The Changing Face of Software Architecture ...

... and what that means for educators

David Garlan

12 March 2010

A Brief Personal History

- 1987 PhD at CMU
- 3 years at Tektronix developing a product line
- 1990 joined faculty at CMU
  - Began collaboration with Mary Shaw
  - Became involved in the Master in Software Engineering program
- 1992 taught first course in Architectures for Software Systems with Mary Shaw
- 1996 published book with Mary Shaw
- 2003 published book with Paul Clements, Len Bass, Judy Stafford, and others

Acknowledgements

... Mary Shaw

... Tony Lattanze

... other co-instructors of Software Architecture course

... members of the ABLE research group

... funders (NSF, DARPA, ARO, Siemens, and many others)

This Talk

- Software Architecture: past and present
  - What is software architecture?
  - Evolution of the field and its role in software engineering
- What should software engineers know about software architecture?
  - Elements of a course on software architecture
  - Architectural thinking
- Emerging trends and Issues
  - Architecture evolution
  - Architecture conformance
  - Frameworks, platforms, and ecologies
  - Conway’s law revisited
- Some questions to ponder

Examples of Software Architecture

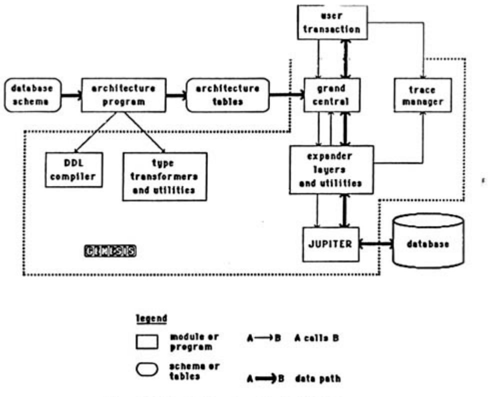


Figure 3.1 The Configuration of the GENESIS Prototype

Genesis: A Reconfiguration Database Management System, D. S. Batory, J.R. Barnett, J.F. Garza, K.P. Smith, K. Tsukuda, B.C. Twichell, T.E. Wise, Department of Computer Sciences, University of Texas at Austin,

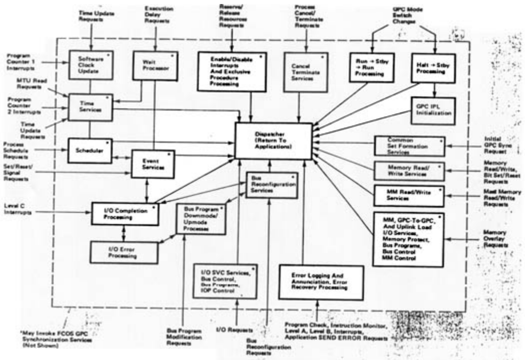

Time Update Requests
Execution Delay Requests
Reserve/ Release Resources Requests
Process Cancel/ Terminate Requests
GPC Mode Switch Changes
Program Counter 1 Interrupts
Software Clock Update
Wait Processor
Enable/Disable Interrupts And Exclusive Procedure Processing
Cancel Terminate Services
Run → Stby → Run Processing
Halt → Stby Processing
MTU Read Requests
Program Counter 2 Interrupts
Time Services
Time Update Requests
GPC IPL Initialization
Dispatcher (Return To Applications)
Process Schedule Requests
Scheduler
Event Services
Common Set Formation Services
Initial GPC Sync Request
Set/Reset/ Signal Requests
Memory Read/ Write Services
Memory Read/Write, Bit Set/Reset Requests
Bus Reconfiguration Services
MM Read/Write Services
Mast Memory Read/Write Requests
Level C Interrupts
I/O Completion Processing
Bus Program Downmode/ Upmode Processes
MM, GPC-To-GPC, And Uplink Load I/O Services, Memory Protect, Bus Programs, Bus Control MM Control
Memory Overlay Requests
I/O Error Processing
I/O SVC Services, Bus Control, Bus Programs, IOP Control
Error Logging And Annunciation, Error Recovery Processing
*May Invoke FCOS GPC Synchronization Services (Not Shown)
Bus Program Modification Requests
I/O Requests
Bus Reconfiguration Requests
Program Check, Instruction Monitor, Level A, Level B, Interrupts, Application SEND ERROR Requests

FIGURE 7. Flight Computer Operating System (The FCOS dispatcher coordinates and controls all work performed by the on-board computers.)

Communications of the ACM, "Architecture of the Space Shuttle Primary Avionics Software Systems," Gene D. Carlow, September 1984, Vol. 27, No. 9, P. 933

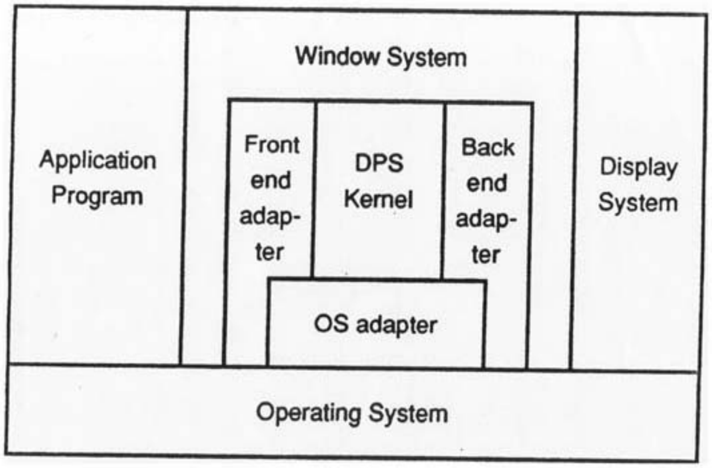
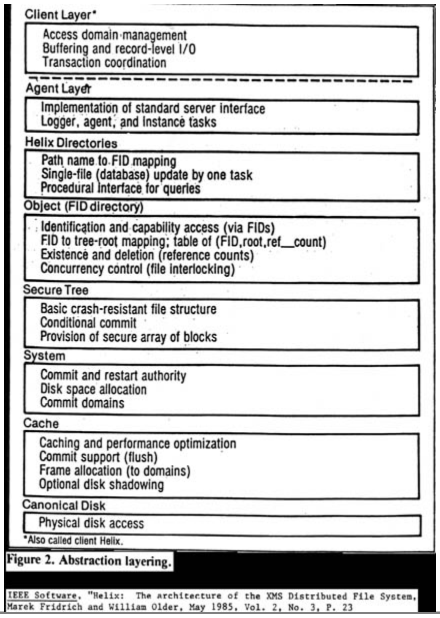


Figure 2. Display PostScript interpreter components.

An Overview of the DISPLAY POSTSCRIPT^{TM} System, Adobe Systems Incorporated, March 16, 1988, P. 10



Client Layer*

Access domain management
Buffering and record-level I/O
Transaction coordination

Agent Layer

Implementation of standard server interface
Logger, agent, and instance tasks

Helix Directories

Path name to FID mapping
Single-file (database) update by one task
Procedural interface for queries

Object (FID directory)

Identification and capability access (via FIDs)
FID to tree-root mapping; table of (FID,root,ref__count)
Existence and deletion (reference counts)
Concurrency control (file interlocking)

Secure Tree

Basic crash-resistant file structure
Conditional commit
Provision of secure array of blocks

System

Commit and restart authority
Disk space allocation
Commit domains

Cache

Caching and performance optimization
Commit support (flush)
Frame allocation (to domains)
Optional disk shadowing

Canonical Disk

Physical disk access

*Also called client Helix.

Figure 2. Abstraction layering.

