

Next-Generation Architecture to Support Simulation-Based Acquisition

Dr. Bipin Chadha
Principal Member
Lockheed Martin ATL
(856) 338-3865
bchadha@atl.lmco.com

John Welsh
Manager
Lockheed Martin ATL
(856) 338-4223
jwelsh@atl.lmco.com

Abstract-For simulation-based acquisition to effectively lower life-cycle cost and cycle time, the ability to make good design decisions early is a significant driver. Building virtual prototypes, enabling one to analyze the impact of decisions, achieves effective simulation-based acquisition processes. Virtual prototypes need to support a comprehensive set of analyses that will be performed on the product; hence, all aspects of product data and behavior need to be represented. Building virtual prototypes of complex systems being designed by a multi-organizational team requires new architectural concepts and redesigned processes. Implementation of these new architectures is complex and needs to leverage commercial technologies to achieve feasible solutions. One must also carefully consider the state of the current commercial technologies and frameworks as well as the organizational and cultural aspects of organizations that use these systems. This paper describes key architectural principles that one must address for a cost-effective implementation. The paper then discusses key architectural concepts and trade-offs that are necessary to support virtual prototypes of complex systems.

I. INTRODUCTION

Lockheed Martin, government and industry partners, and supply chain members are developing and manufacturing large, complex systems. Using simulation-based acquisition and design strategies to develop cost-effective virtual prototypes of these systems present enormous challenges. One will use virtual prototypes, intended to support a product's entire life-cycle, for multiple purposes, including system engineering during conceptual design and for warfighters during training. Therefore, virtual prototypes must capture all of the information related to a product's definition and it must provide mechanisms to incorporate all aspects of a product's behavior. Product complexity forces organizations to share the design and manufacture of these products; therefore, the virtual prototype must be shared. Detailed knowledge of a subsystem typically resides with the supplying organization, so it is critical that the organization create and manage the virtual prototype of that subsystem. Management of the virtual prototype's complexity is important to ensure that developments focus on

areas of sufficient benefit; otherwise, the implementation cost of the virtual prototype could be unbounded. Well-designed architectures are key to managing complexity. The following key goals will ensure a cost-effective virtual prototype:

- Create a multi-domain product model that is open, extensible, integrated, and synchronized.
- Integrate equations and/or behaviors with the relevant product data, thereby enabling evaluating functional performance and optimizing life-cycle costs.
- Create executable representations of the system with multi-level fidelity to support multi-level analysis.
- Execute simulation for analysis and demonstration.

II. MYTH OF COMMONALITY IN ARCHITECTURE

Information Technologies are trending toward common systems, tools, and processes. While that is a trend away from disjointed systems, it is unrealistic to assume that there will ever be a single process or system that satisfies every need in an organization with multiple businesses. Figure 1 shows diminishing returns as the degree of commonality increases beyond a limit. Figure 2 shows that hidden or ignored qualitative factors increase costs while easily quantifiable and visible metrics decrease costs. In this example, maintenance, deployment, and integration represent quantifiable costs, while coordination and changes and implementation delays represent qualitative factors that cause significant cost penalties in overall solutions. Most approaches ignore qualitative factors that become important as commonality approaches 100 percent. Existing architectures only account for technical aspects and ignore cultural and organizational aspects.

We need better virtual prototyping architectures to address this dichotomy among business needs and capabilities available with commercial off-the-shelf (COTS) solutions. A federated approach [1] provides the flexibility to deal with these current trend factors. Federation implies that one has established a degree of interoperability among business systems. Interoperability enables a shared information area and it provides a set of operations that can be initiated in a controlled fashion by one system operating on information in the shared/private area of another system. Members can join a federation to share information and functionality on a

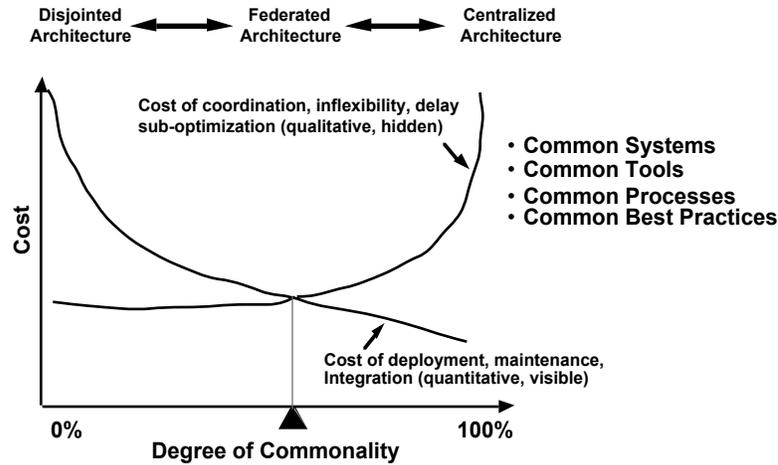


Fig. 1. Federated architecture provides an optimal solution.

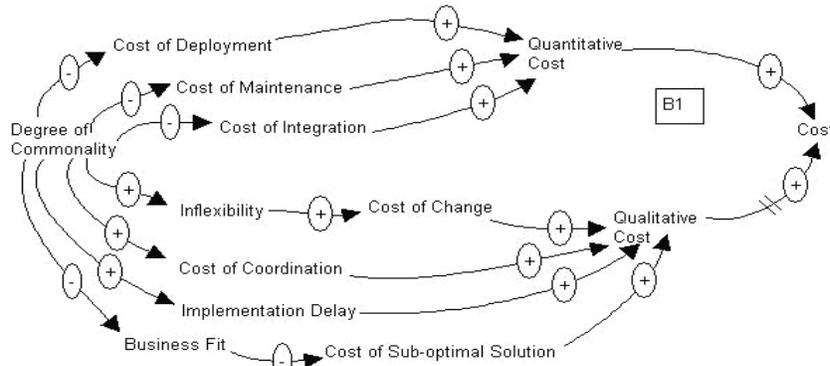


Fig. 2. Architecture trade-off model.

program and they can leave when the effort is finished. Federation provides a low-risk alternative to existing technology-investment strategies, and it enables organizations to bring new technologies in a modular fashion instead of requiring a total upgrade of the internal business system architecture (the *big bang* approach). A federated architecture enables programs to implement a total-systems view and it also provides the ability to optimize supply chains at a global level.

III. GUIDING PRINCIPLES

Several principles guide the development of the architecture. These principles are useful in information technology and other domains, such as lean manufacturing [4] and systems thinking [5]:

- Leverage COTS tools and technologies to the maximum extent possible.
- Focus on federated approaches instead of homogeneous information-system approaches.
- Provide information (and resources) only when needed.
- Provide only needed information (no more, no less).
- Do not carry Information defects to the next step.

- Put in place a process that discourages the generation of defects.
- Information should be owned by the entity that is most suited to keep it current/accurate.
- Information must be accessible by those who have or may have a need.
- Account for soft/qualitative factors.

The focus is on *guiding principle* as opposed to hard-design constraints. This allows one to realize benefits in most cases and to avoid penalties associated with exceptional cases.

IV. ARCHITECTURE FOR SIMULATION-BASED ACQUISITION

The principle objective of the virtual prototyping architecture is to reduce the complexity of the system being designed to implement the virtual prototype. The architecture must also help achieve modularity and reuse of components that make up that architecture. One must apply the principles of Cost as an Independent Variable (CAIV) analysis for systems design to the architecture for the virtual prototype.

