



IBM Software Group | Rational Software



# Best Practices in Game Development

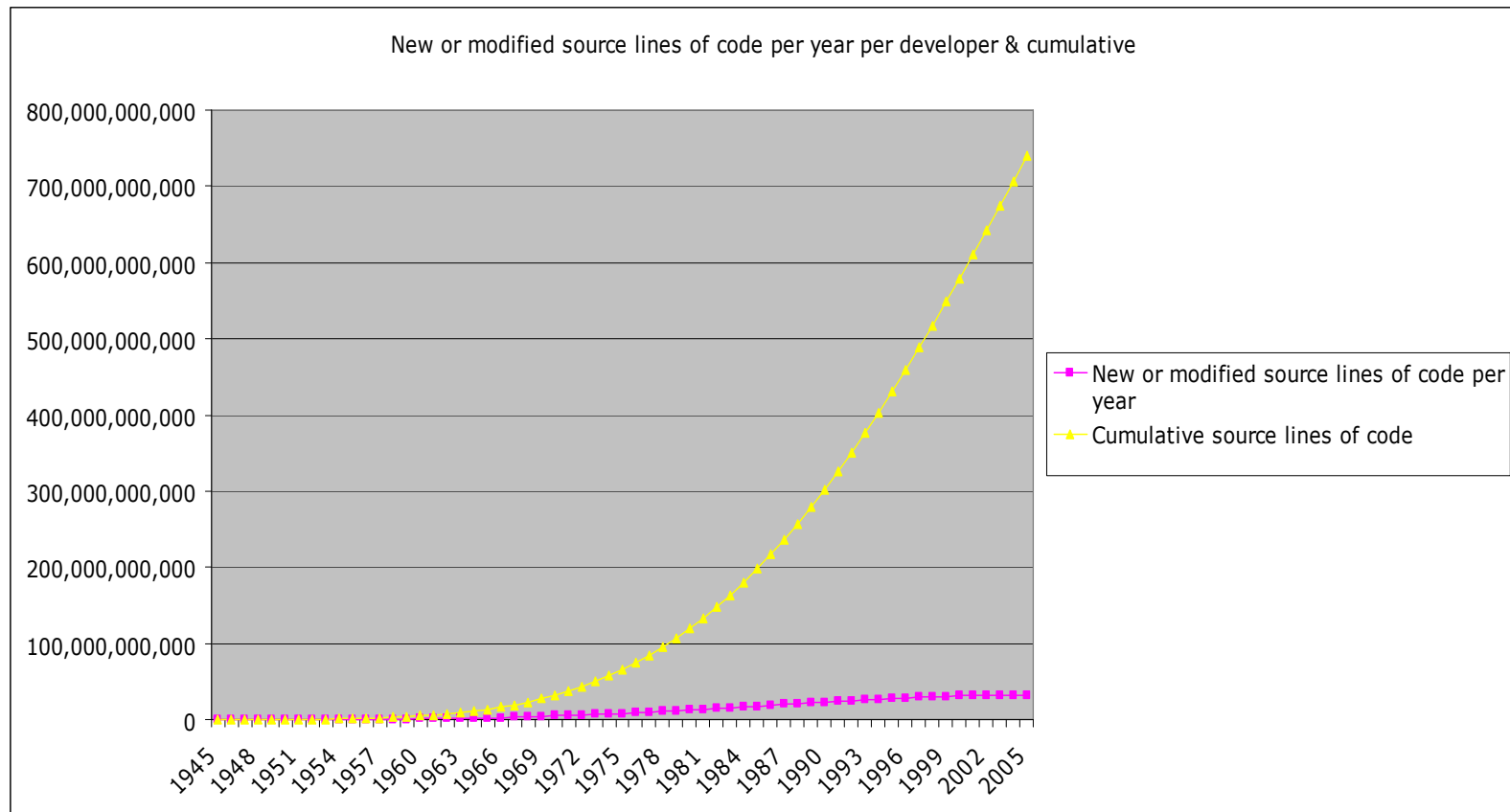
Grady Booch  
IBM Fellow & Free Radical

March 28, 2006

© 2005 IBM Corporation

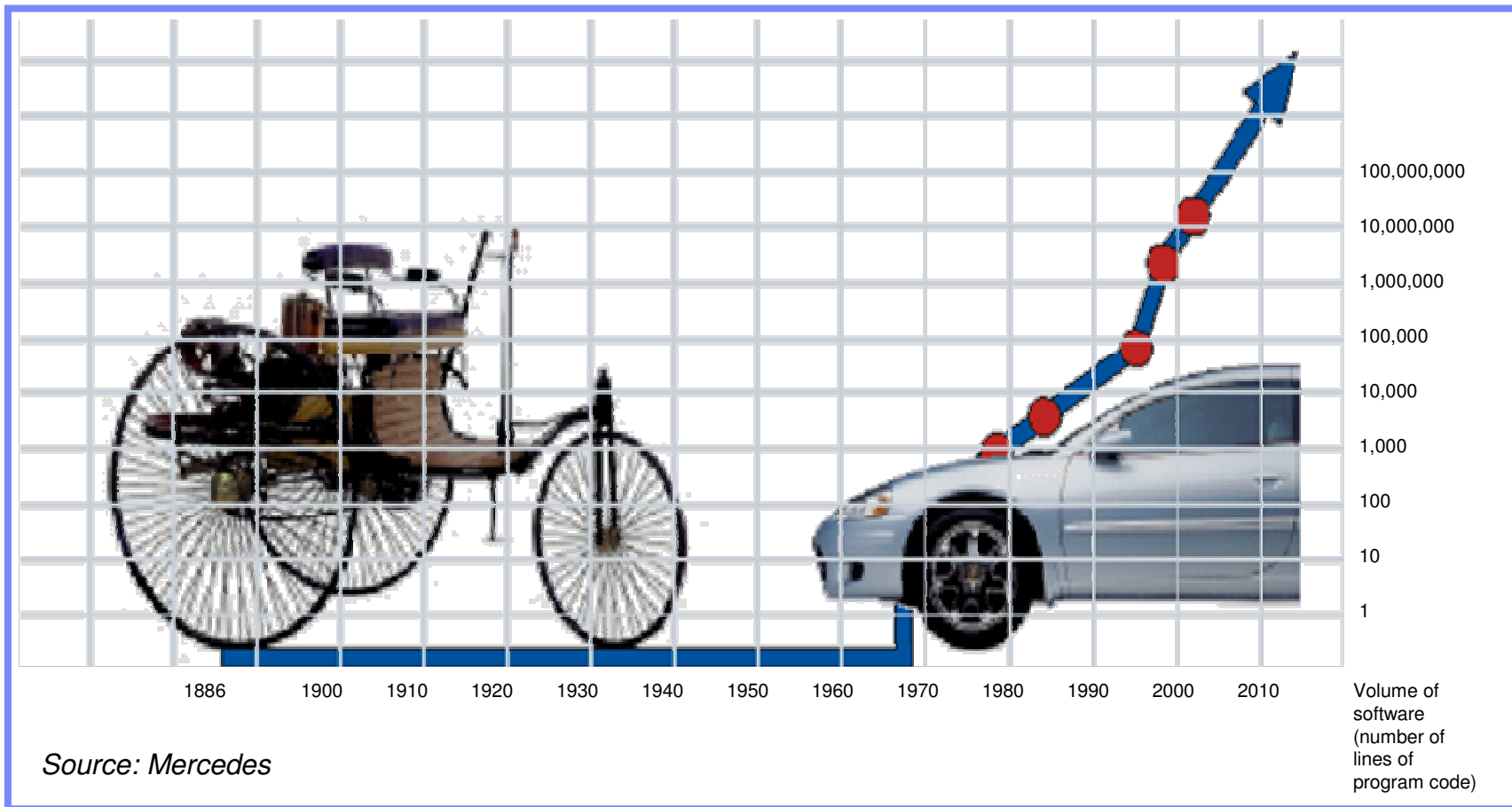


## New Or Modified SLOC/Year And Cumulative



<http://www.booch.com/architecture/systems.jsp>

# Growth Of Software-Intensive Systems

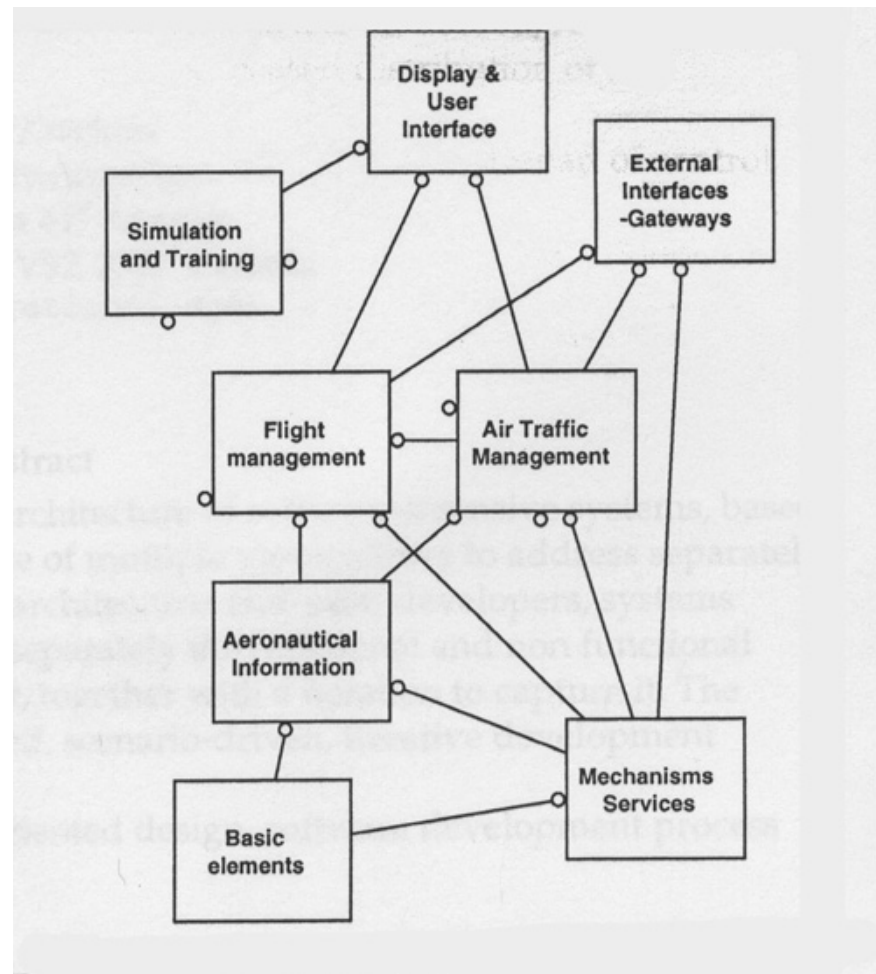


Gartner, April 2003, *Embedded Software Development and Management – Automotive Industry*



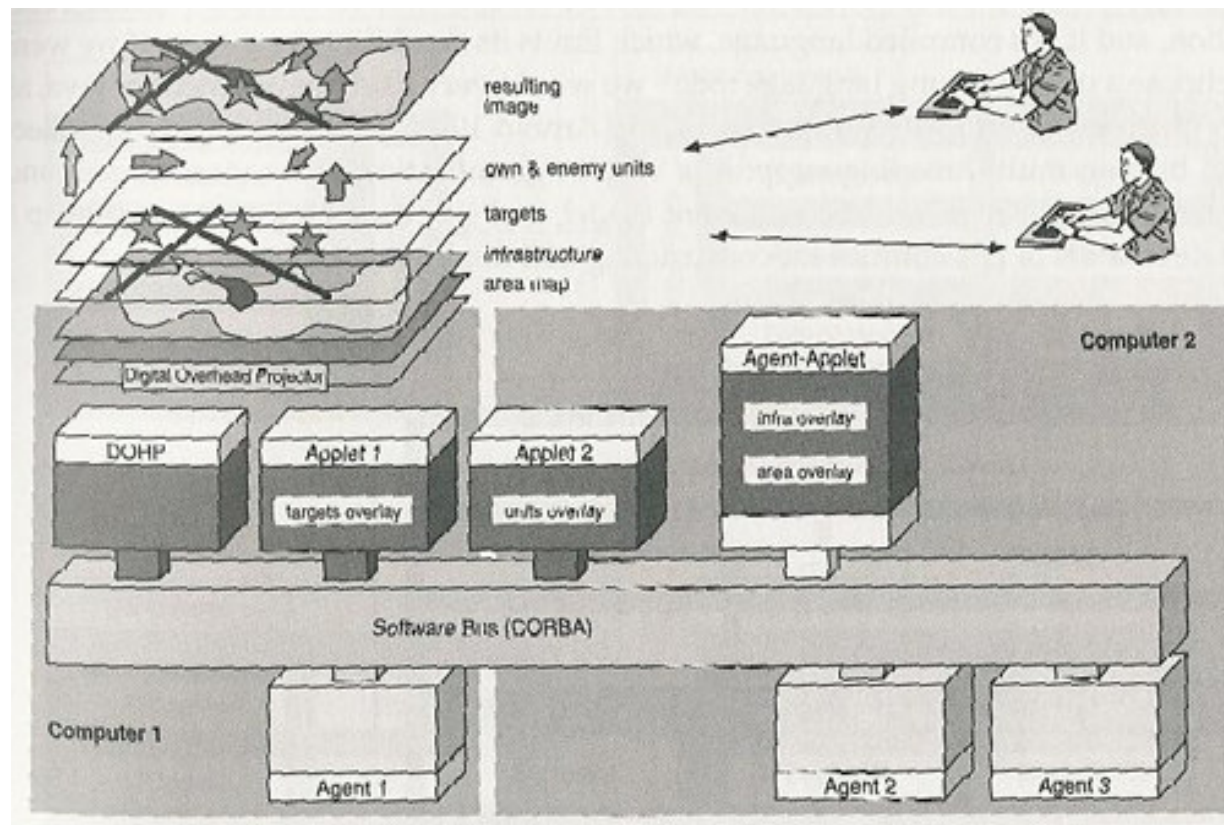
You can't build an enduring a business  
just by hiring bright people  
throwing them in a room together  
and hoping that they'll do great things

## Gallery of Software Architecture - Air Traffic Control



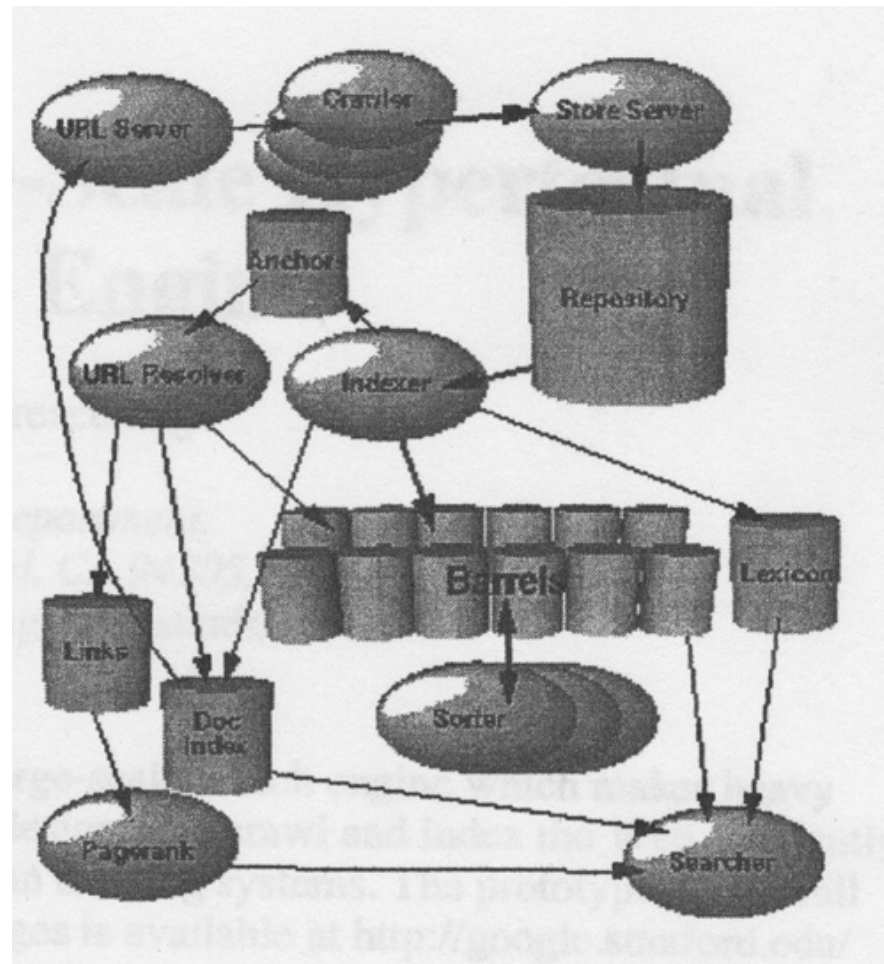
<http://www.booch.com/architecture/architecture.jsp?part=Gallery>