IEEE Software, "Helix: The architecture of the XMS Distributed File System, Marek Fridrich and William Older, May 1985, Vol. 2, No. 3, P. 23

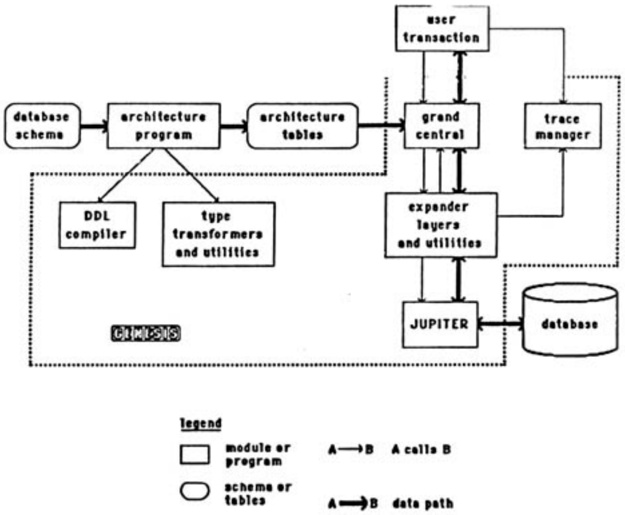


Figure 3.1 The Configuration of the GENESIS Prototype

Genesis: A Reconfiguration Database Management System, D. S. Batory, J.R. Barnett, J.F. Garza, K.P. Smith, K. Tsukuda, B.C. Twichell, T.E. Wise, Department of Computer Sciences, University of Texas at Austin,

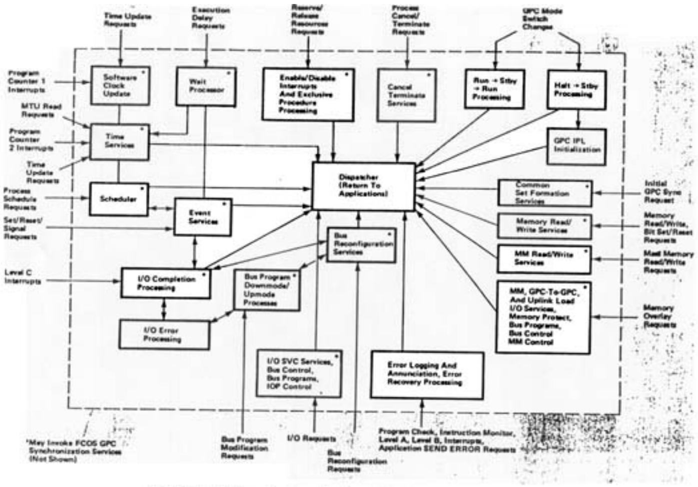
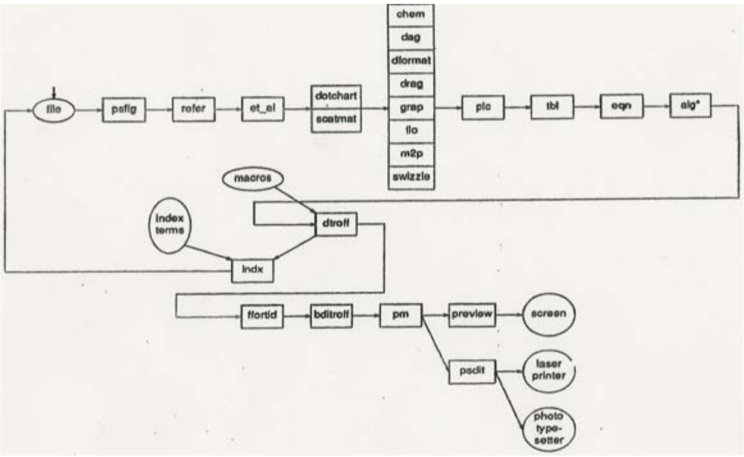

Time Update Requests
Execution Delay Requests
Reserve/ Release Resources Requests
Process Cancel/ Terminate Requests
GPC Mode Switch Changes
Program Counter 1 Interrupts
MTU Read Requests
Program Counter 2 Interrupts
Time Update Requests
Process Schedule Requests
Set/Reset/ Signal Requests
Level C Interrupts
Software Clock Update *
Wait Processor *
Enable/Disable Interrupts And Exclusive Procedure Processing *
Cancel Terminate Services *
Run → Stby → Run Processing
Halt → Stby Processing
GPC IPL Initialization
Time Services *
Dispatcher (Return To Applications)
Scheduler *
Event Services *
Common Set Formation Services *
Memory Read/ Write Services *
MM Read/Write Services *
MM, GPC-To-GPC, And Uplink Load I/O Services, Memory Protect, Bus Programs, Bus Control MM Control *
Bus Reconfiguration Services *
I/O Completion Processing *
Bus Program Downmode/ Upmode Processes *
I/O Error Processing
I/O SVC Services, Bus Control, Bus Programs, IOP Control *
Error Logging And Annunciation, Error Recovery Processing
Initial GPC Sync Request
Memory Read/Write, Bit Set/Reset Requests
Mad Memory Read/Write Requests
Memory Overlay Requests
*May Invoke FCOS GPC Synchronization Services (Not Shown)
Bus Program Modification Requests
I/O Requests
Bus Reconfiguration Requests
Program Check, Instruction Monitor, Level A, Level B, Interrupts, Application SEND ERROR Requests

FIGURE 7. Flight Computer Operating System (The FCOS dispatcher coordinates and controls all work performed by the on-board computers.)

Communications of the ACM, "Architecture of the Space Shuttle Primary Avionics Software Systems," Gene D. Carlow, September 1984, Vol. 27, No. 9, P. 933



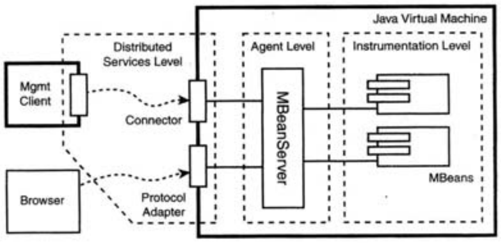
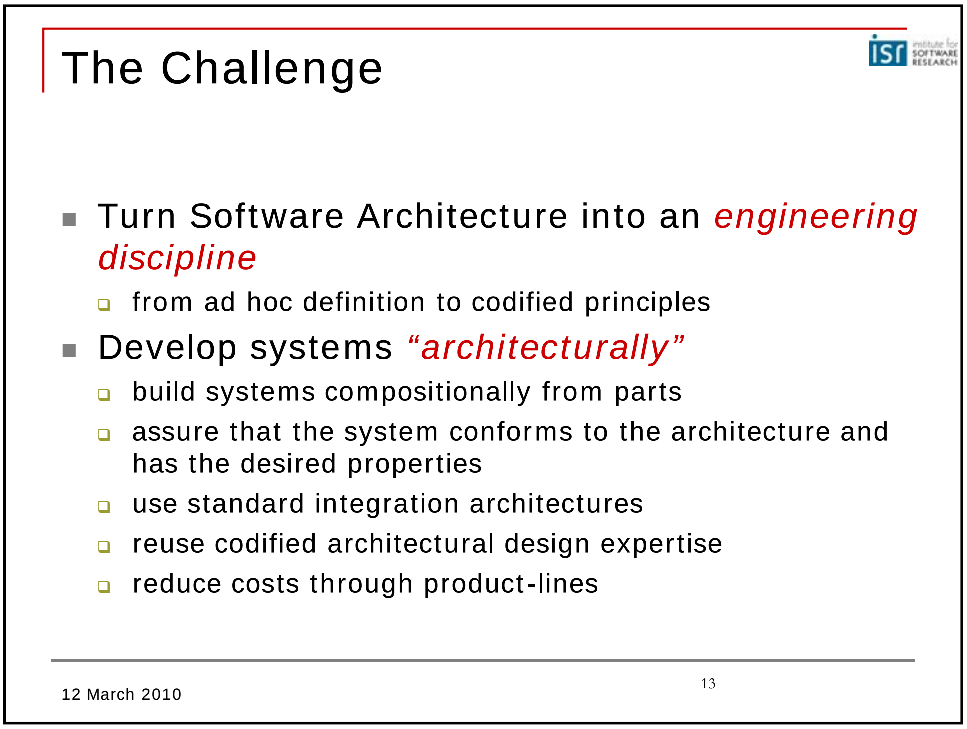
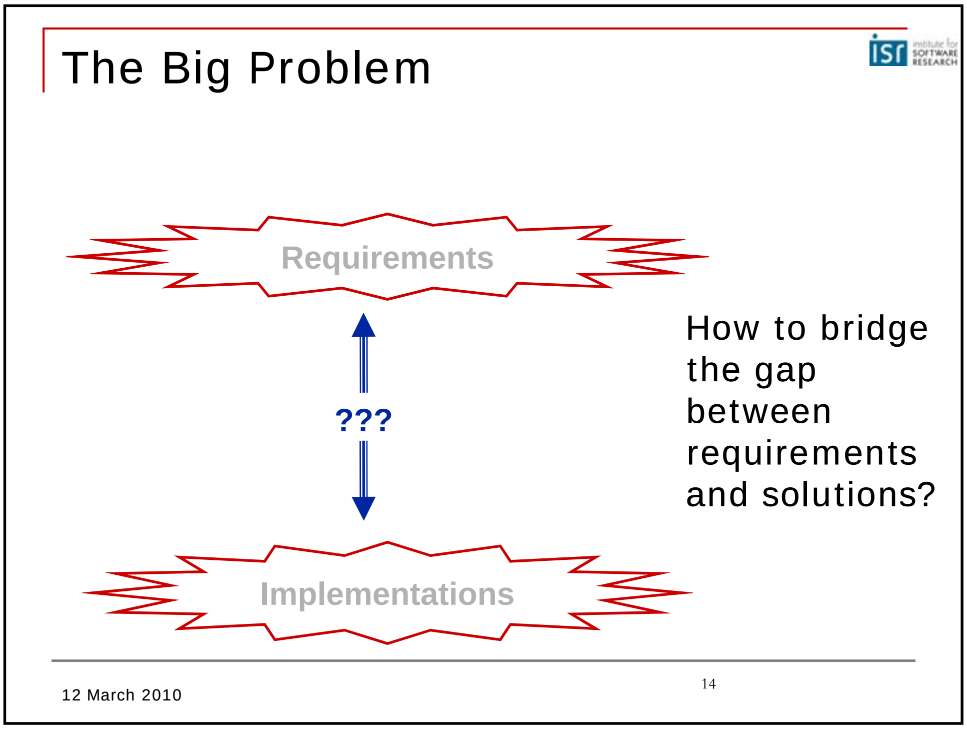


FIGURE 2.1
JMX Management Architecture.



