

Architectural Governance

Grady Booch
IBM Fellow

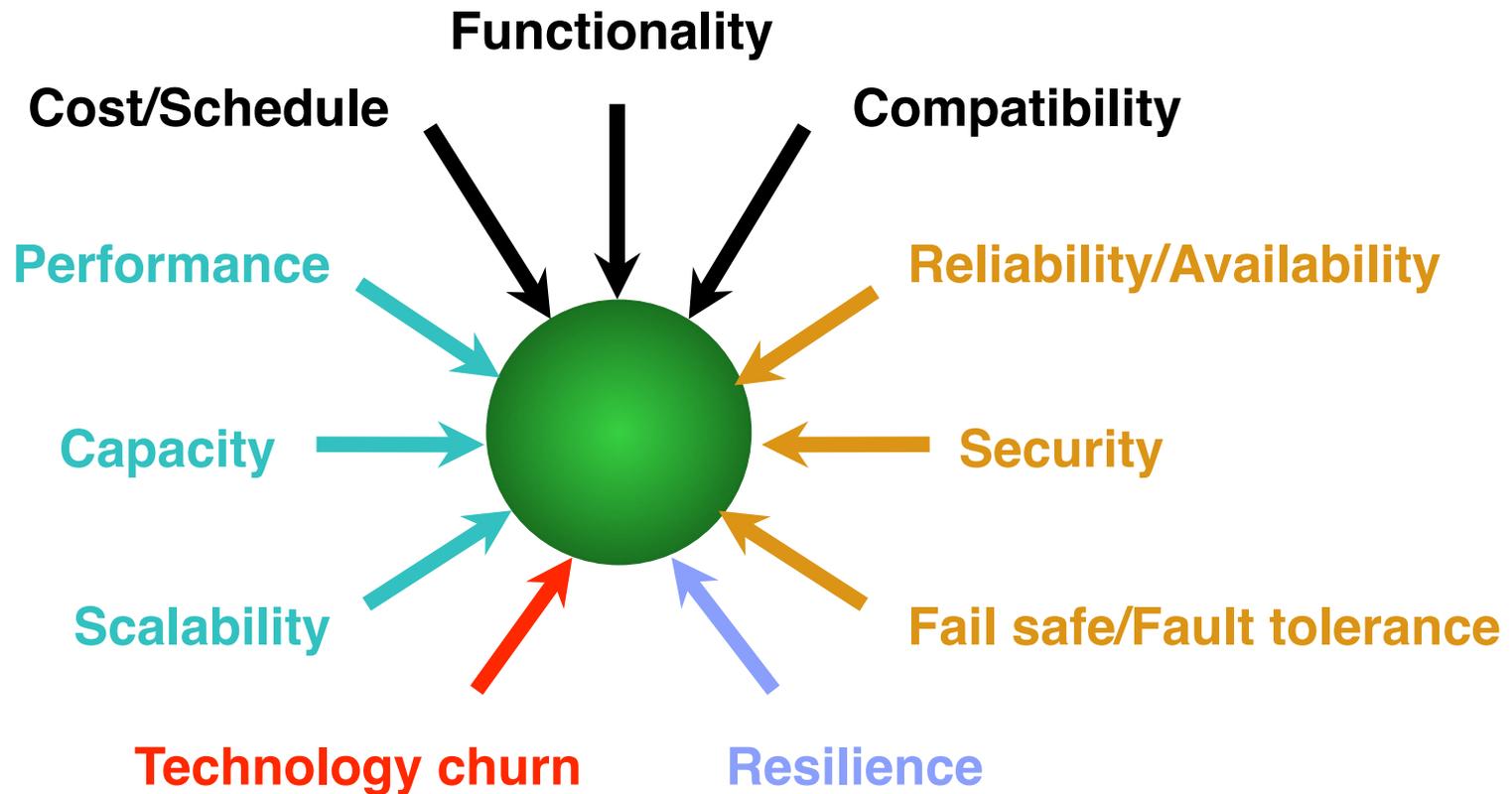


Outline

- **The context for architecture**
- **Architecture defined**
- **Representing architecture**
- **Architectural governance**
- **Organizational best practices**