## Gallery of Software Architecture - C3I



<http://www.booch.com/architecture/architecture.jsp?part=Gallery>

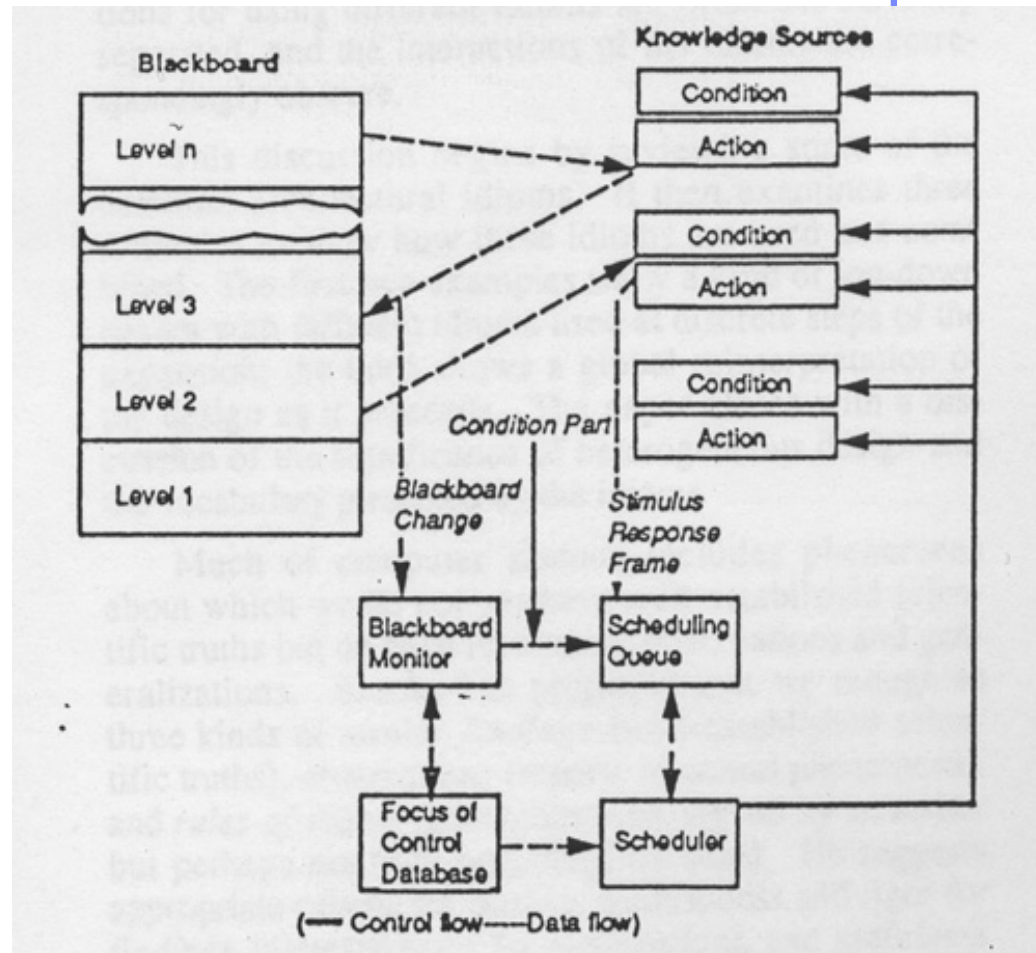
## Gallery of Software Architecture - Google



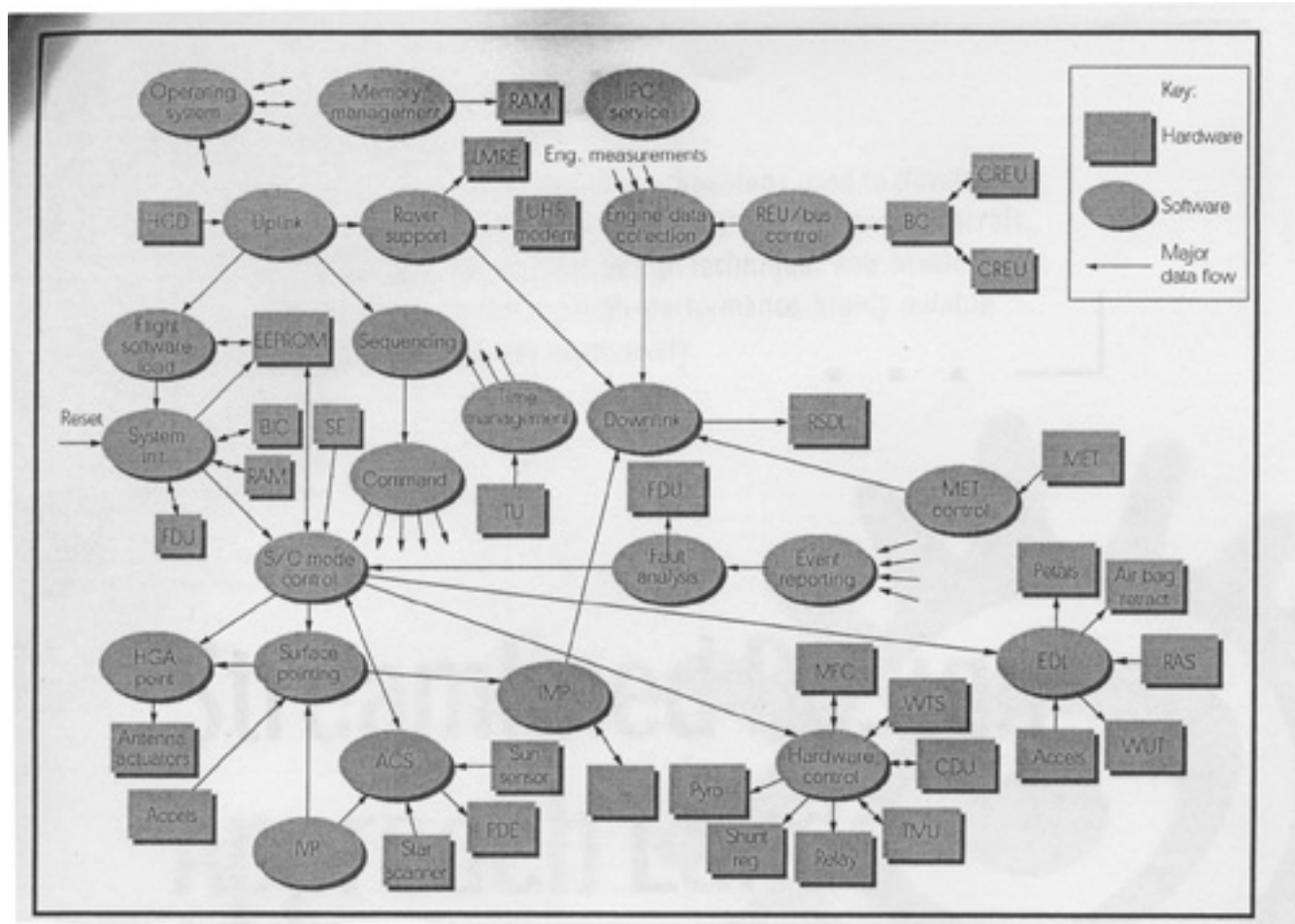
<http://www.booch.com/architecture/architecture.jsp?part=Gallery>



# Gallery of Software Architecture - Speech Recognition

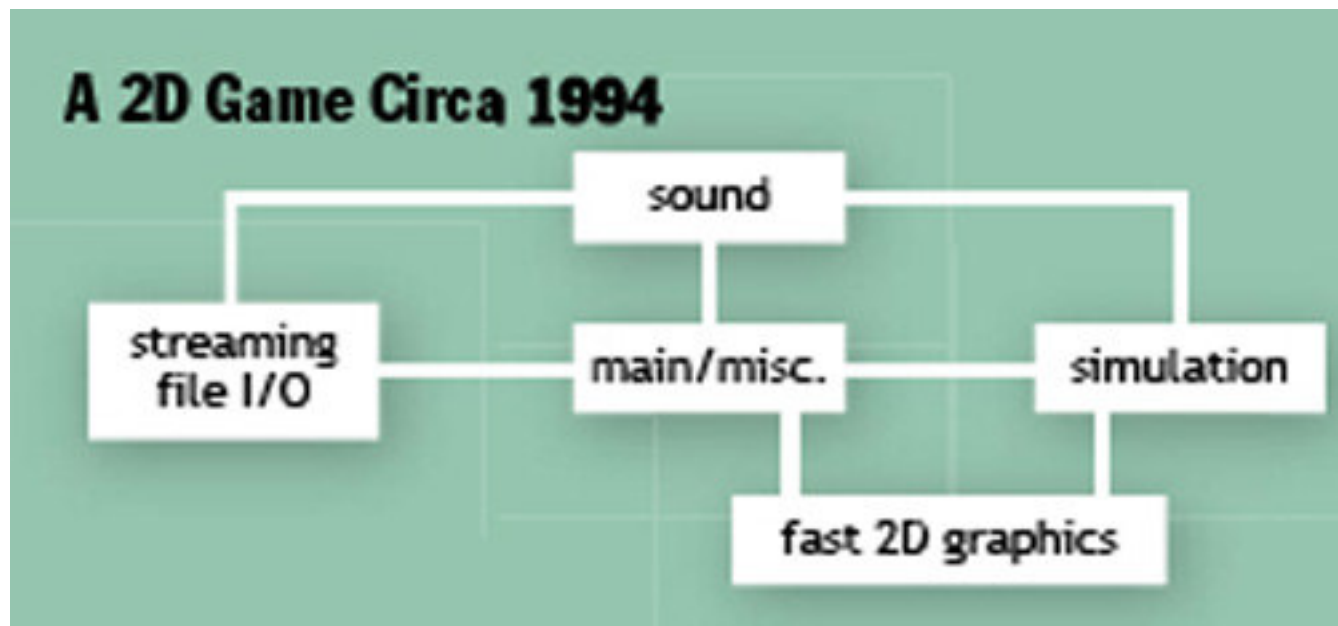


## Gallery of Software Architecture - Pathfinder



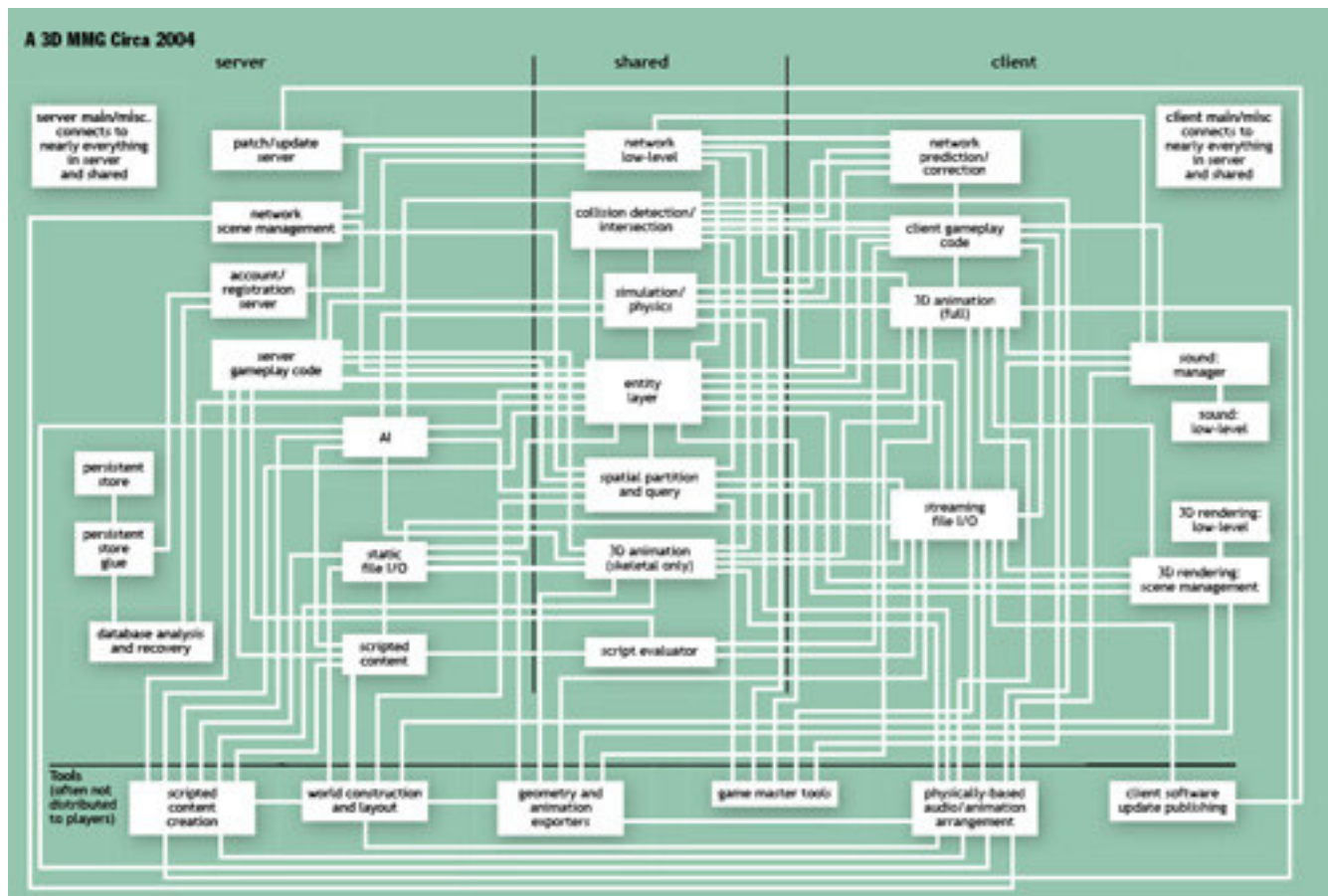
<http://www.booch.com/architecture/architecture.jsp?part=Gallery>