The Challenge

- Turn Software Architecture into an *engineering discipline*
  - from ad hoc definition to codified principles
- Develop systems *“architecturally”*
  - build systems compositionally from parts
  - assure that the system conforms to the architecture and has the desired properties
  - use standard integration architectures
  - reuse codified architectural design expertise
  - reduce costs through product-lines



The Big Problem

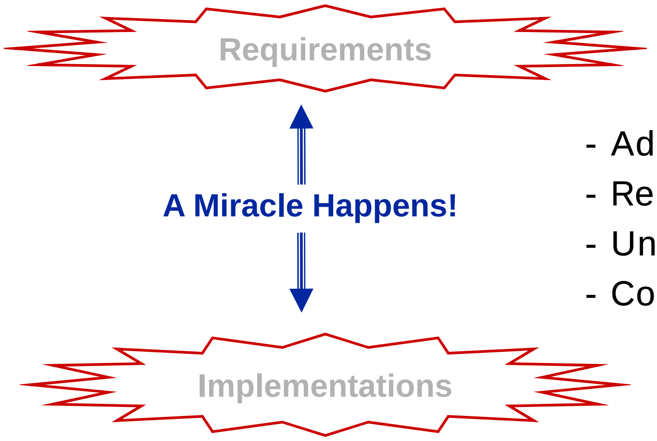
Requirements

???

Implementations

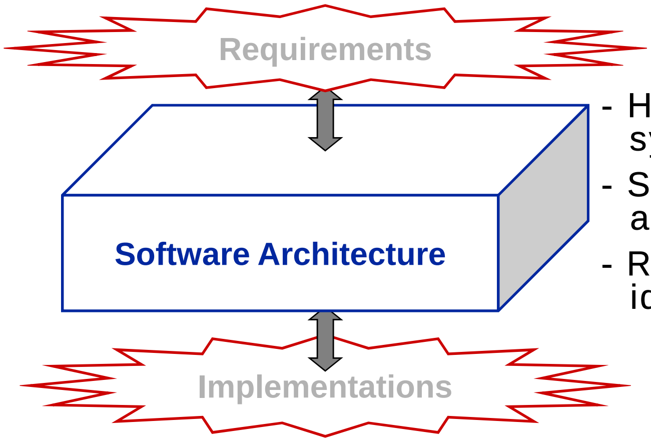
How to bridge the gap between requirements and solutions?

One Possible Answer

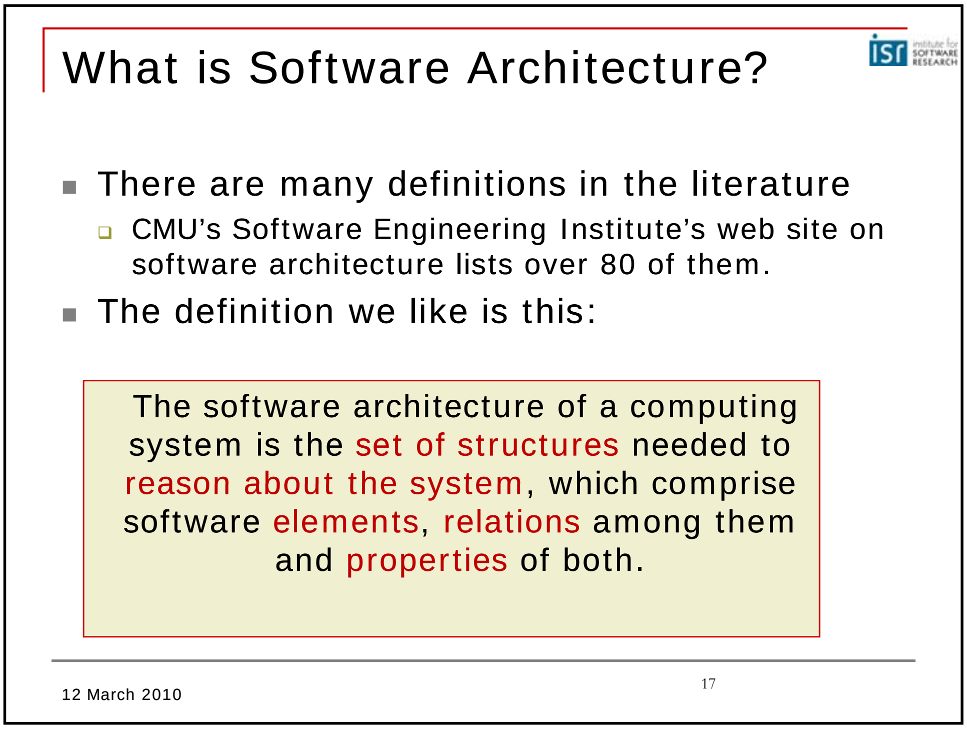


- Ad hoc
- Requires gurus
- Unpredictable
- Costly

The Role of Software Architecture



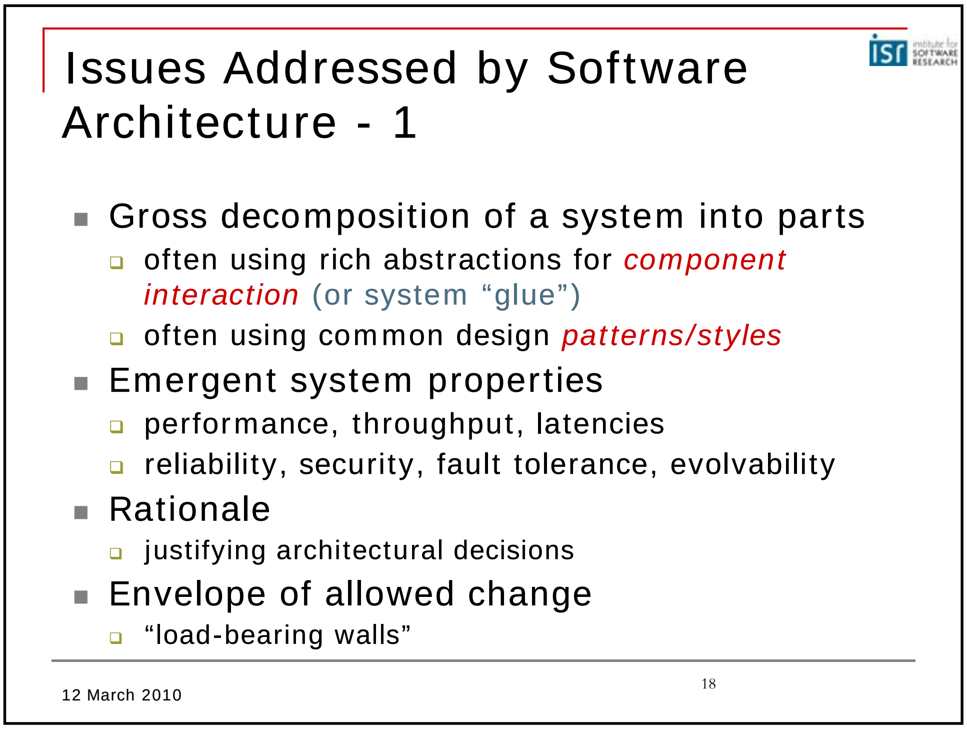
- High level of system design
- System-level abstractions
- Reuse design idioms



What is Software Architecture?

