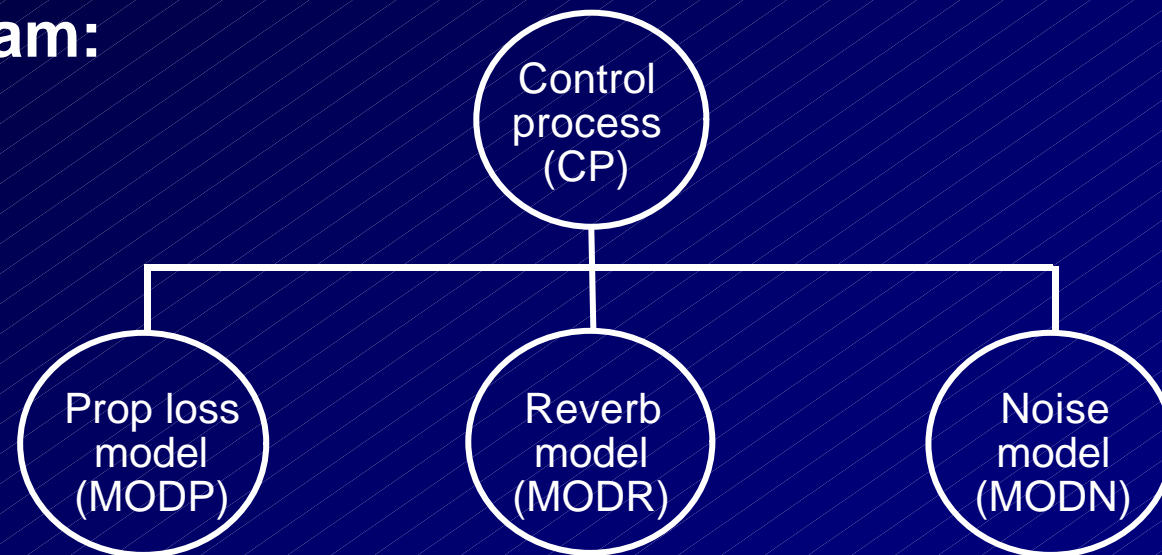


# Some Usual Descriptions of Architecture

**“Components and connectors”**

**“Overall structure of system”**

**A diagram:**





# What's Wrong with “Components and Connectors?”

## What kind of component?

- task? process?
- object? program? function?
- library? compilation unit?
- processor?

## What kind of connector?

- calls? invokes? signals? uses? data flow?
- subclass?
- runs with? excludes?
- co-located with?



# What's Wrong with “Overall Structure?”

***Which* structure? Software is composed of *many* structures.**

- module
- task
- uses
- logical
- functional

**When seeing boxes and lines, we must ask**

- What do the boxes represent?
- What do the arrows mean?



# What's Wrong with the Diagram?

**Same questions as the previous slide.**

- **What kind of components?**
- **What kind of connectors?**
- **What structures?**
- **What do the boxes and arrows mean?**

**Plus new questions**

- **What is the significance of the layout?**
- **Why is control process on a higher level?**

**Box and arrow drawings alone are not architectures;  
rather, they are a starting point.**





# The Definition of Software Architecture

**The software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them.**

**Notice this means that**

- **box-and-line drawings alone are *not* architectures, but a starting point.**
- **architecture includes behavior of components**



# Architectural Style -1

**Architectural style: *a description of component and connector types and a pattern of their runtime control and/or data transfer* [Shaw 96]**

**Architectural styles are a set of canonical architectural solutions to problems.**

**Styles are underspecified architectures. They suggest patterns of runtime interaction, and topologies of components.**



# Architectural Style -2

**A style may be thought of as**

- **a set of constraints on an architecture**
- **an abstraction for a set of related architectures**

**Styles appearing in the literature include**

- **client server**
- **cooperating process**
- **data-centered**
- **layered**



# Schedule and Outline

0900 - 0915 Introductions

0915 - 0945 The Architecture Business Cycle

0945 -1000 What is architecture?

**1000 - 1030 Why is architecture important?**

1030 - 1045 Break

1045 - 1115 Architectural structures

1115 - 1200 New developments in software architecture





# Importance of Architecture to a Development Organization's Business

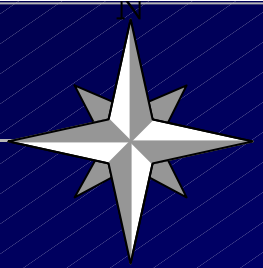
## **Software for a system or group of systems**

- **provides leverage over a marketplace**
- **provides a vehicle for management oversight**
- **provides for the scoping of products**
- **can be used as a sales tool (e.g., conforms to industry standards)**

## **Enterprise architectures enable**

- **shorter learning time**
- **specialized tool support**
- **sharing of infrastructure costs among systems**





# Important of Architecture to a Development Project

**Architecture is important for three primary reasons.**

- 1. It provides a vehicle for communication among stakeholders.**
- 2. It is the manifestation of the earliest design decisions about a system.**
- 3. It is a transferable, reusable abstraction of a system.**



# Communication Vehicle

**Architecture is a frame of reference in which competing interests may be exposed, negotiated.**

- **negotiating requirements with users**
- **keeping customer informed of progress, cost**
- **implementing management decisions and allocations**

**Architecture constrains the implementation and therefore the implementors**

- **implementations must conform to architecture**
- **(global) resource allocation decisions constrain implementations of individual components**



# Result of Early Design Decisions -1

**The architecture dictates organizational structure for development/maintenance efforts. Examples include**

- **division into teams**
- **units for budgeting, planning**
- **basis of work breakdown structure**
- **organization for documentation**
- **organization for CM libraries**
- **basis of integration**
- **basis of test plans, testing**
- **basis of maintenance**

**Once decided, architecture is extremely hard to change!**



# Result of Early Design Decisions -2

**Architecture permits/precludes achievement of a system's desired quality attributes. For example:**

| <b>If you desire</b>     | <b>Examine</b>  |
|--------------------------|---|
| <b>performance</b>       | <b>inter-component communication</b>  |
| <b>modifiability</b>     | <b>component responsibilities</b>   |
| <b>security</b>          | <b>inter-component communication,<br/>specialized components (e. g., kernels)</b> |
| <b>scalability</b>       | <b>localization of resources</b>  |
| <b>ability to subset</b> | <b>inter-component usage</b>  |
| <b>reusability</b>       | <b>inter-component coupling</b>   |

**The architecture influences qualities, but does not guarantee them.**





# Result of Early Design Decisions -3

**An architecture helps users reason about and manage change (about 80% of effort in systems occurs *after* deployment).**

**Architecture divides all changes into three classes.**

- ***local***: modifying a single component
- ***non-local***: modifying several components
- ***architectural***: modifying the gross system topology, communication, and coordination mechanisms

**A good architecture is one in which the most likely changes are also the easiest to make.**





# Reusable Model

**An architecture is an abstraction: a one-to-many mapping (one architecture, many systems).**

**Architecture is the basis for product (system) commonality. Entire product lines can share a single architecture.**

**Systems can be built from large, externally developed components that are tied together via architecture.**



# Schedule and Outline

**0900 - 0915** Introductions

**0915 - 0945** The Architecture Business Cycle

**0945 -1000** What is architecture?

**1000 - 1030** Why is architecture important?

**1030 - 1045** Break

**1045 - 1115** Architectural structures

**1115 - 1200** New developments in software architecture



# Schedule and Outline

**0900 - 0915** Introductions

**0915 - 0945** The Architecture Business Cycle

**0945 -1000** What is architecture?

**1000 - 1030** Why is architecture important?

**1030 - 1045** Break

**1045 - 1115** Architectural structures

**1115 - 1200** New developments in software architecture



# Architectural Structures -1

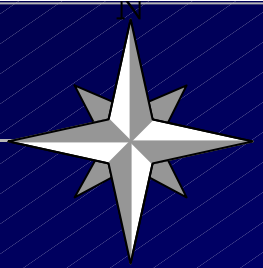
**In a house, there are plans for**

- **rooms**
- **electrical wiring**
- **plumbing**
- **ventilation**

**Each of these constitutes a “view” of the house.**

- **used by different people**
- **used to achieve different qualities in the house**
- **serves as a description and prescription**

**So it is with software architecture.**



# Architectural Structures -2

**Which structures are used, and why?**

**Common structures include**

- **module**
- **process**
- **uses**
- **calls**
- **data flow**
- **class**
- **physical**

**A structure provides a *view* of the architecture.**





# Module Structure

**Components: modules, work assignments**

**Relations: “is a submodule of,” “shares a secret with”**

**Used: as a basis of team structure and resource allocation**

**Affected attributes include: maintainability, understandability**



# Process Structure

**Components: tasks, processes**

**Relations: “synchronizes with,” “excludes,”  
“preempts”**

**Used: to tune system runtime performance, exploit  
multiprocessing hardware**

**Affected attributes include: performance**



# Uses Structure

**Components: procedures**

**Relations: “assumes the correct presence of”**

**Used: to engineer subsets, supersets**

**Affected attributes include: reusability, testability,  
incremental development**



# Calls Structure

**Components: procedures**

**Relation: invokes**

**Used: to trace control flow; for debugging**

**Affected attributes include: buildability, testability,  
maintainability, understandability**



# Data Flow Structure

**Components: programs, modules**

**Relation: “may send data to”**

**Used: for traceability of functionality**

**Affected attributes include: performance, correctness, accuracy**





# Class Structure

**Components: objects**

**Relation: “inherits from,” “is instance of”**

**Used: to exploit similarity among objects**

**Affected attributes include: development time,  
maintainability**



# Physical Structure

**Components: tasks, processes, processors**

**Relation: “resides on same processor”**

**Used: to manage process-to-processor allocation**

**Affected attributes include: performance**



# What Are Structures Used For?

**Descriptive: documentation vehicle for**

- **current development**
- **future development**
- **managers**
- **customers**

**To document the architecture, document the views.**

**Prescriptive: engineering tool to help achieve qualities**



# Architectural Structures Summary

**Structures are related to each other in complicated ways.**

**In some systems, different structures collapse into a single one. (For example, process structure may be the same as module structure for extremely small systems.)**