## Gallery of Software Architecture - Games



<http://www.booch.com/architecture/architecture.jsp?part=Gallery>

## Gallery of Architecture - Games



<http://www.booch.com/architecture/architecture.jsp?part=Gallery>



## The Limits Of Software

The laws of physics

The laws of software

The challenge of algorithms

The difficulty of distribution

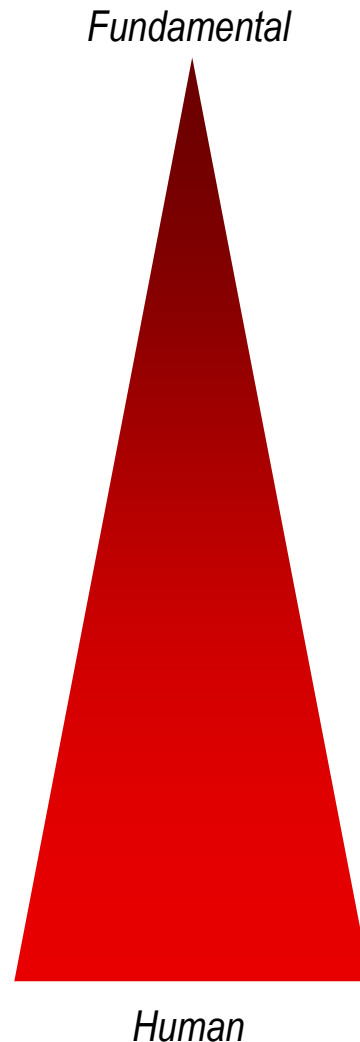
The problems of design

The importance of organization

The impact of economics

The influence of politics

The limits of human imagination



Most hyperproductive organizations  
grow their architecture through the  
incremental and iterative release  
of executables





copyright 1991 philg@mit.edu





## From Vision To Execution



<http://www.gehrytechnologies.com/>

## Early Architecture



### Progress

- Limited knowledge of theory

# Contemporary Architecture



## Progress

- Advances in materials
- Advances in analysis

## Scale

- 5 times the span of the Pantheon
- 3 times the height of Cheops



# Movements In Civil Architecture

- Egyptian/Babylonian, Assyrian, Persian
- Classical Indian
- Dynastic China
- Grecian/Roman
- Early Christian/Byzantine
- Islamic
- Romanesque
- Gothic
- Renaissance
- Palladian
- Neoclassical
- Picturesque
- Mannerism
- Baroque
- Engineering/Rational/National/Romantic
- Art Noveau
- Modernism

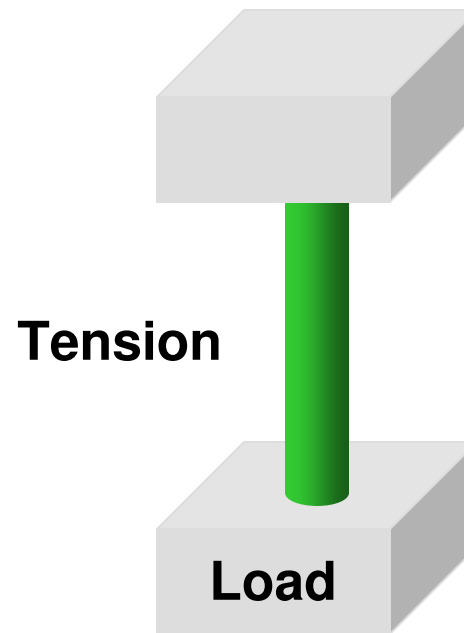
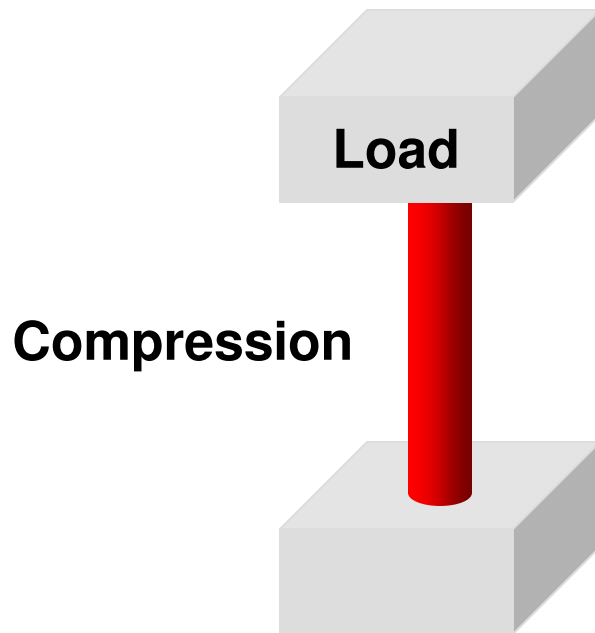
## Progress

- Imitation of previous efforts
- Learning from failure
- Integration of other forces
- Experimentation

## Architects

- Imhotep
- Vitruvius
- Michelangelo
- Palladio
- Wright
- Wren
- LeCorbusier
- Geary
- Libeskind

## Forces In Civil Architecture



### Kinds of loads

- Dead loads
- Live loads
- Dynamic loads

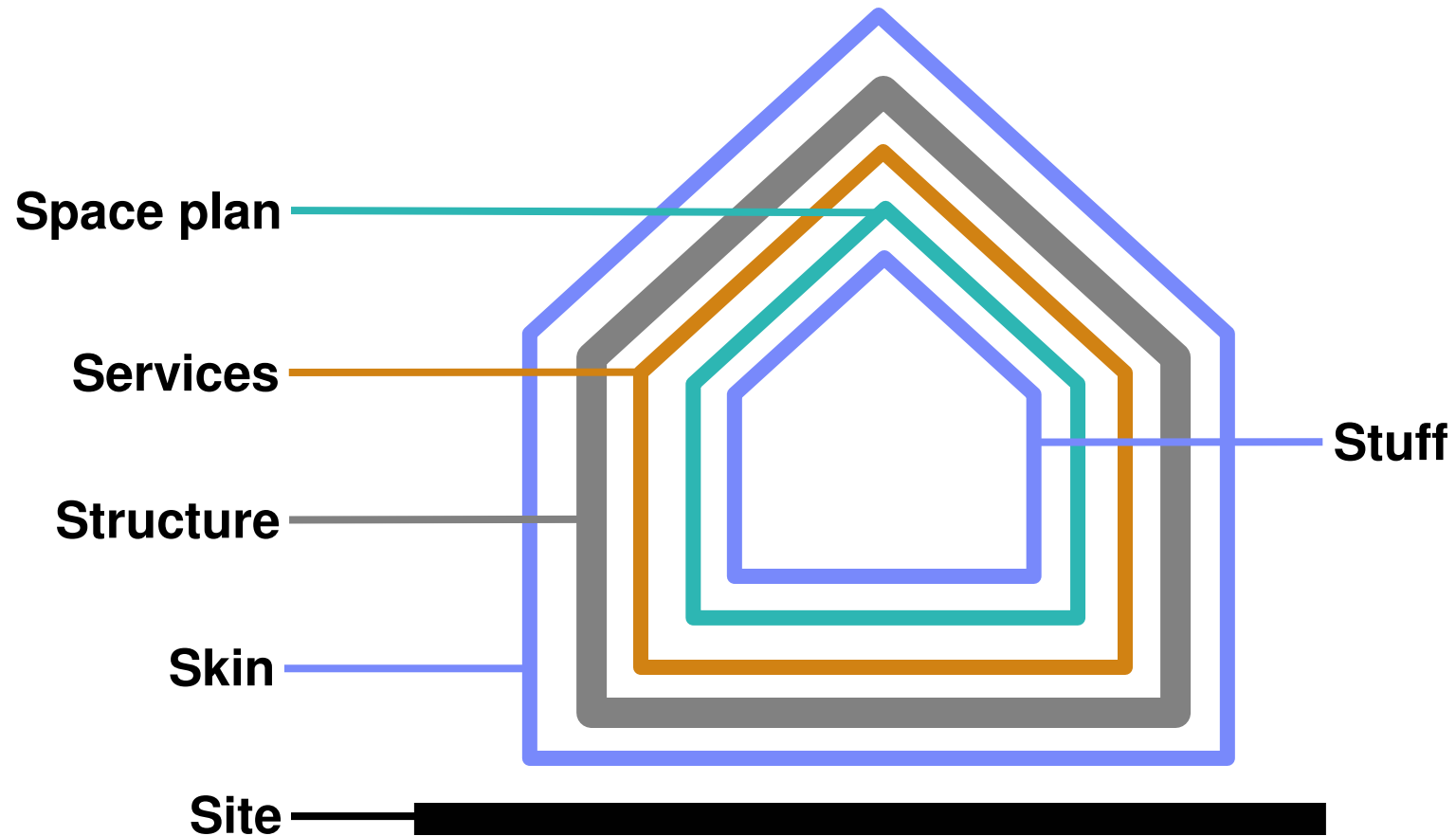
### Avoiding failure

- Safety factors
- Redundancy
- Equilibrium

**Any time you depart from established practice, make ten times the effort, ten times the investigation. Especially on a very large project.**

- LeMessurier

## Shearing Layers Of Change



## Patterns In Physical Systems

- Calloway and Cromley's *The Elements of Style*
- Alexander's *The Nature of Order*
- *The Phaidon Atlas of Contemporary World Architecture*
- Perry's *Chemical Engineers' Handbook*
- Sclater and Chironis' *Mechanism and Mechanical Devices Sourcebook*
- Christiansen's *Electrical Engineers' Handbook*
- *ICRF Handbook of Genome Analysis*



## Physical Systems

- **Mature physical systems have stable architectures**
  - Aircraft, cars, and ships
  - Bridges and buildings
- **Such architectures have grown over long periods of time**
  - Trial-and-error
  - Reuse and refinement of proven solutions
  - Quantitative evaluation with analytical methods
- **Mature domains are dominated by engineering efforts**
  - Analytical engineering methods
  - New materials and manufacturing processes

## Movements in Web-centric Architectures

- **Simple documents**
- **Colorful clients**
- **Simple scripting**
- **Rise of middleware**
- **Rise of simple frameworks**
- **Emergence of dynamic frameworks**
- **Semantic web**

# First Generation: Simple Documents

- Hyperlinks
- Simple formatting



**Welcome to Amazon.com Books!**

*One million titles,  
consistently low prices.*

(If you explore just one thing, make it our personal notification service. We think it's very cool!)

**SPOTLIGHT! -- AUGUST 16TH**

These are the books we love, offered at Amazon.com low prices. The spotlight moves **EVERY** day so please come often.

**ONE MILLION TITLES**

Search Amazon.com's [million title catalog](#) by author, subject, title, keyword, and more... Or take a look at the [books we recommend](#) in over 20 categories... Check out our [customer reviews](#) and the [award winners](#) from the Hugo and Nebula to the Pulitzer and Nobel... and [bestsellers](#) are 30% off the publishers list...

**EYES & EDITORS, A PERSONAL NOTIFICATION SERVICE**

Like to know when that book you want comes out in paperback or when your favorite author releases a new title? Eyes, our tireless, automated search agent, will send you mail. Meanwhile, our human editors are busy previewing galleys and reading advance reviews. They can let you know when especially wonderful works are published in particular genres or subject areas. Come in, [meet Eyes](#), and have it all explained.

**YOUR ACCOUNT**

Check the status of your orders or change the email address and password you have on file with us. Please note that you **do not** need an account to use the store. The first time you place an order, you will be given the opportunity to create an account.

## Second Generation: Colorful Clients

- More engaging content
- More precise formatting
- Rise of eye candy





## Third Generation: Simple Scripting

- **Rise of scripting languages**
  - Perl, PHP, JavaScript, ...
- **DHTML**
  - ASP/JSP

## Fourth Generation: Rise Of Middleware

- **.Net**
- **WebSphere**
- **JBoss**

## Fifth Generation: Rise Of Simple Frameworks

- **Struts**
  - Codification of MVC pattern

## Sixth Generation: Emergence Of Dynamic Frameworks

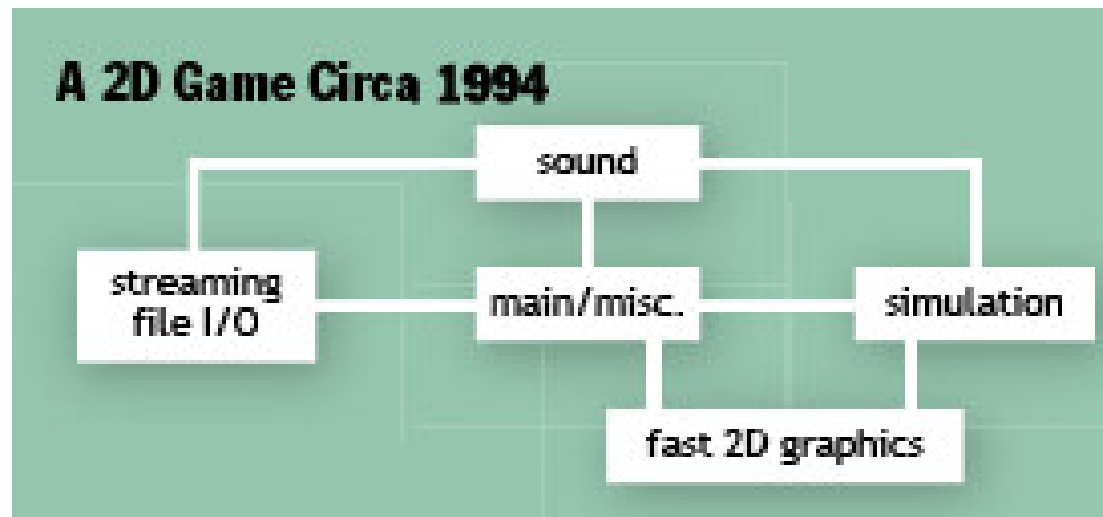
- **Ajax**
  - HTML + DOM + XMLHttpRequest
- **SOA**

