Model-based Approaches for Service Oriented Architectures

Mel Greer
Bob Epps
Report Documentation Page

Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

1. REPORT DATE  
MAY 2011

2. REPORT TYPE

3. DATES COVERED
00-00-2011 to 00-00-2011

4. TITLE AND SUBTITLE
Model-based Approaches for Service Oriented Architectures

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

6. AUTHOR(S)

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:
   a. REPORT unclassified
   b. ABSTRACT unclassified
   c. THIS PAGE unclassified

17. LIMITATION OF ABSTRACT
   Same as Report (SAR)

18. NUMBER OF PAGES 27

19a. NAME OF RESPONSIBLE PERSON

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18
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

Architects give models to developers

Architects create the models

Developers create artifacts based on the models (maybe)

Developers provide feedback to the architects (maybe)

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

Architects create the models

Using a standard-conforming UML modeling tool

Technologists specify the implementation platform

Architects validate the models by executing them in a simulated test environment

Using a standard-conforming UML execution tool

The models are provisioned as executing artifacts on the target platform

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.

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

System engineers create the models

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

**Business Concerns**

- Business Model
  - Enterprise Services (e-SOA)
  - Roles, Collaborations & Interactions
  - Process, Information & Rules

- Logical System Model
  - Technology Services (t-SOA), Components, BPM
  - Interfaces, Messages & Data

- Technology Specification
  - JMS, JEE, Web Services, .NET
  - WS*, BPEL, XML Schema

MDA Terms

Computation Independent Model

Platform Independent Model

Platform Specific Model

Refinement & Automation

Line-Of-Sight
Relating the Parts for Model Driven SOA

Open Source MDA Tooling

Modeling Provisioning Engine

Uses

Uses

SoaML Cartridge for JEE

Provisioning Profile

OMG SoaML UML Profile

Implements

Uses

Users SOA Model

Provisioning Model UML Tool

Manual Platform Application Artifacts

Automated Platform Application & IDE Artifacts

Platform & Tools (E.G. Eclipse/Netbeans/.NET)

Application Deploy
Value derived from the architecture with MDA

Business Concerns
- Business Model
  - Enterprise Services (e-SOA)
  - Roles, Collaborations & Interactions
  - Process & Information

Logical System Model
- Technology Services (t-SOA), Components, BPM
- Interfaces, Messages & Data

Technology Specification
- JMS, JEE, Web Services
- WS*, BPEL, XML Schema

Business Driven Technology
Facilitating Business Processes
System Modeling

Real World → Math Models → Difference Equations → Algorithms → Architecture

Each transition point introduces constraints and assumptions.
# System Modeling

![Diagram showing the relationship between real world and math models](Image)

## Sample Domains

<table>
<thead>
<tr>
<th>Sample Domains</th>
<th>Sample Constraint</th>
<th>Sample Assumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>Airborne</td>
<td>Non-Linear PDE/DE</td>
<td>Coordinate System selection</td>
</tr>
<tr>
<td>Medical</td>
<td>Conceptual Data Model</td>
<td>Handles Data &amp; Images</td>
</tr>
<tr>
<td>Information Technology (IT)</td>
<td>Conceptual Data Model</td>
<td>Handles Data &amp; Images</td>
</tr>
</tbody>
</table>
System Modeling

Math Models

Difference Equations

<table>
<thead>
<tr>
<th>Sample Domains</th>
<th>Sample Constraint</th>
<th>Sample Assumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>Airborne</td>
<td>Non-Linear PDE/DE</td>
<td>Approximation Non-Linear System of Equations</td>
</tr>
<tr>
<td>Medical</td>
<td>Step not applicable</td>
<td>Step not applicable</td>
</tr>
<tr>
<td>Information Technology (IT)</td>
<td>Step not applicable</td>
<td>Step not applicable</td>
</tr>
</tbody>
</table>
## System Modeling

### Difference Equations

### Algorithms

<table>
<thead>
<tr>
<th>Sample Domains</th>
<th>Sample Constraint</th>
<th>Sample Assumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>Airborne</td>
<td>Non-Linear PDE/DE</td>
<td>Approximation Non-Linear System of Equations</td>
</tr>
<tr>
<td>Medical</td>
<td>Step not applicable</td>
<td>Step not applicable</td>
</tr>
<tr>
<td>Information Technology (IT)</td>
<td>Step not applicable</td>
<td>Step not applicable</td>
</tr>
</tbody>
</table>
System Modeling

<table>
<thead>
<tr>
<th>Sample Domains</th>
<th>Sample Constraint</th>
<th>Sample Assumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>Airborne</td>
<td>Decisions on Packaging</td>
<td>Level of Fidelity and error propagation rate</td>
</tr>
<tr>
<td>Medical</td>
<td>Logical Data Models</td>
<td>Data &amp; Images representations</td>
</tr>
<tr>
<td>Information Technology (IT)</td>
<td>Decisions on Packaging</td>
<td>Acceptable response time</td>
</tr>
</tbody>
</table>
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 Algoritms 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”
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.
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

For Processes, Procedures, Methodologies

Applying High Tech - Low Cost Techniques

For Proposals, Contracts, Services

Supplying Low Cost-High Utility Solutions

For Research & Development

Developing High Tech-High Utility Products
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