ENSE 622: System Requirements, Design and TradeOff

Model-Based Systems Engineering

Mark Austin

E-mail: austin@isr.umd.edu

Institute for Systems Research, University of Maryland, College Park
Topics:

1. Goals for Model-Based Systems Engineering
2. Basic System Concepts (e.g., definition, emergent properties).
3. System Structure, System Behavior, System Complexity
4. System Interfaces.
5. Strategies for Studying a System
9. Ontologies and Rules
10. Multiple Viewpoints
Model-Based Systems Engineering

Goals

Model-based systems engineering (MBSE) development is an approach to systems-level development in which

... the focus and primary artifacts of development are models as opposed to documents.

Approach and Benefits

MBSE procedures provide a formal basis for:

- Closing the gap between what is needed and how the system will work, and in particular, the development of virtual prototypes.
- Assisting in the management of complex systems.
- Early and formal approaches to system validation and verification.
Approach and Benefits

Virtual prototypes provide a means for ...

... clarifying what an engineering system needs to provide and how that purpose might be accomplished.

When models are based on rigorous mathematical formalisms,

... algorithms can be developed to detect errors and omissions in design before the project resources are committed to a full implementation.
Model-based systems engineering process at Vitech
Model-Based Systems Engineering

Multi-level approach model-based systems engineering.

Orchestration of Good Design Solutions

- Semi-Formal Models; Formal Models; Abstraction; Decomposition.
Model-Based Systems Engineering

INCOSE’s Roadmap of MBSE Capability 2007 through 2025...

DRAFT
June 24, 2007
Basic System Concepts
Definition of a System

For our purposes, a system is:

... a collection of components (some of which can be modules and sub-systems) that are interconnected so that the system can perform a function which cannot be performed by the components alone.

Systems may consist of products, people and processes.

Elements of a System
Basic System Concepts

Key points:

1. A boundary separates the system from its external environment (e.g., walls in a building; starting and finishing times for a numerical analysis).

2. Inputs are elements that enter the system (e.g., raw materials entering a manufacturing plant).

3. Outputs are the finished products and consequences of being in the system. New cars leaving a car assembly plant is an example of finished products. An example of “consequence of being” is the ability of a highway bridge system to carry traffic.

4. System threats are those things that can potentially affect acceptability of the system configuration – for example, a lack of knowledge, insufficient time to build, lack of finance etc...
Dichotomies of System Classification

• **Artificial versus Natural**
  Artificial systems are man-made.
  Natural systems are not.

• **Physical versus Conceptual**
  Physical systems operate on matter (or from matter) in the physical environment.
  Conceptual systems exist abstractly as ideas, plans, or information.

• **Open versus Closed**
  Open systems interact with the surrounding environment through a boundary.
  Closed systems do not.
Emergence and Emergent Properties

Emergence is the way in which complex systems and patterns arise out of a multiplicity of relatively simple interactions.

Two examples from nature: (1) parallel lines in sand caused by water and wind; (2) axes of symmetry in crabs, butterflies and bugs.
From a systems engineering standpoint ...

... we would like to engineer emergence to achieve pre-specified emergent behavior.

**Example 1.** Emergent Properties in Architecture.

**Example 2.** Emergent Properties in Bridge Engineering.

Typical: Aesthetics, load carrying capacity, physical symmetries, resistance to aeroelastic flutter.
Warning. Failure to understand emergent properties can be catastrophic!

Tacoma Narrows Bridge
One difficulty in predicting the emergent behavior of systems is ...

... knowing where their boundary is within their environment.

Consider, for example, trying to define the boundary of the US economy, the boundary of an individual, the boundary of the Universe?

Source: Johnson, 2006.
Hierarchy Structure

A hierarchy is ...

... an arrangement of items in which the items are represented as being above, below, or at the same level as one another.

Example
Benefits of the Hierarchy Structure

For designers the hierarchy structure is a powerful abstraction mechanism ...

- The hierarchy viewpoint enables a designer to visualize an entire related aspect of the system without the confusing detail of subparts and without the unrelated and distracted generality of super-parts.

- By reducing the distracting detail to a single object that is lower in the hierarchy, one can greatly simplify many system development operations.
  For example, simulation, verification, design-rule checking, and layout constraints can all benefit from hierarchical representation, which makes them much more computationally tractable.
Layered Structure

A layered system is ...

... one where the hierarchy of system components is clustered into horizontal strata.

**Example 1.** Open systems interconnection (OSI) model for computer communications.

<table>
<thead>
<tr>
<th>Layer</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Application</td>
</tr>
<tr>
<td>6</td>
<td>Presentation</td>
</tr>
<tr>
<td>5</td>
<td>Session</td>
</tr>
<tr>
<td>4</td>
<td>Transport</td>
</tr>
<tr>
<td>3</td>
<td>Network</td>
</tr>
<tr>
<td>2</td>
<td>Data Link</td>
</tr>
<tr>
<td>1</td>
<td>Physical</td>
</tr>
</tbody>
</table>

Open Systems Interconnection Model for computer communications.
Example 2. Layered organization of multi-dimensional attributes in spatial data.
Network Structure

A network is a ... 

... set of elements (or modules or nodes or devices) that are connected by a set of interfaces (or links or communication channels). Formally, a network is a graph.

The modules may be computers, mechanical machines, etc...