## Next Generation: Semantic Web

- **XML + XML Schema + RDF + RDF Schema + OWL**

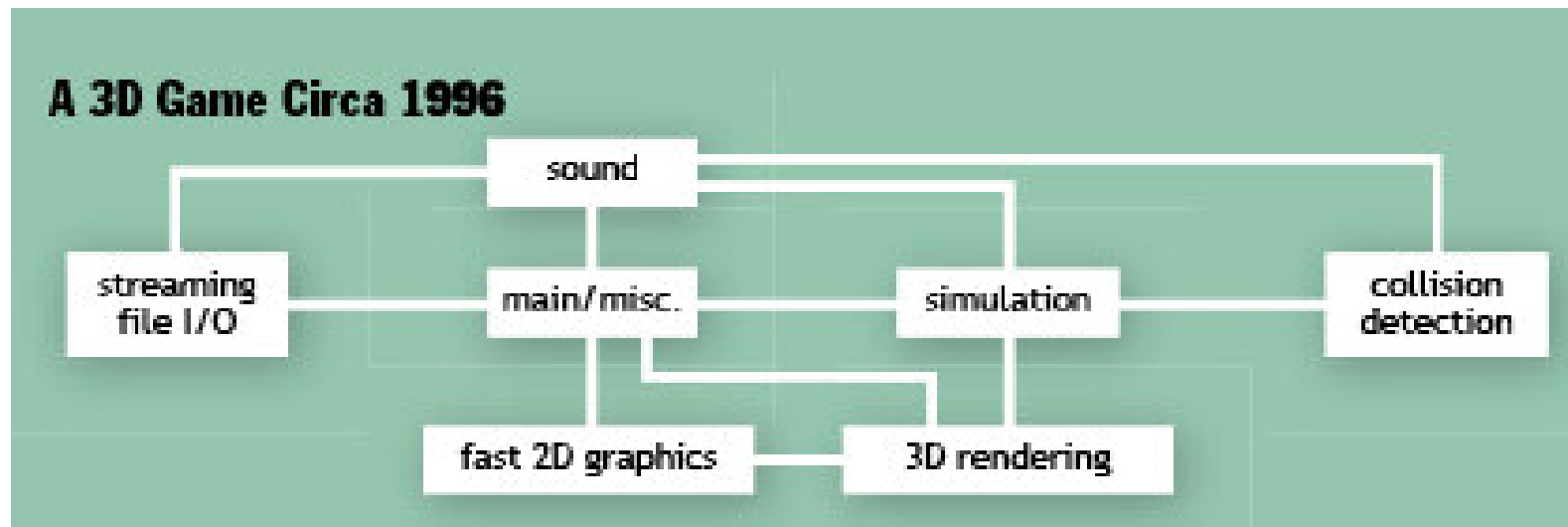


## Movements In Game Architectures



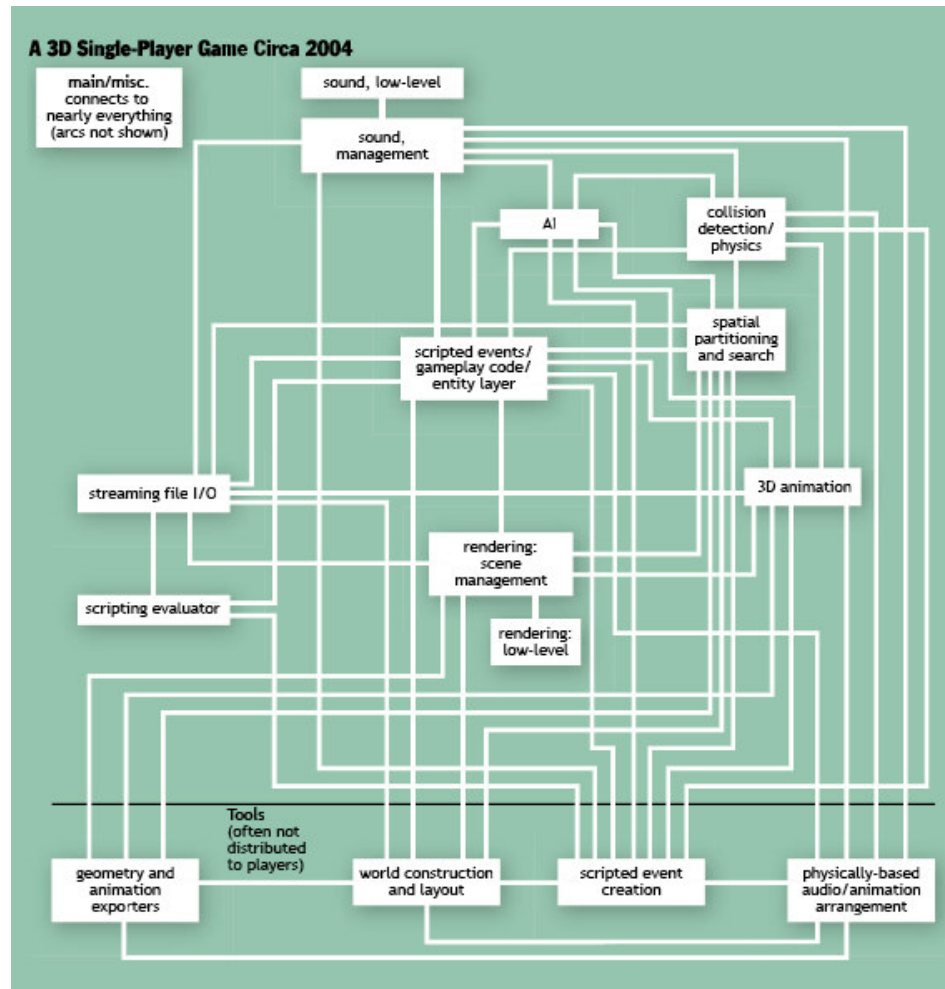
Blow, J. "Game Development: Harder Than You Think." *ACM Queue*

## Movements In Game Architectures



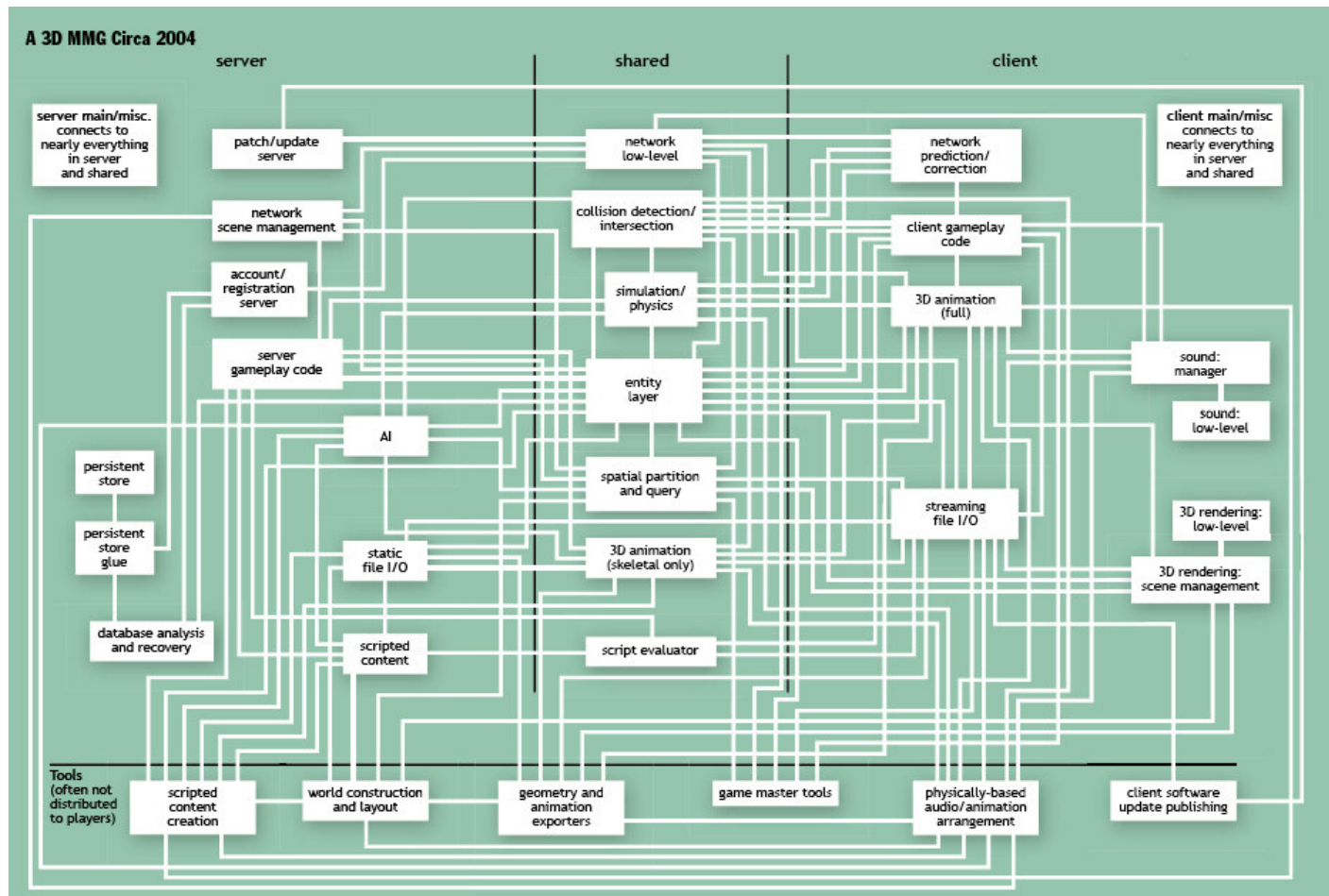
Blow, J. "Game Development: Harder Than You Think." *ACM Queue*

# Movements In Game Architecture



Blow, J. "Game Development: Harder Than You Think." *ACM Queue*

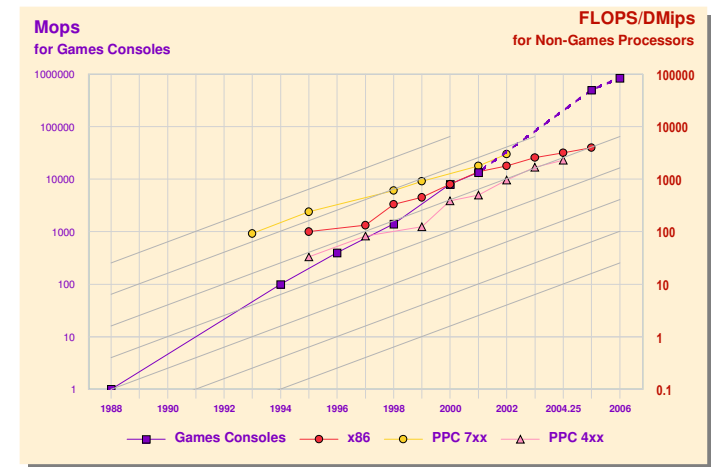
# Movements In Game Architecture



Blow, J. "Game Development: Harder Than You Think." *ACM Queue*

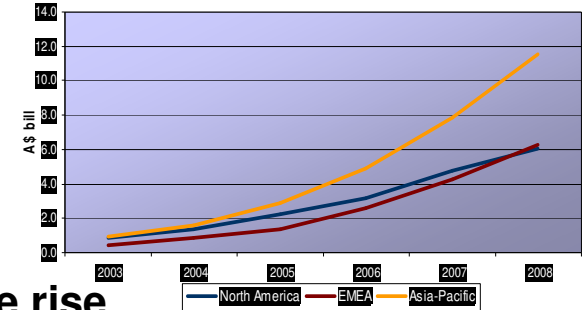
## Forces In Game Architecture

- **Currently at an inflection point driven by:**
  - Introduction of new consoles
  - Dramatic technology shifts
  - New genres of game
- **Development studios must:**
  - Learn new programming models
  - Purchase new development tooling
  - Move from a single threaded, linear, execution model to a multithreaded, parallel, execution model





## Forces In Game Architecture



- **Broadband penetration and game subscription on the rise**
  - Game companies are challenged with managing millions of users
- **Alternative revenue needed to offset development costs**
  - Advertising revenue integrated
  - Licensed content
- **Games appealing to broader demographic, becoming main stream**
  - Casual gamers the fastest growing segment
  - Women over 40 are another fast growing segment
- **Consolidation: publishers are buying studios**
- **Increase in collaboration between art and engineering**
- **Wireless and mobile games a growing segment**
  - Mobile games must be developed for various devices, often in Java
  - All game aspects are integrated into the wireless providers infrastructure
- **Telcos and cable companies want to get in the game**

## Game Genres (By Platform)

- Four types of Games –
  - PC: Windows based and games written in C/C++
  - PC Online: Windows based and games written in C/C++
  - Console: Xbox, PlayStation, and GameCube
  - Handheld/Mobile
- Each game market is its own distinct market segment.  
No segment reduces market share in the other as it increases.
- Gamers tend to play games on more than one platform



Online games



PC



Sony  
PlayStation3



Nintendo  
"Revolution"



Microsoft  
Xbox 360



## Genres (By Subject)

- **Action**
- **Strategy**
- **Role-playing**
- **Sports**
- **Vehicle**
- **Construction & management**
- **Adventure**
- **Artificial life**
- **Online**

## The Average Game (In 2000)

- **Cost \$5-10 million to develop**
- **Required 1-3 years development time**
- **Involved a team of 10-50 developers and artists**
- **Produced 500meg of data**

## Process

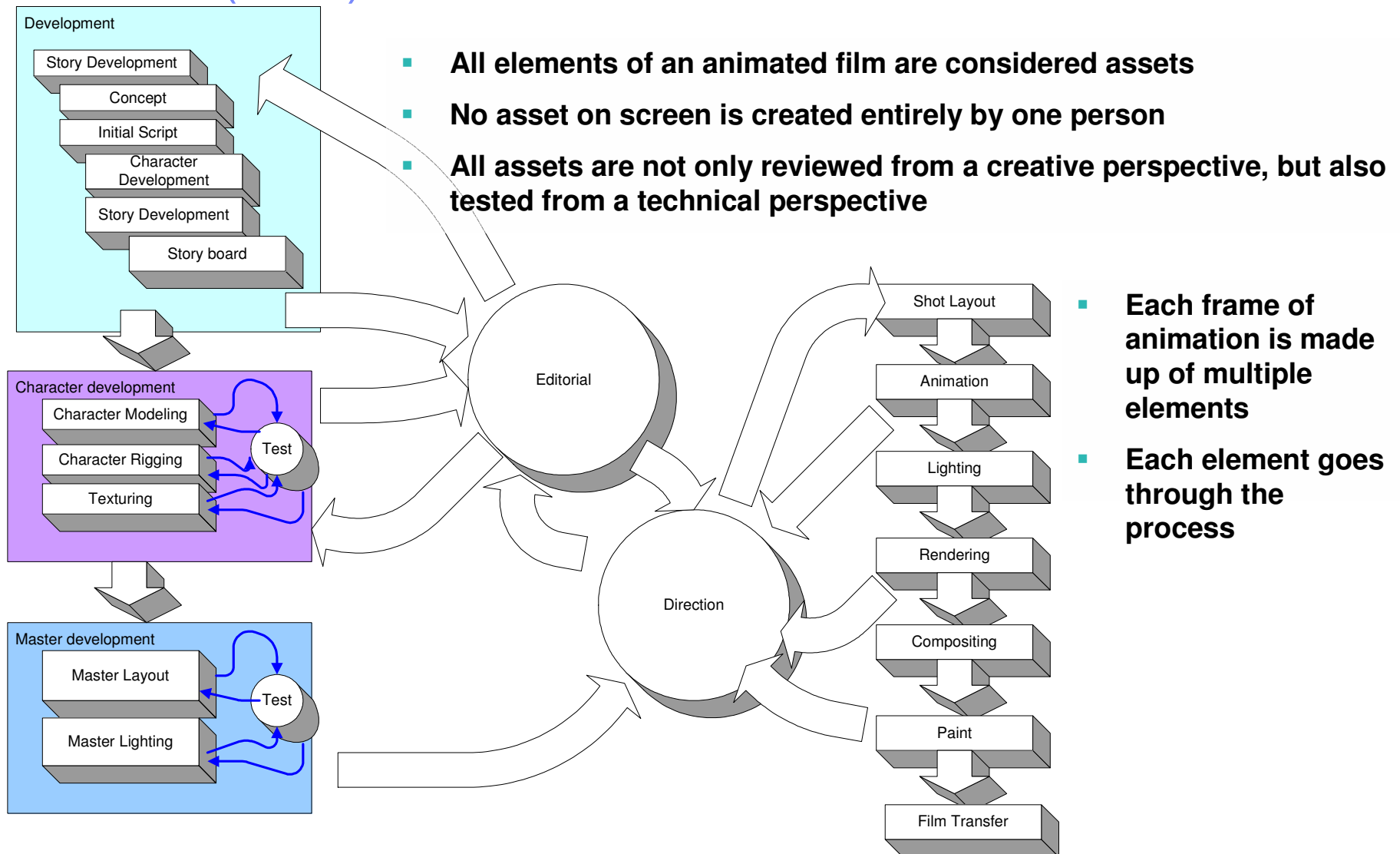
- **Insane development schedules**
  - You have to make your Christmas Ship date
  - Titles tend to be on a 12, 18, 24 or 36 dev cycle
  - Only the largest studios can afford to spend 36 months
- **And it's only getting worse...**
  - Increased complexity
  - Game transitioning from one CPU
  - Explosion of Art
    - Expensive to create, takes a lot of space, can not be automated
    - Tooling for collaborative art development is lagging tooling for SW development