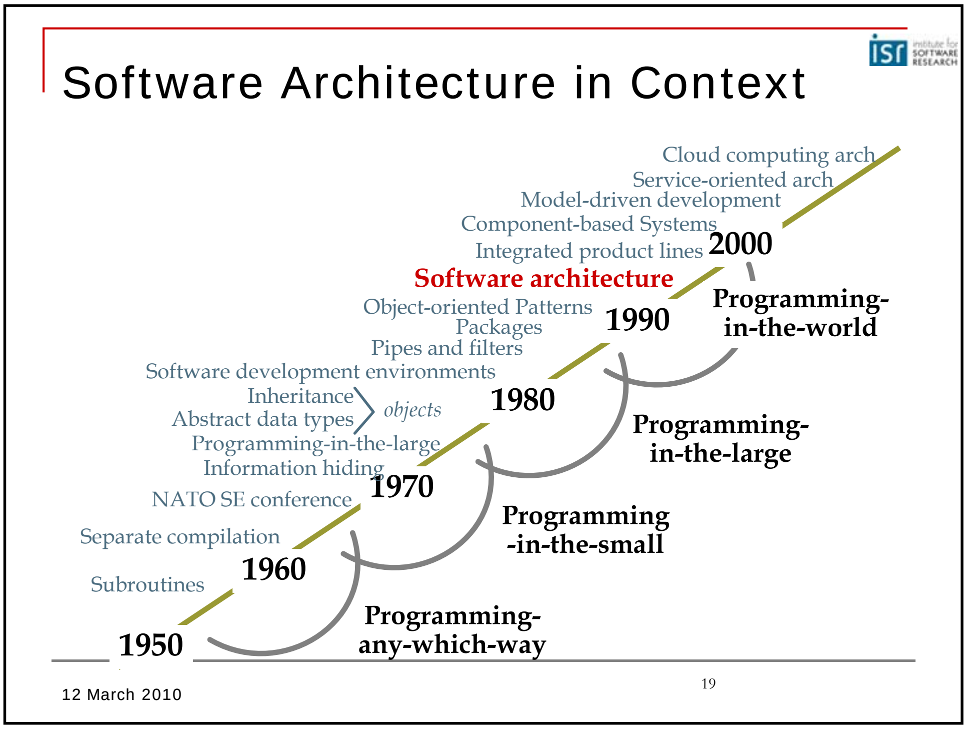
- There are many definitions in the literature
  - CMU's Software Engineering Institute's web site on software architecture lists over 80 of them.
- The definition we like is this:

> The software architecture of a computing system is the set of structures needed to reason about the system, which comprise software elements, relations among them and properties of both.



Issues Addressed by Software Architecture - 1

- Gross decomposition of a system into parts
  - often using rich abstractions for *component interaction* (or system "glue")
  - often using common design *patterns/styles*
- Emergent system properties
  - performance, throughput, latencies
  - reliability, security, fault tolerance, evolvability
- Rationale
  - justifying architectural decisions
- Envelope of allowed change
  - "load-bearing walls"

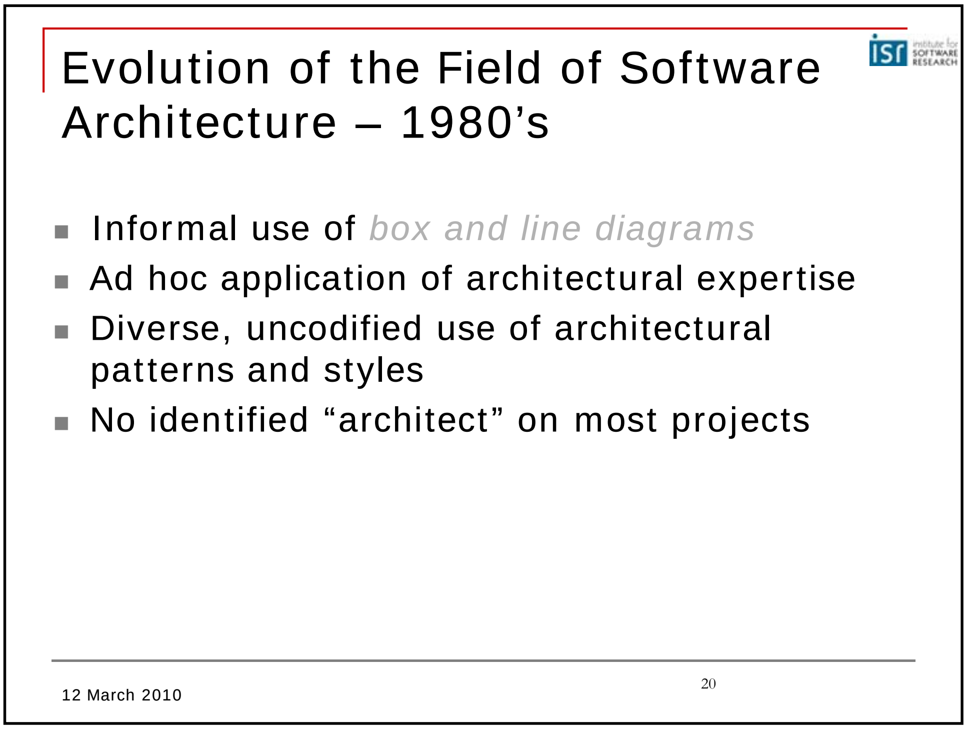


Software Architecture in Context

Cloud computing arch
Service-oriented arch
Model-driven development
Component-based Systems
Integrated product lines
2000
Software architecture
Object-oriented Patterns
Packages
Pipes and filters
1990
Programming-in-the-world
Software development environments
Inheritance
Abstract data types
objects
1980
Programming-in-the-large
Information hiding
1970
Programming-in-the-large
NATO SE conference
Separate compilation
Programming -in-the-small
Subroutines
1960
1950
Programming-any-which-way

12 March 2010

19



Evolution of the Field of Software Architecture – 1980's

- Informal use of *box and line diagrams*
- Ad hoc application of architectural expertise
- Diverse, uncodified use of architectural patterns and styles
- No identified "architect" on most projects

12 March 2010

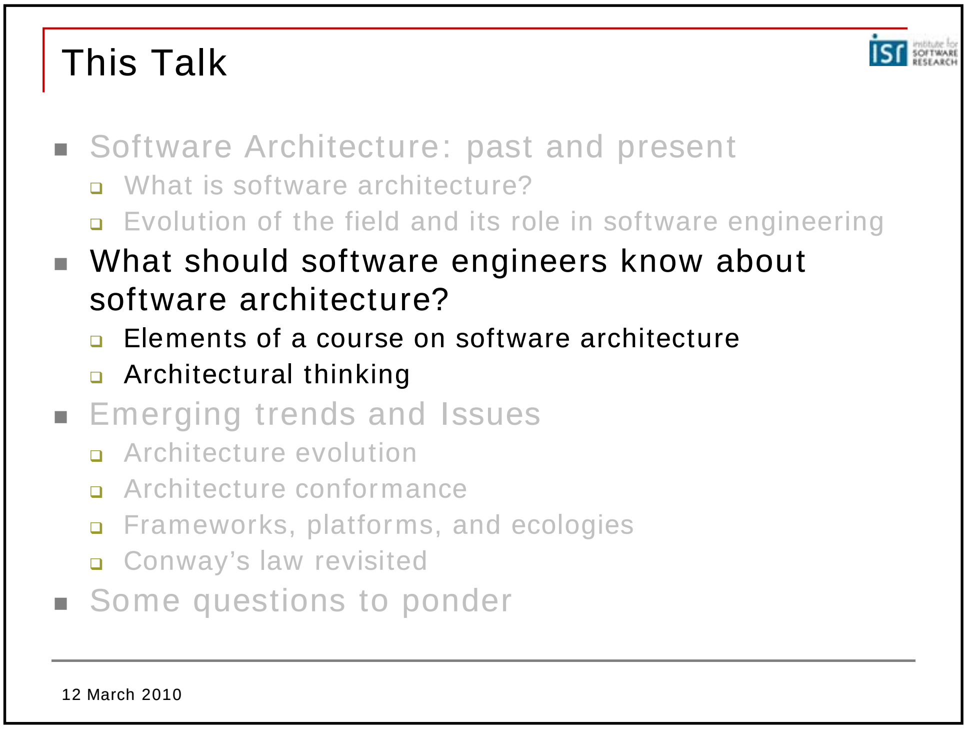
20

1990’s

- Recognition of the value of *architects* in software development organizations
- *Processes* requiring architectural design reviews & explicit architectural documentation
- Use of *product lines*, commercial architectural *standards*, component *integration frameworks*
- *Codification* of vocabulary, notations & tools for architectural design
- *Books/courses* on software architecture

