# Model-based Approaches for Service Oriented Architectures

Mel Greer

Bob Epps

# Report Documentation Page

*Form Approved*
*OMB No. 0704-0188*



| 1. REPORT DATE | 2. REPORT TYPE | 3. DATES COVERED |
|---|---|---|
| **MAY 2011** | | **00-00-2011 to 00-00-2011** |

| 4. TITLE AND SUBTITLE | 5a. CONTRACT NUMBER |
|---|---|
| **Model‐based Approaches for Service Oriented Architectures** | 5b. GRANT NUMBER |
| | 5c. PROGRAM ELEMENT NUMBER |
| 6. AUTHOR(S) | 5d. PROJECT NUMBER |
| | 5e. TASK NUMBER |
| | 5f. WORK UNIT NUMBER |
| 7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Lockheed Martin,Cherry Hill,NJ,08002** | 8. PERFORMING ORGANIZATION REPORT NUMBER |
| 9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) | 10. SPONSOR/MONITOR'S ACRONYM(S) |
| | 11. SPONSOR/MONITOR'S REPORT NUMBER(S) |

12. DISTRIBUTION/AVAILABILITY STATEMENT
**Approved for public release; distribution unlimited**

13. SUPPLEMENTARY NOTES
**Presented at the 23rd Systems and Software Technology Conference (SSTC), 16-19 May 2011, Salt Lake City, UT. Sponsored in part by the USAF. U.S. Government or Federal Rights License**

14. ABSTRACT

15. SUBJECT TERMS

| 16. SECURITY CLASSIFICATION OF: | | | 17. LIMITATION OF ABSTRACT | 18. NUMBER OF PAGES | 19a. NAME OF RESPONSIBLE PERSON |
|---|---|---|---|---|---|
| a. REPORT **unclassified** | b. ABSTRACT **unclassified** | c. THIS PAGE **unclassified** | **Same as Report (SAR)** | **27** | |

# Why Model?

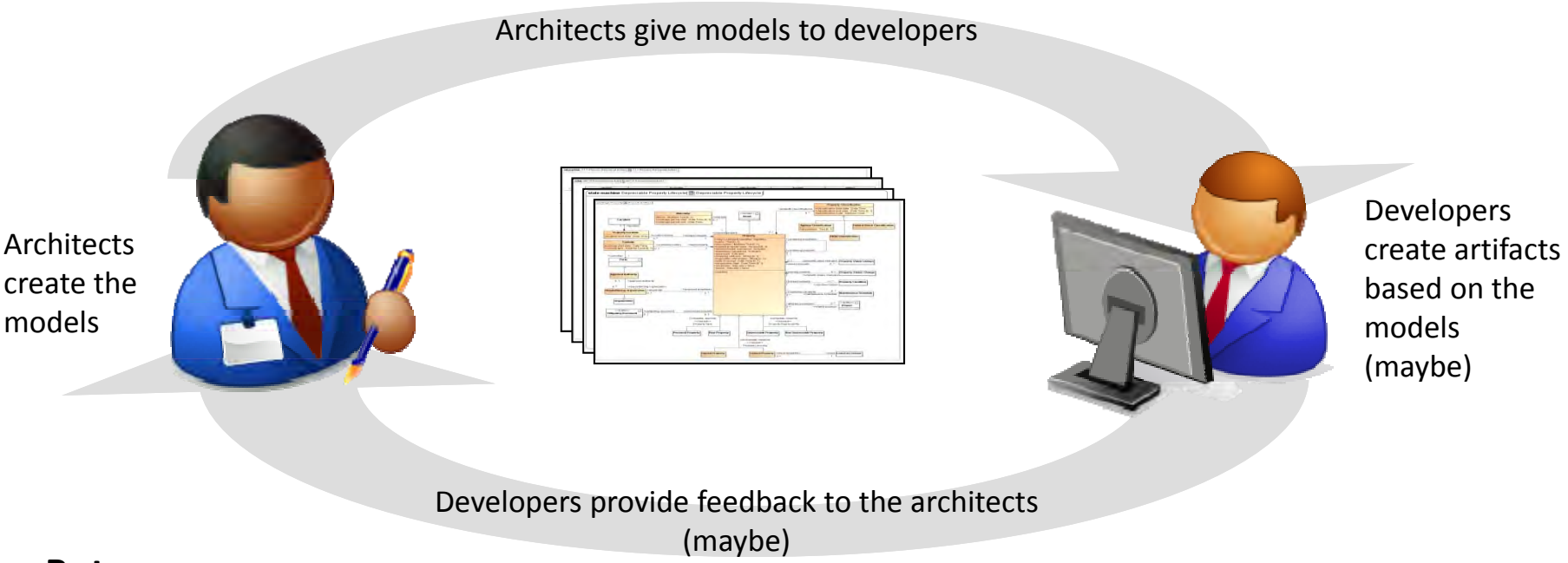
- A *model* is a set of statements in some modeling language made *about* some system or domain.
  - Standard modeling languages: Unified Modeling Language (UML), Business Process Modeling Notation (BPMN), Systems Modeling Language (SysML), Service Oriented Architecture Modeling Language (SoaML), etc.
- A model may be used to *describe* a domain or system under study or to *specify* a (business, software and/or hardware) system to be built.
  - Descriptive models are generally used for analysis.
  - Specification models are generally used for engineering.
- Models are intended to *represent* and *communicate* the results of analyses and proposals for new syntheses.
  - No model can represent *everything* – but, to be useful, a model must effectively promote general understanding and communicate important details.

# Why Execute Models?

- A model may specify the *behavior* of a system, that is, how the system interacts with external entities and changes its state over time.
- A behavioral model is *executable* if it is complete enough that the specified behavior can be *enacted* or *simulated* by an automated execution tool.
- Model execution may be used to:
  - Explore possible (desirable and undesirable) behaviors of a system
  - Validate the behavioral specification for a system
  - Actually act as the implementation of the system (particularly for business processes or software systems)