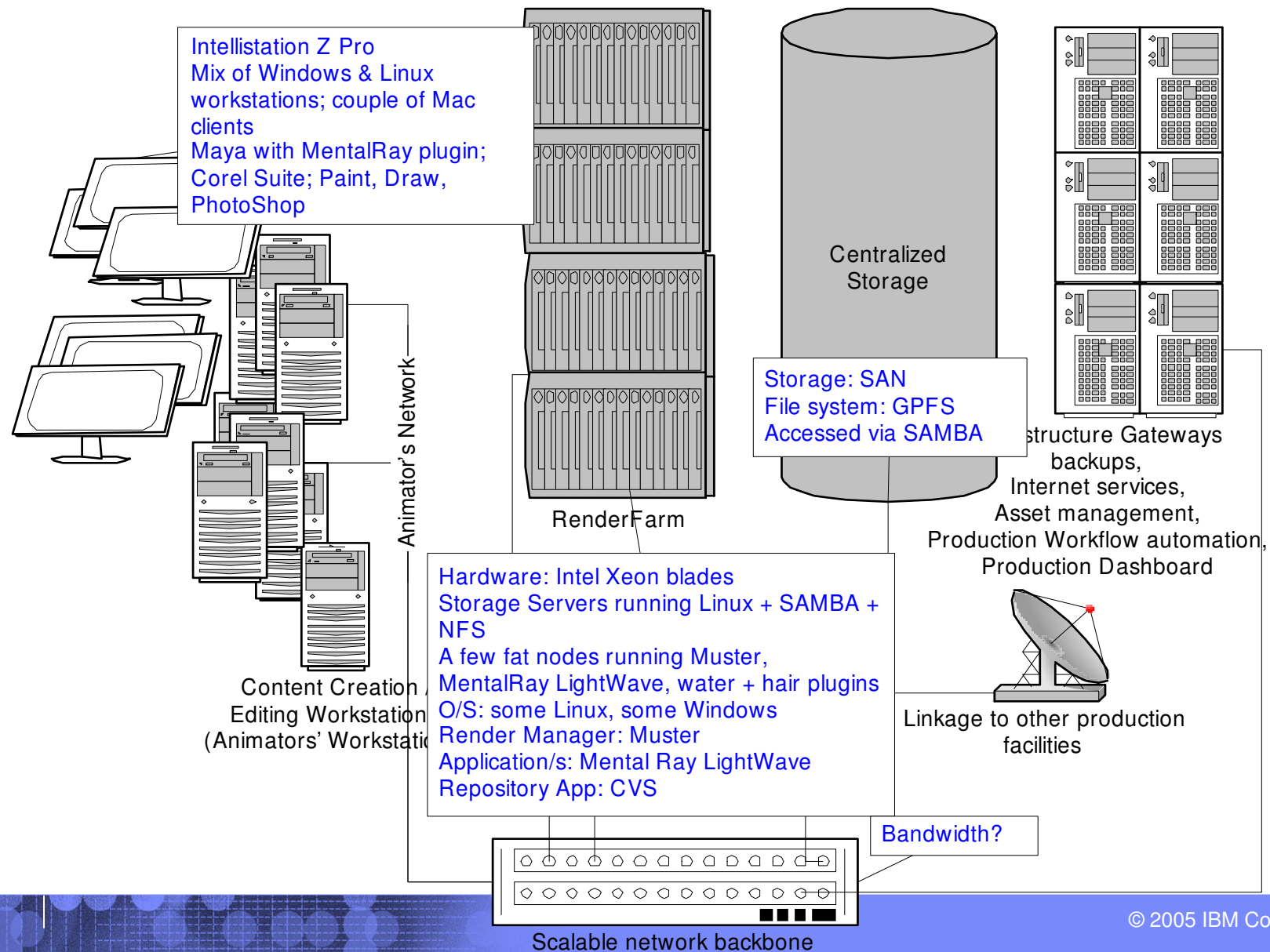
## What Went Wrong

- ***Unreal Tournament***: No central design document
- ***Tropico***: Lack of up-front design work
- ***Trespasser***: Game design problems
- ***Deus Ex***: Didn't front-load all of our risks
- ***Star Wars Starfighter***: Initial lack of detailed design
- ***Rainbow Six***: Lack of up-front design
- ***Soldier of Fortune***: Unfocused design
- ***Asheron's Call***: No feature iteration during development

# Process (Film)



# Development Platform (Film)



## Forces In Software



<http://www.booch.com/architecture/architecture.jsp?part=Limits-Forces>

## Fundamentals

- **Crisp abstractions**
  - **Clear separation of concerns**
  - **Balanced distribution of responsibilities**
  - **Simplicity**
- 
- **Grow a system through the iterative and incremental release of an executable architecture**



## Software Patterns

- Buschman, *Pattern-Oriented Software Architecture*
- Dyson, *Architecting Enterprise Solutions*
- Fowler, *Patterns of Enterprise Application Architecture*
- Gamma et al, *Design Patterns*
- Hohpe et al, *Enterprise Integration Patterns*
- Kircher, *Pattern-Oriented Software Architecture*
- Schmidt, *Pattern-Oriented Software Architecture*
- Shaw/Garland *Software Architecture*
- Towbridge et al, *Integration Patterns*

## Architecting Software Is Different

- **No equivalent laws of physics**
- **Transparency**
- **Complexity**
  - Combinatorial explosion of state space
  - Non-continuous behavior
  - Systemic issues
- **Requirement and technology churn**
- **Low replication and distribution costs**

The entire history of software engineering  
Is one of rising levels of abstraction

**Languages:** Assembly -> Fortran/COBOL -> Simula -> C++ -> Java  
**Platforms:** Naked HW -> BIOS -> OS -> Middleware -> Domain-specific  
**Processes:** Waterfall -> Spiral -> Iterative -> Agile  
**Architecture:** Procedural -> Object Oriented -> Service Oriented  
**Tools:** Early tools -> CLE -> IDE -> XDE -> CDE  
**Enablement:** Individual -> Workgroup -> Organization

## Why Architecture?

- **In hyperproductive projects**
  - Process centers around growing an executable architecture
  - Well-structured systems are full of patterns
- **Why architecture?**
  - Risk-confrontive
  - Simplicity
  - Resilience

# The Role Of The Architect

QuickTime™ and a  
DV/DVCPRO - NTSC decompressor  
are needed to see this picture.



## What We Know We Know

- **Every software-intensive system has an architecture**
- **We generally understand what software architecture is and what it is not**
- **Different stakeholders have different concerns and therefore different viewpoints**
- **All well-structured software-intensive systems are full of patterns**

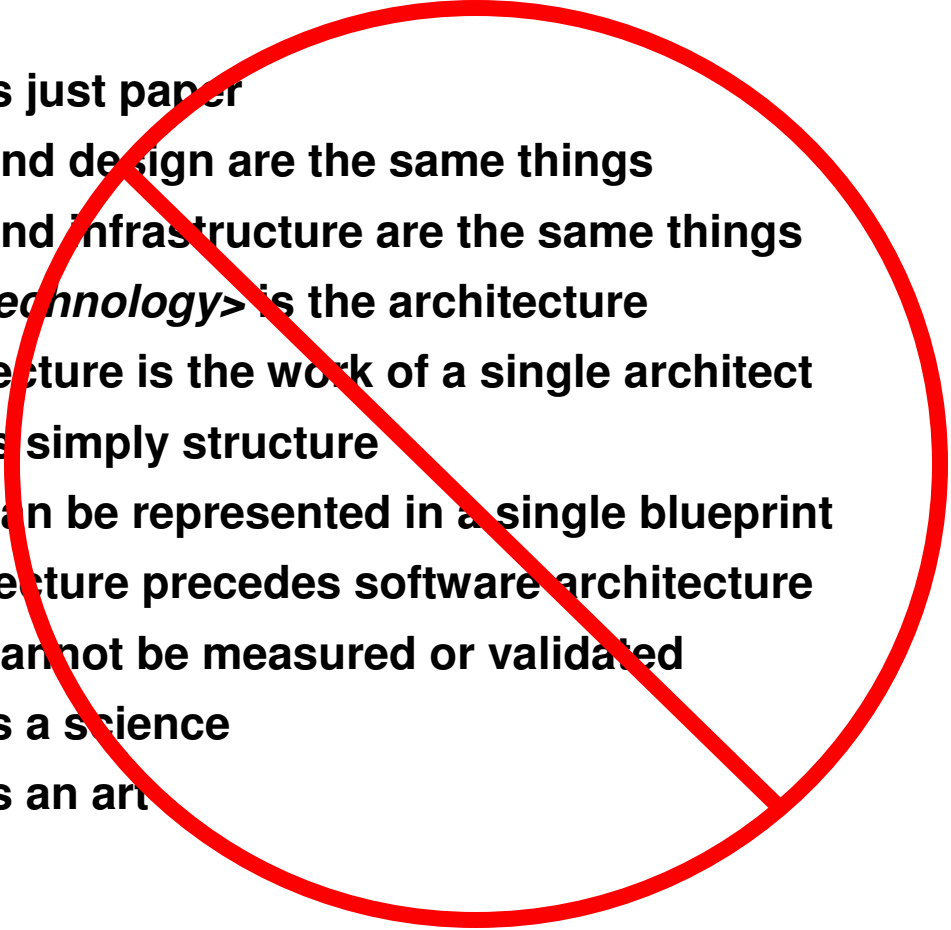
## What We Are Fairly Certain We Know

- **We are starting to develop a profession of software architecture**
- **We are starting to see the emergence of domain-specific software architectures**

## What We Know We Don't Know

- **We still don't have formal architectural representations that scale**
- **We don't yet have a good understanding of the architectural patterns that are found among domains.**

## Misconceptions About Architecture

- 
- **Architecture is just paper**
  - **Architecture and design are the same things**
  - **Architecture and infrastructure are the same things**
  - **<my favorite technology> is the architecture**
  - **A good architecture is the work of a single architect**
  - **Architecture is simply structure**
  - **Architecture can be represented in a single blueprint**
  - **System architecture precedes software architecture**
  - **Architecture cannot be measured or validated**
  - **Architecture is a science**
  - **Architecture is an art**

## Architecture Defined

- **Perry and Wolf, 1992**
- **Boehm et al., 1995**
- **Clements et al., 1997**
- **Common elements**
  - Architecture defines major components
  - Architecture defines component relationships (structures) and interactions
  - Behavior of components is a part of architecture insofar as it can be discerned from the point of view of another component
  - Every system has an architecture (even a system composed of one component)
  - Architecture defines the rationale behind the components and the structure
  - Architecture definitions do not define what a component is
  - Architecture is not a single structure -- no single structure is the architecture

<http://www.sei.edu/architecture/definitions.html>

# Architecture Defined

- **IEEE 1471-2000**

- Software architecture is the fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution

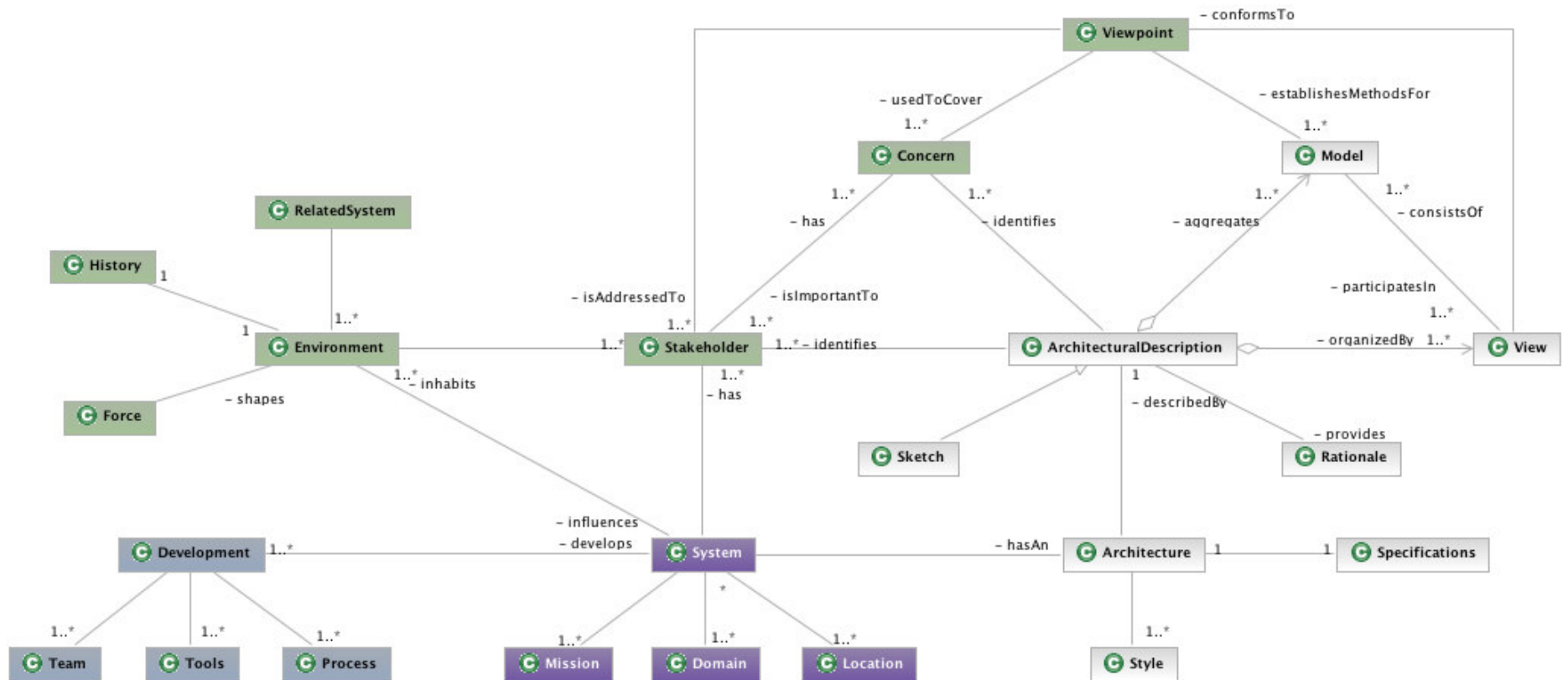
- **Software architecture encompasses the set of significant decisions about the organization of a software system**

- Selection of the structural elements and their interfaces by which a system is composed
- Behavior as specified in collaborations among those elements
- Composition of these structural and behavioral elements into larger subsystems
- Architectural style that guides this organization

Booch, Kruchten, Reitman, Bittner, and Shaw

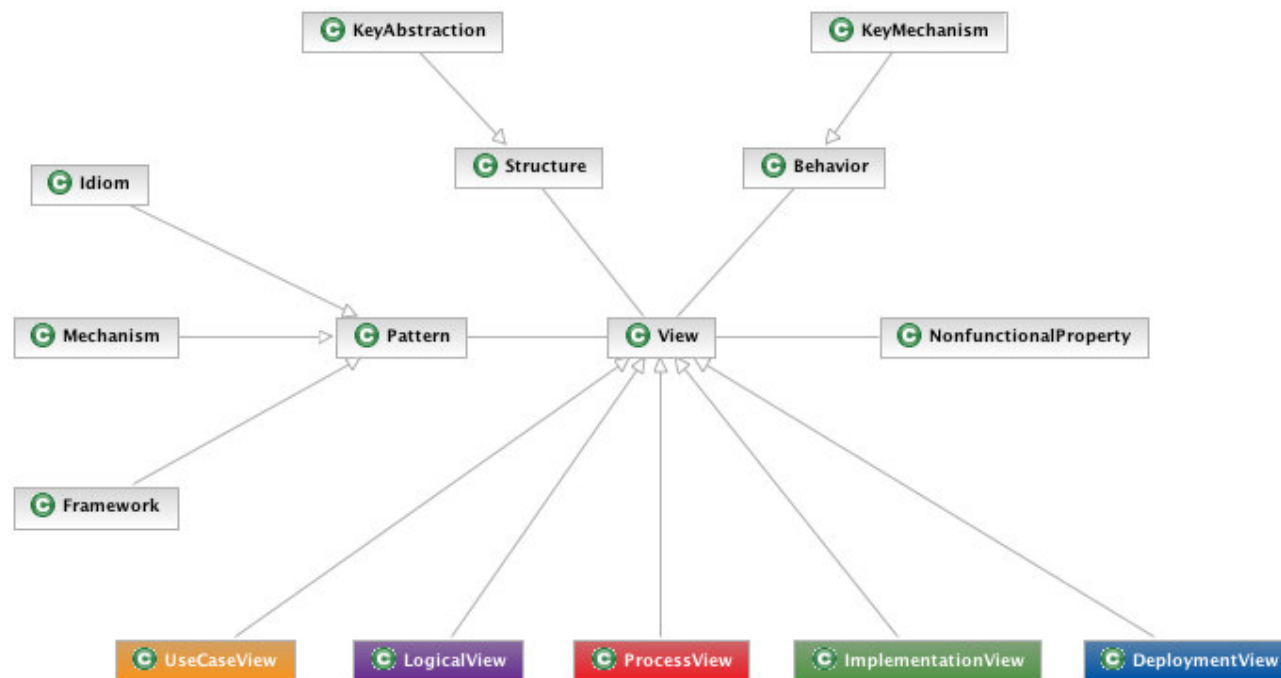


# Software Architecture Metamodel



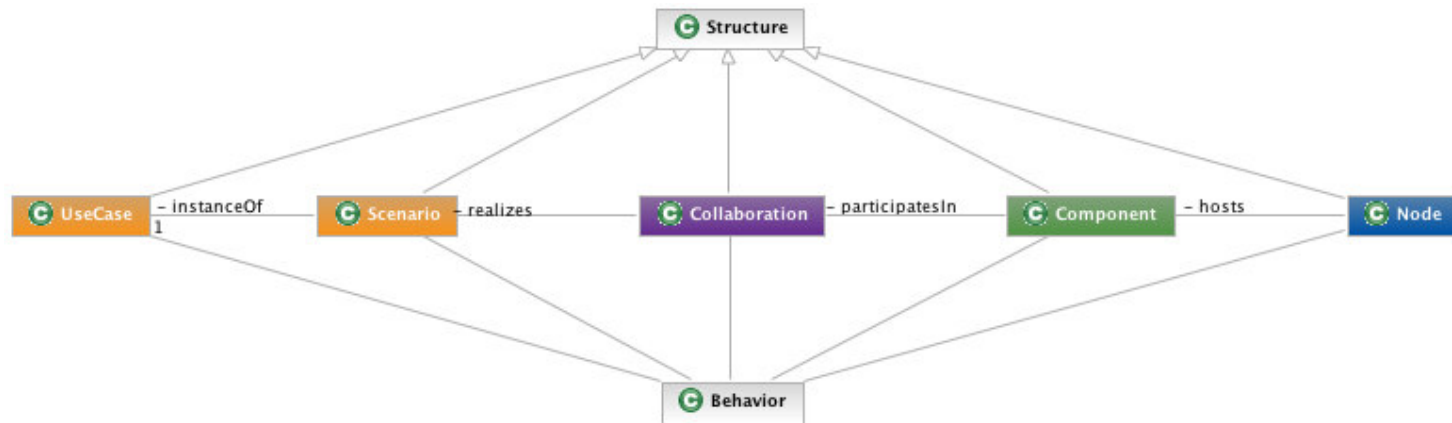
<http://www.booch.com/architecture/architecture.jsp?part=Metamodel>