# Which Views Should I Use?

**Rational Unified Process: 4+1 views**

**Siemens 4-view model**

**(C4ISR framework prescribes 3 views, but these are not views of the *software* architecture. More later.)**

**What to do? Choose the structures that are useful to the system being built and to the achievement of qualities that are important to you.**





# Architectural Structures Example: A-7E Corsair II Aircraft

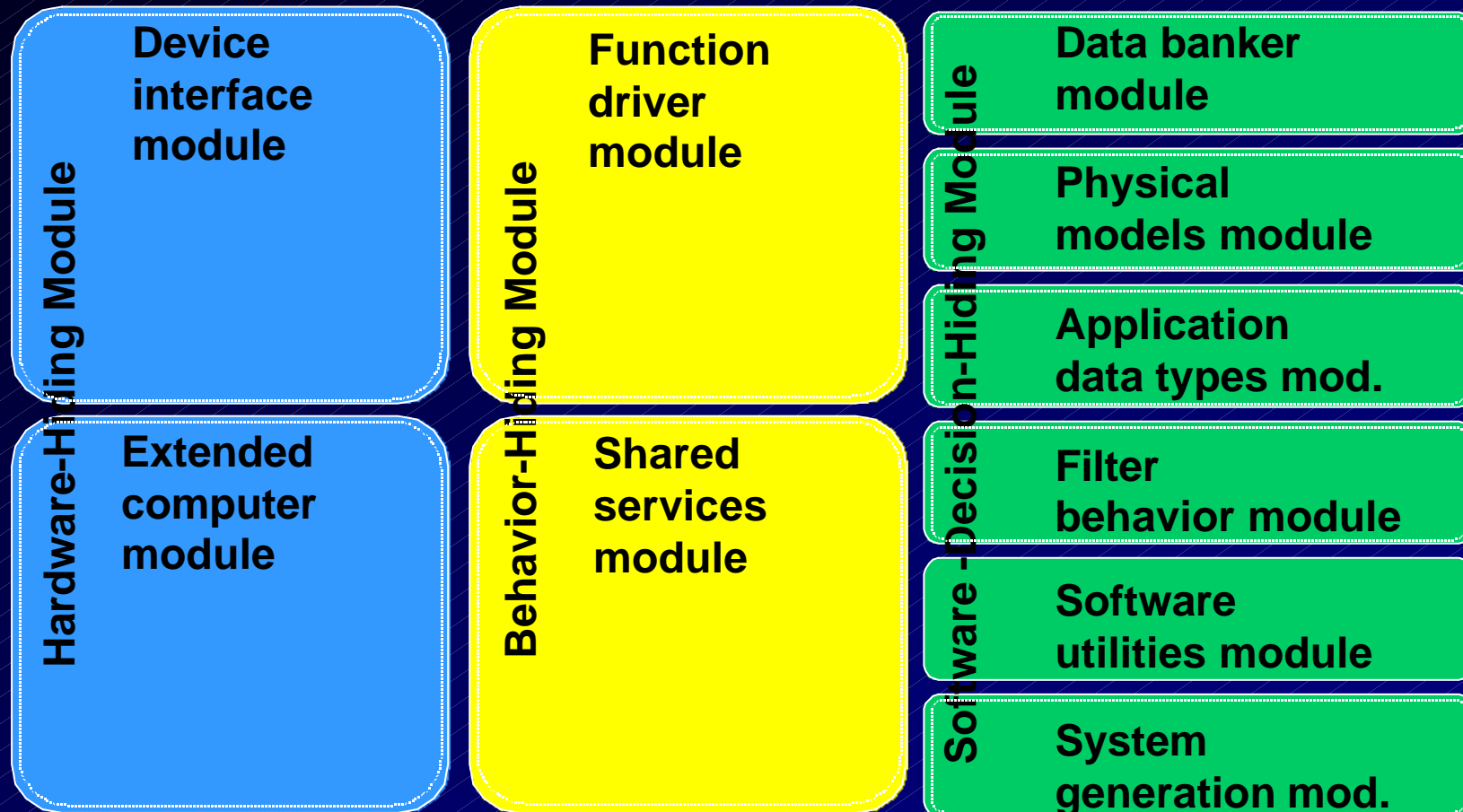
**U. S. carrier-based, light attack aircraft, used from the 1960s through the 1980s**