# Modeling for Software Development

How it usually works without executable models

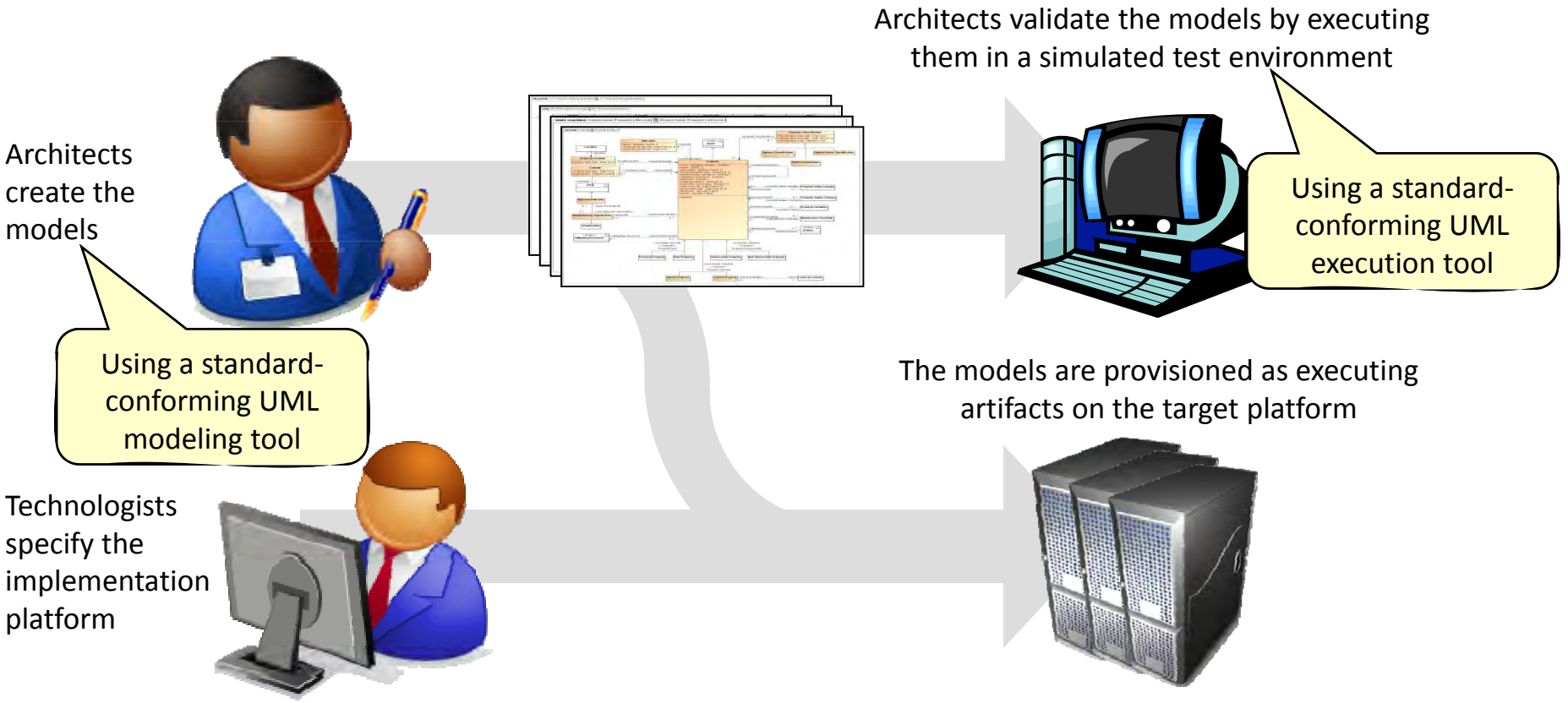



**But…**

- It is hard to validate the correctness of the models before development.
- The developers may not follow the models, without providing feedback.
- It is hard to keep the models and development artifacts in sync during development (and maintenance).

# Executable Modeling for Software Development

How it works with executable models



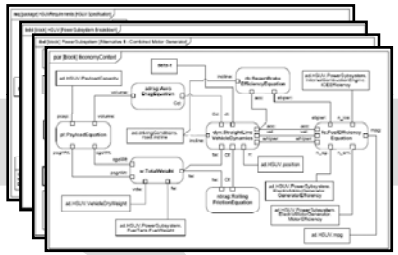

The models *are* the source code.

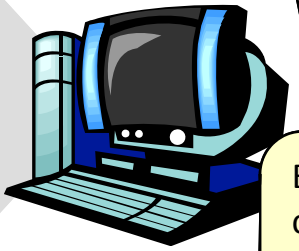
# Executable Modeling for System Engineering

Using a standard-conforming SysML execution tool

System engineers analyze, simulate and validate the system design, and allocate requirements to components.

System engineers create the models

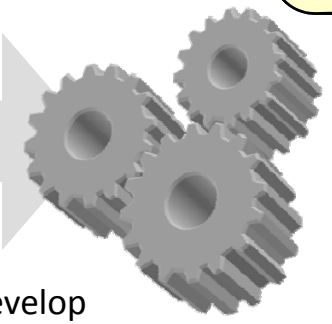




Execution artifacts could include:
- System behavior
- Timing
- Statistics

Using a standard-conforming SysML modeling tool

Models can include *both* hardware and software components.

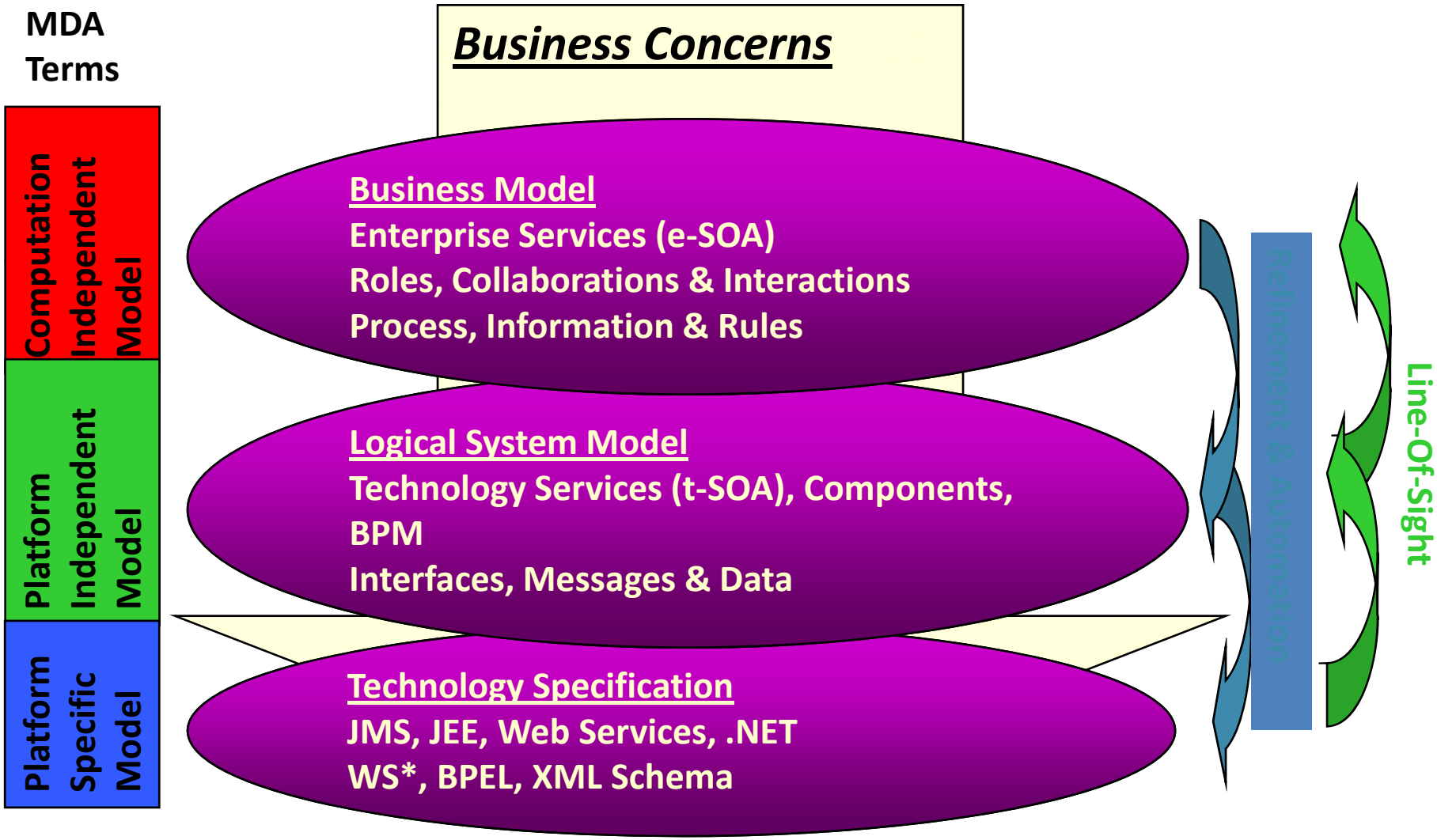


- Hardware and software engineers develop components to satisfy the requirements.
- Test engineers develop the test environment to verify the requirements.