# Software Architecture Metamodel



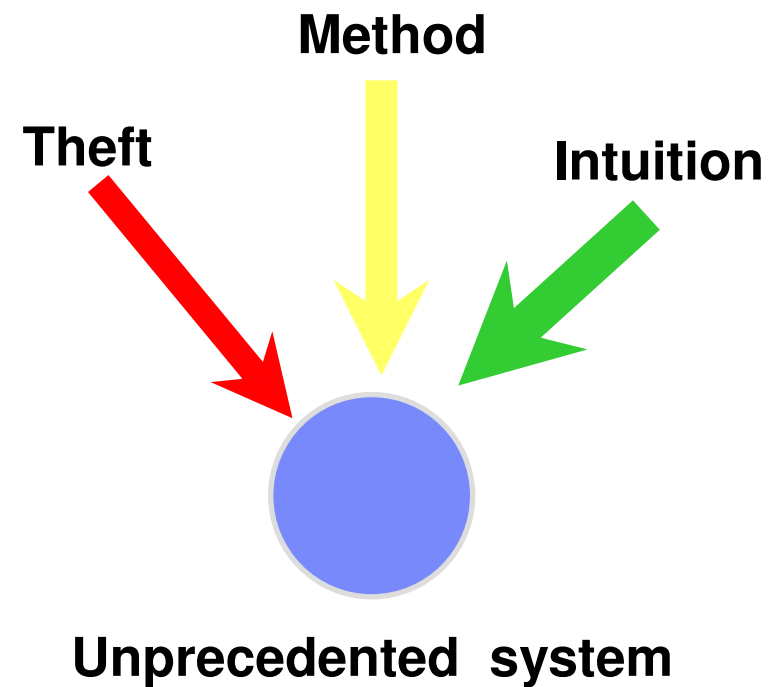
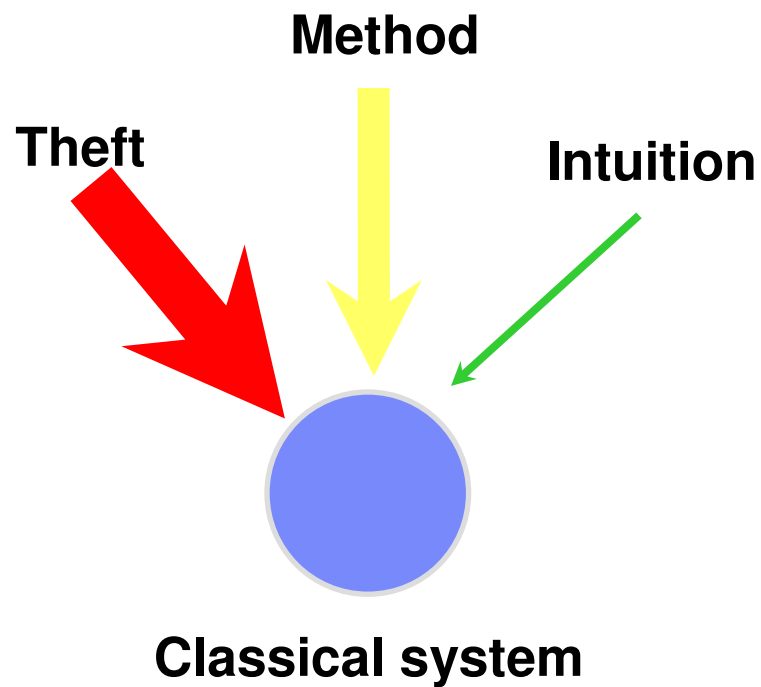
<http://www.booch.com/architecture/architecture.jsp?part=Metamodel>

# Software Architecture Metamodel



<http://www.booch.com/architecture/architecture.jsp?part=Metamodel>

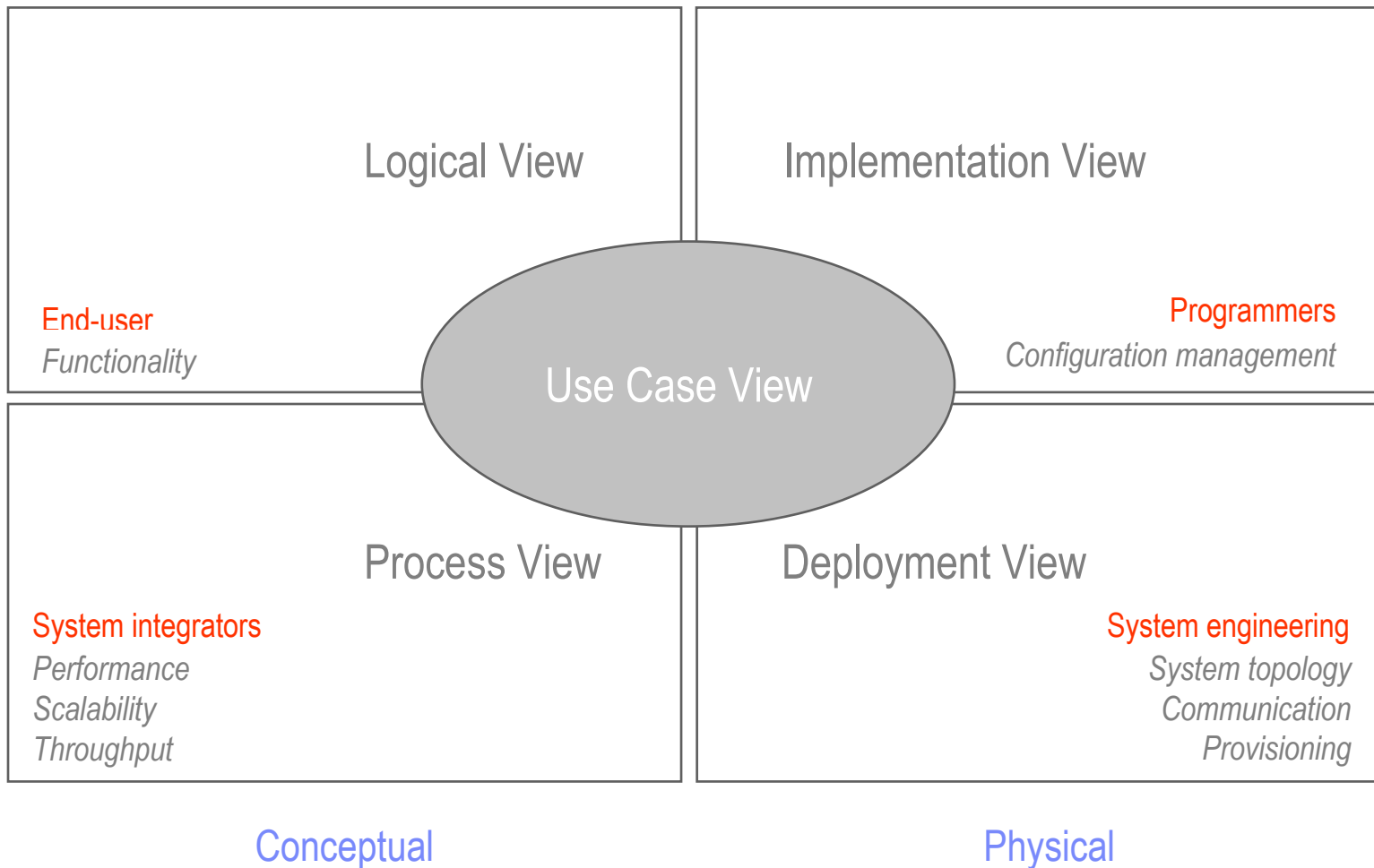
## Sources Of Architecture



## Architectural Style Defined

- **Style is the classification of a system's architecture according to those with similar patterns**
- **A pattern is a common solution to a common problem; patterns may be classified as idioms, mechanisms, or frameworks**

# Representing Software Architecture



Kruchten, "The 4+1 Model View"



## Cross-functional Mechanisms

- **Some structures and behaviors crosscut components**
  - Security
  - Concurrency
  - Caching
  - Persistence
- **Such elements usually appear as small code fragments sprinkled throughout a system**
- **Such elements are hard to localize using traditional approaches**

## ABIO Deployment View

### Proto-type: Mechanical Design

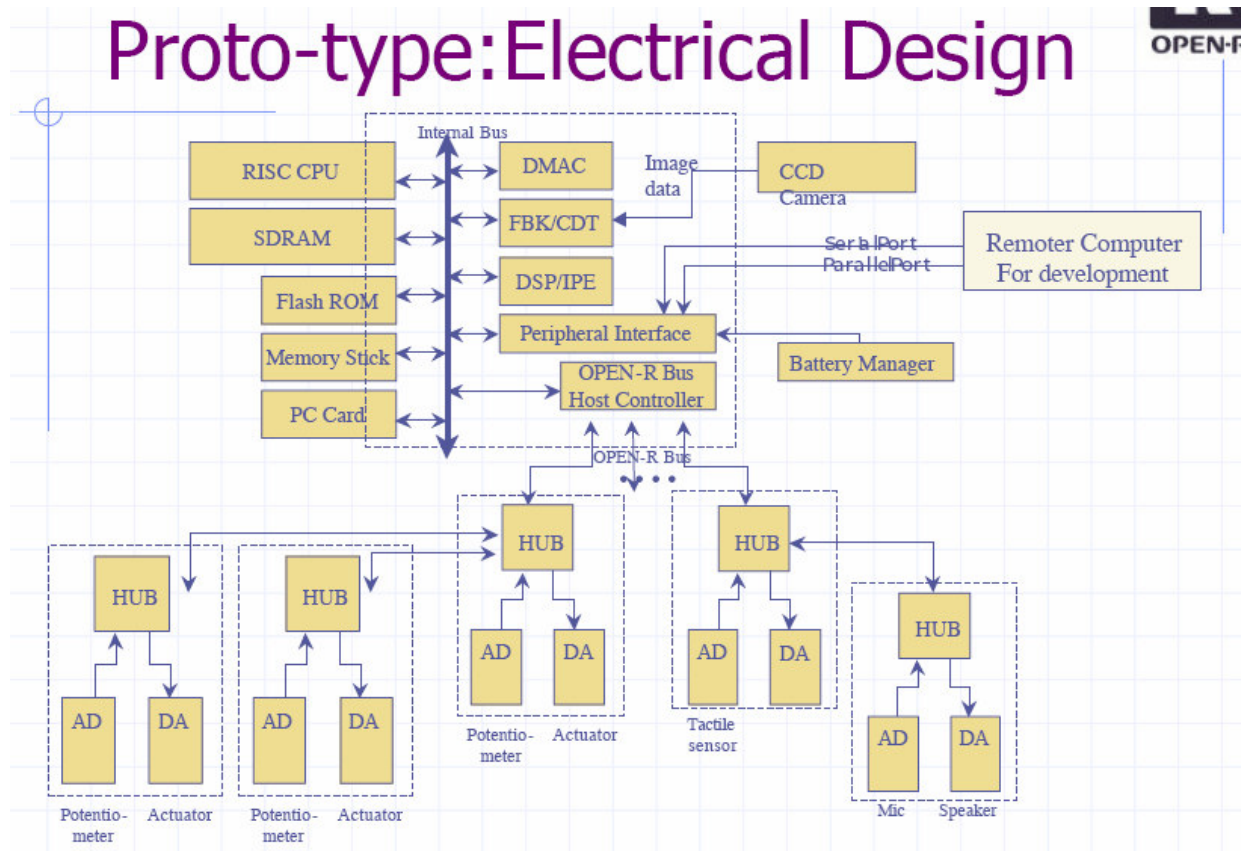
Reconfigurable Robot  
Creating robot (1)