A. Architecture Concepts

Rapidly evolving software, network, and associated technologies require innovative concepts to leverage their benefits in the architectural approach. This section provides an overview of these concepts, which represent foundational technologies for virtual prototyping architectures.

1. Internet Architecture

The virtual prototype architecture will need to be highly scalable and evolvable. The Internet provides an architecture that is highly distributed and scalable. It derives these properties via simple mechanisms, such as hyperlinking information across web servers. Nodes on networks can be clients and/or servers, and each can perform the role that is necessary for operation. The ability of enterprise agents to hyperlink to each other across the network and to travel across clients and servers will provide essentially unlimited scalability for the architecture.

2. Common Object Request Broker Architecture

The Object Request Broker (ORB) is a basic architectural construct sits between clients and makes requests. Figure 3 [2] shows the objects that will service those requests. The ORB mechanisms can support all interactions among various architectural components.

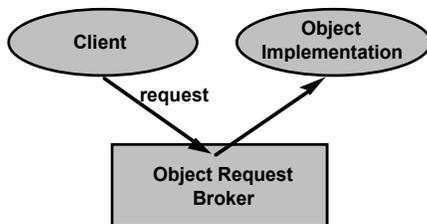


Fig. 3. Object request broker concept.

One can define interfaces statically in an interface definition language (object management group IDL), which defines the types of objects according to operations that may be performed on them and the parameters to those operations. The IDL is language independent, and bindings to several programming languages are available, such as C++ and Java. Note that ORBs do not provide complete data-management functions or object-domain functionality. They facilitate the communication without necessarily understanding what is being communicated. They also interoperate via the Internet Inter-ORB Protocol (IIOP), which specifies a standardized interoperability protocol for the Internet.

3. Federated Architecture

A federation implies a loosely coupled system distributed across the Internet or an Intranet, where the participants join

or leave the federation without breaking it and where they function on their own when not a part of the federation (Figure 4).

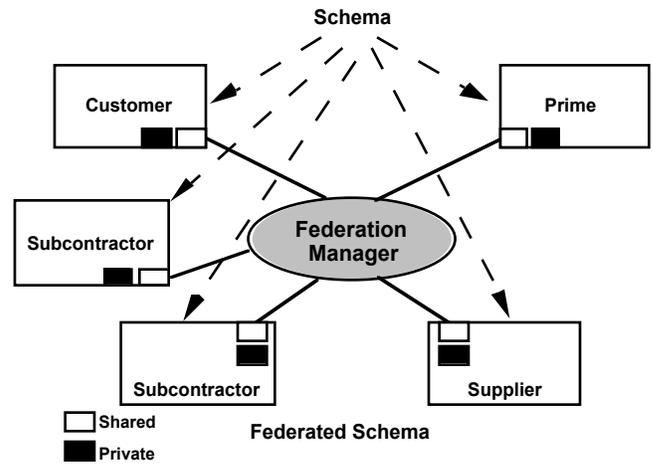


Fig. 4. Federated architecture concept.

The federation provides a low-risk alternative to existing technology investment strategies, and it enables organizations to bring new technologies in a modular fashion as opposed to the *big bang* approach, which requires replacement of most or all of the information systems for a technology upgrade. Mechanisms, such as smart-proxy objects, achieve federation.

4. Smart Proxy Objects

An innovative concept is the “proxy” object, which provides a surrogate or placeholder for another object to control access to it (Figure 5) [3]. A proxy object resides in the local environment and represents an object residing in a remote environment. The proxy handles requests made to it and forwards them to the real object for further processing, which returns the results back to the proxy object. The proxy then forwards the results to the requester. Therefore, clients only work with the proxy object in the local environment. They are insulated from the details and how-to of dealing with objects in remote environments. These proxies are not the same as those in the ORB; however, they will use proxy objects and services provided by the ORBs for their operation. Proxy objects provide tremendous flexibility in implementation approaches, enabling variable degrees of information replication and flexible behavior via tailorable business rules. Proxies can be implemented to various levels of complexity, starting from simple uniform resource locators (URLs) to complex, tightly-coupled proxy objects.

5. Component-Based Architecture

A key concept that enables realistic and affordable development of the architecture is the ability to build in

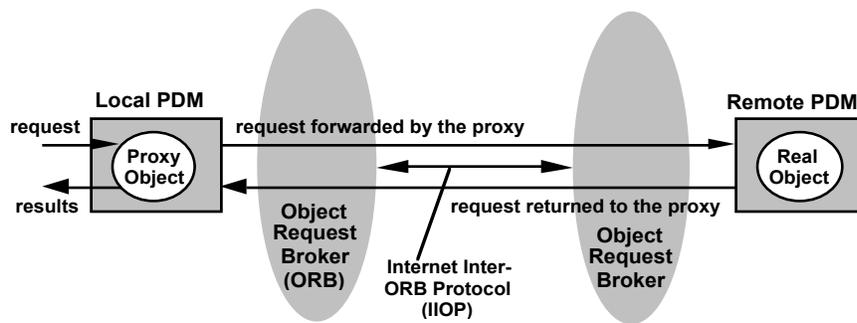


Fig. 5. Proxy object concept.

modular, reusable components. A component is a reusable software program that can be easily added to or combined with other software programs, enabling construction of sophisticated programs in a modular fashion. One can divide the architecture into components via well-defined interfaces. Interfaces allow small components to be grouped to create complex systems that would otherwise be impractical to build. Concepts of interfaces allow objects to be typed across many dimensions. Objects can adhere to many different interfaces without the burden of multiple inheritance. This also allows one to build information models from the bottoms-up and not use the monolithic, top-down information model. The impact of component-based architectures is a significant gain in productivity. The use of this design strategy is rapidly gaining acceptance by the information-technology community. Both COM+ and Java use component-based architectures.

V. IMPLEMENTATION ISSUES

In addition to typical cultural and organizational issues that one must address in architecture strategies [1], a key implementation issue is the use of a top-down versus a bottom-up approach. We recommend a hybrid of the two based on the following observations:

- Bottom up is relevant when there is little understanding of the problem.
- Top down can completely address the problem (no items are missed).
- Bottom up enables the problem to be addressed in small chunks.
- Top down works in relatively stable environments; bottom up is suitable for rapidly changing environments.
- Top down is more efficient when there is a low probability of going astray.
- Bottom up is more forgiving.
- Top down is about “knowing it all” at the start; whereas, bottom up is about “learning it” along the way.
- Top down has slower cycles; whereas, bottom up has faster cycles.

- Apply top down at the conceptual level but apply bottom up for details.

A hybrid of both approaches fits most real-world situations. Most projects start top-down before shifting to bottoms up at implementation, where the focus is on reuse of available, existing, or previously developed components. Strategies can also shift emphasis among top down and bottoms up at multiple points in the development process.

VI. SUMMARY

Virtual prototyping is a critical to the cost-effective design of complex systems. New architectural concepts are emerging to support virtual prototyping of complex systems across company boundaries. When implementing virtual prototyping architectures, realize there are limits to benefits achieved through commonality across distributed implementations and that soft/qualitative factors play an important role in architectural trade-offs. This paper outlined key architectural principles and concepts that improved the practicality of large-scale virtual prototyping efforts. Many COTS tools and technologies are emerging to support implementation of these concepts. Effectively leveraging these tools and technologies will manage the scope of the effort while maintaining cost and schedule constraints. Bottom-up and top down design approaches have advantages, but a hybrid approach is best for most implementations.