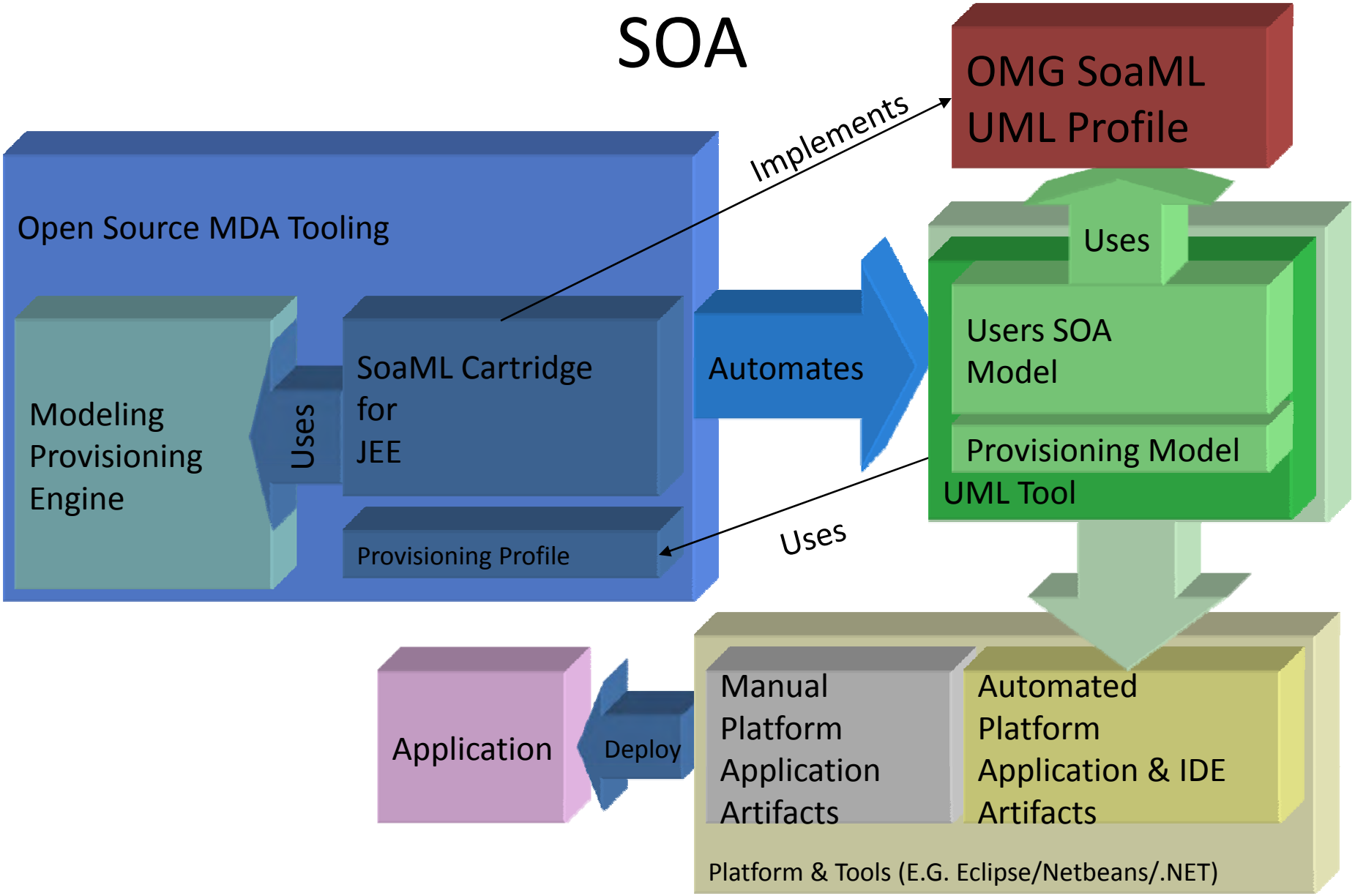
# What is SoaML?

- An OMG Standard for Modeling Service Oriented Architectures
  - Adopted from the UML® Profile for Modeling Services (UPMS) RFP
  - SoaML supports the "A" in SOA
  - Used for modeling SOA at the business, enterprise and technology levels
  - Leverages Model Driven Architecture
- A "Profile" of the Unified Modeling Language™
  - Can be used with off-the-shelf UML tools as well as customized tooling
- In the "finalization" stage of the OMG process – essentially an adopted "beta" specification
  - Finalization with minor clean-up expected to complete this year
- Tool support & implementations already exist
  - Tool support – making it easy to create services models
  - MDA Implementations – provisioning web services, business artifacts and implementations from SoaML models