Sabe, *OPEN-R*, Sony Intelligent Dynamics Laboratory

# ABIO Deployment View

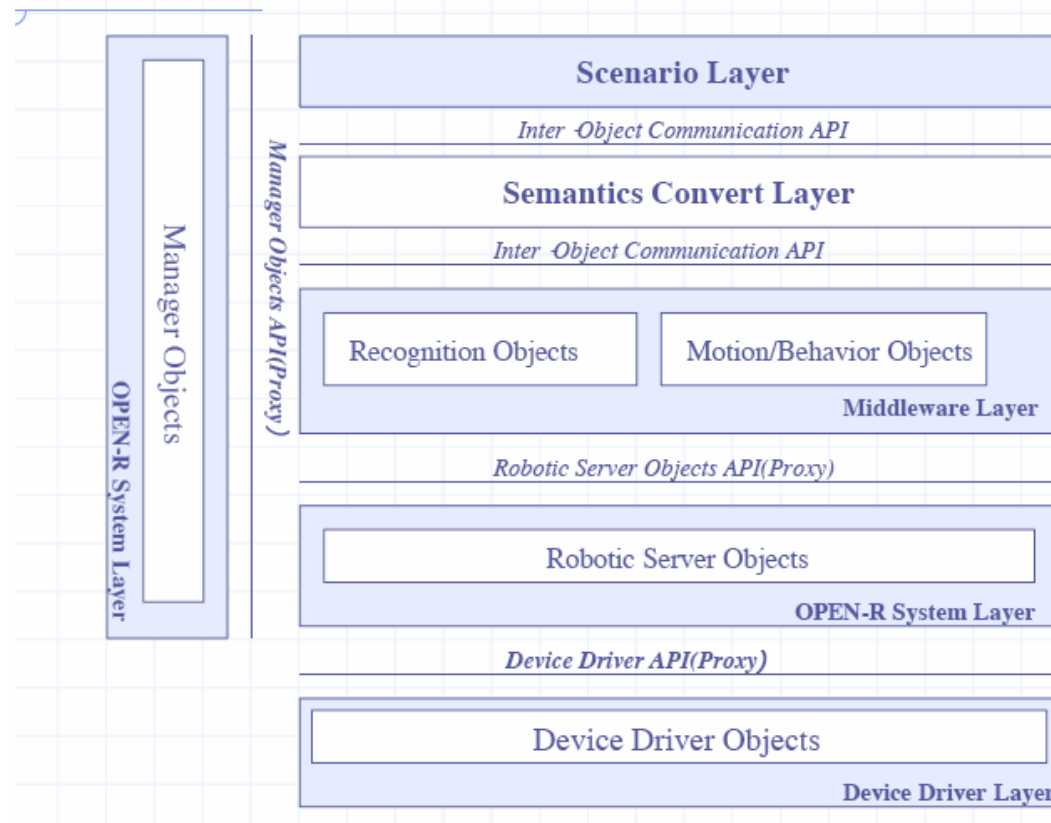
## Proto-type:Electrical Design



Sabe, *OPEN-R*, Sony Intelligent Dynamics Laboratory

## ABIO Logical View

### Proto-type: Software Overview

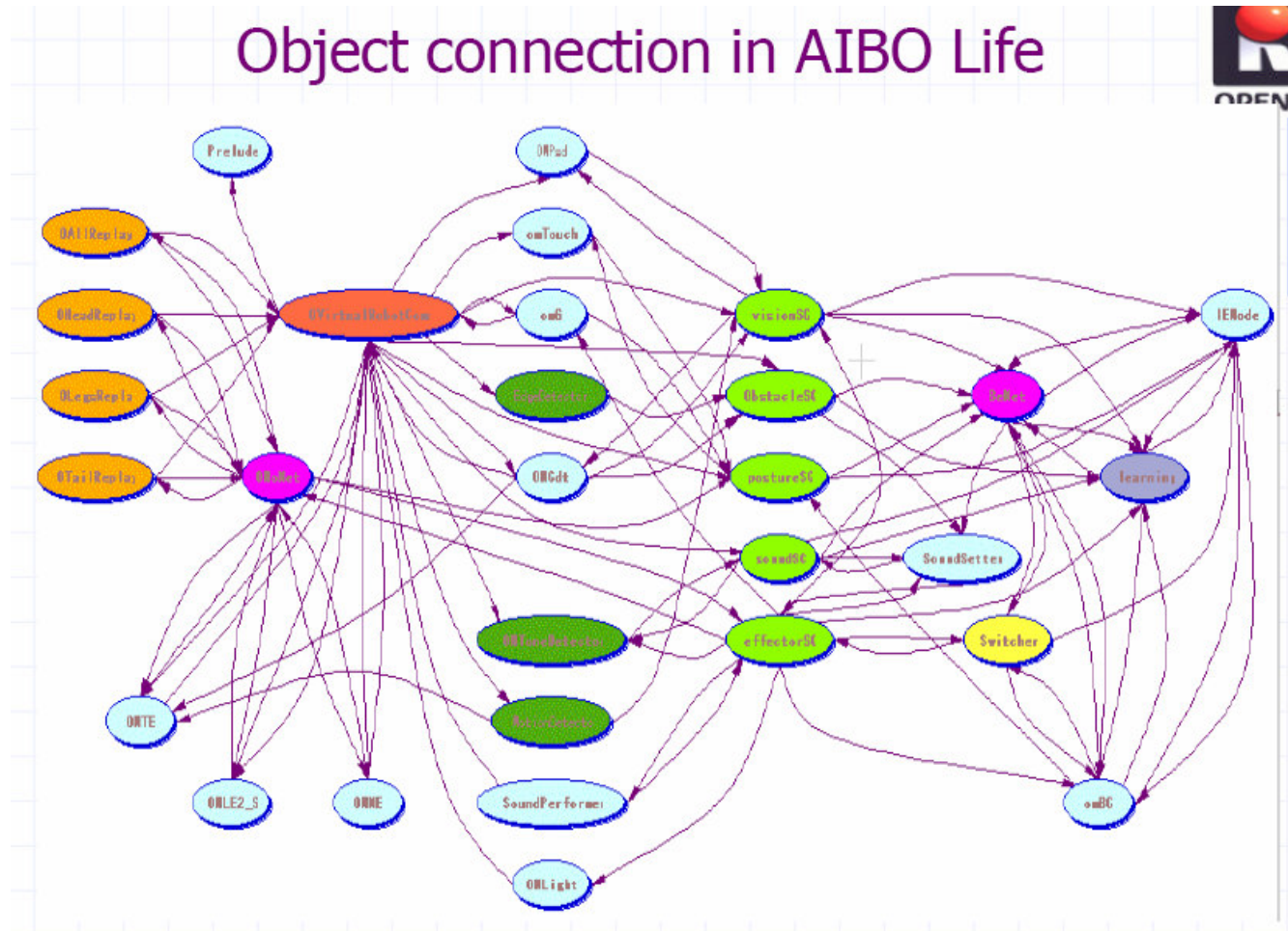


Sabe, *OPEN-R*, Sony Intelligent Dynamics Laboratory



# ABIO Logical View

## Object connection in AIBO Life



Sabe, *OPEN-R*, Sony Intelligent Dynamics Laboratory

# Eclipse

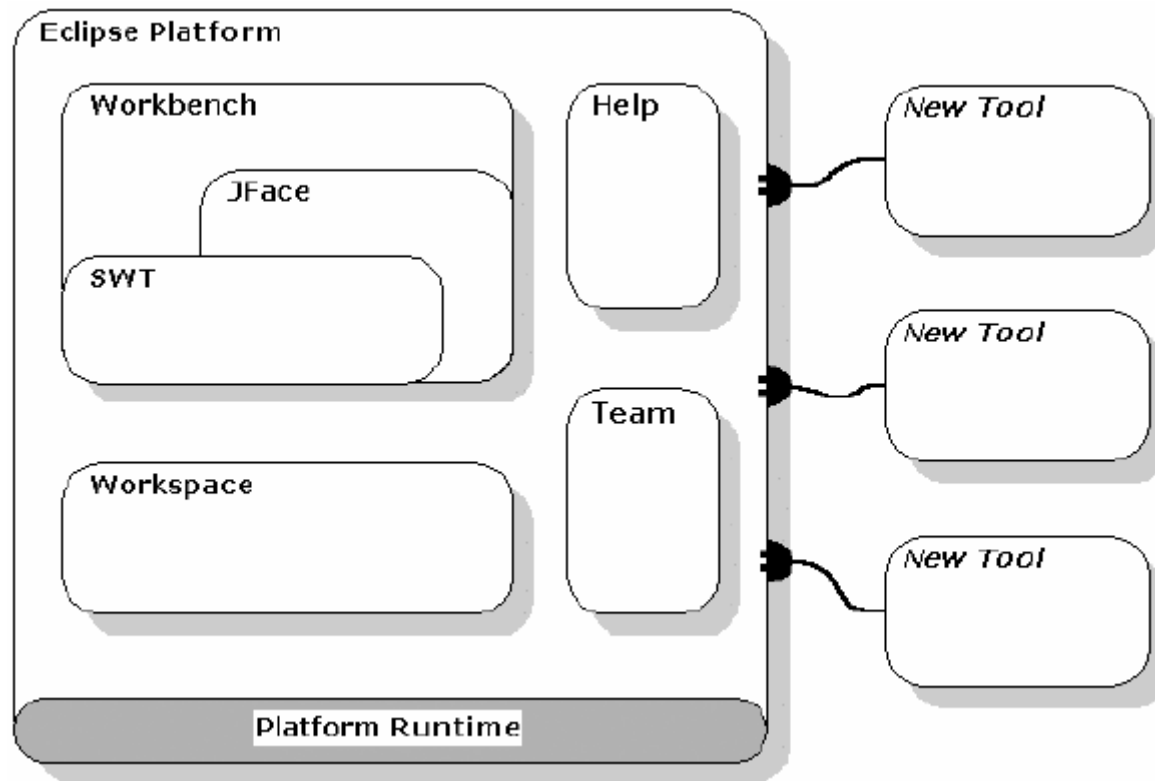
- **[www.eclipse.org](http://www.eclipse.org)**
- **Eclipse was started about 2 yrs go - when IBM made a \$40M contribution to the main code base – but is now an independent entity**
- **The principal architects were John Wiegand, Dave Thomson, John Duimovich all part of the OTI team which jump-started Eclipse.**



## Eclipse Artifacts

- ***Eclipse Platform Technical Overview***
- ***How To Use The Eclipse API***
- ***Eclipse Overview***
- **More detailed information exists for each of the subprojects.**

# Eclipse Architecture



## Eclipse Use Case View

- **Check In/Out Resource**
- **Close Perspective**
- **Close Window**
- **Display Help**
- **Invoke New Tool**
- **Open Perspective**
- **Open Window**
- **Refresh Workspace**
- **Shutdown Workbench**
- **Startup Workbench**

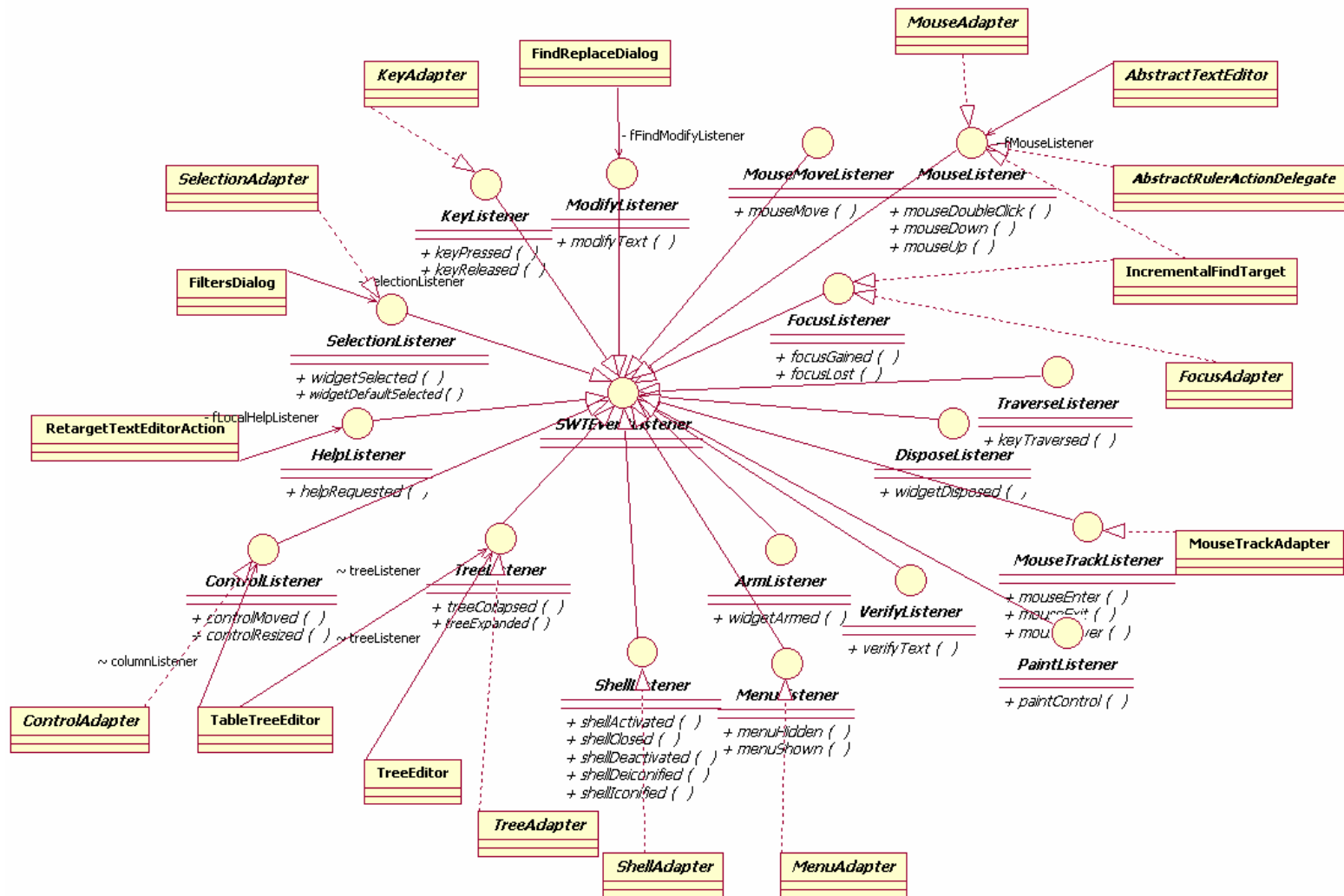
## Eclipse Implementation View

ant.jar anrunner.jar antsupport.jar antsupportlib.jar appserver.jar boot.jar bootstrap.jar catalina.jar commons-logging.jar compare.jar cvs.jar dtcore.jar dtui.jar editors.jar externaltools.jar forms.jar help.jar helpworkbench.jar helpworkbenchwin32.jar jakarta-regexp-1.2.jar jasper-compiler.jar jasper-runtime.jar jdi.jar jdimodel.jar jdui.jar jdt.jar jdtCompilerAdapter.jar	jdtcore.jar jface.jar jfacetext.jar jsp.jar junit.jar junit.support.jar launching.jar launchingsupport.jar lucene-1.2.jar naming-common.jar naming-factory.jar naming-resources.jar optional.jar parser.jar pde.jar pde-ant.jar pdebuild.jar pdebuild-ant.jar pdecore.jar pdert.jar pdeui.jar pdeuiant.jar resources.jar resources-ant.jar runtime.jar search.jar servlet.jar	servlets.jar servlets-common.jar servlets-default.jar servlets-invoker.jar servlets-manager.jar servlets-webdav.jar slimlauncher.jar snippetsupport.jar startup.jar swt.jar team.jar teamcvssh.jar teamcvsui.jar tomcat-coyote.jar tomcat-http11.jar tomcat-util.jar tomcatwrapper.jar ui.jar updatecore.jar update-servlets.jar updateui.jar update32.jar versioncheck.jar views.jar webapp.jar workbench.jar workbenchwin32.jar
--	---	---

## Eclipse Logical View

- **Plugin Development Environment (PDE)**
- **Workbench**
- **Team**
- **Debug**
- **Ant**
- **Help**
- **Java Development Tools (JDT)**

## SWT





## Lessons Learned

- **A lot of design information lives in tribal memory**
- **There were very high level architectural views and very low level design views, but little in between**
- **Reconstructing the deployment and implementation views are easy**
- **Reconstructing the use case view is possible**
- **Reconstructing the logical and process views are hard**
- **Harvesting patterns is harder still**

## Towards An Architecture Handbook

- **Workshops conducted by Bruce Anderson**
  - OOPSLA '91 & '92
  - Beck, Casseiman, D'Souza, Gamma, Gilbert, Gossain, Helm, Lea, Lee, Menga, Paulisch, Smooty, Vlissides, Johnson, Griss
- **Classification**
  - Configurations: patterns of things or relationships between things
  - Blocks: things which provide functionality
  - Mechanisms: ways of doing things