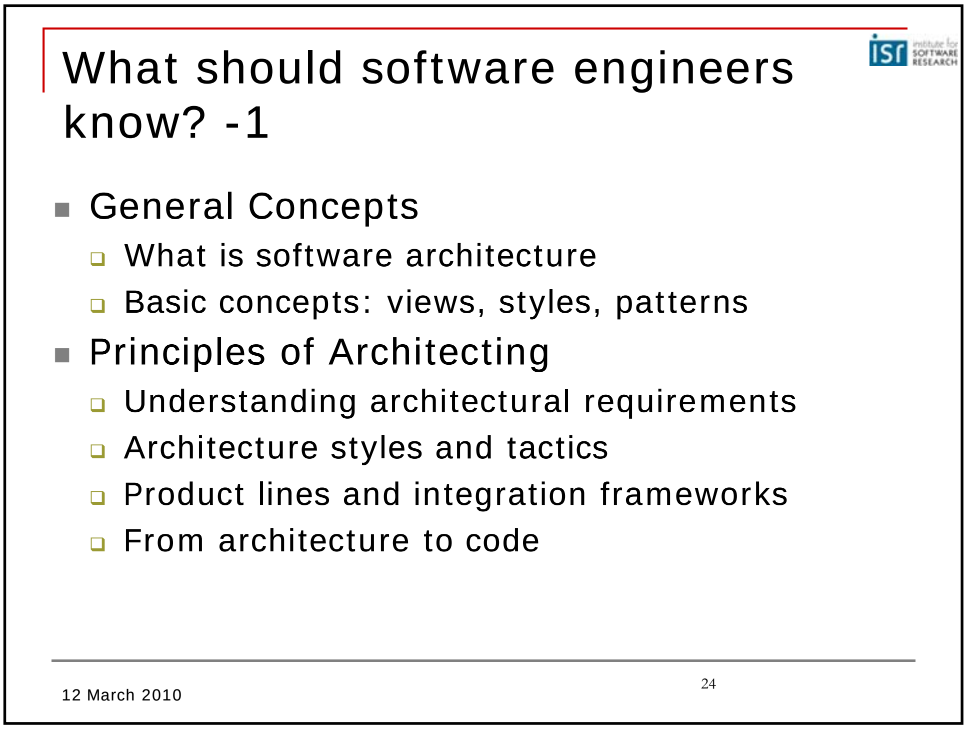
2000’s

- Incorporation of architectural notions into mainstream *design languages* and *tools* (e.g., UML-2)
- *Methods* based on architectural design and refinement (e.g., Model-Driven Design)
- Some architecture *analysis tools*
- Architectural *standards* for Enterprise Systems (e.g., RM-ODP, TOGAF)
- Architectural *frameworks* (e.g., SOA)



This Talk

- Software Architecture: past and present
  - What is software architecture?
  - Evolution of the field and its role in software engineering
- What should software engineers know about software architecture?
  - Elements of a course on software architecture
  - Architectural thinking
- Emerging trends and Issues
  - Architecture evolution
  - Architecture conformance
  - Frameworks, platforms, and ecologies
  - Conway's law revisited
- Some questions to ponder



What should software engineers know? -1

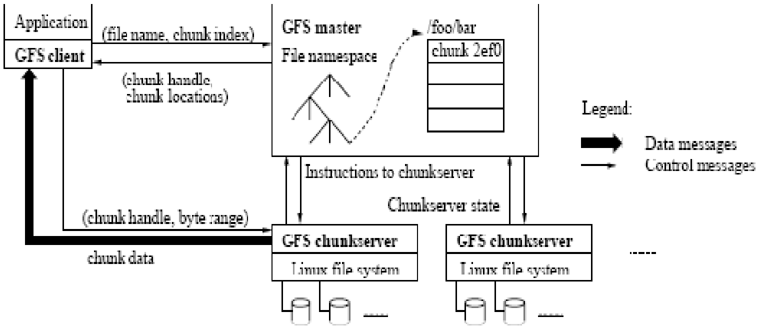
- General Concepts
  - What is software architecture
  - Basic concepts: views, styles, patterns
- Principles of Architecting
  - Understanding architectural requirements
  - Architecture styles and tactics
  - Product lines and integration frameworks
  - From architecture to code

What should software engineers know? -2

- Architecture in Practice
  - Evaluating architectural designs
  - Handling architectural problems
  - Documenting a software architecture
  - Presenting an architecture to others
  - Architecture for X (security, usability, reliability, etc.)

Architectural Thinking - 1

An engineering mindset



Source: “The Google File System” Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung

Figure 1: GFS Architecture

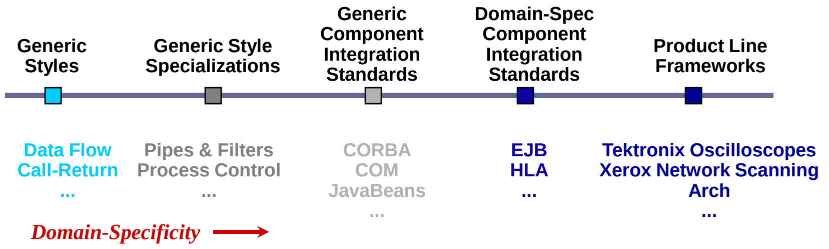
Architectural Thinking - 2

Different issues for architecture & programs

Architecture	*Programs*
interactions among parts	implementations of parts
structural properties	computational properties
declarative	operational
mostly static	mostly dynamic
system-level performance	algorithmic performance
outside module boundary	inside module boundary

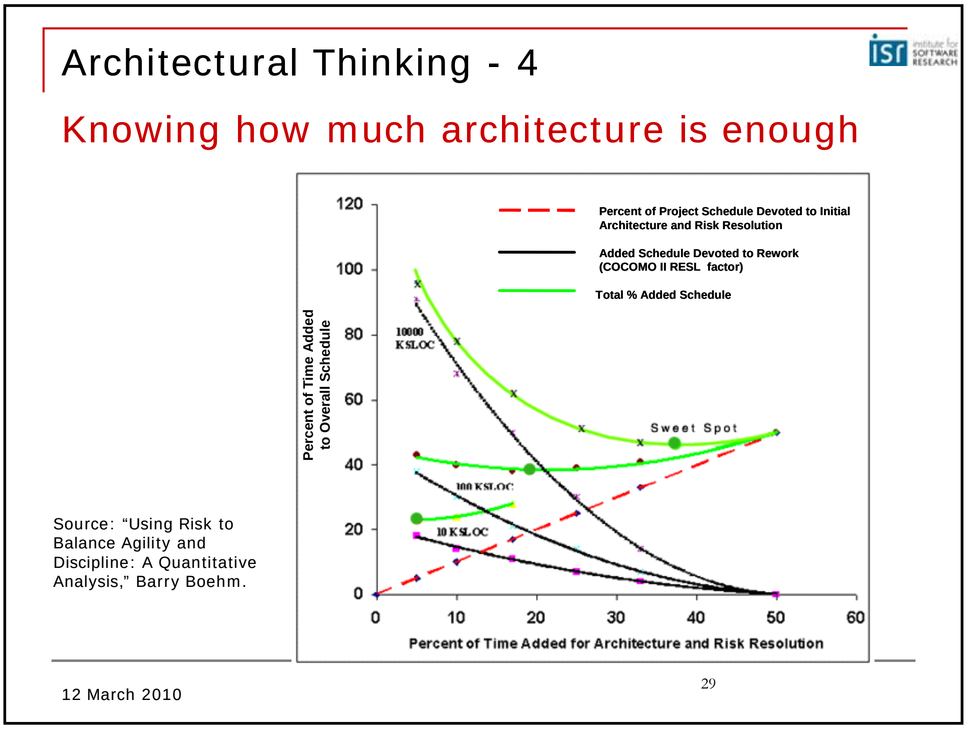
Architectural Thinking - 3

Product lines, platforms, and styles



Generic Styles	Generic Style Specializations	Generic Component Integration Standards	Domain-Spec Component Integration Standards	Product Line Frameworks
Data Flow Call-Return ...	Pipes & Filters Process Control ...	CORBA COM JavaBeans ...	EJB HLA ...	Tektronix Oscilloscopes Xerox Network Scanning Arch ...