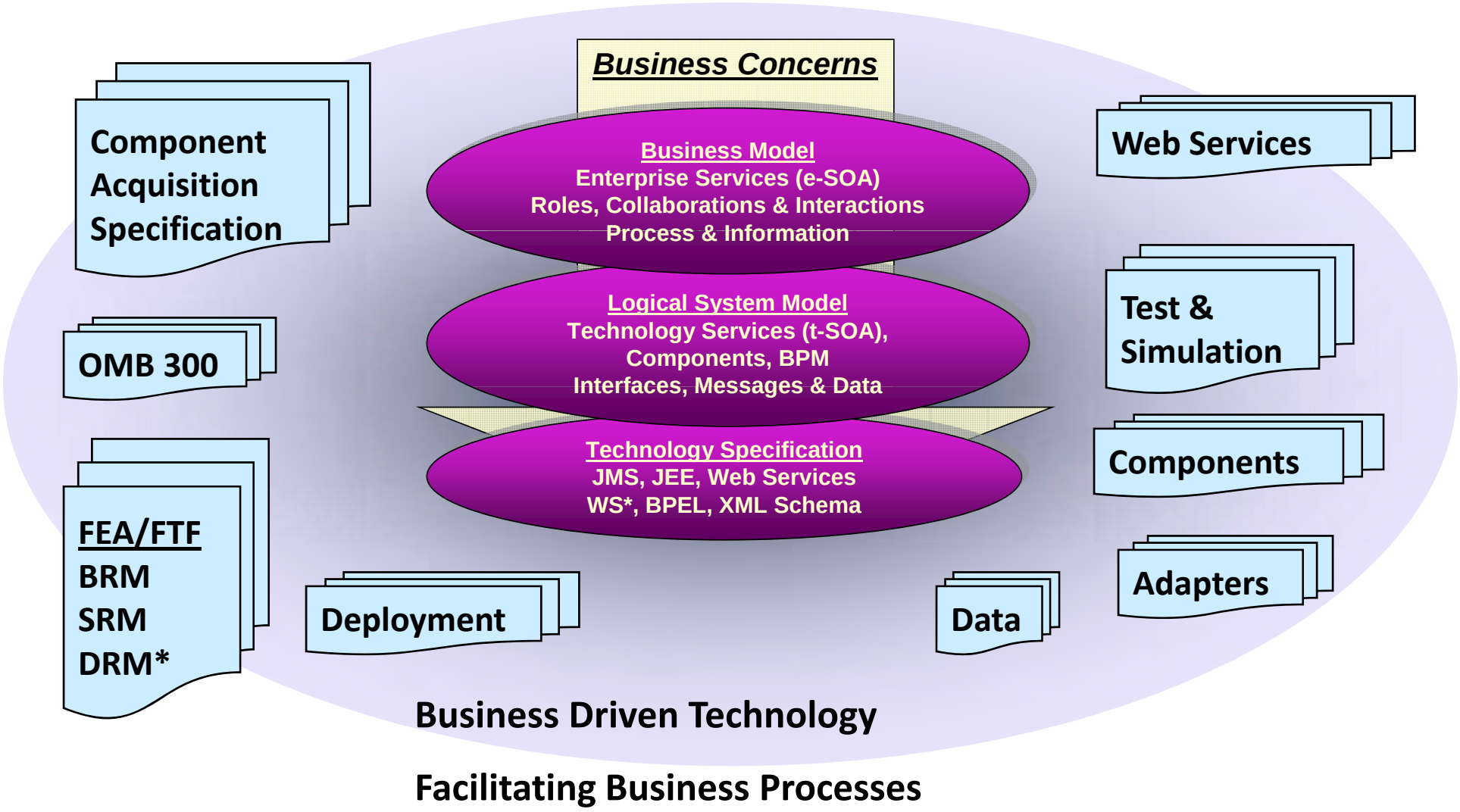
# Context for Enterprise SOA



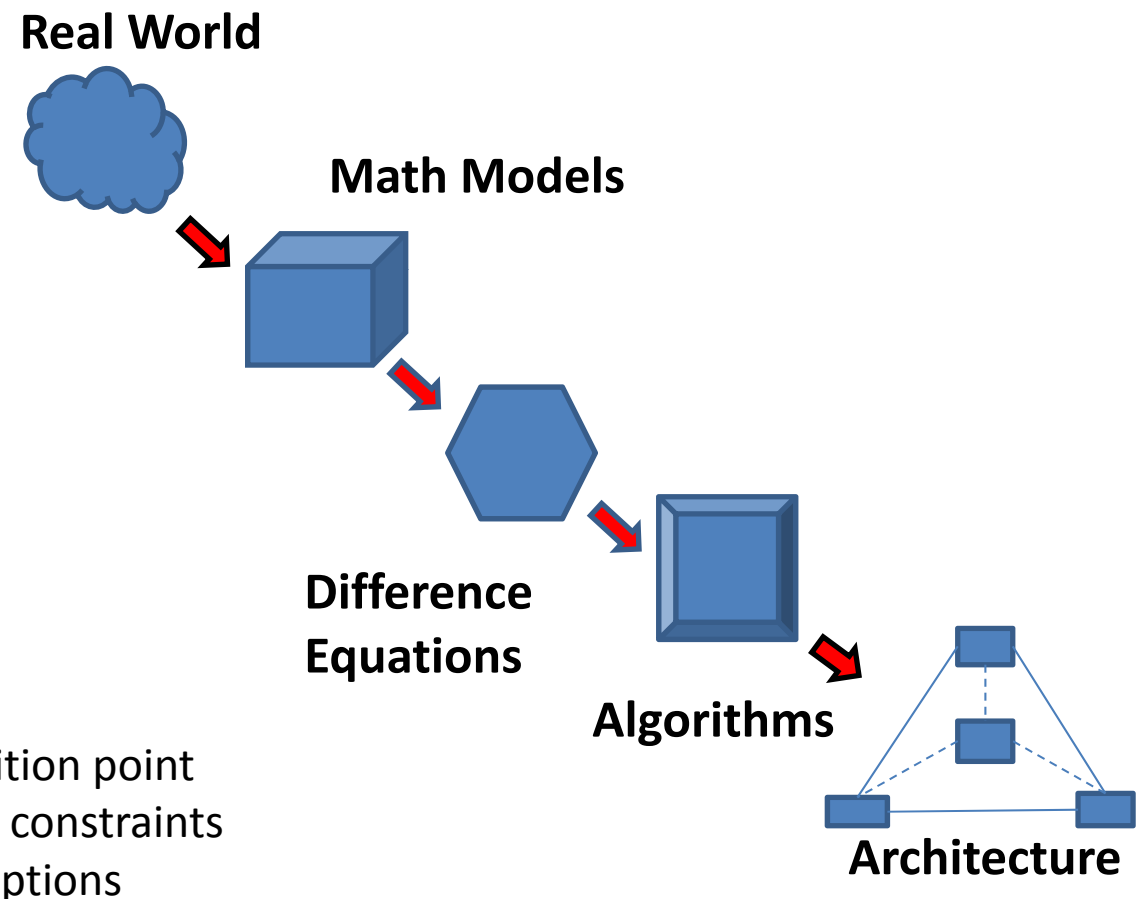
# Relating the Parts for Model Driven SOA



# Value derived from the architecture with MDA

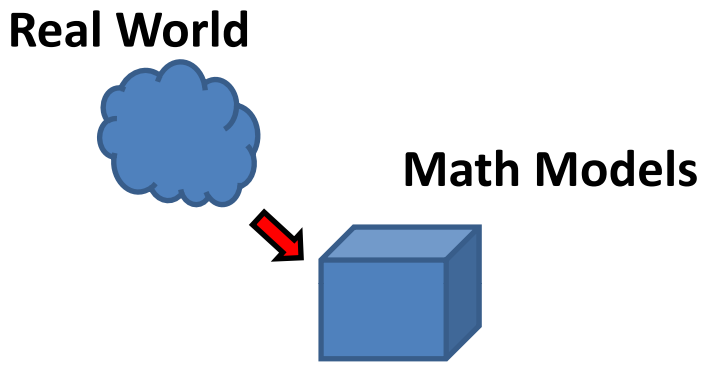


# System Modeling




Each transition point introduces constraints and assumptions

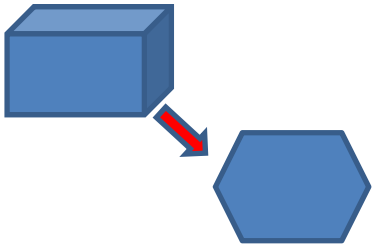
# System Modeling




| Sample Domains | Sample Constraint | Sample Assumption |
|---|---|---|
| Airborne | Non-Linear PDE/DE | Coordinate System selection |
| Medical | Conceptual Data Model | Handles Data & Images |
| Information Technology (IT) | Conceptual Data Model | Handles Data & Images |

# System Modeling

**Math Models**

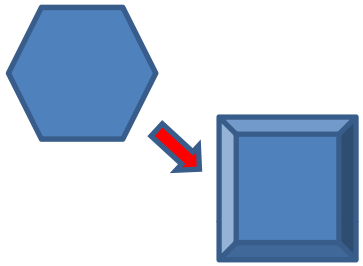


**Difference Equations**

| Sample Domains | Sample Constraint | Sample Assumption |
|---|---|---|
| Airborne | Non-Linear PDE/DE | Approximation Non-Linear System of Equations |
| Medical | Step not applicable | Step not applicable |
| Information Technology (IT) | Step not applicable | Step not applicable |