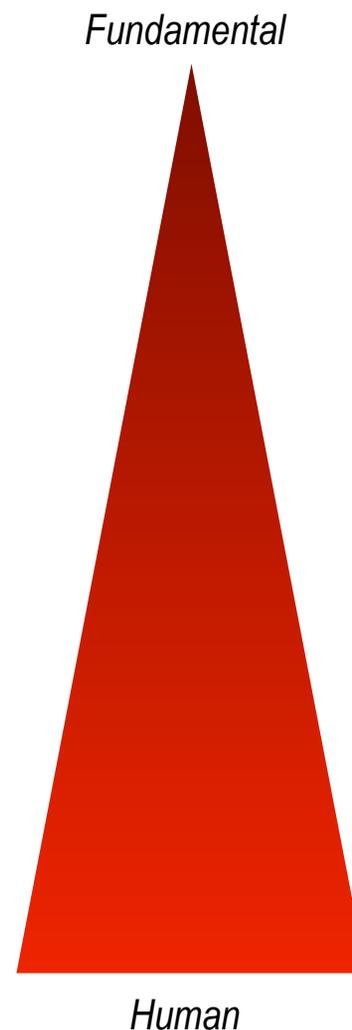
The context for architecture

Forces in software



The limits of technology

- **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**



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
 - New manufacturing processes

Software-intensive systems

- **A system in which software is the dominant, essential, and indispensable element**
 - E-commerce system
 - IT (business) system
 - Telephone switch
 - Flight control system
 - Real-time control system (e.g. industrial robot)
 - Sophisticated weapons system
 - Software development tools
 - System software (e.g. operating systems or compilers)

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**

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

Architecture defined (very informally)

- **Software architecture is what software architects do**

Software architecture (more formally)

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

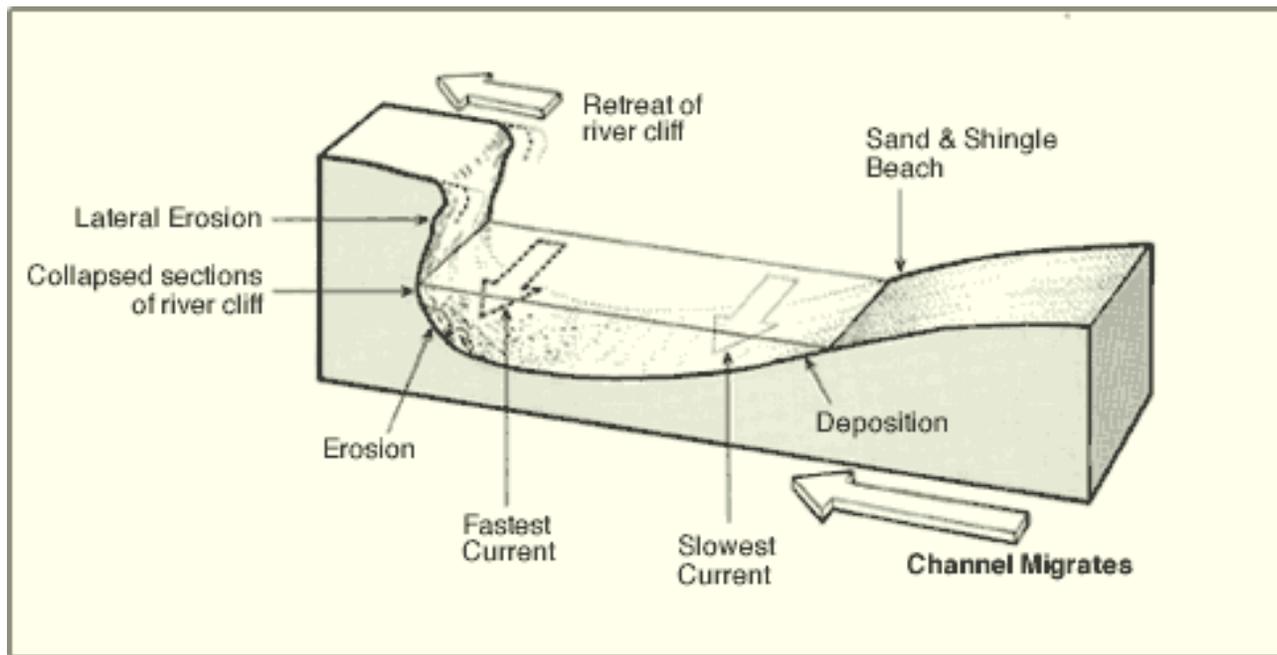
Architecture defined (operational)