**Small computer on board for navigation, weapons delivery**



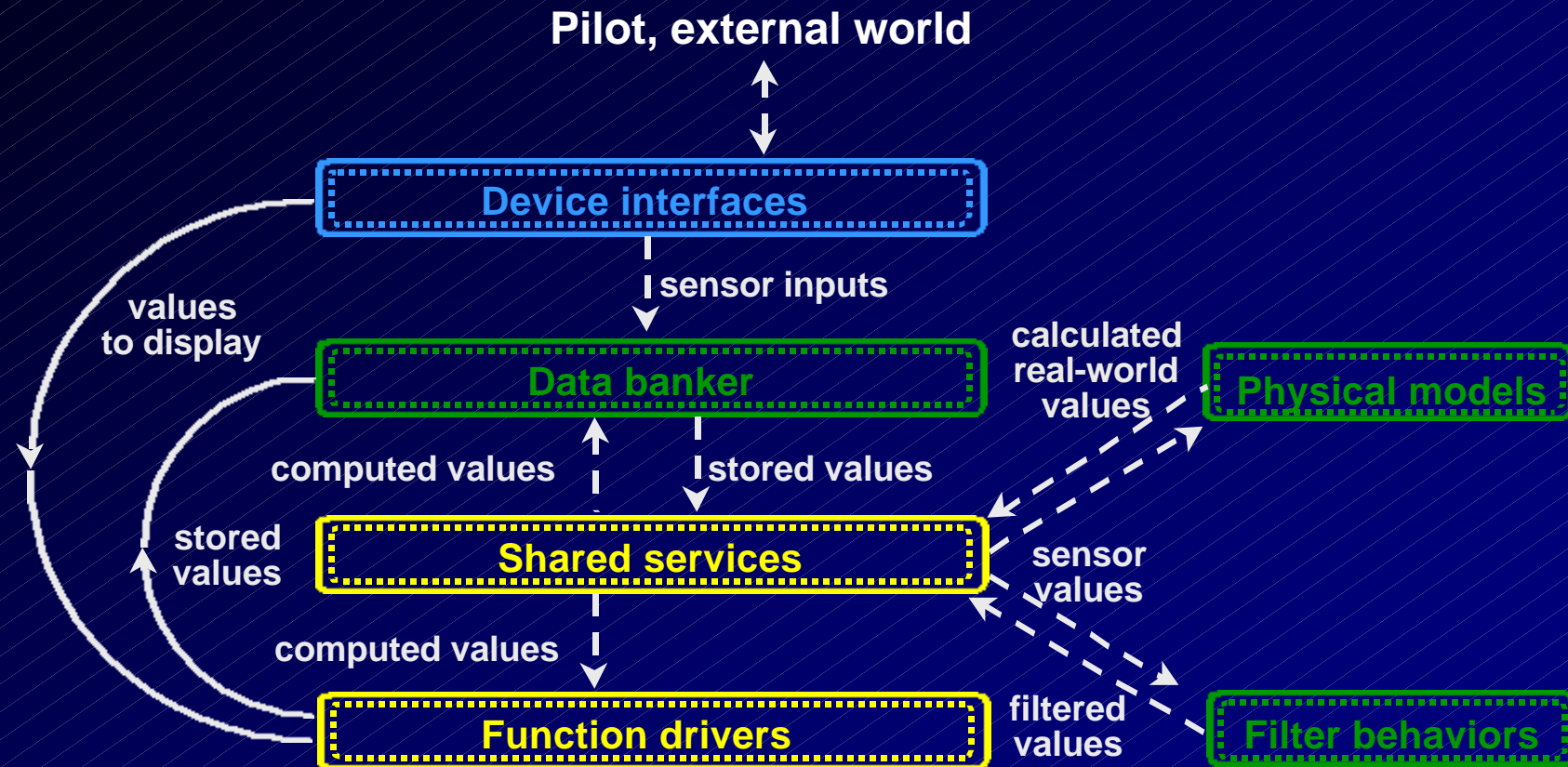


# A-7E Module Structure (2 Levels)



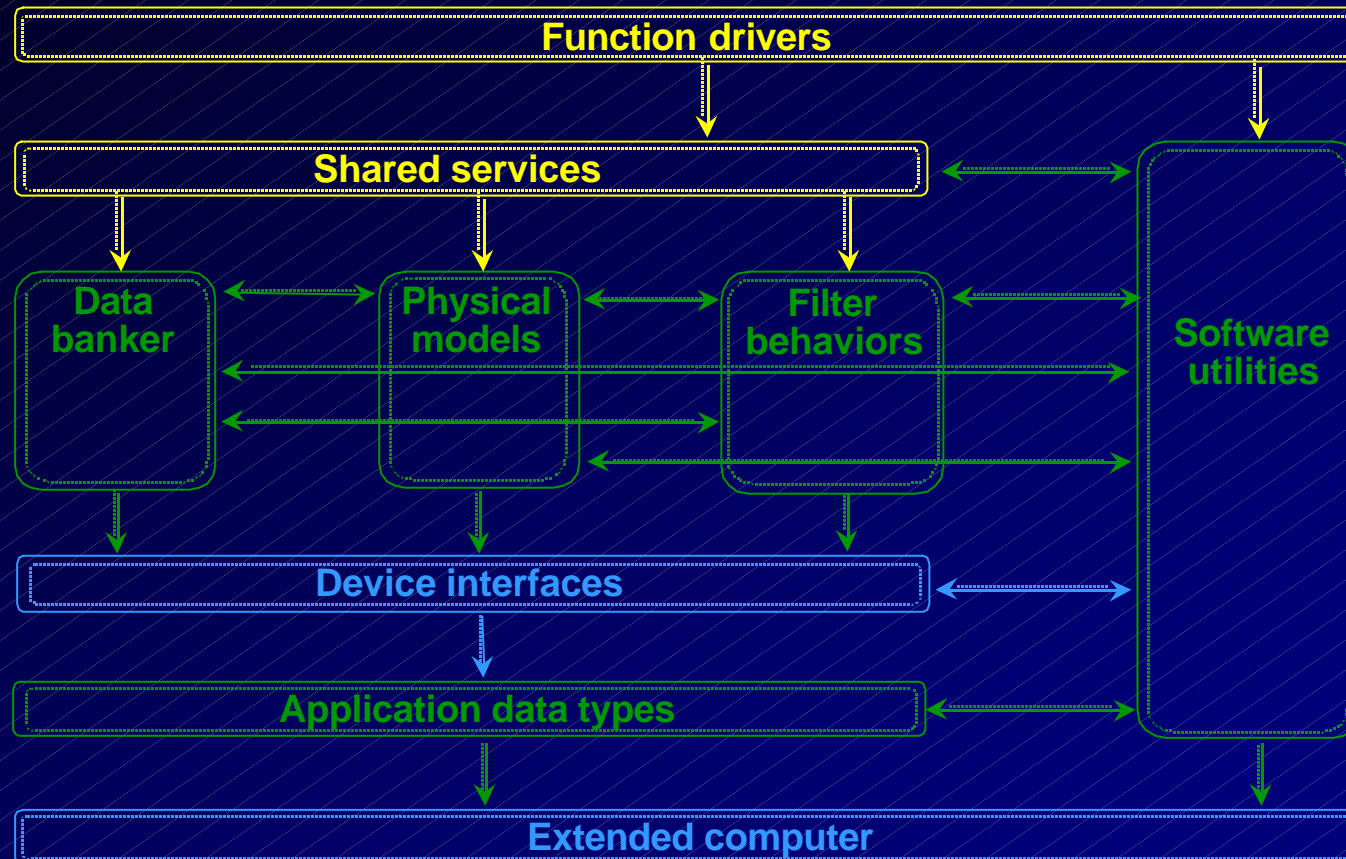


# Data Flow View





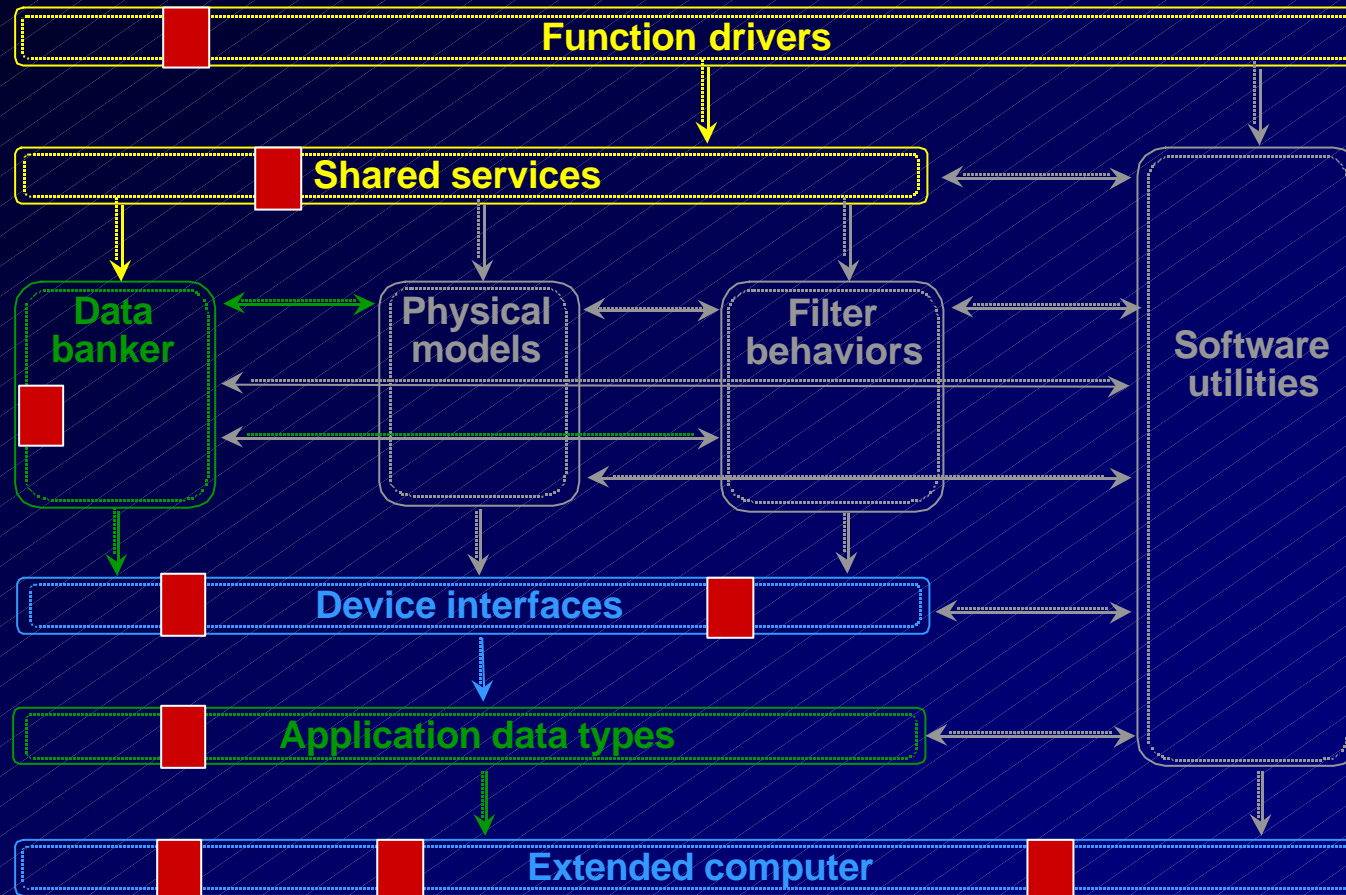
# “Uses” View







# A-7E Subset: Display HUD Altitude







# Lecture Summary -1

**Architecture is important because**

- **it provides a communication vehicle among stakeholders**
- **it is the result of the earliest design decisions about a system**

**An architecture is composed of many structures, documented as *views*, which are software components and their relationships.**

**Each structure provides engineering leverage on different qualities. Engineer and document the structures that help to achieve your desired qualities.**



# Schedule and Outline

**0900 - 0915** Introductions

**0915 - 0945** The Architecture Business Cycle

**0945 - 1000** What is architecture?

**1000 - 1030** Why is architecture important?

**1030 - 1045** Break

**1045 - 1115** Architectural structures

**1115 - 1200** New developments in software architecture

- **Software product lines**
- Aspect-oriented software development
- Predictable assembly from certifiable components



# CelsiusTech Systems

**Swedish defense contractor**

- approximately 2000 employees
- about \$300 million in annual sales

**Long-time supplier of naval shipboard command and control systems**





# 1985: Disaster Struck!

**CelsiusTech marketers landed two large contracts simultaneously.**

- **1,000,000 SLOC each (estimated)**
- **greater complexity of requirements than before**

**CelsiusTech realized they could not fulfill both contracts unless they started doing business in a totally new way.**

- **Earlier systems were troublesome to integrate and had cost schedule overruns.**
- **Hiring was not an option: there was a shortage of engineers.**





# CelsiusTech's Response

## **Business strategy**

- **create a product family**
- **make the software scaleable over a wide range of systems**

## **Technical strategy**

- **create a new generation of system**
  - **hardware, software**
  - **supporting development approach**
- **configure systems from product family; each new project was added to the family**





# What CelsiusTech Did

**Assembled a small expert architecture team with**

- **extensive domain knowledge**
- **previous systems experience**
- **Objective: produce architecture that would suffice for *both* systems *plus* new systems in the same domain.**

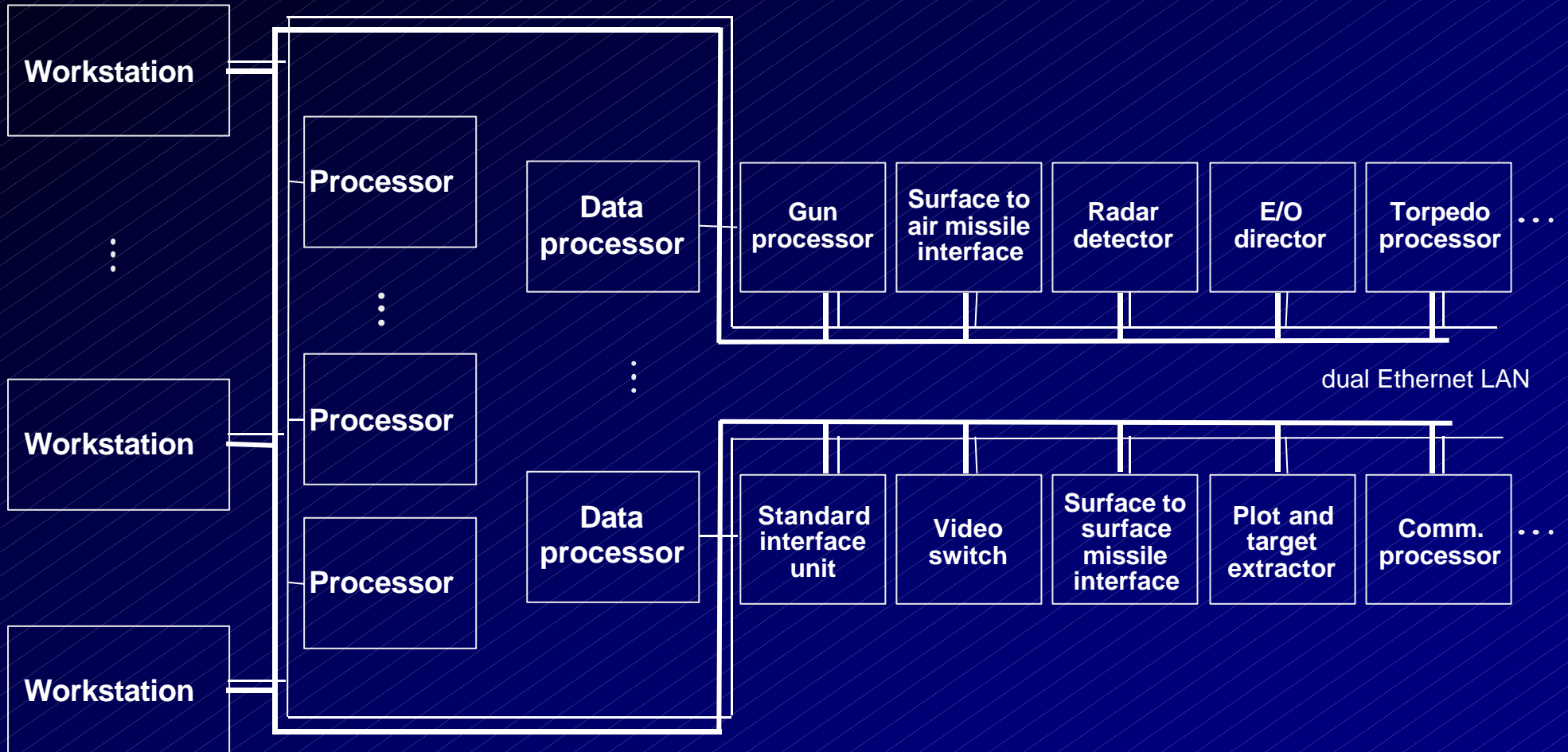
**Produce software components that populated this architecture**