# System Modeling

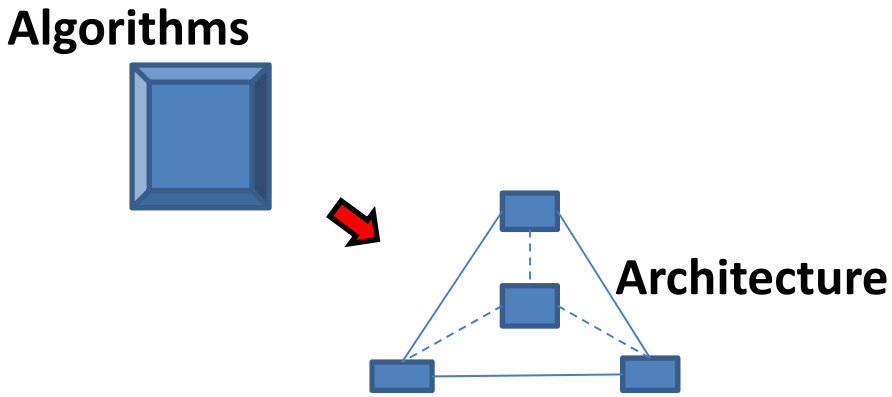
**Difference Equations**



**Algorithms**

| Sample Domains | Sample Constraint | Sample Assumption |
|---|---|---|
| Airborne | Non-Linear PDE/DE | Approximation Non-Linear System of Equations |
| Medical | Step not applicable | Step not applicable |
| Information Technology (IT) | Step not applicable | Step not applicable |

# System Modeling




| Sample Domains | Sample Constraint | Sample Assumption |
|---|---|---|
| Airborne | Decisions on Packaging | Level of Fidelity and error propagation rate |
| Medical | Logical Data Models | Data & Images representations |
| Information Technology (IT) | Decisions on Packaging | Acceptable response time |

# Architecture Element

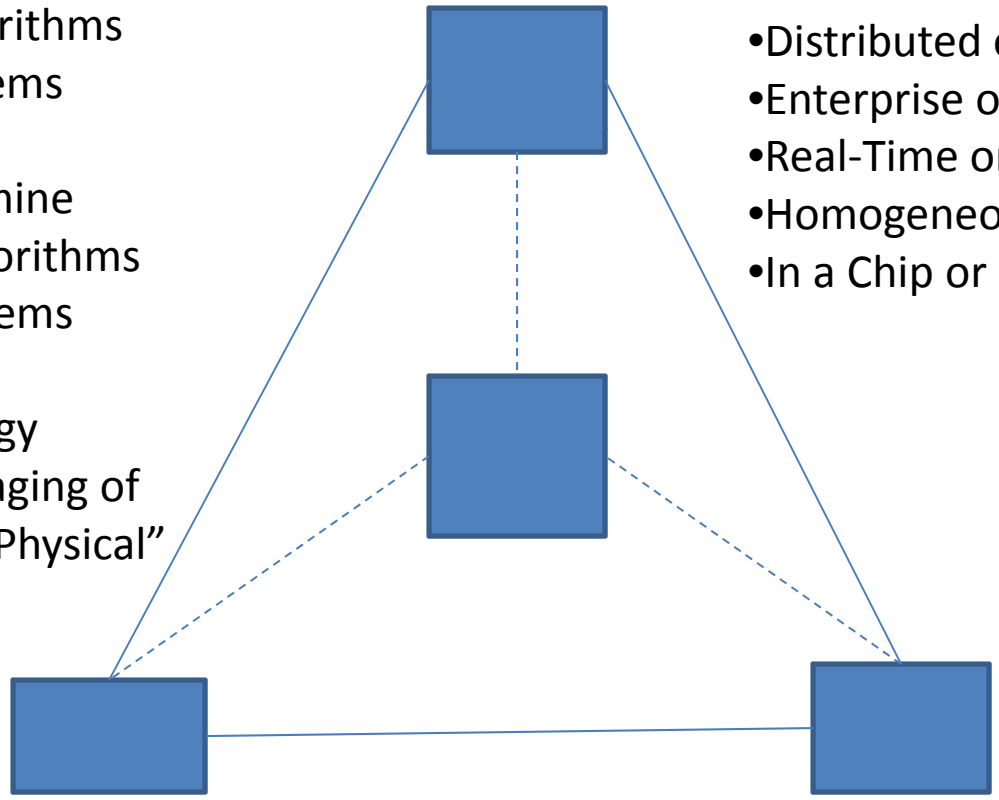
## Where the rubber meets the road: Partitioning

Partitioning driven by:

1. Packaging of Algorithms Into “Physical” Items

2. Interfaces determine packaging of Algorithms into “Physical” Items

3. Integration strategy determines packaging of Algorithms into “Physical” Items



Accommodates:

- Distributed or Centralized
- Enterprise or Embedded
- Real-Time or Non-Real-Time
- Homogeneous or Heterogeneous
- In a Chip or In the “Cloud”

# Architecture Element

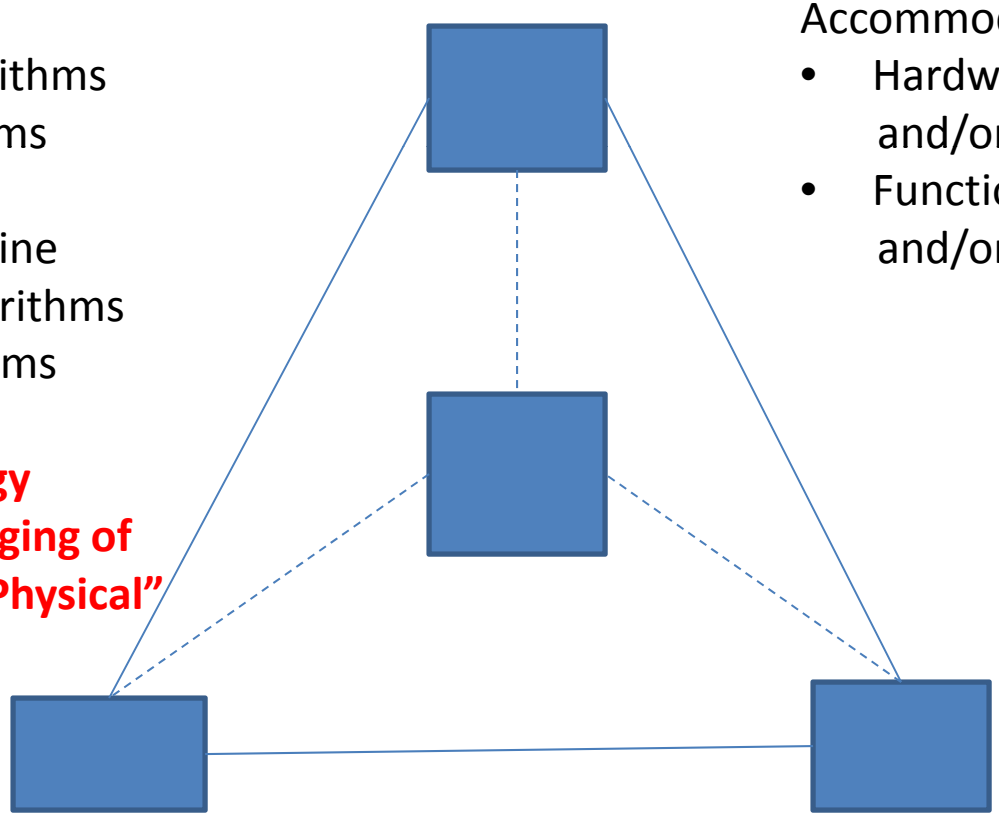
## Battle begins with Domain Allocation and Granularity

Partitioning driven by:

1. Packaging of Algorithms Into "Physical" Items

2. Interfaces determine packaging of Algorithms into "Physical" Items

3. **Integration strategy determines packaging of Algorithms into "Physical" Items**