- **All architecture is design; not all design is architecture. A system's architecture is defined by its significant design decisions (where "significant" is measured by the cost of change).**
- **Every software-intensive system has an architecture, forged from the hundreds of thousands of small decisions made every day.**
- **The code is the truth, but not the whole truth: most architectural information is preserved in tribal memory.**

An Classic Analogy



A Fresh Analogy (A Snapshot In Time)



A Fresh Analogy (A Broader Expanse)



Therefore

- **The architecture of an enterprise's software intensive systems is akin to the instantaneous structure and behavior of a river**
- **The lifecycle of that architecture is akin to the intentional and accidental morphing of those instantaneous architectures over a region of time.**

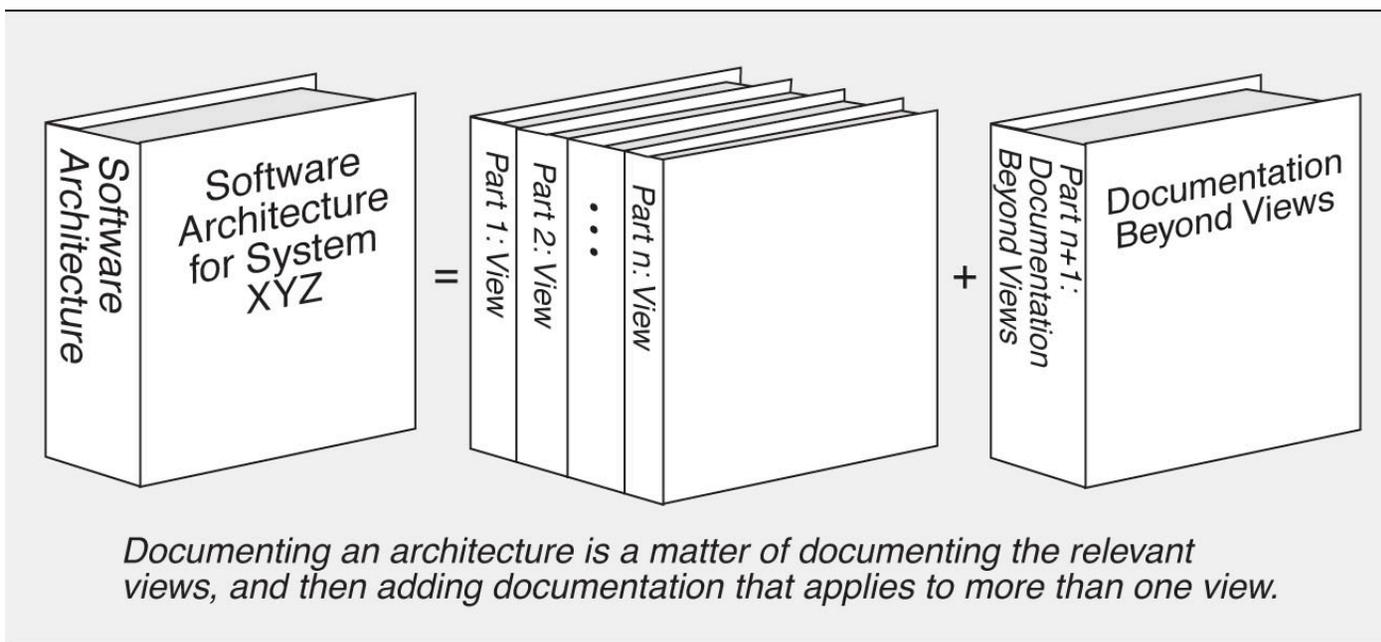
Representing architecture

Views and documentation

- **Architecture documentation is an essential governance artifact**
- **An architecture is a multidimensional construct, too involved to be seen all at once**
- **Systems are composed of many structures**
 - Design-time concerns – showing modules, their composition/ decomposition and mapping to code units; how teams cooperate to build the system; ...
 - Run-time concerns - processes and how they synchronize; programs and how they call or send data to each other; how components and connectors work; ...
 - Software in its environment - how software is deployed on hardware; network connections and ports;...
- **A view is a representation of a structure. We use views to manage complexity by separating concerns**

View-based documentation

Views give us our basic principle of architecture documentation



Adapting views

- **Not all systems require all views**
 - Single process (ignore process view)
 - Small program (ignore implementation view)
 - Single processor (ignore deployment view)
- **Some systems require additional views**
 - Data view
 - Security view
 - Other aspects