REFERENCES

- [1] Chadha, B., “A Federated PIM for Supply Chains”, DH Brown Symposium, 1997.
- [2] “The Common Object Request Broker: Architecture and Specification”, Revision 2.0, OMG, July 1995, Updated July 1996.
- [3] Gamma, E., et. al., “Design Patterns: Elements of Reusable Object-Oriented Software”, Addison-Wesley, 94.
- [4] Womack, J.P., Jones, D.T., Roos, D., “The Machine that Changed the World”, MIT Press, 1990.
- [5] Senge, P.M., et. al., “The Fifth Discipline Fieldbook”, Currency Doubleday, 1999.

LOCKHEED MARTIN

Systems Integration



Next Generation Architecture to Support Simulation Based Acquisition



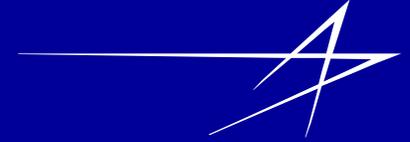
Advanced Technology Laboratories

James Saultz, Dr. Bipin Chadha, and John Welsh

jwelsh@atl.lmco.com

**1 Federal Street • A&E Building 2W
Camden, New Jersey 08102**

Agenda



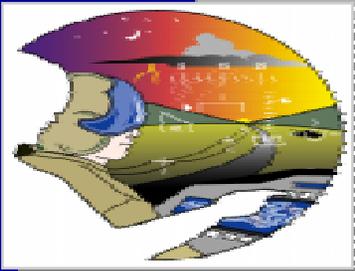
- **Advanced Technology Laboratories (ATL)**
- **Architecture trade-off**
- **Architecture concepts**

Advanced Computing Solutions ...

Leveraging technology into business



TECHNOLOGY



Artificial Intelligence

- Intelligent Agents
- Situation Assessment
- Associate Technology
- Data Fusion



Distributed Processing

- Component Systems
- Distributed Objects
- Integrated QoS
- Distributed Management/Planning



Embedded Processing

- Advanced Simulation/Autocode
- Digital Signal Processing
- Multiprocessor Architectures
- Enterprise Process and Information Management

BUSINESS

Intelligent Systems

- Operator Assistants
- C4I Software/Testbeds
- Agent Information Access
- Sensor Data Fusion

Information Infrastructure

- RT/FT QoS Software
- Component Architectures
- Distributed Frameworks

Embedded Training

- Auto-Assessment/Measurement
- Team Performance Enhancement
- Scenario-Based Learning

Virtual Prototyping

- Rapid Prototyping Tools/Services
- Advanced Simulation Testbeds
- Embedded Multiprocessors
- Smart Product Models

Enterprise Technology Capabilities

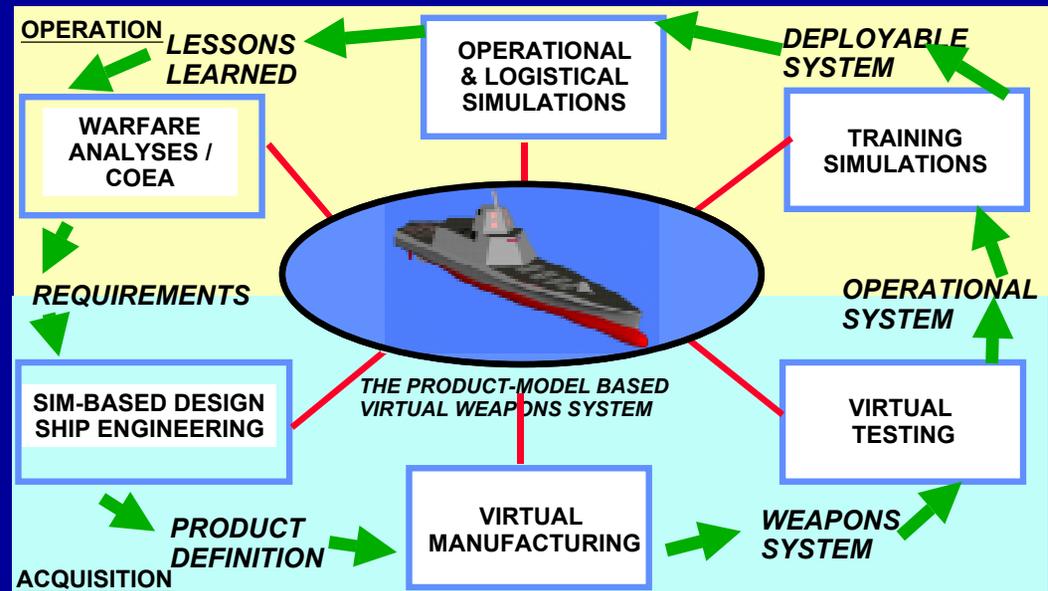


- **Key ATL capabilities**

- Process reengineering
- Product data management
- Reuse technology: classification hierarchy models and domain specific processes
- Manufacturing and supply chain technologies

- **SBA/Smart Product Model - ATL roles**

- Core SPM architecture development
- Systems engineering tradeoff and cost analysis capabilities



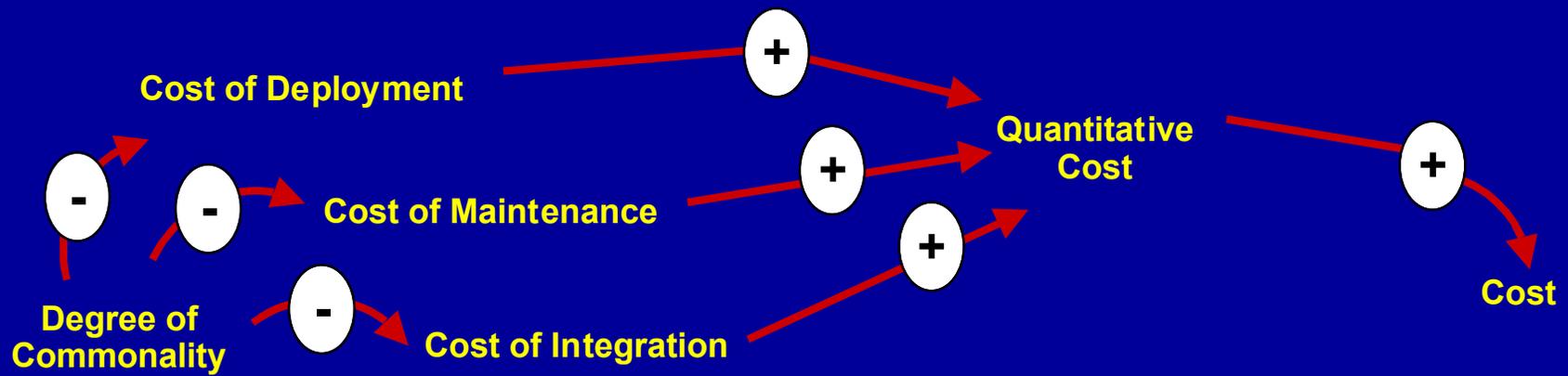
Leveraging Success: SBA/Smart Product Model strategy builds on PDM, COTS product integration and CAIV experience from RASSP and other programs

