Applying Enterprise Architecture to Model Based Systems Engineering

A pragmatic approach to modeling large systems
Project and system information is now **document centric**: Relies on textual, tabular, and graphic documents from a *variety of sources* which means…

*interpretation* of the information is difficult and *consistency* hard to maintain, especially under changing conditions.

**Model centric** approach relies far less on documents:

Depends instead on a *single repository* for storing project information and enforces *formal semantics* of its expression…

Promotes a more *common interpretation and consistency* of project information, even under very dynamic conditions.
Presentation objectives

- Provide conceptual understandings of **modeling** and **enterprise architecture**: as applied to **systems engineering** and **project management** of large efforts.

- Share our experiences in applying UPDM and SysML to components of NASA’s Constellation Program.

- Enable you to make informed decisions regarding the applicability of these approaches to your situations.
Some basic SysML semantics
Theoretical aspects of modeling and enterprise architectures
Additional SysML and UPDM semantics
Demonstration of a SysML tool and models
Q & A, and discussion
DEA Tool Functional Use Cases

- **Query DEA Database**
  - ID: 2.1.1
  - Text: "The tool shall provide users the ability to query the database subject to appropriate permissions."

- **Create, Update, Delete DEA's**
  - ID: 2.1.2
  - Text: "The tool shall provide users the ability to create, update and delete DEA's."

- **DEA Lifecycle**
  - ID: 2.1.3
  - Text: "The tool shall support a formal workflow lifecycle process as part of the update/change features."

- **DEA Electronic Signature**
  - ID: 2.1.4
  - Text: "The tool shall provide an electronic signature."

- **Print DEA**
  - ID: 2.2
  - Text: "The tool shall be able to print a DEA in RTF format."

- **Lists of DEA's**
  - ID: 2.5
  - Text: "The tool shall output DEA's in an Excel compatible format."

- **Maintain User Accounts**
  - ID: 3.1
  - Text: "The tool shall provide facilities for creating, updating and deleting user accounts."

- **Database Management**
  - ID: 3.2
  - Text: "The tool shall provide facilities for maintaining the database; for example, updating enumeration lists to reflect changing business requirements."

- **Import DEA**
  - ID: 3.3
  - Text: "The tool shall provide the ability to batch import DEA's from Excel and comma separated value formatted files."
## Conclusion

- **Why you don’t want to model**
  - Modeling is hard
  - Modeling tools are difficult
  - Modeling will likely require cultural changes

- **Why you do want to model**
  - It increases the rate of communications
  - It increases the precision of communications
  - It reduces tacit information
  - It promotes a common understanding of your project

- **Benefit:** Stakeholders having a common understanding of how the project is organized and its objectives will work together more effectively and make better decisions.
Relationship between systems engineering and project management

EA and Modeling: definitions in the context of UPDM/SysML

The relationship between EA, UPDM and SysML
While systems engineering and project management share many of the same qualities, there are important differences.
Similarities between PM & SE

- Lots of behaviors having complex relationships
- Lots of entities having complex relationships
- Many relationships and dependences between behaviors and entities.
 Systems engineering

- Systems engineering is a multidisciplinary approach for developing balanced systems solutions in response to diverse stakeholder needs.

- It includes the application of both management and technical processes to achieve balance and mitigate project risk.

- The management process is applied to ensure that development cost, schedule and technical performance objectives are met.

Source: A Practical Guide to SysML, copyright © 2008 by Elsevier, Inc. All rights reserved.
## Differences between PM & SE

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Project Management</th>
<th>Systems Engineering</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scheduling</td>
<td>Absolute times</td>
<td>Relative times</td>
</tr>
<tr>
<td>Economic cost</td>
<td>Mandatory/aggregation centric</td>
<td>Optional/unit centric</td>
</tr>
<tr>
<td>People</td>
<td>Individuals and roles</td>
<td>Role centric</td>
</tr>
<tr>
<td>Entities and behaviors</td>
<td>Notional</td>
<td>Precise</td>
</tr>
</tbody>
</table>
Project Management Models

Temporal Models

Gantt, PERT Charts

Structural Models

Informs

Behavioral Models

SysML

Informs
Being a successful PM or SE means:

- Understanding the project/system information and how it fits together.