- **Components were flexible, configurable across a wide variety of envisioned uses**

**System-building became a matter of integration, not construction.**



# SS2000 System Architecture





# Typical System Configuration

**15-30 nodes on LAN**

**30-70 CPUs**

**100-300 Ada programs**

**1-1.5 million Ada SLOC**



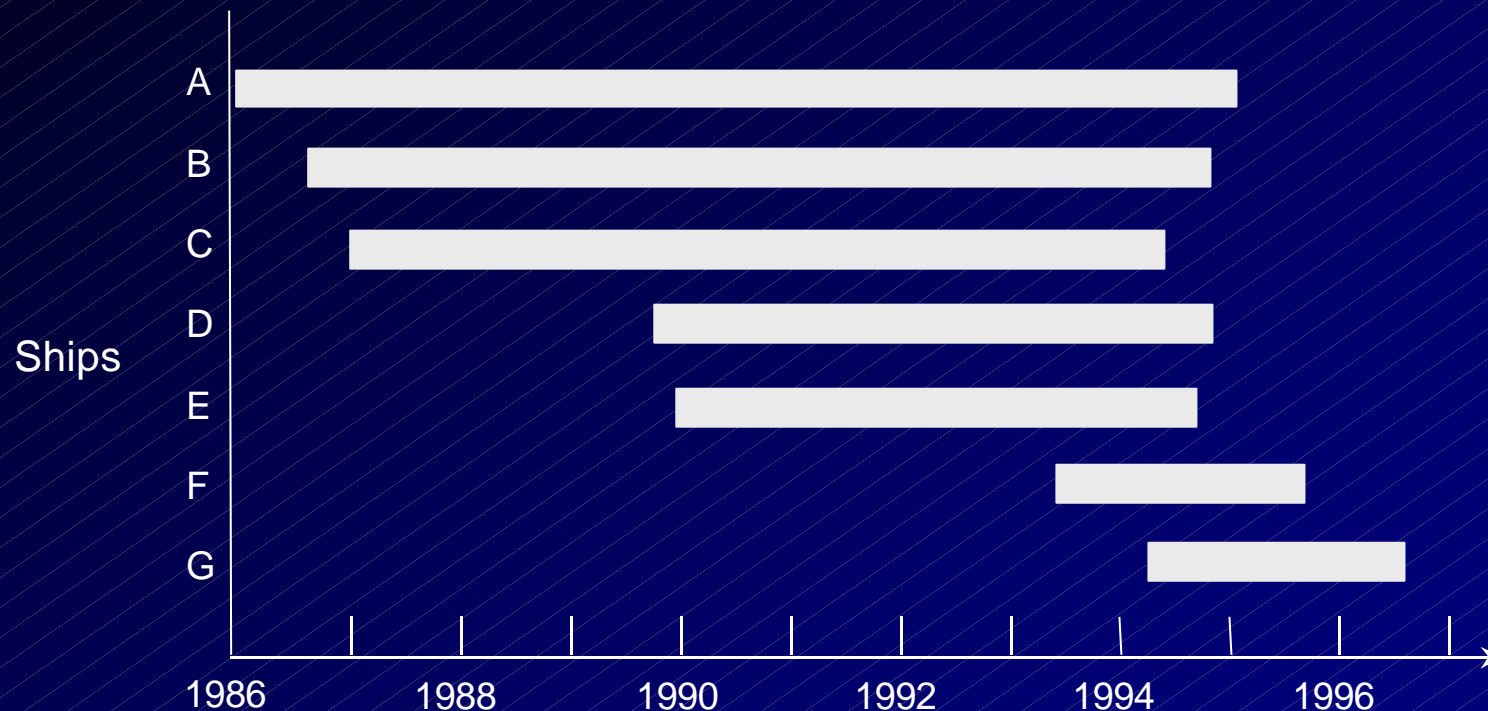
# Members of SS2000 Product Family

## Over 55 variants

- **Swedish Goteborg class Coastal Corvettes (KKV) (380 tons)**
- **Danish SF300 class multi-role patrol vessels (300 tons)**
- **Finnish Rauma class Fast Attack Craft (FAC) (200 tons)**
- **Australian/New Zealand ANZAC frigates (3225 tons)**
- **Danish Thetis class Ocean Patrol vessels (2700 tons)**
- **Swedish Gotland class A19 submarines (1330 tons)**
- **Pakistani Type 21 class frigates**
- **Republic of Oman patrol vessels**
- **Danish Niels Juel class corvettes**