Rationale

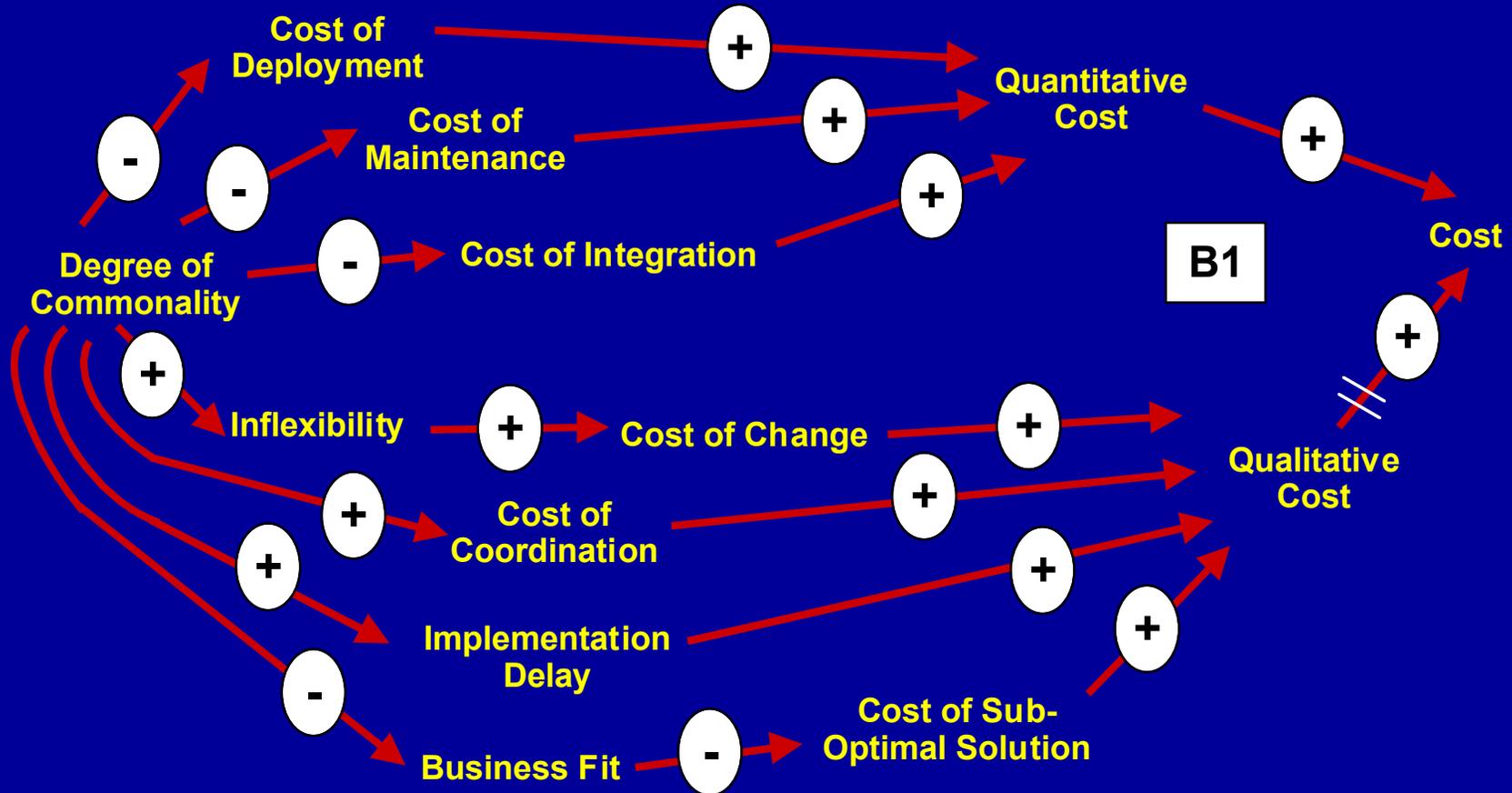


- **All businesses under pressure to cut cost and cycle time**
- **Programs run by teams of different organizations; each has its own tools**
- **Each program may require a different mix of team members and systems**
- **Heterogeneity of business systems cannot be avoided**
- **Cannot (always) enforce use of one set of business systems for each program**
- **Cannot afford to procure/install/deploy/maintain/train on multiple systems**
- **Cannot afford to integrate design tools to multiple systems**