# Handbook Of Software Architecture

- **No architectural reference exists for software-intensive systems**
- **Goals of the handbook**
  - Codify the architecture of a large collection of interesting software-intensive systems
  - Study these architectural patterns in the context of the engineering forces that shaped them
  - Satisfy my curiosity

handbook of software architecture

<http://www.booch.com/architecture>

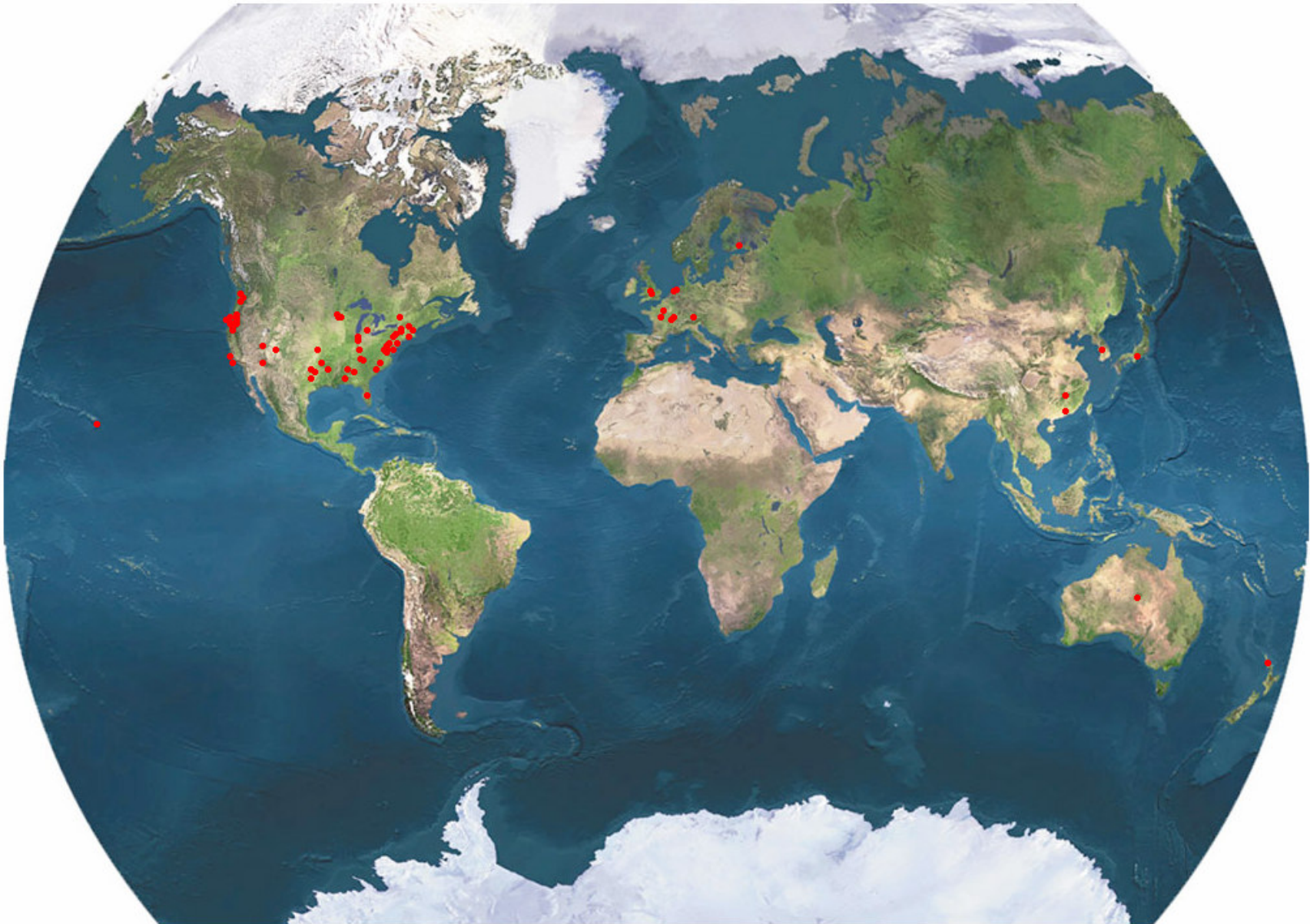
## Genres

- **Artificial intelligence**
- **Commercial & Non-Profit**
- **Communications**
- **Content Authoring**
- **Devices**
- **Entertainment & Sports**
- **Financial**
- **Games**
- **Government**
- **Industrial**
- **Legal**
- **Medical**
- **Military**
- **Operating Systems**
- **Platforms**
- **Scientific**
- **Tools**
- **Transportation**
- **Utilities**

## Representative Systems

- **Canadian Air Traffic Control**
- **Photoshop**
- **Olympics Games Management**
- **Pathfinder**
- **Eclipse**
- **DNA Analyzer**
- **QRIO**
- **eBay**
- **911 Response**
- **AEGIS**
- **Linux**
- **Jason**
- **London Underground**
- **Firefox**
- **DixieNarco DN5000 vender**
- **Massive**
- **WebSphere**
- **Google**

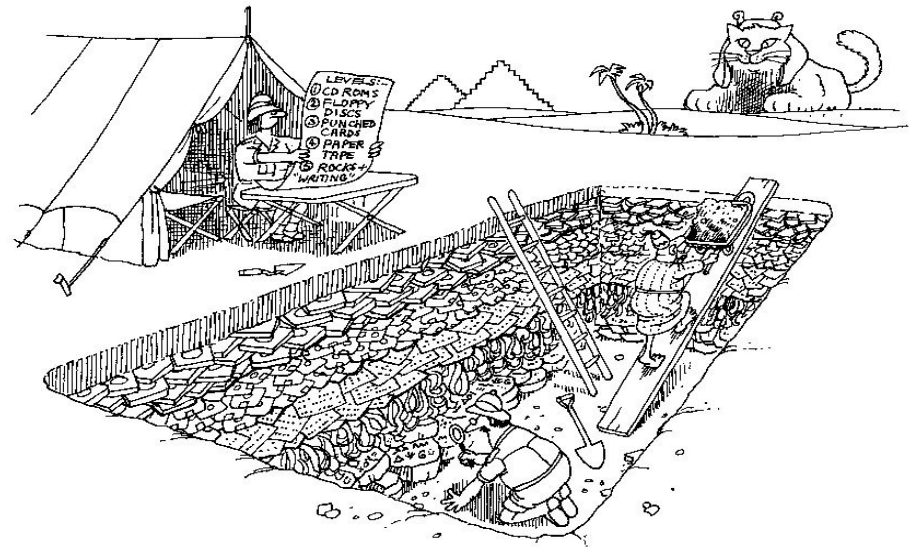






## Software Archeology

- **The recovery of essential details about an existing system sufficient to reason about, fix, adapt, modify, harvest, and use that system itself or its parts.**



## The Process of Archeology

- **Study of the source code**
- **Reverse engineering**
- **Probing and other instrumentation**
- **Review of existing documents**
- **Interviews with tribal leaders**

## Barriers To Software Archeology

- **Concerns over leakage of IP**
- **Accessibility of the development team**
- **Language**

## Preservation Of Classic Software

- **No comprehensive and intentional activity has yet been undertaken to preserve the industry's seminal software artifacts**
- **There are a number of reasons to act now**
  - Many of the authors of such systems are still alive
  - Many others may have the source code or design documents for these systems collecting dust in their offices or garages
  - Time is our enemy



<http://www.computerhistory.org>

## Efficiency: Improving Software Economics

$$\text{Time or Cost To Build} = (\text{Complexity}) (\text{Process}) * (\text{Team}) * (\text{Tools})$$

- Complexity** → Volume of human-generated code
- Process** → Methods, notations, maturity
- Team** → Skill set, experience, motivation
- Tools** → Process automation

## Two dimensions

### Waterfall

Few risk, sequential  
Late integration and testing

### Relaxed

Little documentation  
Light process  
Low ceremony

### Disciplined

Well-documented  
Traceability  
CCB  
High ceremony

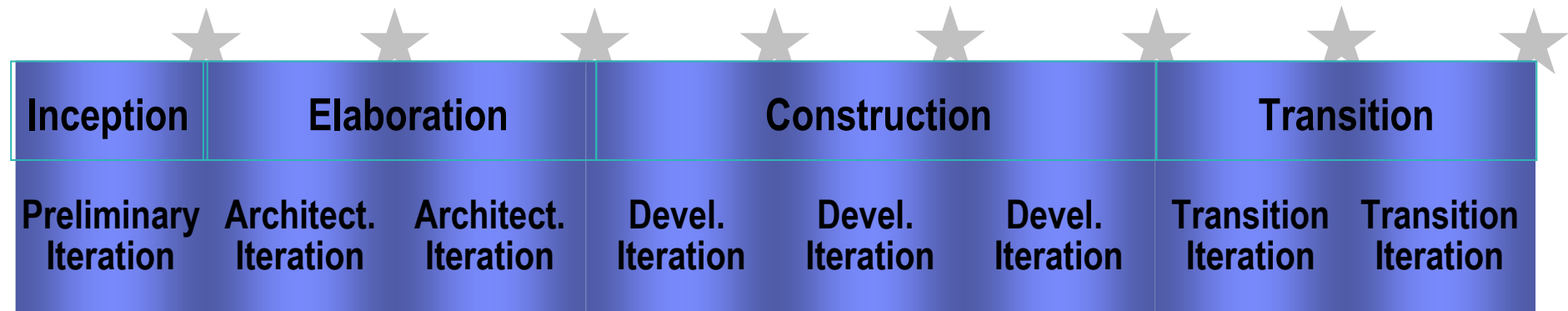
### Iterative

Risk driven  
Continuous integration and testing



# Iterations and Phases

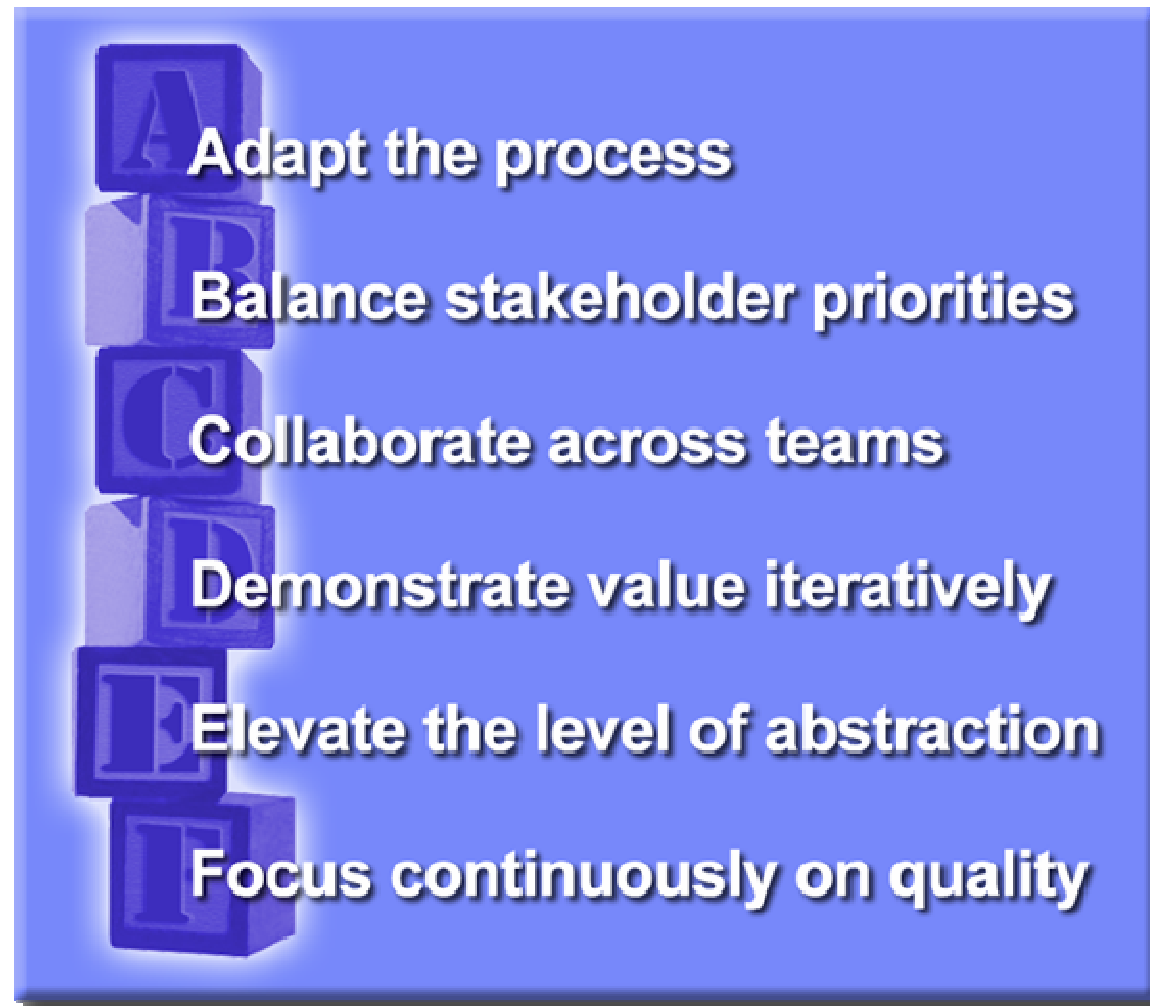
## Executable Releases



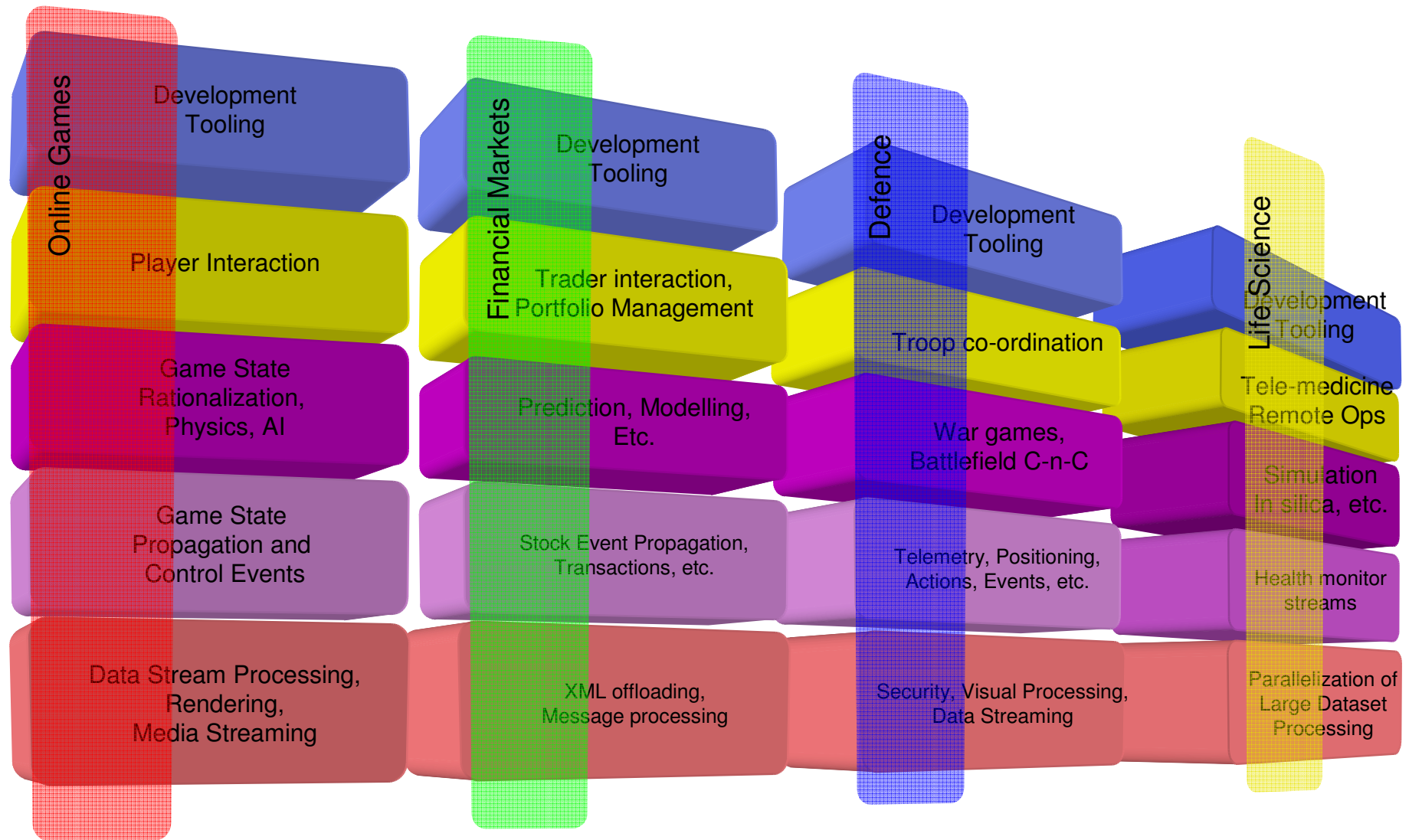
---

An iteration is a distinct sequence of activities with an established plan and evaluation criteria, resulting in an executable release.

## Business-Driven Development: Key Principles



# Common Technology Stacks



## Summary

- **Every interesting software-intensive system has an architecture**
- **The most successful software architectures are intentional, manifest, and visible - and beautiful**

# Thank you!