- Communicating and otherwise making this information available to stakeholders in a timely, consistent fashion in a form relevant to their backgrounds and needs.
EA and Modeling: concepts in the context of UPDM/SysML

- Looking at modeling from SysML’s perspective
- Ideas behind enterprise architectures
SysML modeling is about communications

- It’s a **language** having syntax and structure...
- that uses a **medium**, primarily graphics...
- and has a **methodology**, which currently is largely undefined.
Three main attributes of languages

- **Abstractions** of the world around us
- Some form of **persistence**
- A **shared experience**: a producer and a consumer
Three main attributes of SysML

- Abstractions of systems
- Database persistence
- A shared experience
who’s the truest of us all:

- An electrical schematic of a radio
- An economic model
- A model student
- A non working model airplane
- A novel about present day life the author believes to be possible
- A description of a pencil
Abstraction levels

La Joconde

Femme au Chapeau Orné
Two criticisms of SysML...

- **Its old ideas warmed over** – Abstractions and persistence have been around at least since humans drew pictures on cave walls. **There is nothing new here.**

- **Its simply not practical** – Having my team think abstractly in the same way and put their information into a database in the same fashion is absurd. **It is not workable.**
Ideas Behind Enterprise Architectures

To Be Done By Ian?
An enterprise architecture describes the organizations (and their people), processes, inputs, products and systems that comprise an enterprise and the relations between each of these entities.

It can be used to understand the “as-is” state of the enterprise and plan the “to-be” vision of where the enterprise should be.
Enterprise Architecture Evolution

- Zachman (1987)
- Enterprise Architecture Planning (Spewak 1992)
- TOGAF (1995)
- FEA (1999)
- DoDAF 1.0 (2003)
- DoDAF 1.5 (2007)
- UPDM 1.0 (2008)
- DoDAF 2.0 (2009?)
Benefits of Enterprise Architecture

**Enterprise Architecture**

**«analytical»**
Simulate the architecture and produce artifacts used to execute enterprise processes

**«descriptive»**
Promote a common understanding of the enterprise

**«constraints»**
Determine critical resources, processes and data

**«gaps»**
Determine needs for resources, processes and data

**«objective»**
Optimize resource consumption and production
Benefits of Enterprise Architecture

- Simulate and Execute Processes
- Determine Critical Processes and Data
- Specify Information Systems Needs
- Promotes Common Understanding of Enterprise

Enterprise Architecture
Catalog of DoDAF views

Operations View

Systems View

Technical View
DoDAF Example of Models
OV-2 Operational Node Relationship Description


CoFR System → Review Minutes

Cx Information Element → Cx Software Systems

Color is used for visual enhancement and has no independent semantic meaning.
<table>
<thead>
<tr>
<th>#</th>
<th>Needline Id...</th>
<th>Operational E...</th>
<th>Operational Exchange Type</th>
<th>Operational Exchange Item Name</th>
<th>Sending Node</th>
<th>Receiving Node</th>
<th>Sending Operational Activity</th>
<th>Receiving Operational Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>Information Exchange</td>
<td>Review Package</td>
<td>CoFR System</td>
<td>Review Participants</td>
<td>Prepare Review Package</td>
<td>Analyze Package Data</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>Information Exchange</td>
<td>Review Minutes</td>
<td>Review Participants</td>
<td>Review Decision M...</td>
<td>Analyze Package Data</td>
<td>Make Review Decisions</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>Information Exchange</td>
<td>Review Minutes</td>
<td>Review Participants</td>
<td>CoFR System</td>
<td>Analyze Package Data</td>
<td>Disseminate Review Results</td>
<td></td>
</tr>
</tbody>
</table>
Views into model
# Ranges and Types of Modeling Tools

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Simple Drawing Tools</th>
<th>Enterprise Modeling Suites</th>
</tr>
</thead>
<tbody>
<tr>
<td>Most commonly purchased by</td>
<td>Individual</td>
<td>Department</td>
</tr>
<tr>
<td>Easy to learn and use</td>
<td>Yes¹</td>
<td>No</td>
</tr>
<tr>
<td>Semantic consistency imposed by tool?</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Data based repository for model elements</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Framework support</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Diagrams</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Matrices (tables and/or reports)</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Able to operationalize data</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Examples</td>
<td>PowerPoint, Visio, Gliffy, Canvas</td>
<td>Enterprise Architect, MagicDraw</td>
</tr>
<tr>
<td>Purchase cost</td>
<td>$100,s</td>
<td>$1000s</td>
</tr>
</tbody>
</table>