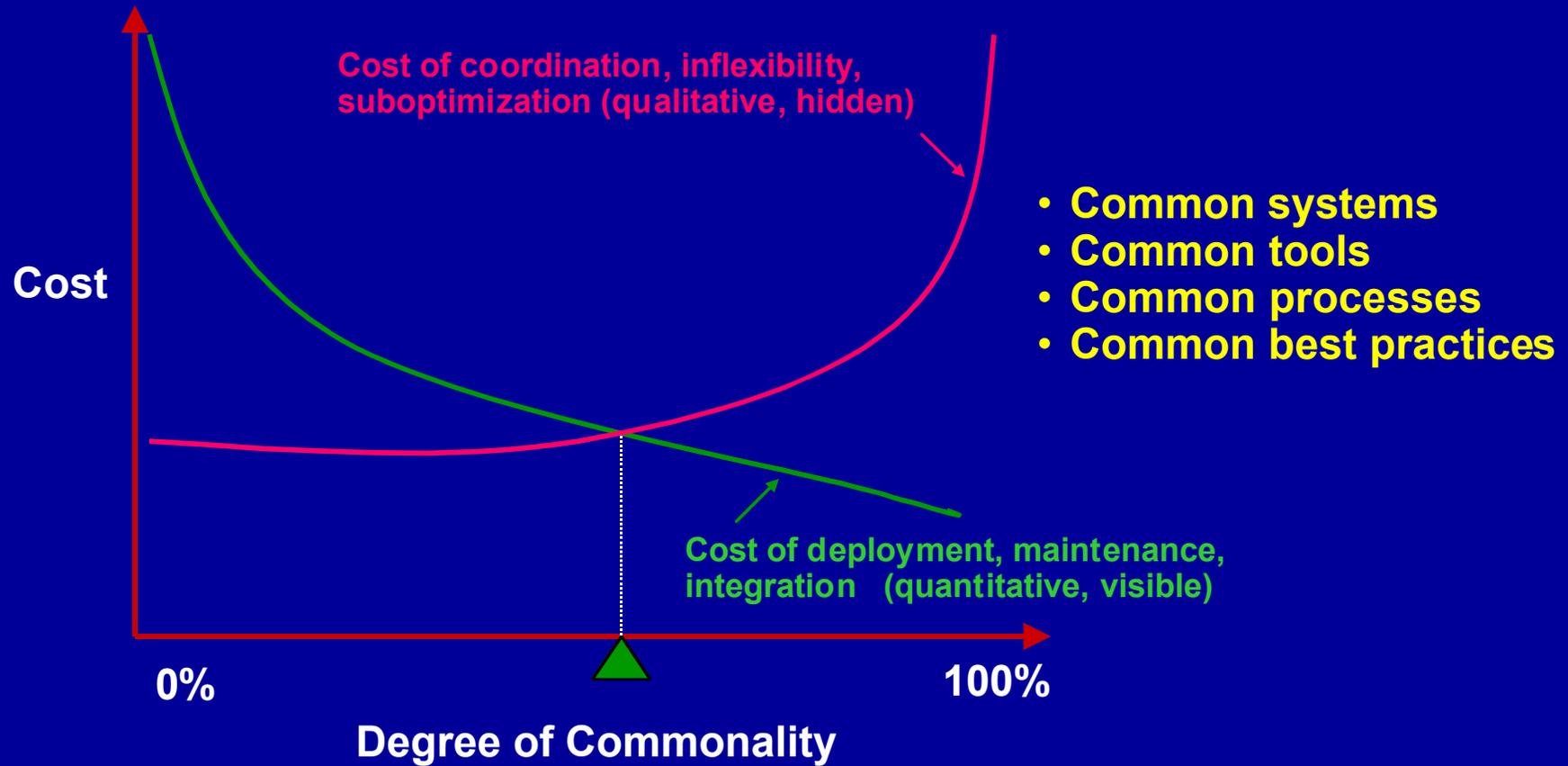
Degree of Commonality Model



Degree of Commonality Model



Myth of Commonality

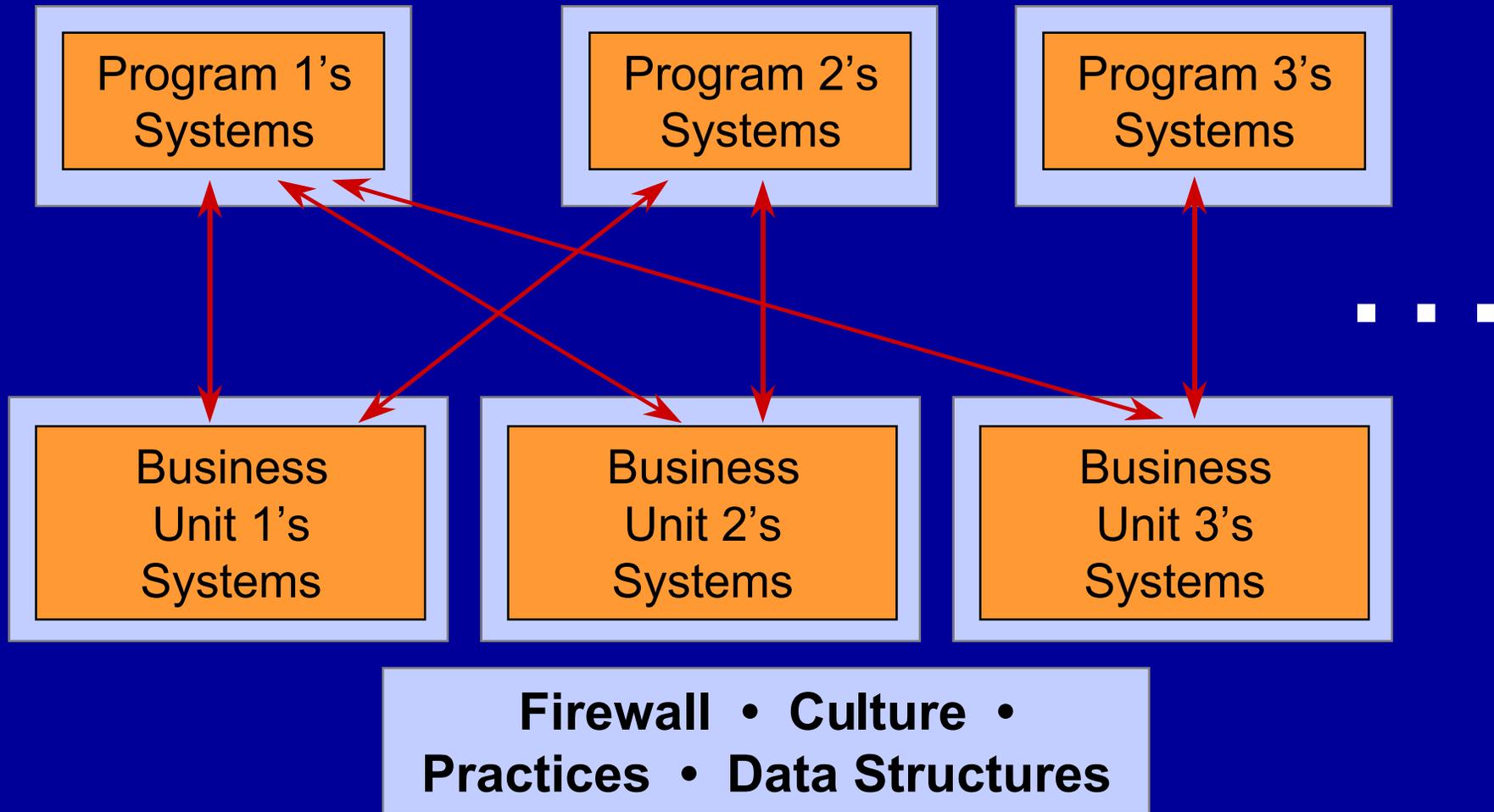


Approach

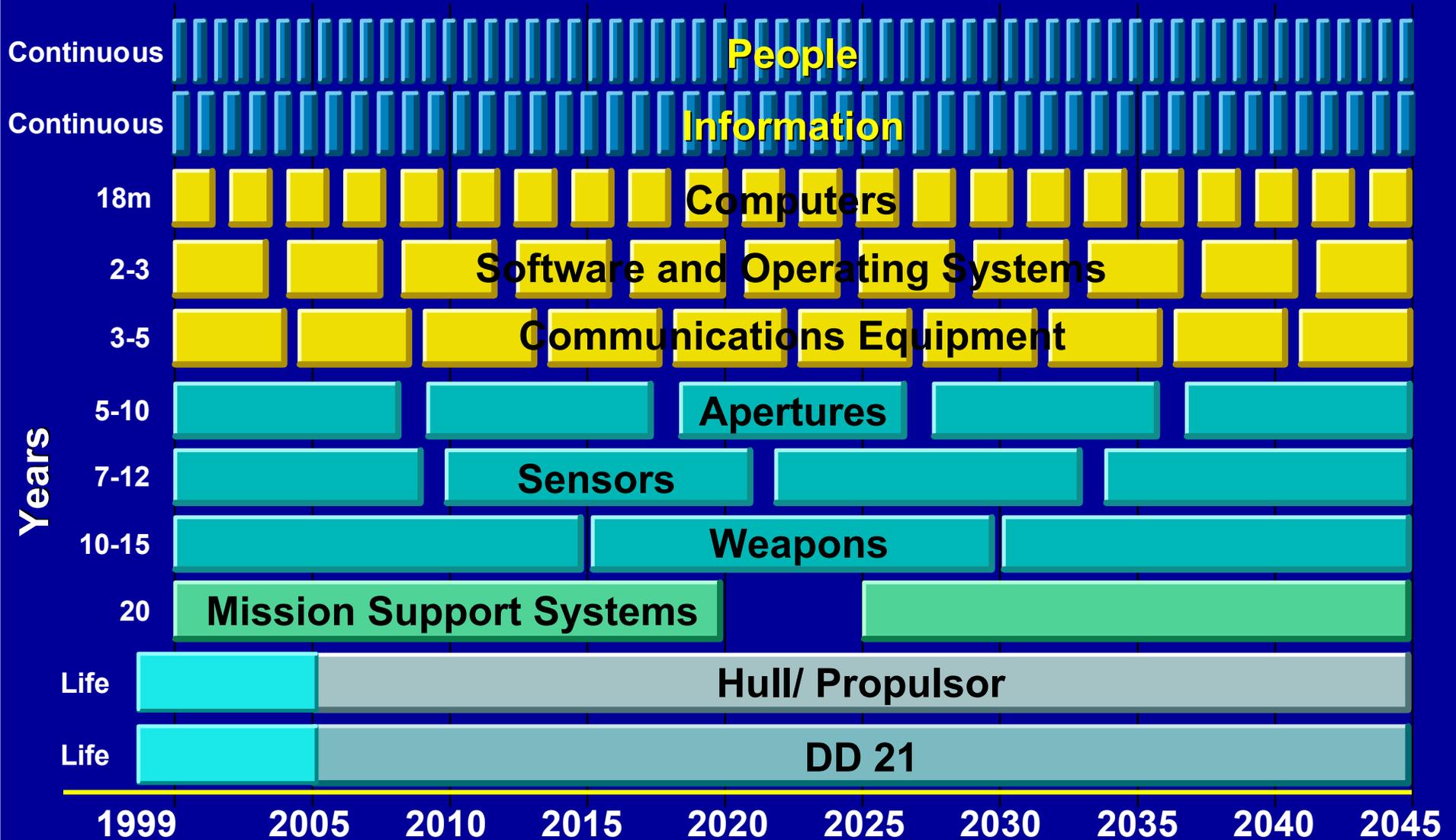


- **Accept diversity of business systems**
- **Let each business maintain its business systems while striving for commonality**
- **Design for change**
- **Be open, flexible, and vendor co-dependent**
- **Develop technologies to enable these business systems to interoperate as dictated by business requirements**