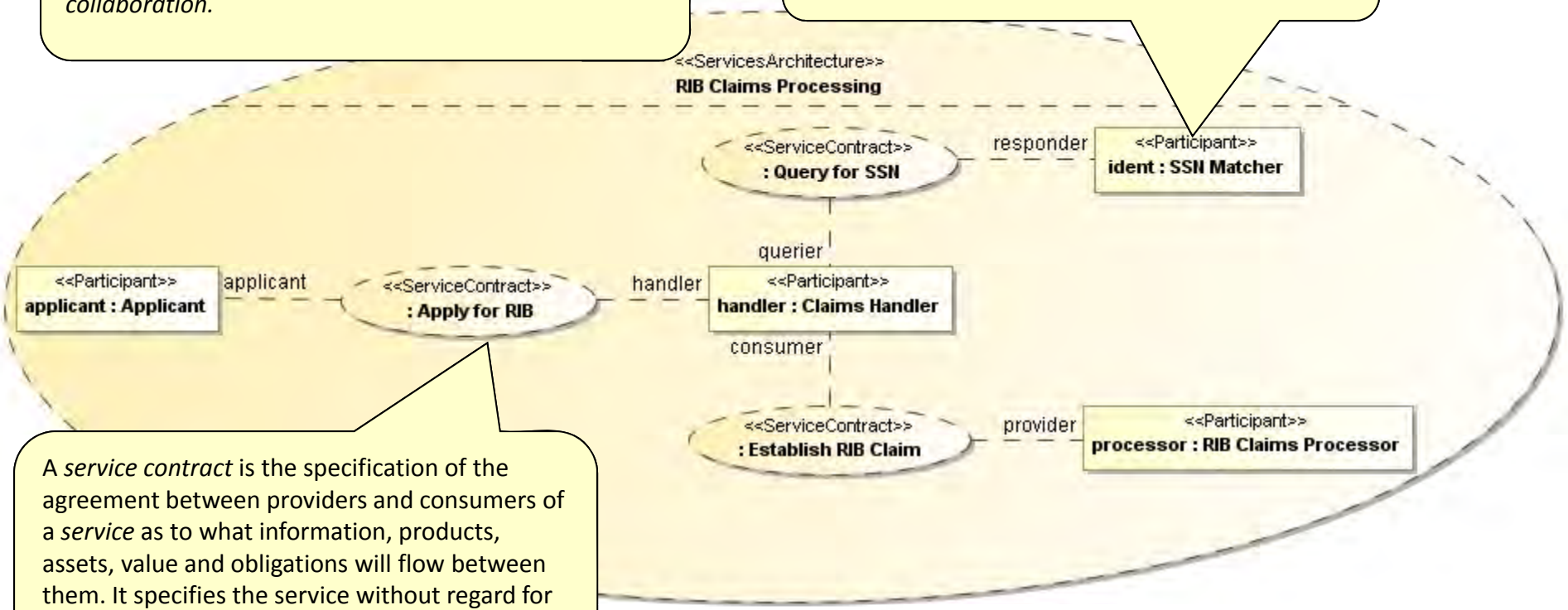
Accommodates:

- Hardware, Software and/or Firmware
- Functions, Objects and/or **"Services"**

# *Example Enterprise Level SOA*
# *Claims Processing* Services Architecture

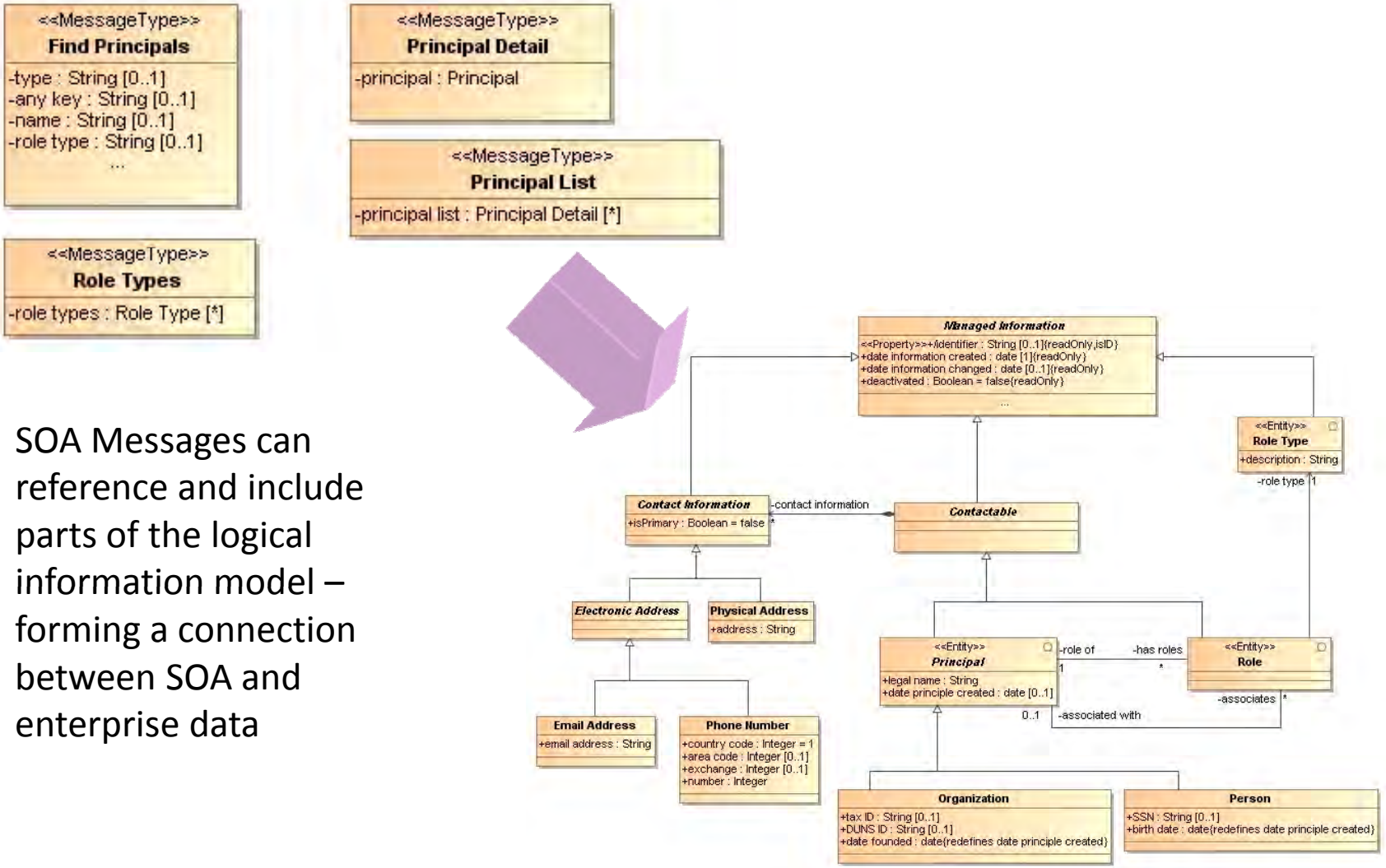
A *services architecture* describes how *participants* work together for a purpose by providing and using services expressed as *service contracts*. It is modeled as a UML *collaboration.*

A *participant* represents some party that provides and/or consumes services. Participants may represent people, organizations or systems.




A *service contract* is the specification of the agreement between providers and consumers of a *service* as to what information, products, assets, value and obligations will flow between them. It specifies the service without regard for realization, capabilities or implementation.