¹Yellow cells indicate the most prominent differentiating advantage for that class of software - From Enterprise Architecture Presentation by Lisa Murphy, Ian Sturken, Andy Schain
SysML and other modeling constructs

Abstraction

More

Less

Concepts

Enterprise architectures

SysML

Domain specific languages

Examples

DoDAF, MoDAF, UPDM, FEA

SysML

UML, Modelica, Simulink, MARTE
Additional SysML and UPDM semantics

- SysML structure and semantics
- UPDM structure and semantics
SysML may be considered in terms of its diagrams, reports and grammar.
SysML Diagram Taxonomy

Source: OMG Specification
Model elements

- Models consist of elements
- Elements must have unique names, within a namespace
- All model elements must reside in (be owned by) one package
Major types of model elements

- Structural
- Behavioral
Packages

- Models are composed of one or more packages
- Packages can contain other packages and/or any collection of model elements
- Package hierarchies define the name space
Packages shown as a containment tree

FIGURE 5.4
Browser view of the model’s package hierarchy.

Source: A Practical Guide to SysML, copyright © 2008 by Elsevier, Inc. All rights reserved.
Packages shown graphically

FIGURE 5.1
An example package diagram.

Source: A Practical Guide to SysML, copyright © 2008 by Elsevier, Inc. All rights reserved.
Dependences and stereotypes

FIGURE 5.10
Example of dependencies in the camera performance view.

Source: A Practical Guide to SysML, copyright © 2008 by Elsevier, Inc. All rights reserved.
Blocks

- Blocks represent structural elements
  - Organizations
  - Data
  - People
  - Airplanes

- Blocks have two main attributes
  - Properties, other structural elements
  - Behavior, either as an intrinsic capability or through behavioral elements
Block property types

- Parts (in UML, composition)
- References (in UML, aggregation)
- Associations
Block definition diagram—Part

The black diamond is used for communicating a Part Property relationship.
No adornment is used for communicating an Association Property relationship.
The white diamond is used for communicating a Reference Property relationship.

FIGURE 6.10
A reference association on a block definition diagram.

Source: A Practical Guide to SysML, copyright © 2008 by Elsevier, Inc. All rights reserved.
Block compartments

**Figure 6.4**
An automobile with four wheels described as separate parts.

Source: A Practical Guide to SysML, copyright © 2008 by Elsevier, Inc. All rights reserved.
Blocks and inheritance

FIGURE 6.35
Example of block specialization.

Source: A Practical Guide to SysML, copyright © 2008 by Elsevier, Inc. All rights reserved.
Activity Diagrams
Sample activity diagram 2

FIGURE 8.1
An example activity diagram.

Source: A Practical Guide to SysML, copyright © 2008 by Elsevier, Inc. All rights reserved.
- Fully and unambiguously specify what the model is to do (functional requirements) and the context in which it is to operate (non-functional requirements).
- Provide concise and unambiguous information showing how requirements relate to each other.
- Provide concise and unambiguous information showing how requirements relate to other model elements—the project lifecycle.
Requirements diagram—Example

Source: A Practical Guide to SysML, copyright © 2008 by Elsevier, Inc. All rights reserved.
Requirements dependences

- «refine»
- «satisfy»
- «deriveReqt»
- «copy»
- «verify»
- «trace»
Refine, satisfy & derive Reqt

Figure 16.4 - Links between requirements and design

Source: OMG Specification
Figure 16.6 - Use of the copy dependency to facilitate reuse
Verify

Source: A Practical Guide to SysML, copyright © 2008 by Elsevier, Inc. All rights reserved.
Trace

**req** [Package] Customer Specification [trace example]

```
<requirement>
Operating Environment

id = "S1"
text = "The system shall be capable of detecting intruders 24 hours per day, 7 days per week, under all weather conditions."
```

```
<document>
Market Survey
```

Source: A Practical Guide to SysML, copyright © 2008 by Elsevier, Inc. All rights reserved.
Use case scope