Business Architecture



Accommodating Change



Everything will change. Things Change at Different Rates

Architecture Strategy



- **Guiding principles**

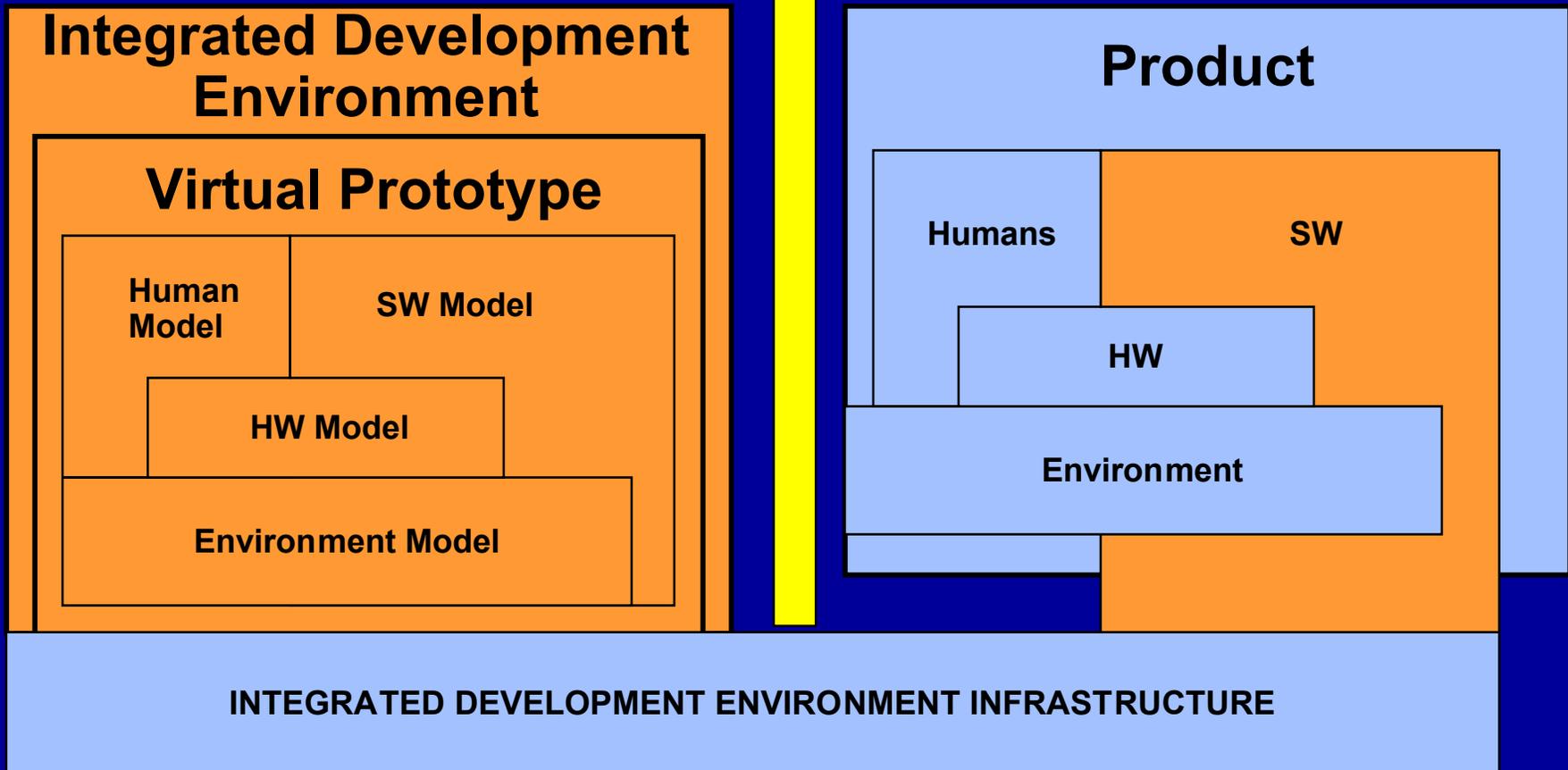
- Leverage commercial, off-the-shelf (COTS) tools
- Provide information (and resources) only when needed
- Provide only the needed information (no more, no less)
- Do not carry information "defects" to the next step
- Put in place a process that discourages generation of defects
- Information should be owned by the entity that is most suited to keep it current/accurate
- Information should be accessible by those who have or may have a need
- Account for soft/qualitative factors

Context for the Architecture

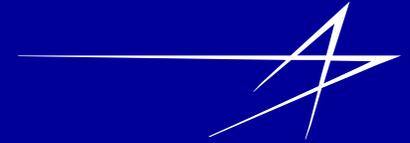


Virtual

Physical/Real

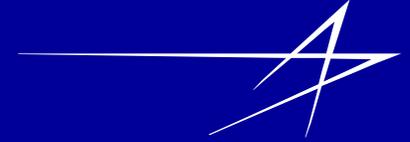


Architecture Concepts



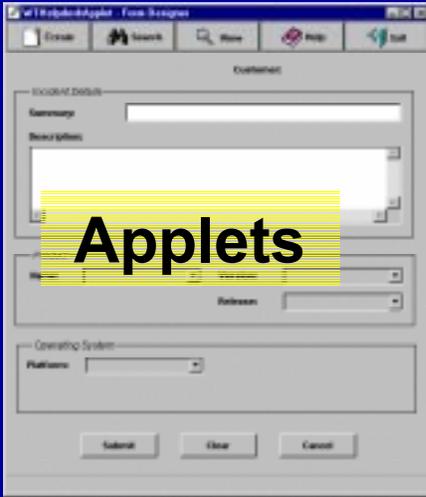
- **Internet-based architecture**
- **Federated architecture**
- **Object-request-broker architecture**
- **Component-based architecture**
- **Smart proxy objects**

Internet-Based Architecture



- **Based on loose coupling of systems connected by ubiquitous standards**
- **Highly scalable, distributed and open architecture**
- **High local control; no central control**
- **Easy access to information**