# Result of Changes: Shrinking, Predictable Schedules

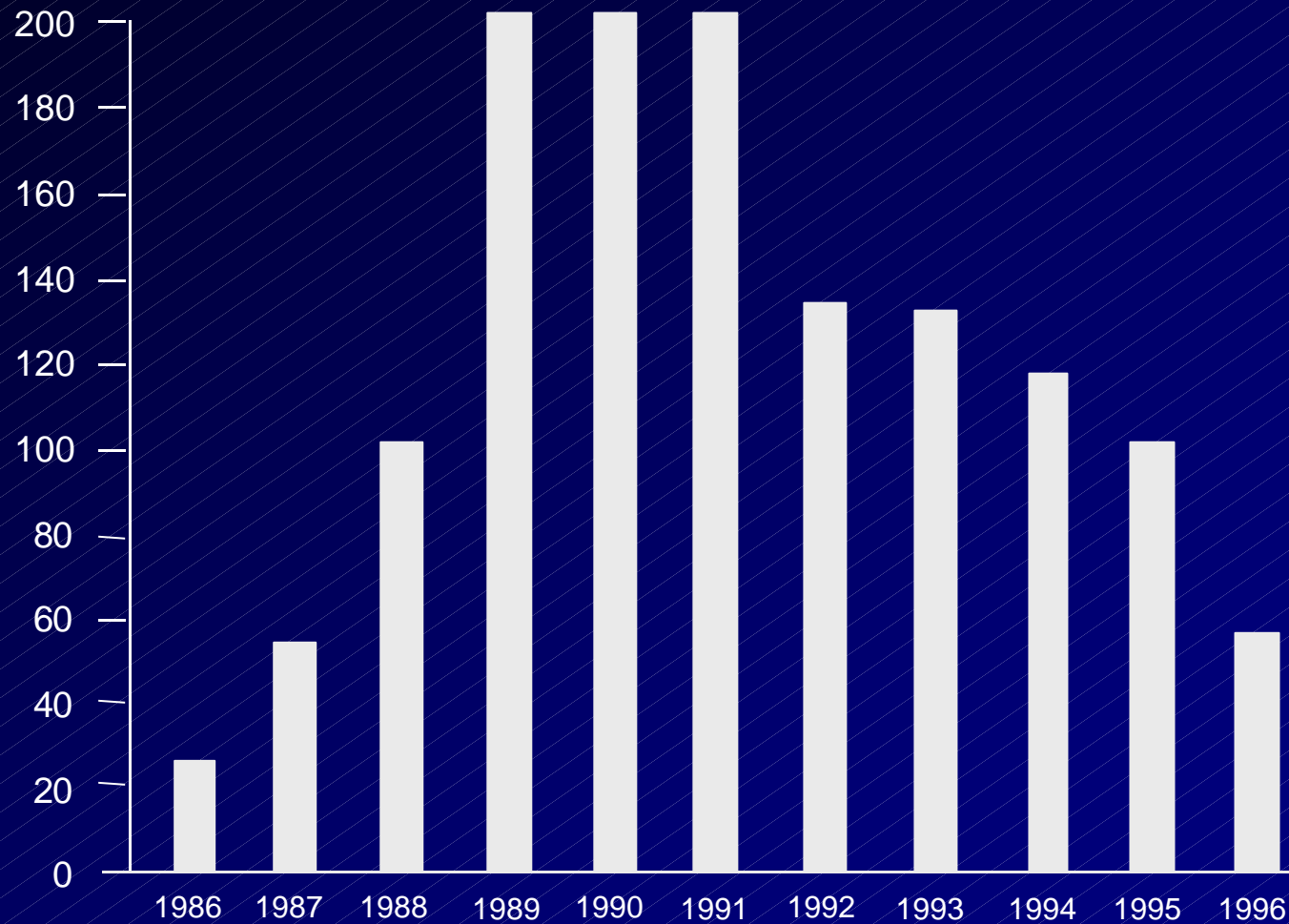


**Hardware-to-software cost ratio changed from 35:65 to 80:20**



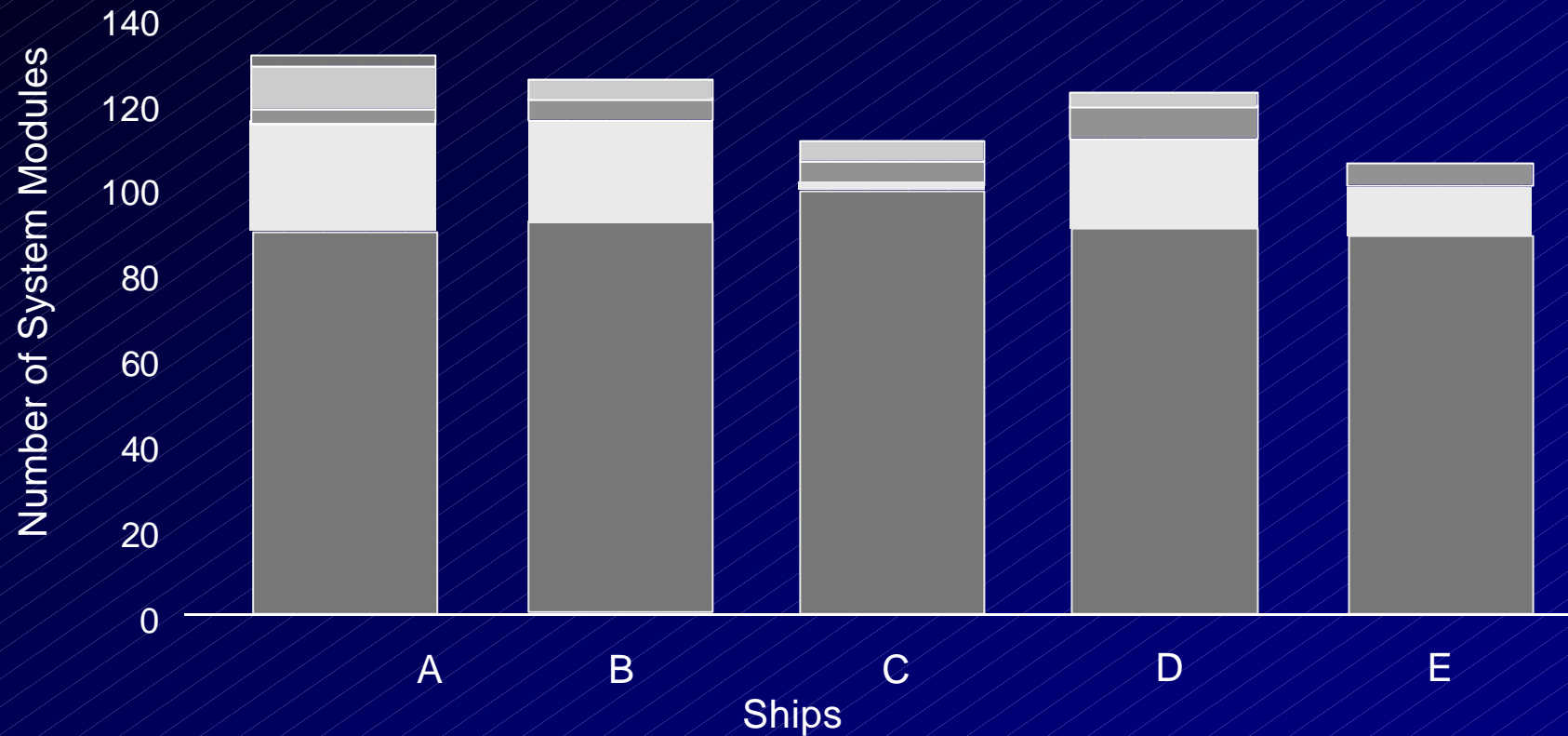


# Result of Changes: Lower Staffing





# Result of Changes: Reuse



Unique application  
National application

Modified  
Reusable application

Common (verbatim)



# Cummins, Inc.

**World's largest  
manufacturer of  
large diesel engines.**





# Complex domain of variation

**Today's diesel engines are driven by software**

- **Micro-control of ignition timing to achieve optimum mix of power, economy, emissions**
- **Conditions change dynamically as function of road incline, temperature, load, etc.**
- **Must also respond to statutory regulations that often change**
- **Reliability is critical! Multi-million dollar fleets can be put out of commission by a single bug**
- **130KSLOC -- C, assembler, microcode**
- **Different sensors, platforms, requirements**





# In 1993, Cummins had a problem

**Six engine projects were underway  
Another 12 were planned.**

**Each project had complete control over its  
development process, architecture, even choice of  
language. Two were trying to use O-O methods.**

**Ron Temple (VP in charge) realized that he would  
need another 40 engineers to handle the new projects  
-- out of the question.**

**This was no way to do business.**





# What Cummins did

**In May, 1994 Temple halted all the projects.**

**He split the leading project.**

- **One half built core assets -- generic software, documentation, and other assets that every product could use**
- **Other half became pilot project for using the core assets to turn out a product**

**In 1995, the product was launched on time (relative to re-vamped schedule) with high quality.**

**Others followed.**



# Cummins' results

**Achieved a product family capability with a breathtaking capacity for variation, or customization**

- 9 basic engine types
- 4-18 cylinders
- 3.9 - 164 liter displacement
- 12 kinds of electronic control modules
- 5 kinds of microprocessors
- 10 kinds of fuel systems
- diesel fuel or natural gas

**Highly parameterized code. 300 parameters are available for setting by the customer after delivery.**



# Cummins' results by the numbers -1

- **20 product groups launched, which account for over 1000 separate engine applications**
- **75% of all software, on average, comes from core assets**
- **Product cycle time has plummeted. Time to first engine start went from 250 person-months to a few person-months. One prototype was built over a weekend.**
- **Software quality is at an all-time high, which Cummins attributes to product line approach.**



# Cummins' results by the numbers -2

- **Customer satisfaction is high. Productivity gains enables new features to be developed (more than 200 to date)**
- **Projects are more successful. Before product line approach, 3 of 10 were on track, 4 were failing, and 3 were on the edge. Now, 15 of 15 are on track.**
- **Widespread feeling that developers are more portable, and hence more valuable.**