Domain-Specificity →

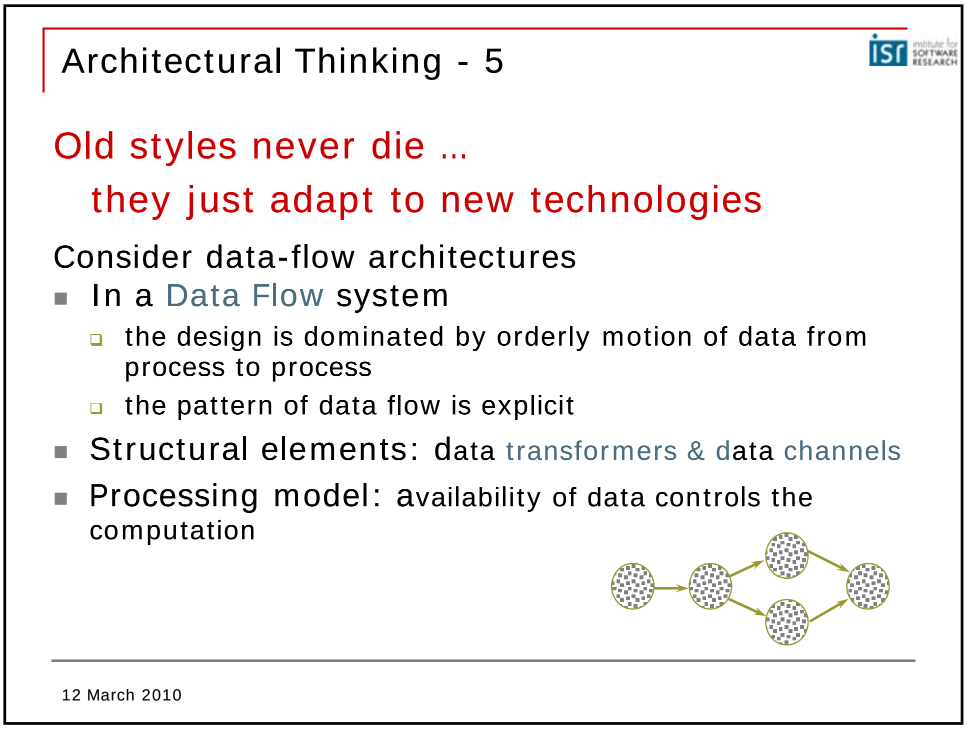


Architectural Thinking - 4

Knowing how much architecture is enough

Source: "Using Risk to Balance Agility and Discipline: A Quantitative Analysis," Barry Boehm.

12 March 2010



Architectural Thinking - 5

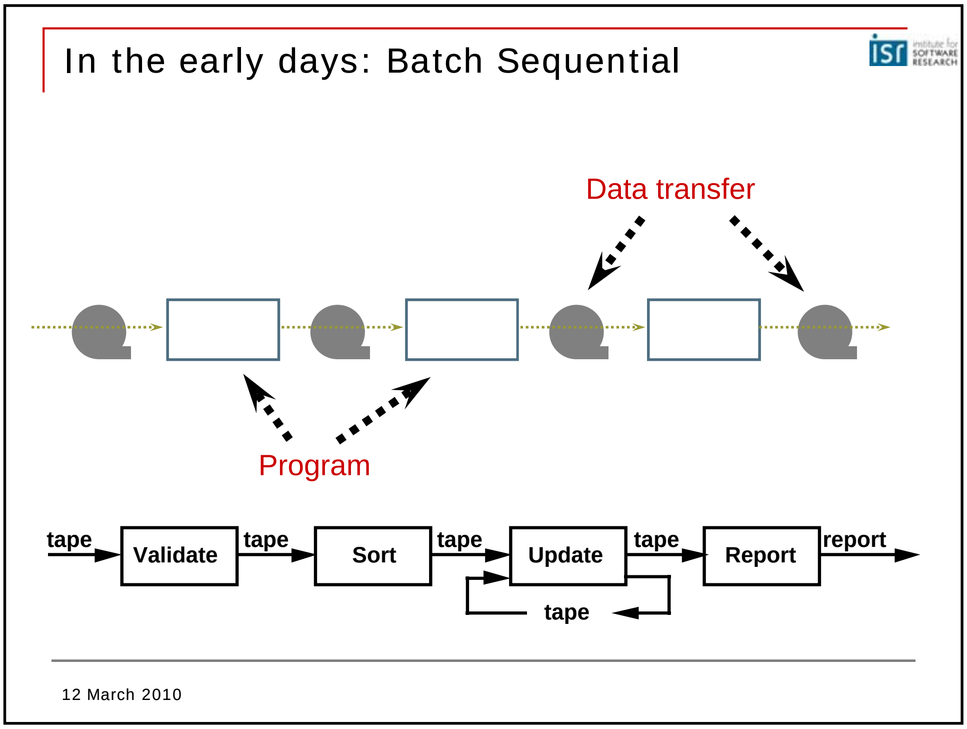
Old styles never die …

they just adapt to new technologies

Consider data-flow architectures

- In a Data Flow system
  - the design is dominated by orderly motion of data from process to process
  - the pattern of data flow is explicit
- Structural elements: data transformers & data channels
- Processing model: availability of data controls the computation

12 March 2010



In the early days: Batch Sequential

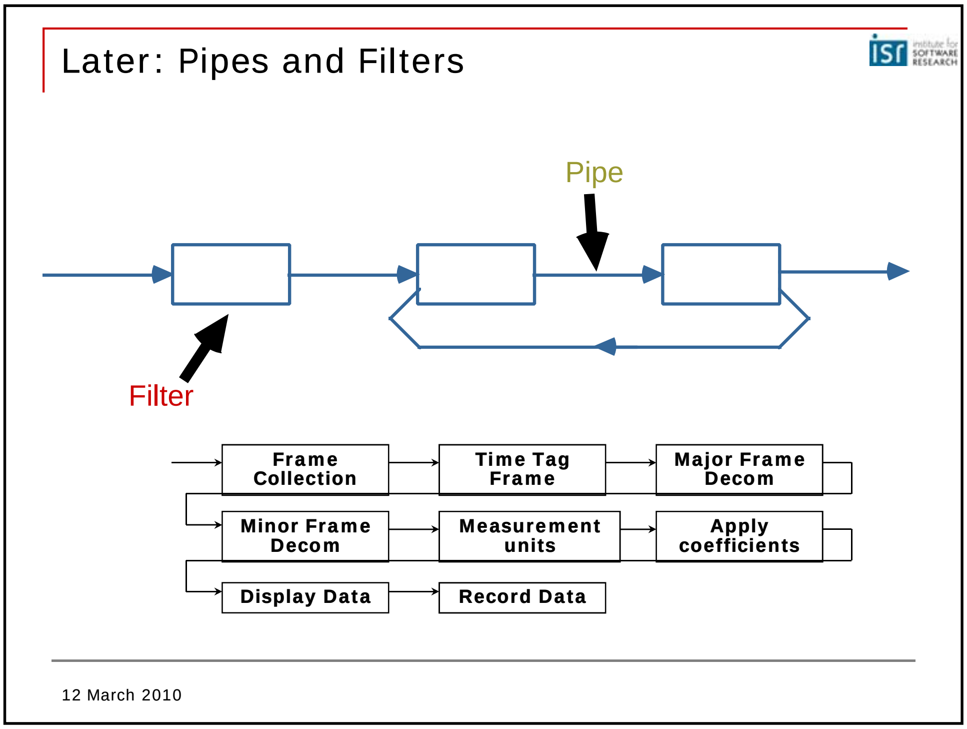
Data transfer

Program

tape → Validate → tape → Sort → tape → Update → tape → Report → report

tape

12 March 2010



Later: Pipes and Filters

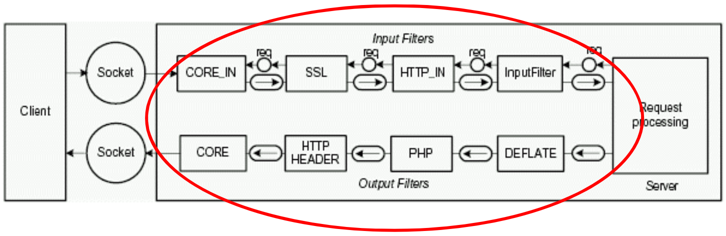
Pipe

Filter

Frame Collection → Time Tag Frame → Major Frame Decom → Minor Frame Decom → Measurement units → Apply coefficients → Display Data → Record Data

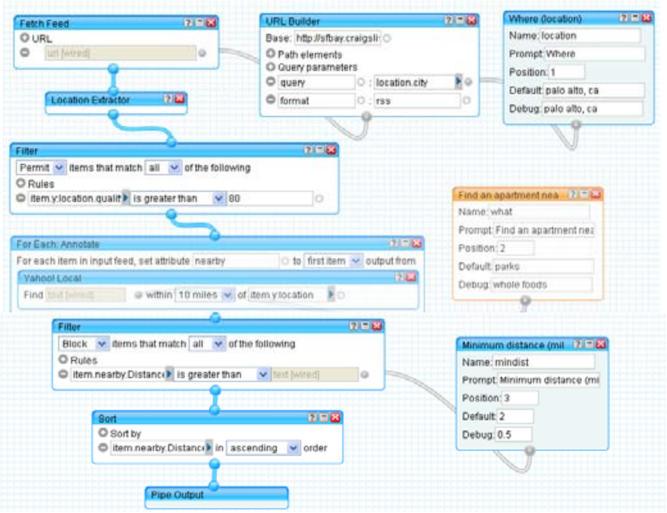
12 March 2010

Example: Apache



Source: Apache Modeling Project: Bernhard Gröne, Andreas Knöpfel, Rudolf Kugel, Oliver Schmidt
http://www.fmc-modeling.org/category/projects/apache/amp/Apache_Modeling_Project.html