Familiar User Interaction



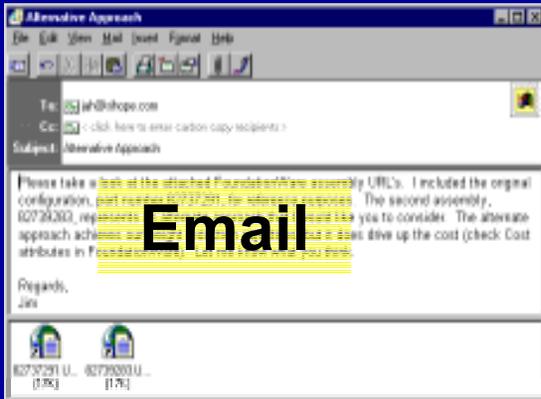
Applets



Search Engines

Search Keywords

Search Results



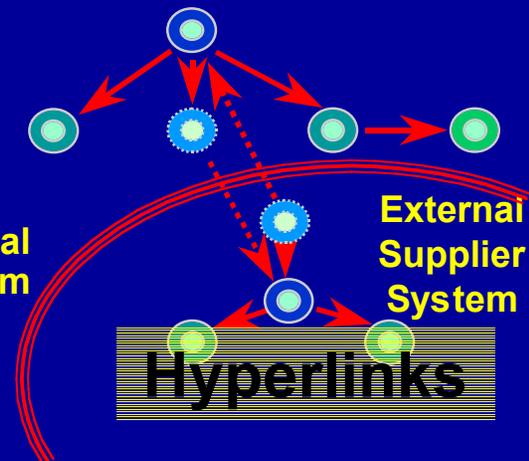
Email



Push

Internal System

External Supplier System

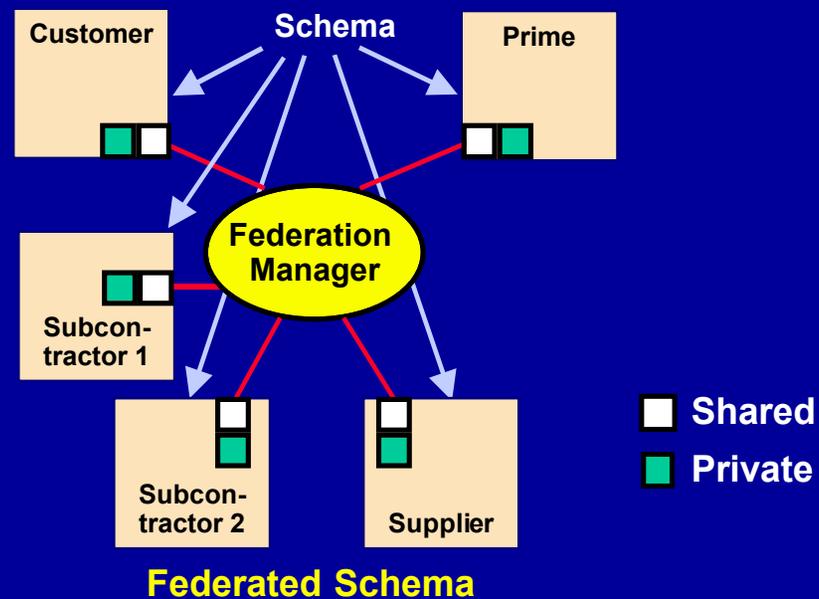
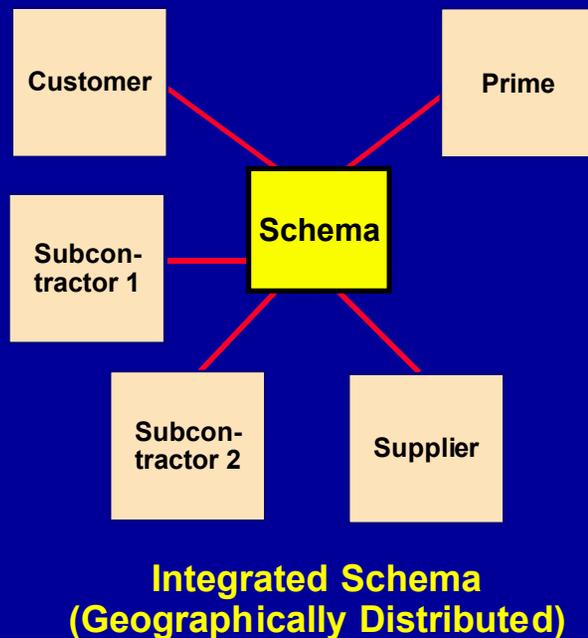


Hyperlinks

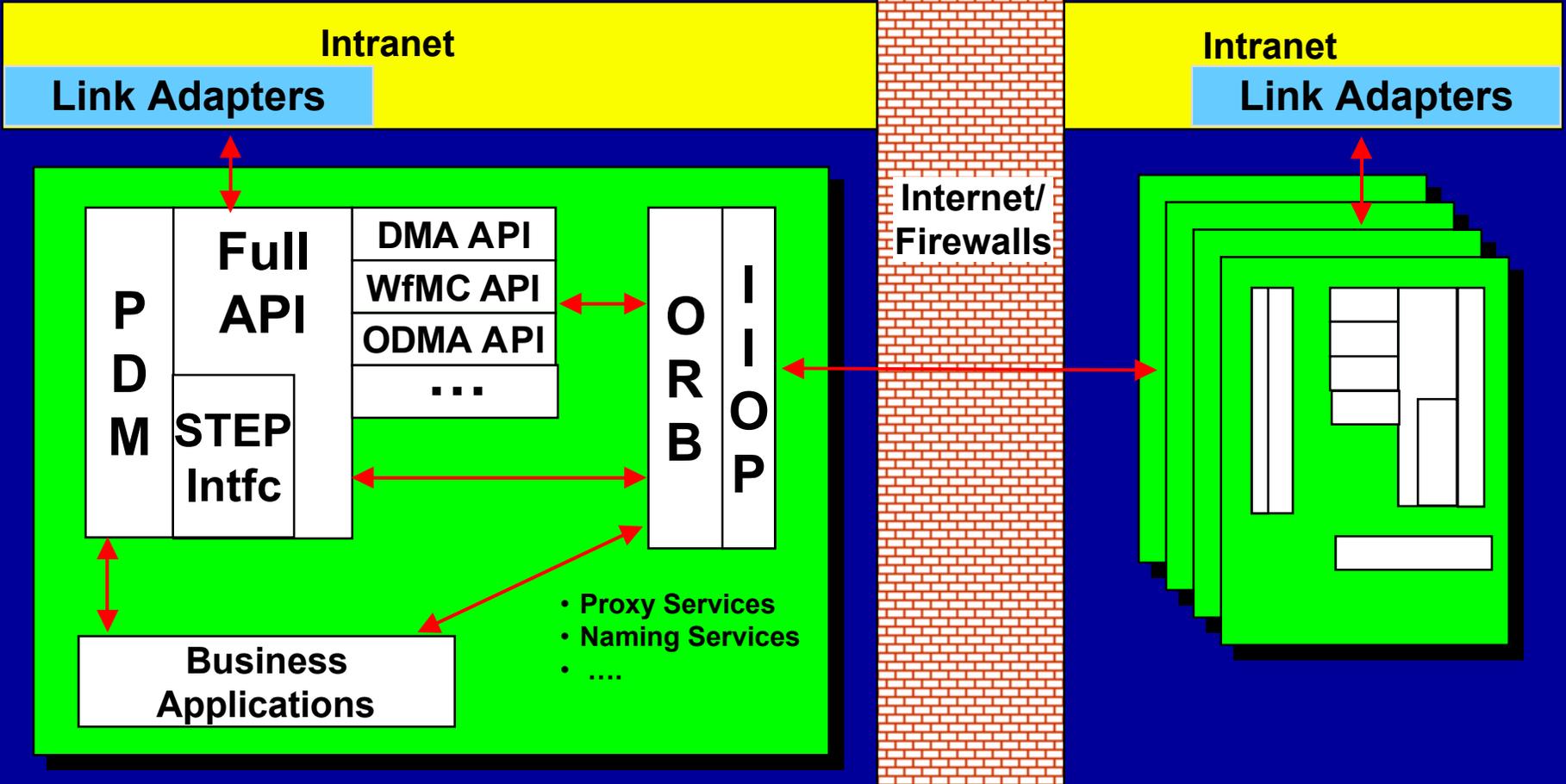
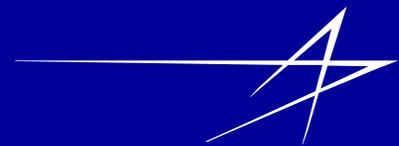
Federated Architecture