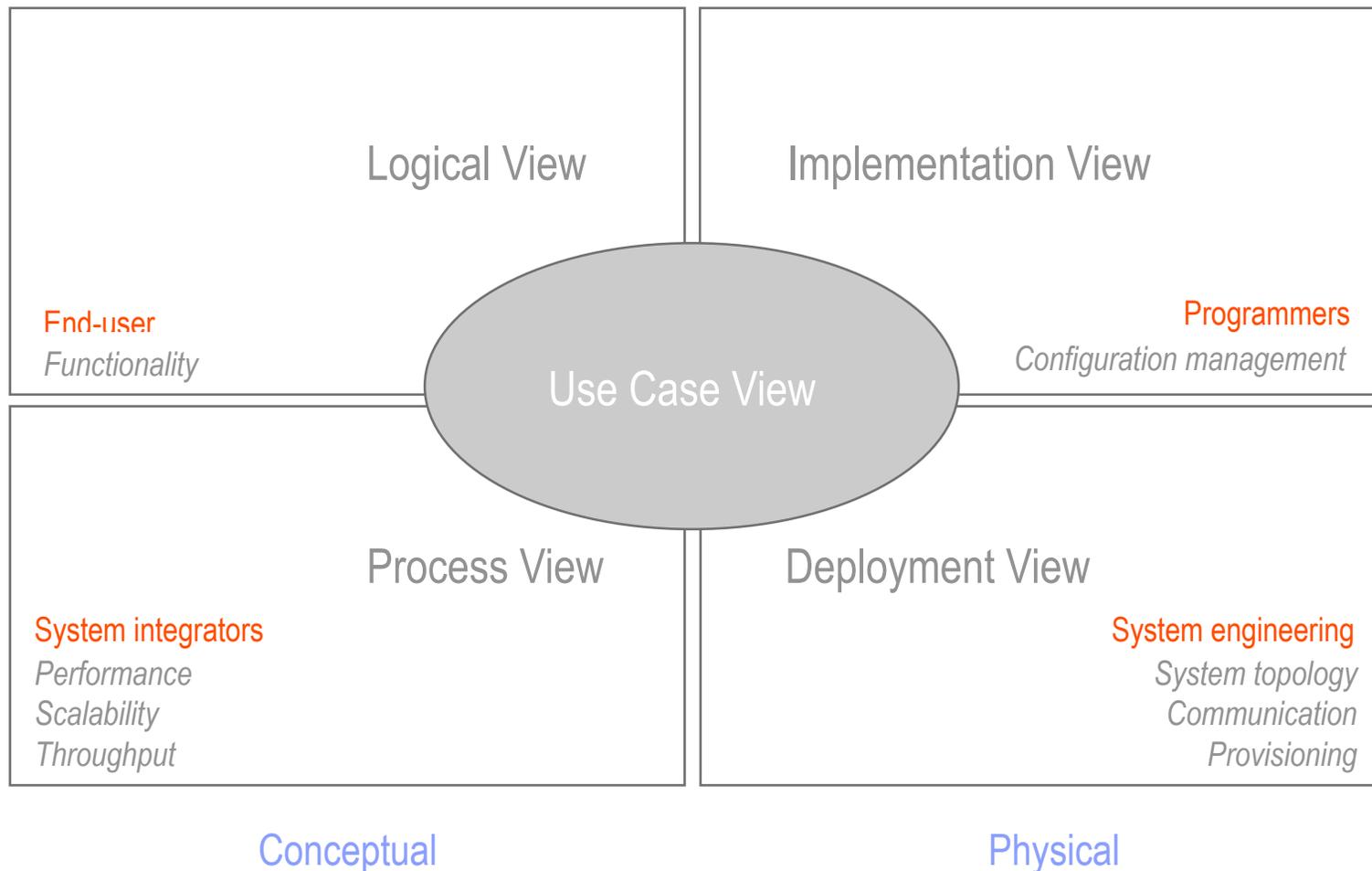
Cross functional mechanisms

- **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**

Alternate model view frameworks

- **ToGAF**
- **DoDAF**
- **MoDAF**
- **NAF**
- **NAE**
- **Zachman**

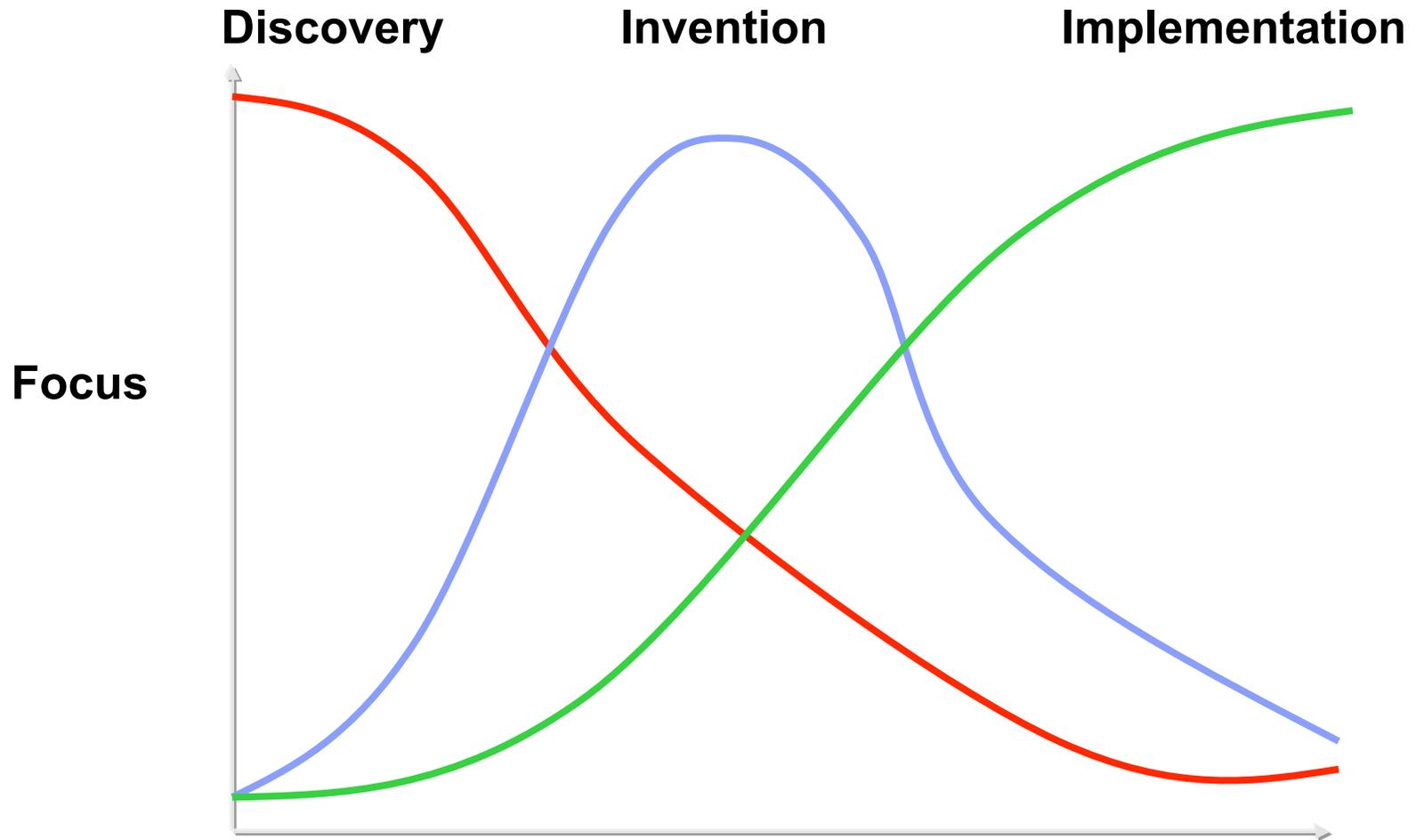
Representing software architecture



Kruchten, 4+1 Model View

Architectural governance

Focus over time



The Enterprise Architecture Lifecycle

- **Phases**

- Inception
- Elaboration
- Construction
- Transition

- **But also**

- Evolution
- Operation

The Enterprise Architecture Lifecycle

- **In my experience**
 - All hyperproductive organizations tend to have a lifecycle that involves the growth of a system's architecture through the incremental and iterative release of testable executables.

- **Not one lifecycle, but many**
 - Different stages of execution, maturation, and quality
 - Harmony, resonance, and cacaphony

Process best practices

- **Grow the architecture of a system through the incremental and iterative release of testable executables**
 - Focus on executables
 - Incremental and iterative progress
 - Architecture as artifact

Process best practices

- **Attack major risks early and continuously or else they will attack you**
- **Ensure that you deliver value to your customer**
- **Have a maniacal focus on working software**
- **Accommodate change early in the project**
- **Baseline an executable architecture early on**
- **Build your system with components**
- **Work closely together as one team**
- **Make quality a way of life, not an afterthought**

Architecture as an artifact of governance

- **Development takes place at two levels: architecture and implementation**
 - Both are ongoing, and they interact with each other strongly. New implementations suggest architectural changes. Architectural changes usually require radical changes to the implementation.

The soul of an architecture is found
in its mechanisms that cut across the components
of the system, thus yielding its essential
structures and behaviors