# Linking messages to business information




SOA Messages can reference and include parts of the logical information model – forming a connection between SOA and enterprise data

# Realizing the Model

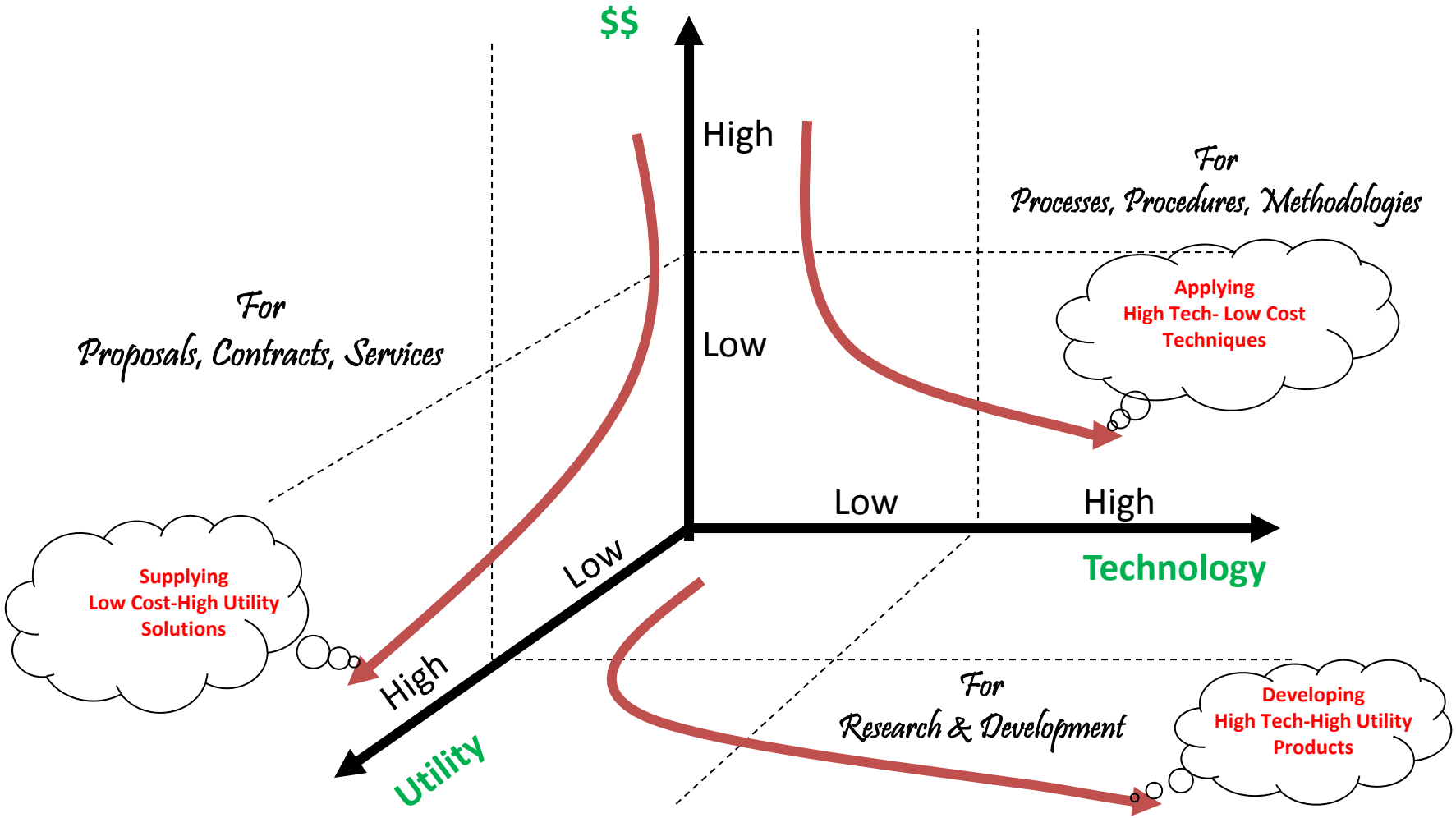
- How to we use I.T. to realize our processes and services?
  - Direct execution frameworks
    - The “no code” approach where the process and services execute directly from the model
    - May use other standards, such as BPEL
  - Wrapping and adapting existing capabilities
    - Automatic or manual creation of “adapter components” that use legacy systems, information or services to create the architected enterprise services
  - Creation of new application components and services
    - Build new capabilities by creating new components and creating composite applications
    - May be visual and declarative or code oriented
- Under the SoaML framework, all of these options can co-exist as a system of systems linked by services

# Intersection of System Modeling & SOA

- Both require an Integration Strategy
- Both require the equivalence of “services” at some level
- Both can accommodate commercially available frameworks

Issue to be solved is finding the appropriate granularity of “services” that allows us to “Construct” systems

# The Affordability Challenge



# Backup Information on System Modeling

- Architecture Element “X”
- Decomposition of Architecture Element “X”
- Establish Structural Model Framework
  - Import Mechanism
  - Control Mechanism
  - Export Mechanism
- Construct Test Scenarios and Capture Test Results

# Architecture Element

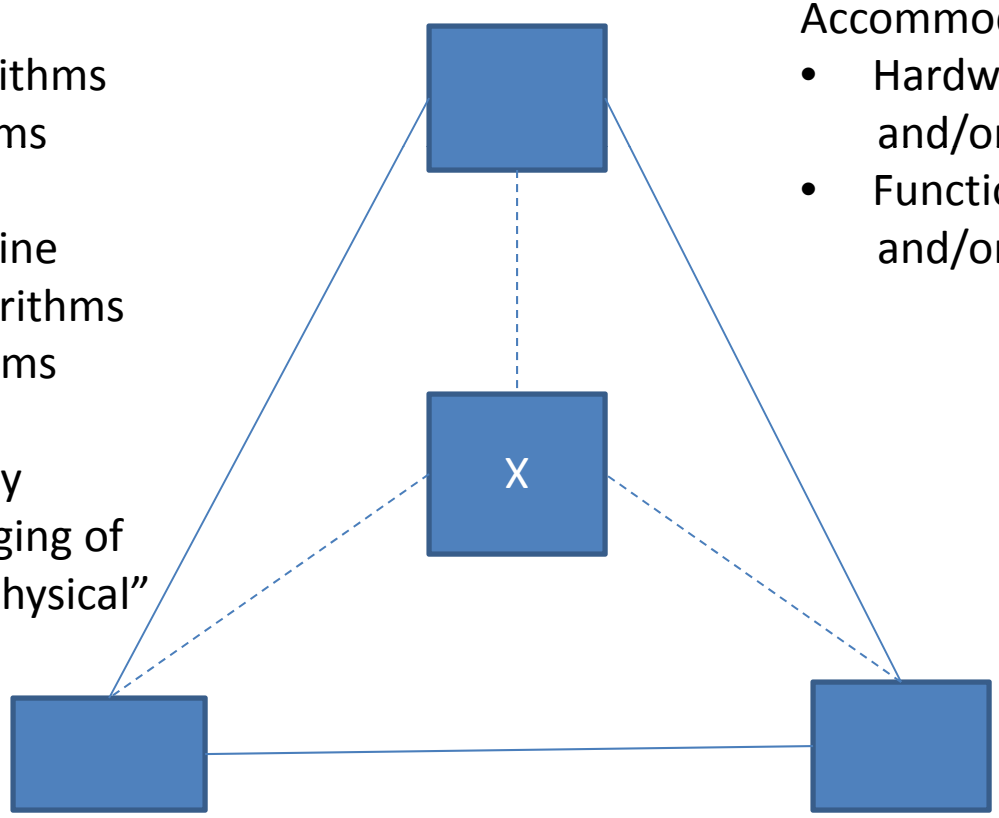
## Battle begins with Domain Allocation and Granularity