- A federation is an arrangement where members join and leave as necessary and can exist as autonomous entities
 - This is the way corporations work when they team for programs
 - Unfortunately, our business systems don't behave this way
 - Enterprise-wide implementations are not necessarily the best route
 - Completely common systems/tools/processes are not necessarily good



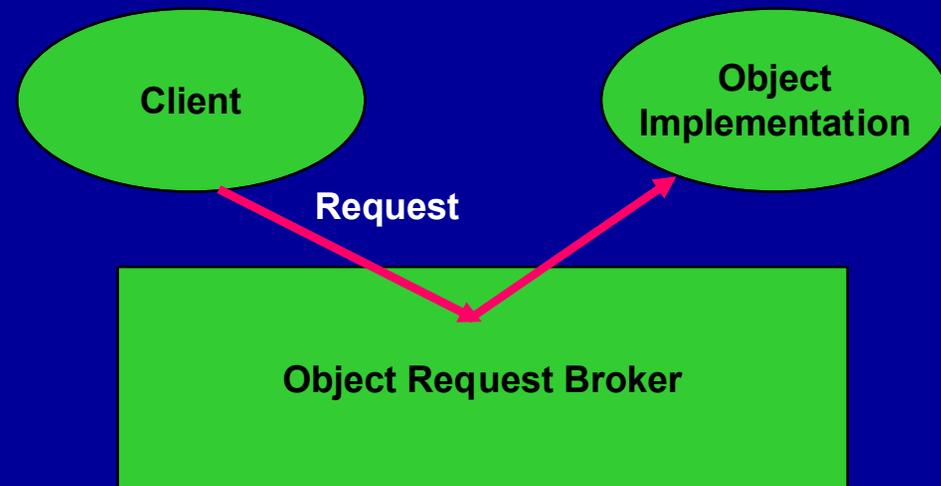
Federation Architecture Implementation Strategies



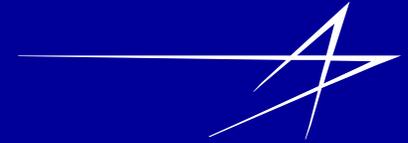
Object-Request-Broker Architecture



- Defined as the CORBA (Common-Object-Request Broker Architecture) standard by Object Management Group (OMG)
- Defines Internet Inter-ORB Protocol (IIOP) as the underlying protocol
- All requests/responses routed through a request broker



Component-Based Architecture

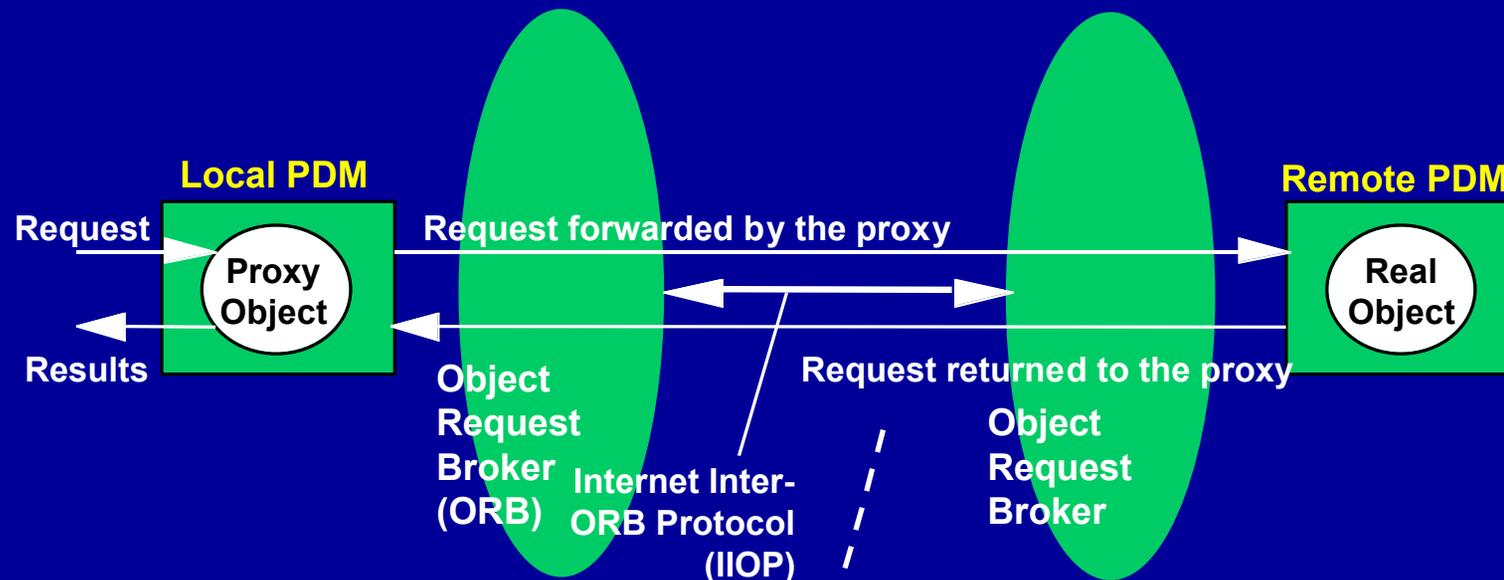


- **System defined as a collection of components**
- **Components have well defined interfaces**
- **Ability to build object models in a bottom-up fashion**
- **Allows bottom-up development of complex systems**
- **Java/remote method invocation and COM+ support component-based architectures**

Smart Proxy Objects



- Local proxy objects act as placeholders for remote objects
- Proxy objects responsible for providing transparent access
- Proxy objects provide implementation flexibility



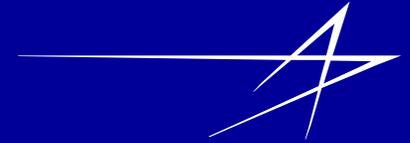
Top-Down or Bottom-Up



- **Use bottom-up when you don't completely understand the problem**
- **Use top-down to completely address the problem and to make sure you don't miss anything**
- **Use bottom-up when the problem needs to be addressed in small chunks**
- **Use top-down in relatively stable environment; use bottom-up in rapidly changing environments**
- **Top-down is more efficient when probability of going astray is low**
- **Bottom-up is more forgiving**
- **Top-down is about knowing it all at the start; bottom-up is about learning along the way**
- **Top-down has slower cycles; bottom-up has faster cycles**
- **Apply top-down at the conceptual level; apply bottom-up near details**

Need a Healthy Mix of Top-Down and Bottom-Up

Summary



- **Virtual prototyping is a key enabler for the simulation-based acquisition**
- **Approaches to large-scale virtual prototypes that require commonality across systems or business are limited by business and technical constraints**
- **New architectural concepts are emerging to facilitate virtual prototyping**
- **Use a hybrid top-down and bottom-up implementation approach**