# Cummins' results by the numbers -3

**Supported Components 1992 1993 1994 1995 1996 1997 1998**

=====

**Electronic control  
modules (ECMs)**

**3 3 4 5 5 11 12**

**Fuel systems**

**2 2 3 5 5 10 11**

**Engines**

**3 3 5 5 12 16 17**

**Features \* ECM**

**60 80 180 370 1100 2200 2400**

**Achieving this flexibility without the product line approach  
would have required 3.6 times the staff Cummins has.**





# Cummins' results by the numbers -4

**Cummins management has a history of embracing change, but carefully targeted change.**

**They estimate that process improvement alone has brought a benefit/cost ration of 2:1 to 3:1.**

**They estimate that the product line approach has brought a benefit/cost ration of 10:1.**

**Product line approach let them quickly enter and then dominate the *industrial* diesel engine market.**

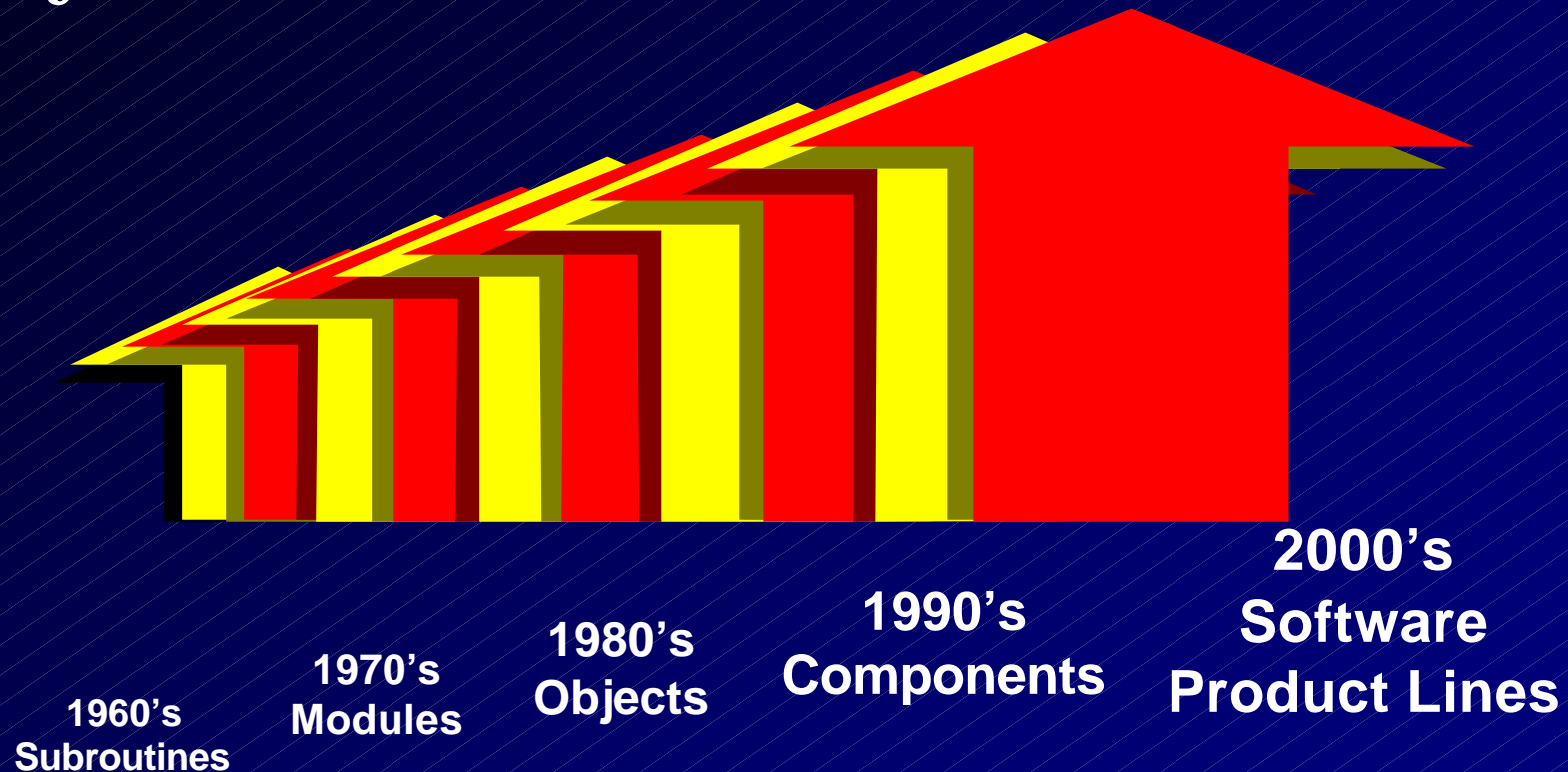


# Two companies, same goals





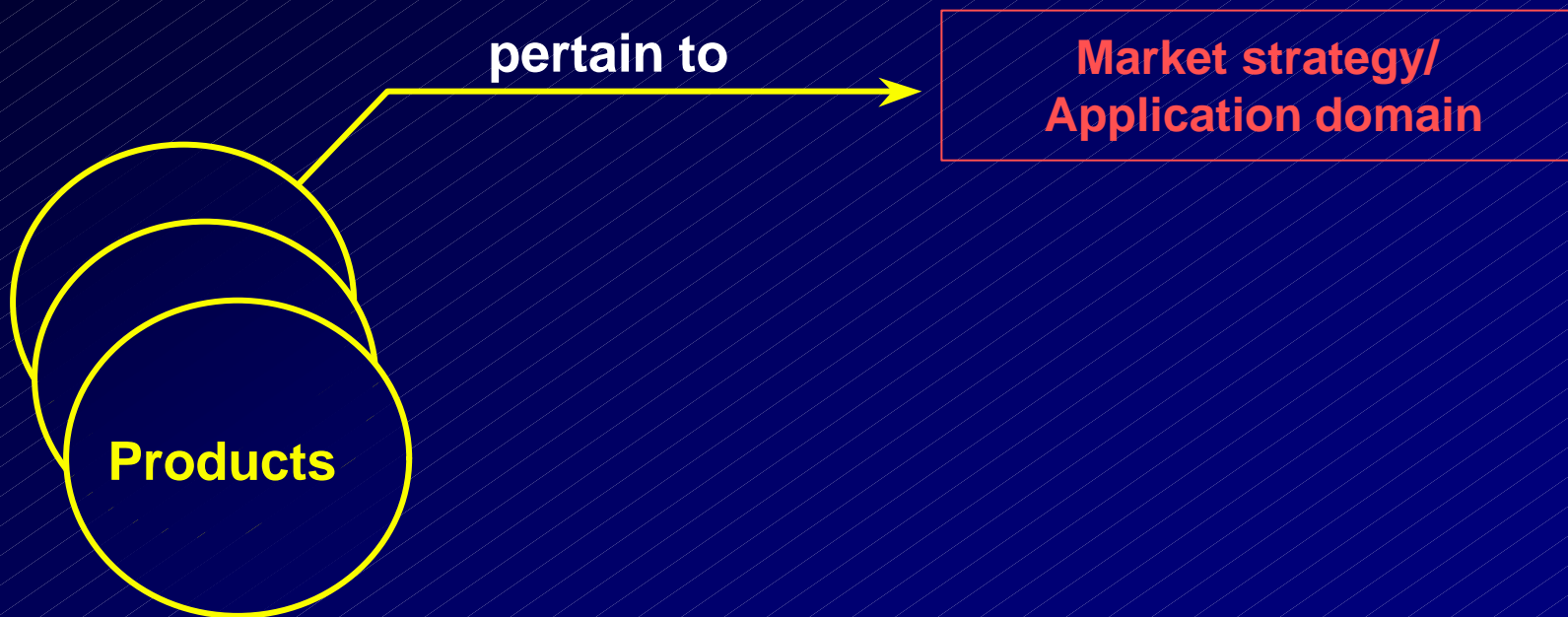
# Reuse History: From Ad-Hoc to Systematic





# What Is a Product Line?

A product line is a group of products sharing a common, managed set of features that satisfy specific needs of a selected market or mission.





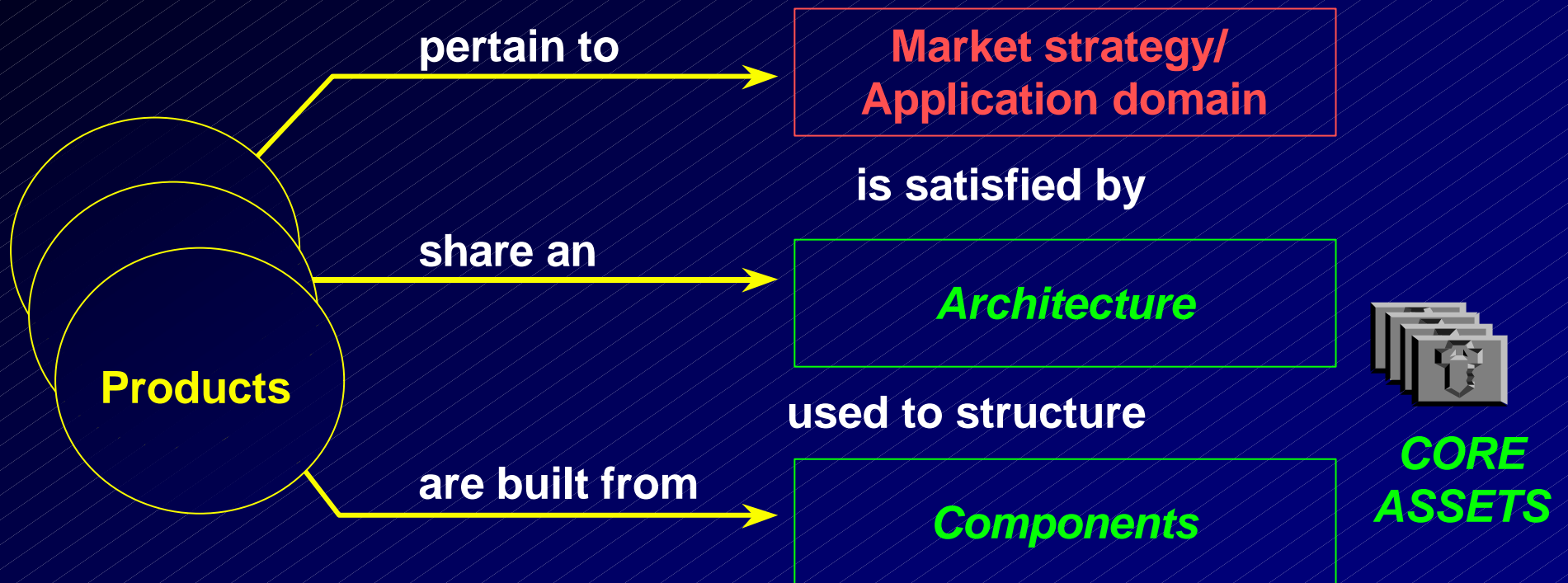
# What is a Software Product Line?

A software product line is a set of **software-intensive systems** sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission **and that are developed from a common set of core assets in a prescribed way.**





# Software Product Lines



## Product lines

- take economic advantage of commonality
- bound variability

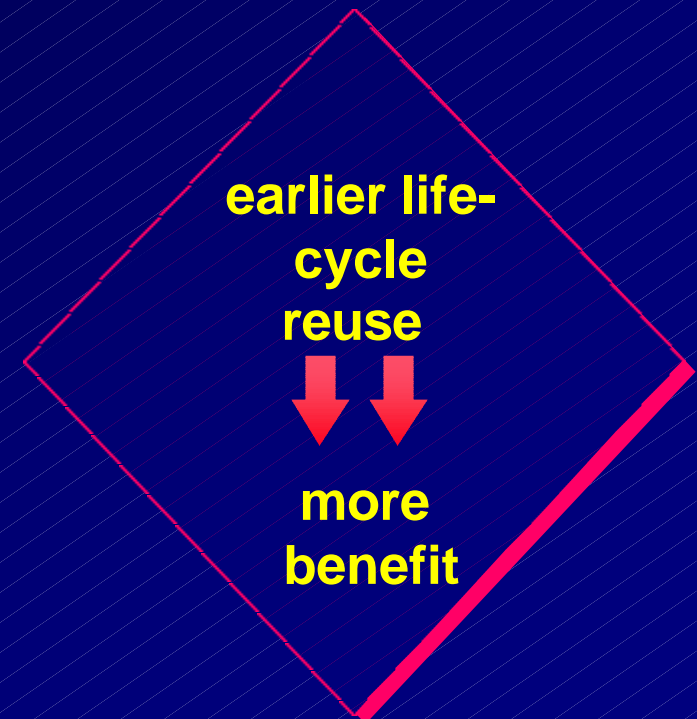


# How Do Product Lines Help?

Product lines amortize the investment in these and other **core assets**:

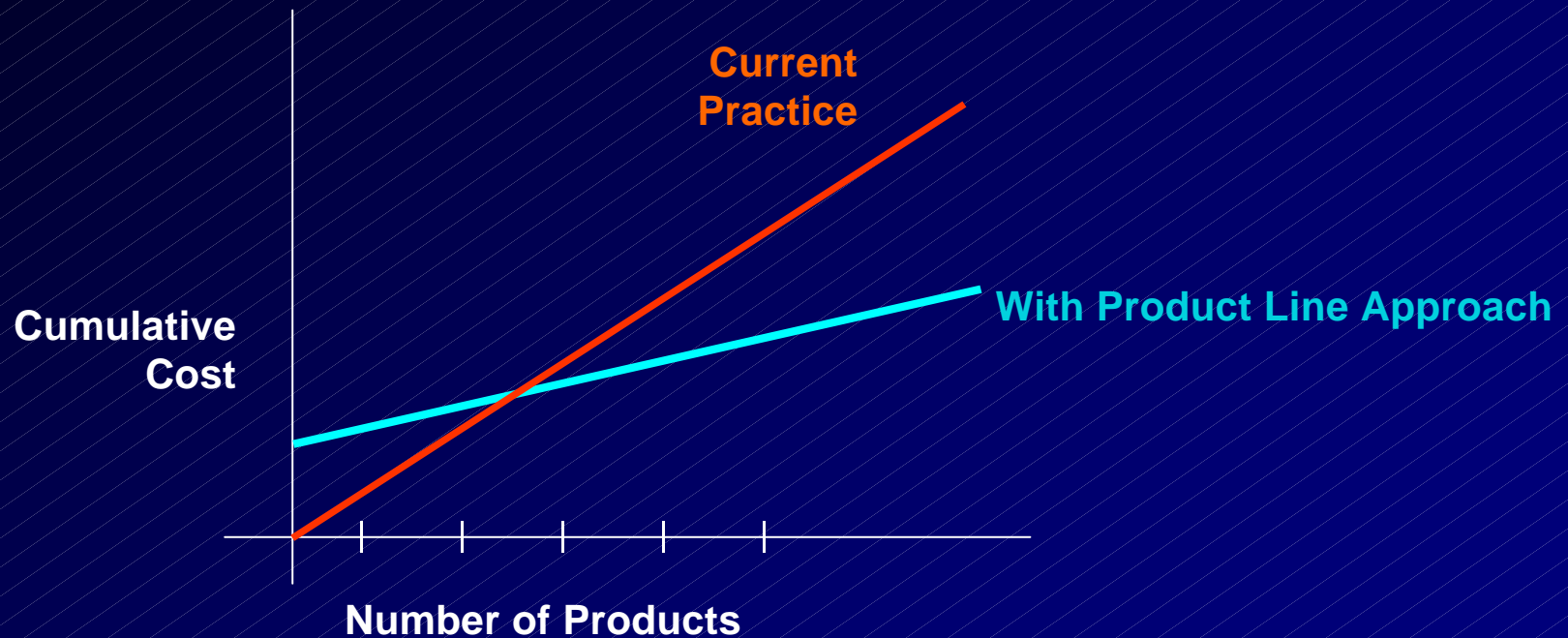
- requirements and requirements analysis
- domain model
- software architecture and design
- performance engineering
- documentation
- test plans, test cases, and data
- people: their knowledge and skills
- processes, methods, and tools
- budgets, schedules, and work plans
- components

***product lines = strategic reuse***





# Economics of Product Lines



Derived from data supplied by  
Lucent Technologies  
Bell Laboratories Innovations



# The Key Concepts

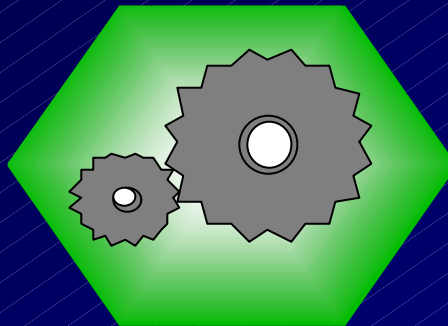
Use of a  
common  
asset base



Architecture



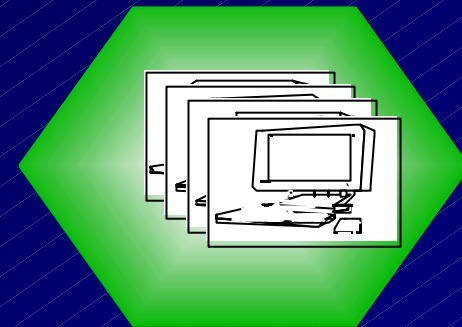
*in  
production*



Production Plan



of a related  
set of  
products



Scope Definition  
Business Case







# Organizational Benefits

**Improved productivity**  
**by as much as 10x**

**Decreased time to market (to field, to launch...)**  
**by as much as an order of magnitude**

**Decreased cost**  
**by as much as 60%**

**Decreased labor needs**  
**by as much as 10X fewer software developers**

**Increased quality**  
**by as much as 10X fewer defects**





# Product Line Practice

Contexts for product lines **vary** widely

- nature of products
- nature of market or mission
- business goals
- organizational infrastructure
- workforce distribution
- process maturity
- artifact maturity

But there are **universal essential** elements and practices.



# CelsiusTech and Cummins both learned vital lessons

## **Lessons in software engineering**

- architectures for product lines
- testing variable architectures and components
- importance of having and capturing domain knowledge
- managing variations
- important of large, pre-integrated chunks

## **Lessons in technical/project management**

- importance of configuration management, and why it's harder for product lines
- product line scoping: What's in? What's out?
- Tool support for product lines

## **Lessons in organizational management.**

- People issues: how to bring about change, how to launch the effort
- Organizational structure: Who builds the core assets?
- Funding: How are the core assets paid for?
- Interacting with the customer has whole new dimension



# Embodying the knowledge: SEI Product Line Practice Framework

**Web-based, evolving document**

**Describes product line essential activities**

- **Core asset development**
- **Product development**
- **Management**

**Describes essential and proven product line practices  
in the areas of**

- **software engineering**
- **technical management**
- **organizational management**



# Framework Goals

**Identify the foundational concepts underlying the software product lines and the essential issues to consider before fielding a product line.**

**Identify practice areas that an organization creating or acquiring software product lines must master.**

**Define practices in each practice area where current knowledge is sufficient to do so.**

**Provide guidance to an organization about how to move to a product line approach for software.**







# SEI Information Sources

**Case studies,  
experience reports,  
and pilots**

**Workshops**



**Surveys**

**Collaborations  
with customers  
on actual product lines**





# Practice Areas

**A practice area is a body of work or a collection of activities that an organization must master to successfully carry out the essential work of a product line.**

- **Are finer chunks than the essential activities**
- **Must be mastered to carry out the essential activities**
- **Provide starting points for organizations wanting to make and measure product line progress**



# Software Engineering Practice Areas

**Architecture Definition**

**Architecture Evaluation**

**Component Development**

**COTS Utilization**

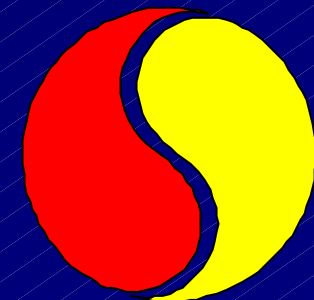
**Mining Existing Assets**

**Requirements Engineering**

**Software System Integration**

**Testing**

**Understanding Relevant Domains**





# Technical Management Practice Areas

**Configuration Management**

**Data Collection, Metrics, and Tracking**

**Make/Buy/Mine/Commission Analysis**

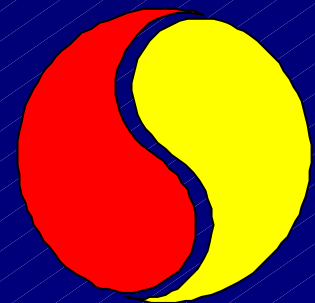
**Process Definition**

**Product Line Scoping**

**Technical Planning**

**Technical Risk Management**

**Tool Support**





# Organizational Management Practice Areas

**Achieving the Right Organizational Structure**

**Building and Communicating a Business Case**

**Customer Interface Management**

**Developing and Implementing an Acquisition Strategy**

**Funding**

**Launching and Institutionalizing a Product Line**

**Market Analysis**

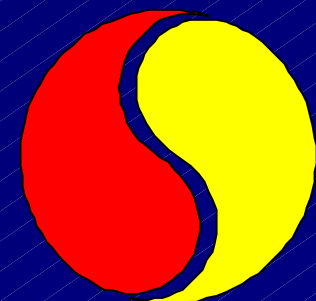
**Operations**

**Organizational Planning**

**Organizational Risk Management**

**Technology Forecasting**

**Training**







# Examples of Product Line Practice - 1

## **Motorola** - FLEXworks Project (family of one-way pagers)

- 4x cycle time improvement
- 80% reuse

## **Nokia** - mobile phones

- went from 4 different phones produced per year to 50 per year

## **National Reconnaissance Office's Control Channel Toolkit** - ground-based satellite systems

- first family member required 1/10 normal number of developers

## **Hewlett Packard** - printer systems

- 2-7x cycle time improvement (some 10x)
- Sample Project
  - shipped 5x number of products
  - that were 4x as complex
  - and had 3x the number of features
  - with 4x products shipped/person