Today: Yahoo! Pipes



Fetch Feed
○ URL
⊖ url [wired]

Location Extractor

URL Builder
Base: http://sfbay.craigsli
○ Path elements
○ Query parameters
⊖ query : location.city
⊖ format : rss

Where (location)
Name: location
Prompt: Where
Position: 1
Default: palo alto, ca
Debug: palo alto, ca

Filter
Permit items that match all of the following
○ Rules
⊖ item.y:location.qualit is greater than 80

Find an apartment nea
Name: what
Prompt: Find an apartment nea
Position: 2
Default: parks
Debug: whole foods

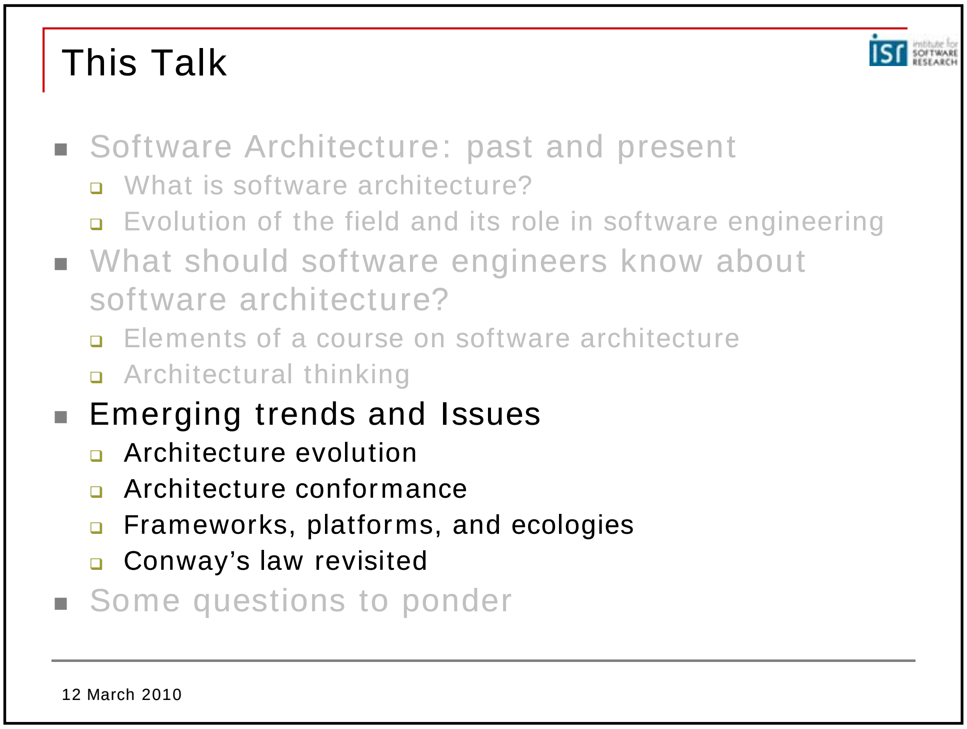
For Each: Annotate
For each item in input feed, set attribute nearby to first item output from
Yahoo! Local
Find text [wired] within 10 miles of item.y:location

Filter
Block items that match all of the following
○ Rules
⊖ item.nearby.Distanc is greater than text [wired]

Minimum distance (mil
Name: mindist
Prompt: Minimum distance (mi
Position: 3
Default: 2
Debug: 0.5

Sort
○ Sort by
⊖ item.nearby.Distanc in ascending order

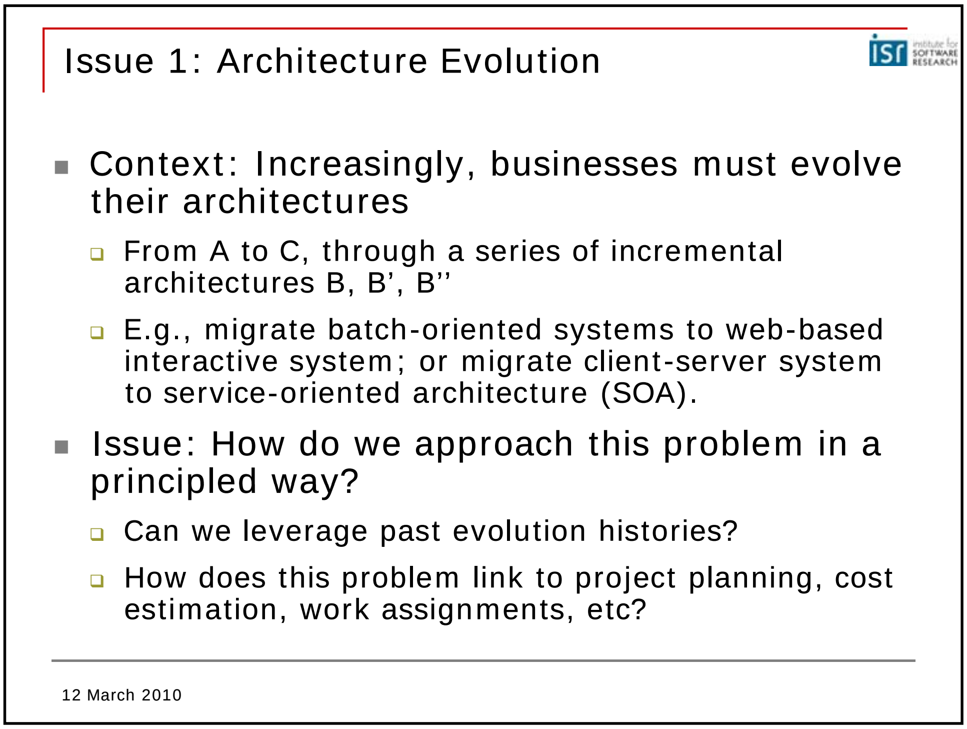
Pipe Output



This Talk

- Software Architecture: past and present
  - What is software architecture?
  - Evolution of the field and its role in software engineering
- What should software engineers know about software architecture?
  - Elements of a course on software architecture
  - Architectural thinking
- Emerging trends and Issues
  - Architecture evolution
  - Architecture conformance
  - Frameworks, platforms, and ecologies
  - Conway’s law revisited
- Some questions to ponder

12 March 2010



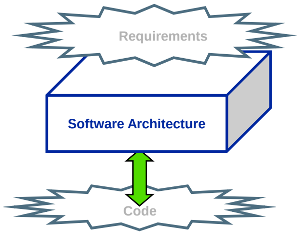
Issue 1: Architecture Evolution

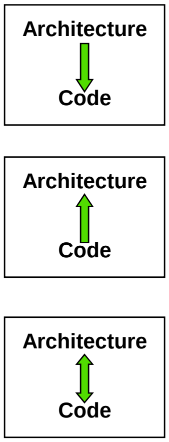
- Context: Increasingly, businesses must evolve their architectures
  - From A to C, through a series of incremental architectures B, B’, B’’
  - E.g., migrate batch-oriented systems to web-based interactive system; or migrate client-server system to service-oriented architecture (SOA).
- Issue: How do we approach this problem in a principled way?
  - Can we leverage past evolution histories?
  - How does this problem link to project planning, cost estimation, work assignments, etc?

12 March 2010

Issue 2: Architecture Conformance

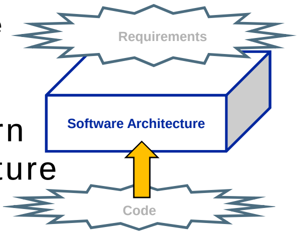
- Context: We would like to make sure that the implementation conforms to architecture (and vice versa)
  - This is the Achilles Heal of software architecture
- Issue: What does it really mean to "conform" and how would we evaluate its satisfaction?