Partitioning driven by:

1. Packaging of Algorithms Into “Physical” Items

2. Interfaces determine packaging of Algorithms into “Physical” Items

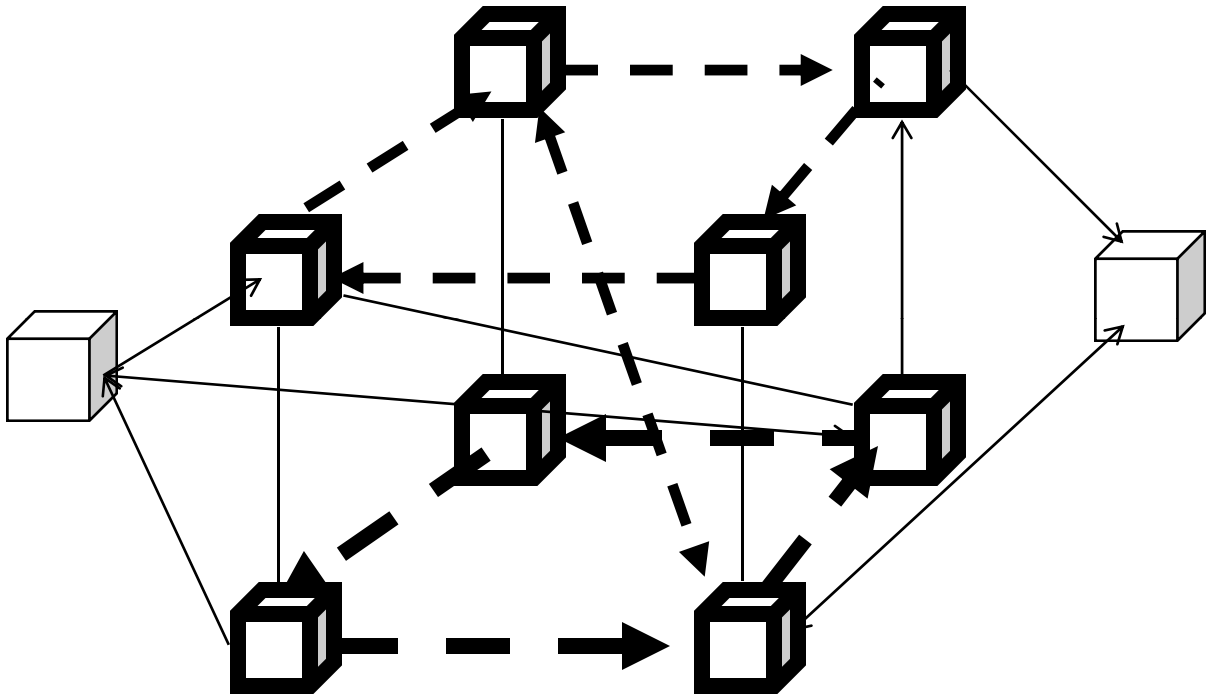
3. Integration strategy determines packaging of Algorithms into “Physical” Items




Accommodates:

- Hardware, Software and/or Firmware
- Functions, Objects and/or “Services”

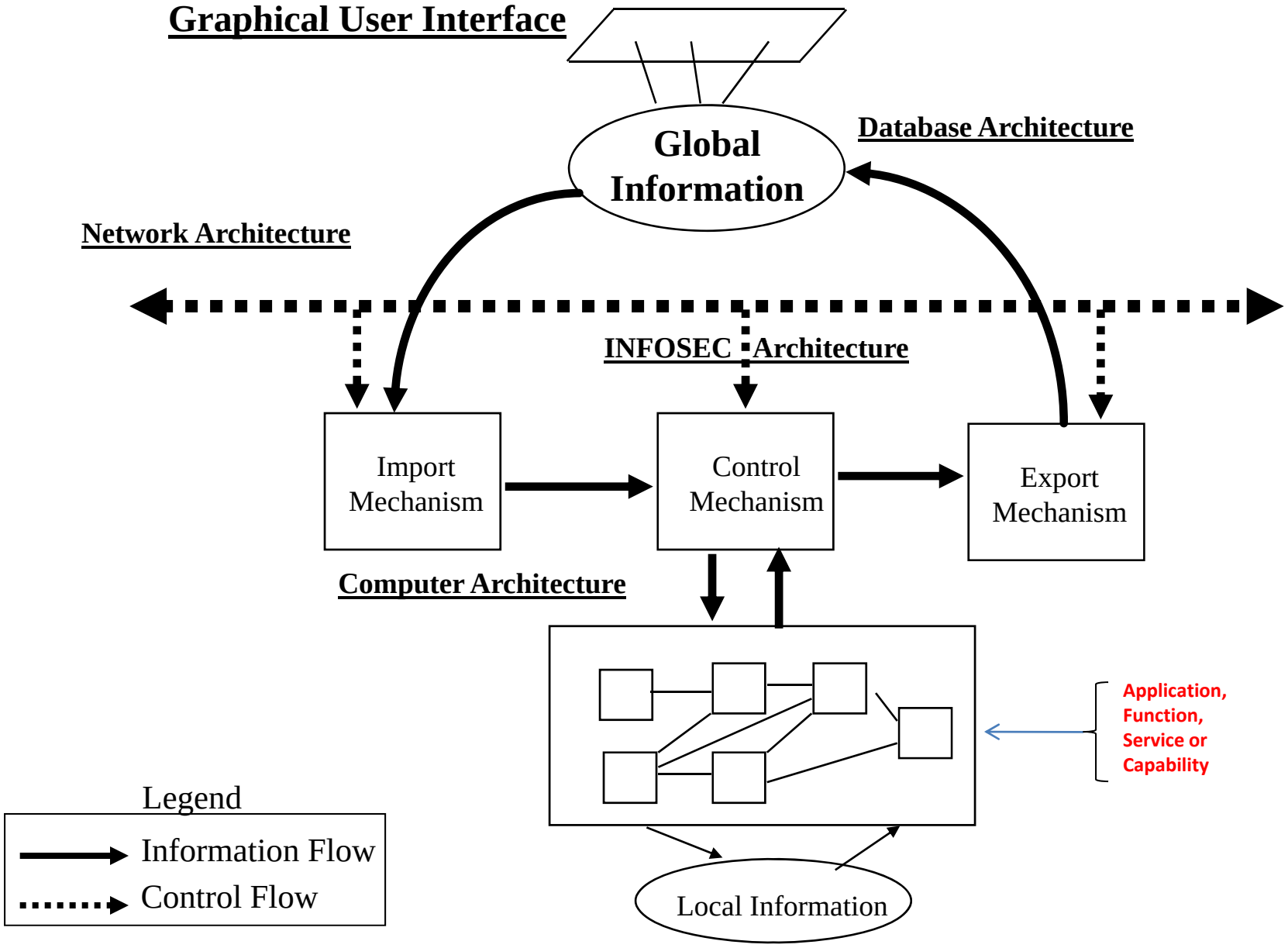
# Architecture Element “X” Decomposition



Control Flow

Information Flow

# Structural Model Framework



# Structural Model Framework