Architecture as artifact

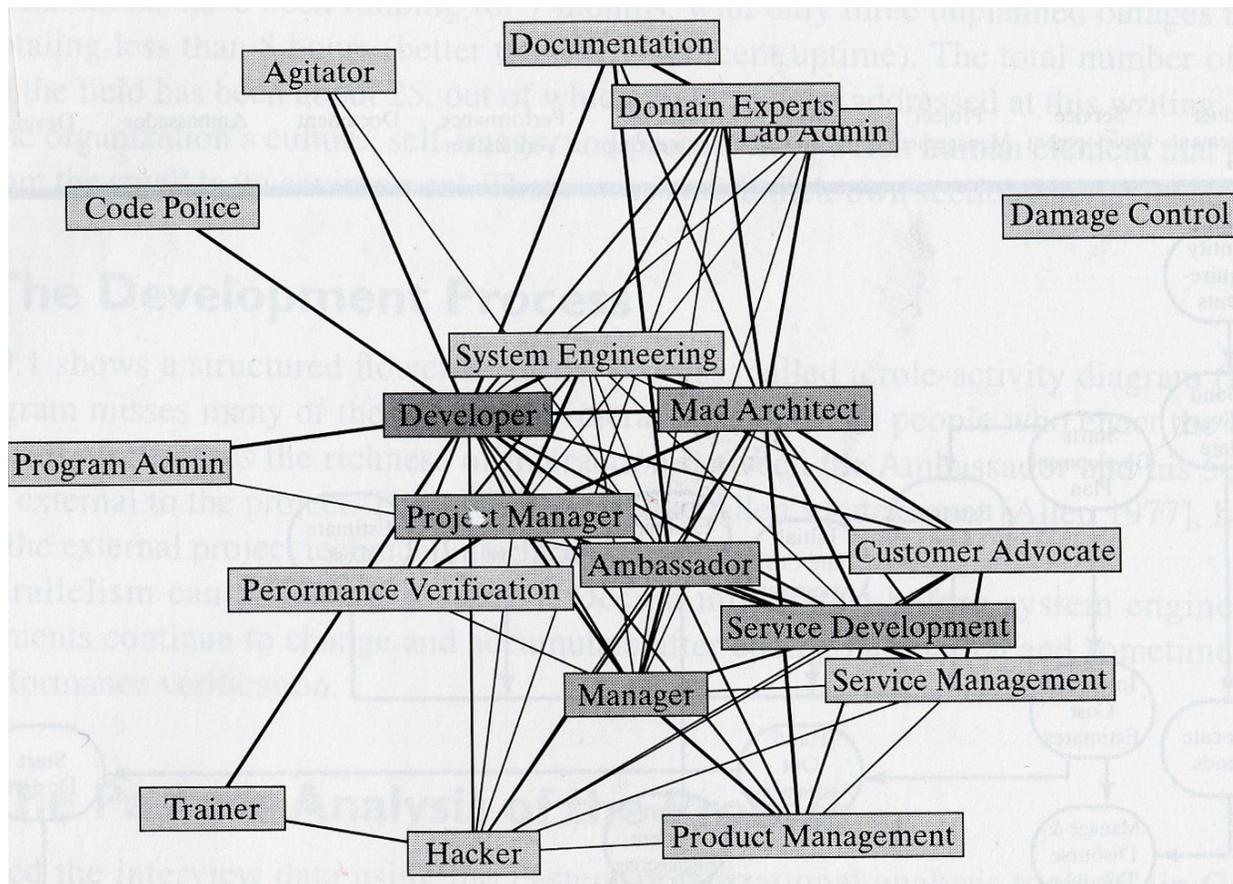
- **Vehicle for communication among stakeholders**
- **Reasoning about an evolving system**
- **Intentional transformation**
- **Mechanism for attacking risk**
- **Accountability**
- **Preservation of tribal memory**