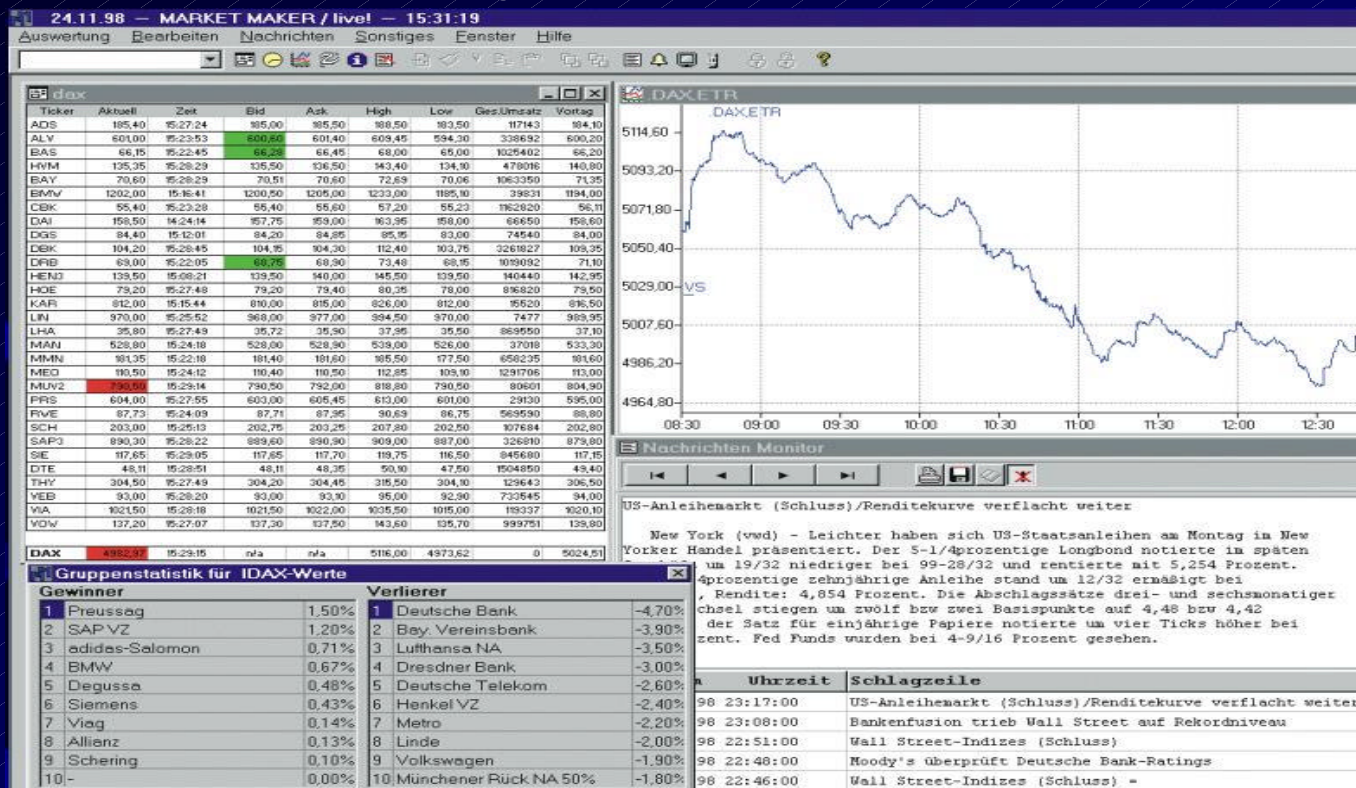




# Examples of Product Line Practice - 3

## MarketMaker Software AG - German financial info provider

- able to field a customer-specific solution in about a week
- small company (under 50 people)





# Adoption strategies

## Proactive (predictive)

- Look ahead, define the product line's scope proactively
- Learn all you can from domain analysis
- Product line adoption is an organization-wide affair
- Cummins and CelsiusTech both took this approach

## Reactive

- Start with 1-2 products
- React to new customers as they arrive

## Extractive

- Extract commonality from existing products
- Form common asset base from what you already have
- Product line adoption can start in small pockets, spread as acceptance grows



# For Additional Information on SEI's Product Line Systems Program

**Dave White**  
**Business Development**  
**Product Line Systems Program**  
**Telephone: 412-268-4796**  
**Email: [dwhite@sei.cmu.edu](mailto:dwhite@sei.cmu.edu)**

**Linda Northrop**  
**Director**  
**Product Line Systems Program**  
**Telephone: 412-268-7638**  
**Email: [lnn@sei.cmu.edu](mailto:lnn@sei.cmu.edu)**

**For questions about this talk:**  
**Paul Clements**  
**Product Line Systems Program**  
**Email: [clements@sei.smu.edu](mailto:clements@sei.smu.edu)**

**U.S. mail:**  
**Software Engineering Institute**  
**Carnegie Mellon University**  
**Pittsburgh, PA 15213-3890**

**World Wide Web:**  
**<http://www.sei.cmu.edu/plp>**

**SEI Fax: 412-268-5758**





# To read more about CelsiusTech

## ***Software Architecture in Practice***

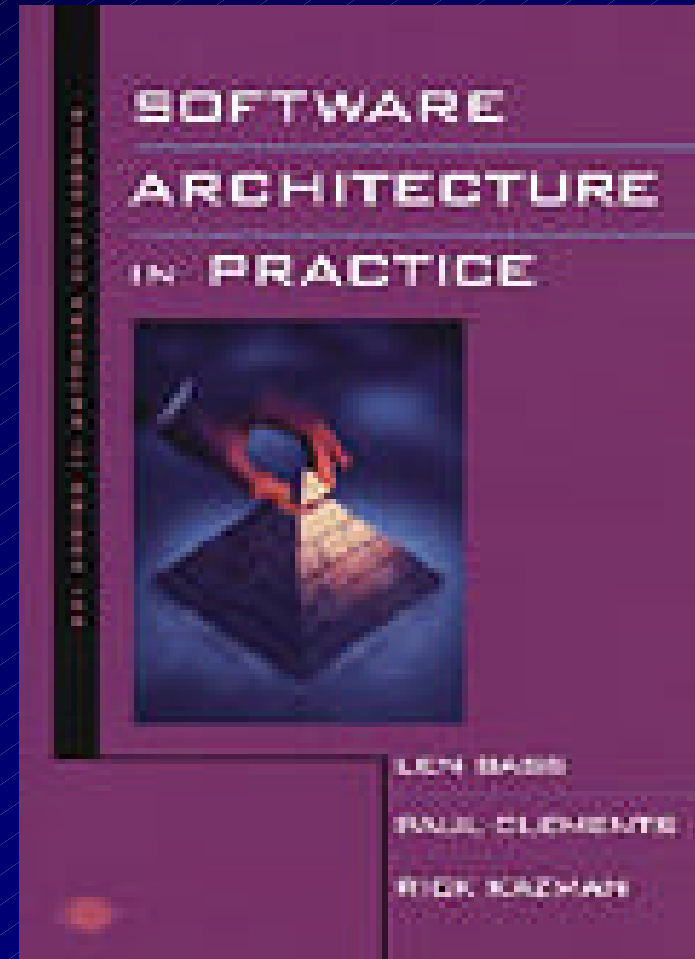
**Len Bass**

**Paul Clements**

**Rick Kazman**

**Addison Wesley 1998**

- Seven case studies in successful software architectures
- Architecture evaluation
- Architecture business cycle
- Achievement of system qualities through architecture





# To read more about Cummins

## ***Software Product Lines: Practices and Patterns***

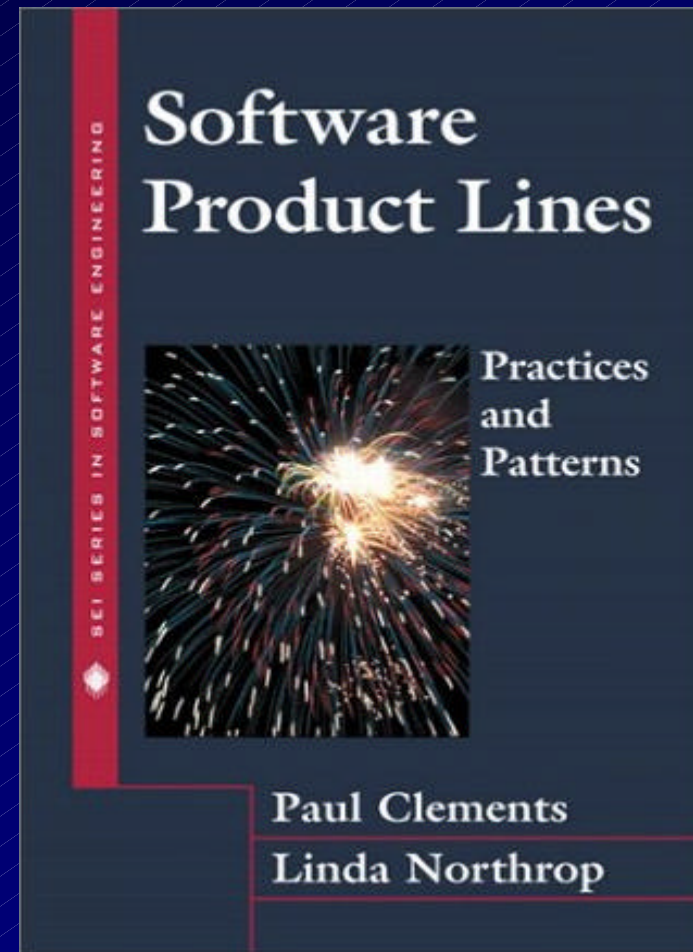
**Paul Clements**

**Linda Northrop**

**Addison Wesley 2001**

**~600 pages**

- **Product line fundamentals and economics**
- **Practice areas described**
- **Patterns for adoption**
- **3 Detailed case studies**
- **Product Line Technical Probe**







# Schedule and Outline

0900 - 0915 Introductions

0915 - 0945 The Architecture Business Cycle

0945 - 1000 What is architecture?

1000 - 1030 Why is architecture important?

1030 - 1045 Break

1045 - 1115 Architectural structures

**1115 - 1200 New developments in software architecture**

- Software product lines
- **Aspect-oriented software development**
- Predictable assembly from certifiable components



# Aspect-Oriented Software Development (AOSD)

**Also called “multi-dimensional separation of concerns.” Recognition that separation of concerns may be performed in many ways.**

## **Example:**

- **Dividing a system into elements based on likely application-based changes**
- **Each element still must reflect**
  - **a particular error-handling philosophy**
  - **an architectural packaging decision**
  - **naming conventions**
  - **interaction protocols**
  - **...and many others**



# AOSD (cont'd.)

**AOSD tools and languages let programmers weave these separate concerns together in discrete elements, so that these global design decisions (that have far-reaching effects) can be changed locally.**

**AOSD represents the introduction of truly architectural thinking into program development.**

**For more information: <http://www.aosd.net>.**



# Schedule and Outline

0900 - 0915 Introductions

0915 - 0945 The Architecture Business Cycle

0945 - 1000 What is architecture?

1000 - 1030 Why is architecture important?

1030 - 1045 Break

1045 - 1115 Architectural structures

**1115 - 1200 New developments in software architecture**

- Software product lines
- Aspect-oriented software development
- **Predictable assembly from certifiable components**





# Predictable Assembly of Certifiable Components (PACC)

**At the vanguard of work on component-based systems.**

**Previous work has concentrated on component selection and qualification, and building frameworks for component-based systems.**

**This work focuses on building systems with provable quality attributes from components.**





# PACC -2

## Two driving questions:

- Given a set of components with certified quality attributes, what can you conclude about the qualities of a system including those component?
- Given a quality attribute need for a system, what must you be able to certify about its components to know you've satisfied that need?

Very early work. Preliminary results with latency, starting to work on reliability.

For more information:

<http://www.sei.cmu.edu/pacc/index.html>