-use case scope diagram with axes:
  - High Level Business Objectives
  - Developers describe general system capabilities
  - Management articulates overall business objectives
  - Developers imagine what/how users will want to do to get their work done
  - Users specify what they need to do to get their work done that day
  - Development managers describe to developers how the work will be implemented
  - Users describe to developers how the work is to be implemented
  - Implementation Details

- Use Case Origin
  - Developer Community
  - User Community
Use cases and actors

- Actors are external entities that interact with your system via use cases.
- Actors can be people, computer systems or really any device that interacts with your system.
- It is highly recommended that Actors be modeled before use cases.
Actors for a household model
Use cases for household food processes
SysML requirements relationships

SysML Requirements Dependency Relationships

- Any behavioral diagram
- <<refine>>
- <<verify>>
- <<allocate>>
- <<satisfy>>
- <<trace>>
- Use Case
- <<requirement>>
- External Information
- <<block>>
- Any entity executing the activity: person, org, machine, software, etc.
UPDM Structure And Semantics

To be done by Ian?
UPDM Overview
OV2 Elements

- Operational Nodes
- Needline
- Information Element
OV3 Elements

- Operational Activity
- Needline
- Item Name
- Operational Node
OV4 Elements
OV5 Elements
OV7 Elements
Use Cases?
Requirements?
Key Features
Conclusion
A Historical Perspective, video clip

End of Part I
Key SysML features & capabilities

- Open standard
- Works equally well with structural and behavioral artifacts
- Treats requirements as first-class modeling elements
- Future plans include simulations
- Formal mechanism for extending SysML semantics
- Formal set of semantically consistent graphical elements.
Conclusion

- **Why you don’t want to model**
  - Modeling is hard
  - Modeling tools are difficult
  - Modeling will likely require cultural changes

- **Why you do want to model**
  - It increases the rate of communications
  - It increases the precision of communications
  - It reduces tacit information
  - It promotes a common understanding of your project

- **Benefit:** Stakeholders having a common understanding of how the project is organized and its objectives will work together more effectively and make better decisions.
A historical perspective
Demonstration

- Ares
- Mission Information System (~CoFR)
- FireSat
Model Organization
# Constellation Program Model

## Table of Contents

### Structural Artifacts

<table>
<thead>
<tr>
<th>Organization Information</th>
<th>Data Taxonomy</th>
<th>Information Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>NASA Organization</td>
<td>High Level Structure</td>
<td>Tools</td>
</tr>
<tr>
<td>Constellation Organization</td>
<td>Operational Data</td>
<td>Tool Org and Functions</td>
</tr>
<tr>
<td>SE&amp;I</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PP &amp;C</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Behavioral Artifacts

#### Virtual Missions and Operations

- GMIP Category Temporal Decomposition
- GMIP Category Information Decomposition
- Design Category Decomposition
- GMIP
  - EVA H/W Processing & Crew Training
  - Integrated Vehicle Engineering Analysis:
  - MPPF
  - Offline Processing
  - CM/SM Stack Integration
  - VAB Integrated Operations

83
Integrated Vehicle Engineering Analysis

![Diagram showing the flow of activities in Integrated Vehicle Engineering Analysis]

- Loads & Dynamics Integrated Analysis
- Aero Thermal Integrated Analysis
- EMC/RF Integrated Analysis
- ECLSS Integrated Analysis

Duration for these concurrent activities is L-210 days to L-120 days
Constellation Data Taxonomy

Constellation Data Taxonomy

```
<<cxData>>
Document
  <<cxData>>
  Requirement Document
dataType = "document"
  <<cxData>>
  Procedure
dataType = "document"

<<cxData>>
Model

<<cxData>>
Process Record
  <<cxData>>
  Data Exchange Agreement
dataType = "processRecord"
  <<cxData>>
  Change Request
dataType = "processRecord"
```
MIS Demonstration
FireSat Demonstration
Backup
Relationship between UML and SysML

Source: OMG Specification
UML/SysML metamodel

Figure 7.8 - An example of the four-layer metamodel hierarchy

Source: OMG Specification
OMG Source material is reprinted with permission. Object Management Group, Inc. (C) OMG. 1997 - 2006.