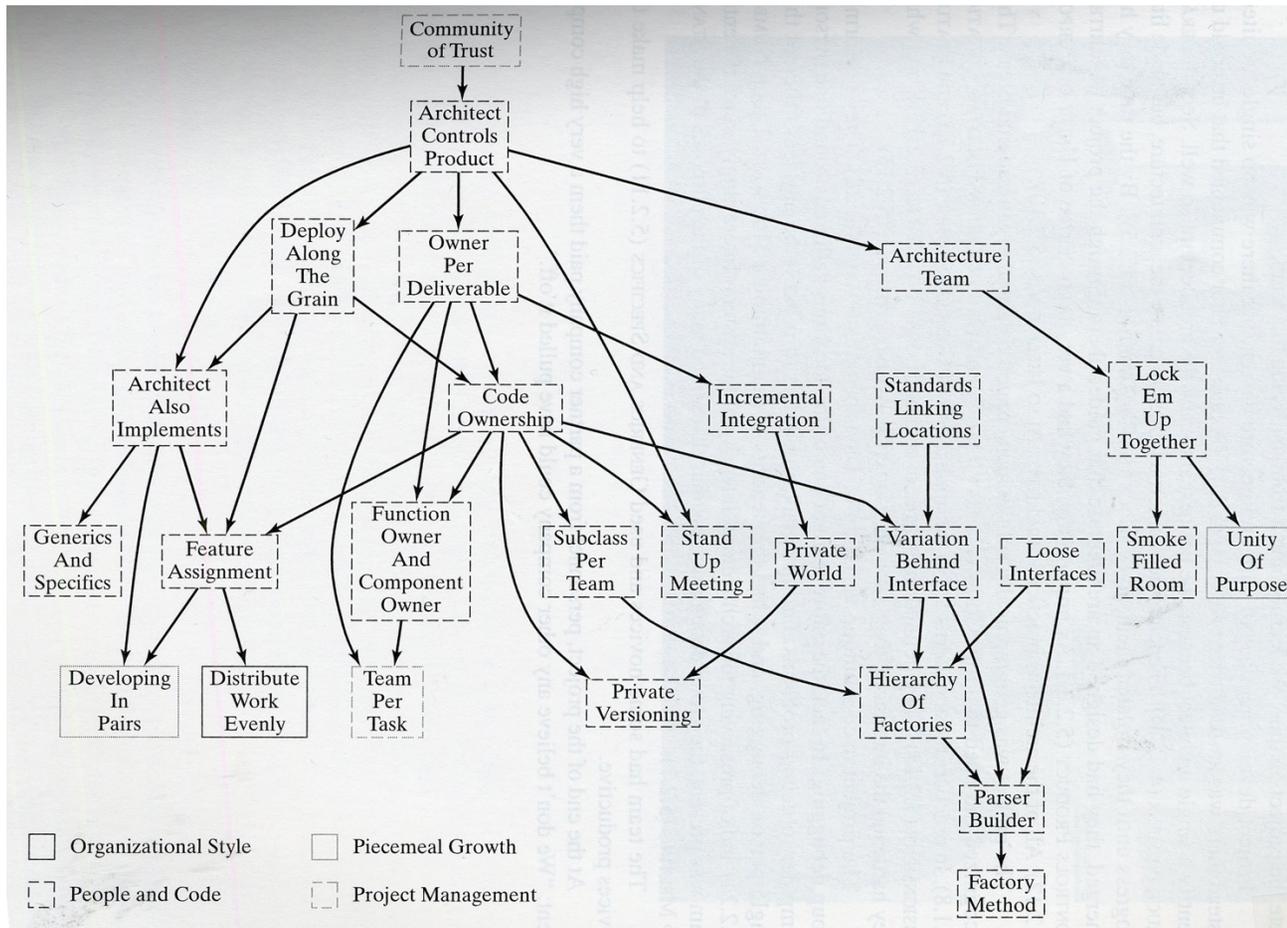
Architectural governance maturity



Ross et al, *Enterprise Architecture as Strategy*

Organizational best practices





Organizational Patterns

- **Big dripping hairball**
- **Senior designer**
- **Chief architect**
- **Architecture team**
- **Architecture control board**

In conclusion

Things You Can Do With Old Software

- **Give it away**
- **Ignore it**
- **Put it on life support**
- **Rewrite it**
- **Harvest from it**
- **Wrap it up**
- **Transform it**
- **Preserve it**

Things You Can Do With An Architecture

- **Reason about its transformation**
 - Evolution, rapid reaction, modernization, merging, acquisition, divestiture

- **Asset identification and repurposing**
 - Product line
 - Strategic thrust

Grady Booch, *IBM Fellow*
gbooch@us.ibm.com

<http://www.handbookofsoftwarearchitecture.com>