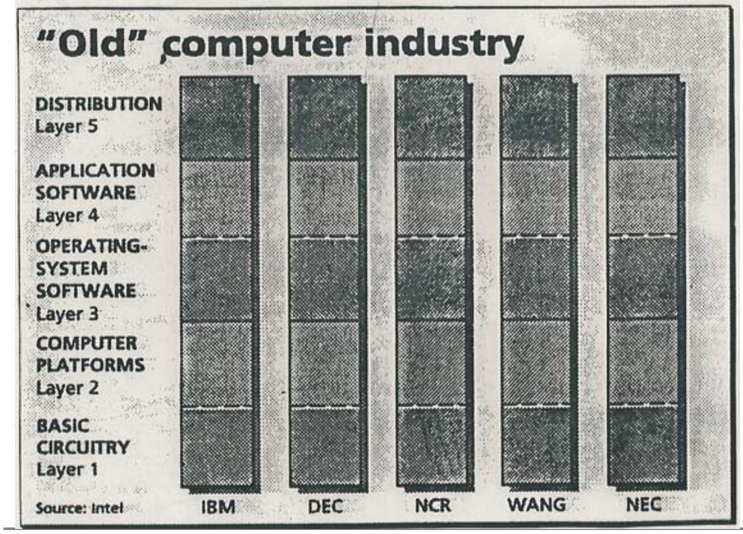


Issue 3: Frameworks, Platforms, and Ecologies

- Context: We have been building on top of platforms and using software frameworks for most of the history of software engineering
  - This introduces an upward constraint on the architecture
  - The nature of such platforms has evolved
- Issue: What is the nature of modern platforms and how should architecture accommodate those?
  - Reflects a historical migration of architectural concerns
  - Needs to be rethought in the presence of the Internet



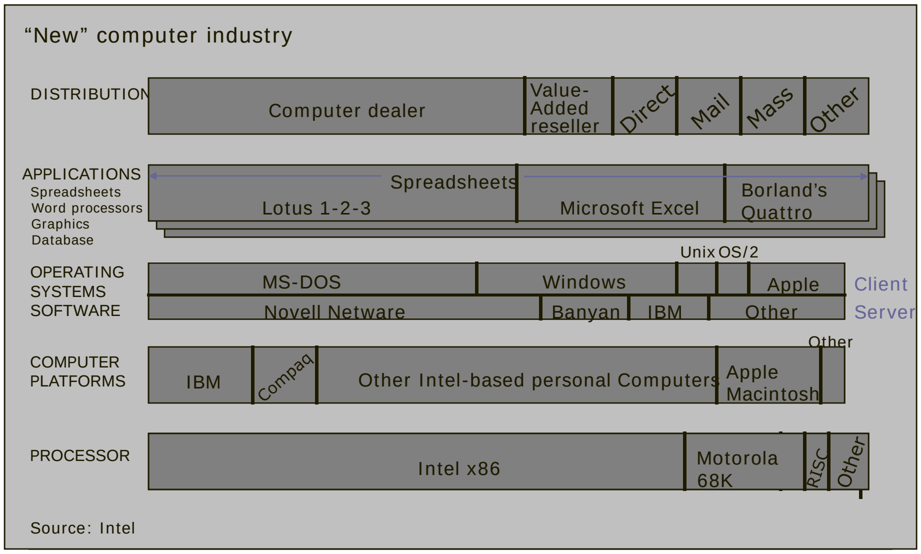
Structure of the Mainframe Computer Industry



Reprinted from *The Economist,* Feb 27, 1993

12 March 2010

New Structure of the Computer Industry



"New" computer industry

DISTRIBUTION: Computer dealer | Value-Added reseller | Direct | Mail | Mass | Other

APPLICATIONS (Spreadsheets, Word processors, Graphics, Database): Spreadsheets — Lotus 1-2-3 | Microsoft Excel | Borland's Quattro

OPERATING SYSTEMS SOFTWARE: MS-DOS | Windows | Unix | OS/2 | Apple — Client
Novell Netware | Banyan | IBM | Other — Server

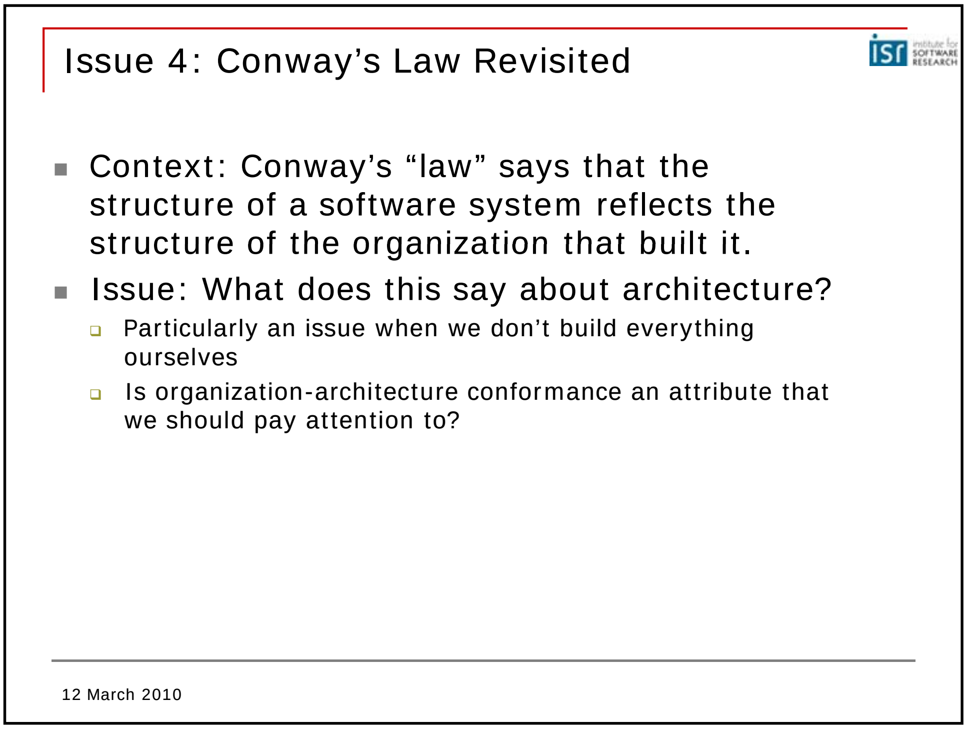
COMPUTER PLATFORMS: IBM | Compaq | Other Intel-based personal Computers | Apple Macintosh | Other

PROCESSOR: Intel x86 | Motorola 68K | RISC | Other

Source: Intel

12 March 2010

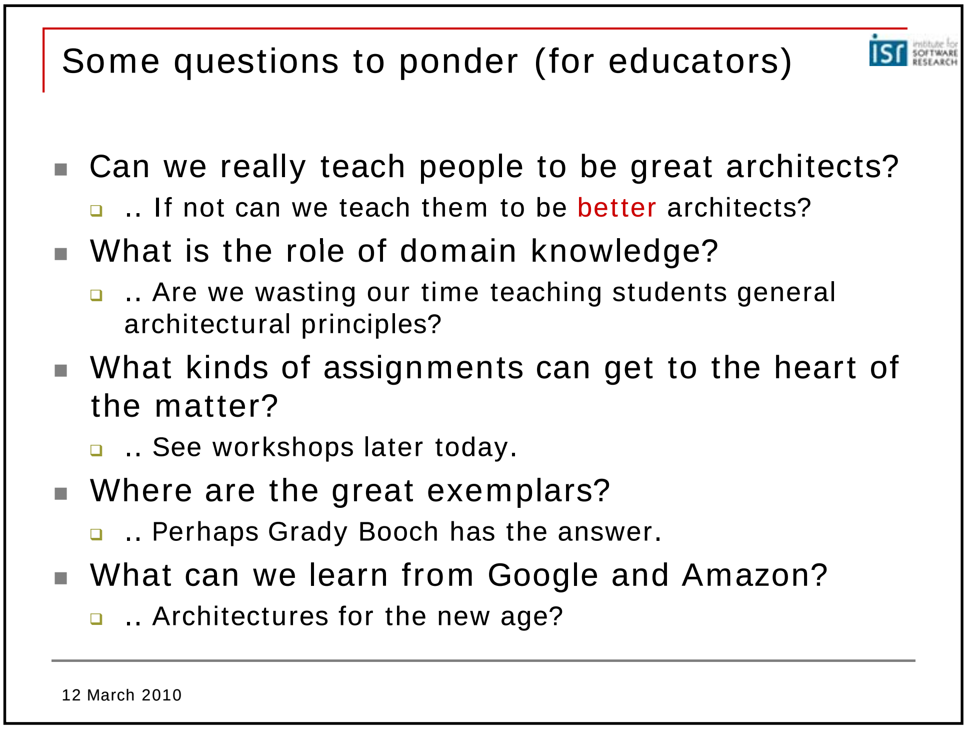
Reprinted from *The Economist,* Feb 27, 1993



Issue 4: Conway's Law Revisited

- Context: Conway's "law" says that the structure of a software system reflects the structure of the organization that built it.
- Issue: What does this say about architecture?
  - Particularly an issue when we don't build everything ourselves
  - Is organization-architecture conformance an attribute that we should pay attention to?

12 March 2010



Some questions to ponder (for educators)

- Can we really teach people to be great architects?
  - .. If not can we teach them to be better architects?
- What is the role of domain knowledge?
  - .. Are we wasting our time teaching students general architectural principles?
- What kinds of assignments can get to the heart of the matter?
  - .. See workshops later today.
- Where are the great exemplars?
  - .. Perhaps Grady Booch has the answer.
- What can we learn from Google and Amazon?
  - .. Architectures for the new age?

12 March 2010