The interfaces may use a variety of communications media.

Example 1. Interacting subsystems in an aircraft.
Example 2. The behavior of many man-made and natural systems can be modeled as networks having cyclic behavior, e.g., the water cycle.
Network Topology

A network topology describes the connectivity (or arrangement) of nodes on a network.

Common network topologies include star, ring, line, bus, and tree configurations:
Network science seeks to discover the common principles, algorithms, and tools that govern network behavior across a wide range of domains.

**Fundamental questions about networks:**

- How big is the network?
- How many hops does it take for a random node A to be connected to node B?
- What is the shortest distance (in terms of edges or cost) from node A to node B?
- From a design standpoint, what are the pros/cons of each network structure?

**More interesting questions:**

- What does nature do? Why?
- What kinds of relationships exist between real-world networks?
- How vulnerable are networks to attack? And how does this change with network structure?
Real-World Networks
Real-World examples of network connectivity

**Right.** Nodal connectivities in four different real-world networks: (a) the Internet; (b) social networking; (c) a random graph; (d) track configuration in a metro system.

**Left.** Connectivity in a typical scale-free network (e.g., air transportation networks).
Basic Questions about Networks

Observation

Some of these questions can be answered through mathematical analysis.

Example: Shortest Path Analysis in an Air Transportation Network

![Diagram of an air transportation network with cities like LAX, JFK, MCO, ORD, ATM, and shortest path examples]

Examples of shortest paths in a graph:

<table>
<thead>
<tr>
<th>source</th>
<th>destination</th>
<th>distance</th>
<th>shortest paths</th>
</tr>
</thead>
<tbody>
<tr>
<td>JFK</td>
<td>LAX</td>
<td>3</td>
<td>JFK-ORD-PHX-LAX</td>
</tr>
<tr>
<td>LAS</td>
<td>MCO</td>
<td>4</td>
<td>LAS-PHX-DFW-HOU-MCO and four others</td>
</tr>
<tr>
<td>HOU</td>
<td>JFK</td>
<td>2</td>
<td>HOU-ATL-JFK and two others</td>
</tr>
</tbody>
</table>
System Structure

Randomness in Connectivity Relationships

![Diagram of connectivity relationships ranging from 0 to 1 in randomness]
Power Law in Scale-Free Networks

The probability that a chosen node has exactly $k$ links follows:

$$P(k) \sim k^\gamma,$$

where $\gamma$ is the degree exponent, with its value for most networks being between 2 and 3.
Appeal of Scale-Free Networks

Scale-free networks were discovered to have a ...  

... **high robustness against random failures and random errors.**

They are, however, ...

... **vulnerable to informed attack.**

**Scale-Free: Yeast Protein Interaction Network**

Networks of Networks

Many large scale systems are intertwined networks of networks.

Understanding the relationships among the networks and their combined behaviors can be very challenging.

Example 1. Buildings have intertwined network structures for:

- The arrangement of spaces,
- Fixed circulatory systems (power, hvac, plumbing), and
- Dynamic circulatory systems (flows of energy through rooms; flows of material).
Example 2. Cascading failure of networks caused by earthquakes.

Christchurch, New Zealand, 4.30 am, September 4, 2010. A magnitude 7.2 earthquake rolls into town ....

20% of homes are uninhabitable. Many transportation links are damaged. Street flooding in low-lying areas → Widespread power outages → Disruption of many services.
Planning for disaster relief needs to look at the connections between network models.

Basic questions:

- What kinds of dependencies exist between the networks?
- How will a failure in one network impact other networks?
- What parts of a system are most vulnerable?
- Does it make sense to stockpile supplies of water and food?
- How much should we spend to prepare for an inevitable attack?
Definition. A transformational system is...

... a process that receives one or more system inputs $I$ from an external environment, transforms them with process $T$, and then releases them as system outputs $O$ to an external environment.

A transformational system generates an output and then terminates.
**Classification:** Single Input/Single Output (SISO)

Input \[\rightarrow\] Transformational Process \[\rightarrow\] Output

**Classification:** Multiple Input/Multiple Output (MIMO)

Input 1 \[\rightarrow\] Transformational Process \[\rightarrow\] Output 1
Input 2 \[\rightarrow\] Transformational Process \[\rightarrow\] Output 2
A reactive system is ...

... a system that, when turned on, is able to create desired effects in its environment by enabling, enforcing, or preventing events in the environment.

Reactive systems are involved in a continuous interaction with the environment. The environment:

... generates input events at discrete intervals through one or more interfaces and the system reacts by changing its state and possibly generating output events.
Typical Classifications

Many reactive systems are:

- **Real-time systems**
  A real-time system is a system in which the correctness of a response depends on the logical correctness and time at which the response is produced.

- **Safety-critical**
  Malfunctioning of the system could lead to a loss of life or property.

- **Embedded systems**
  Software to support a real-time system is often embedded within the system hardware.

- **Control systems**
  Control systems enforce a desirable behavior on their environment.
Reactive Systems

Key Characteristics

- They have behavior defined by continuous, non-terminating, interaction with the surrounding environment.
- If the system terminates during its availability time, then usually this is considered a failure.
- Reactive systems are required to respond to external stimuli as and when they occur. Therefore, reactive systems must be able to respond to interrupts, even when they are doing something else.
- It follows that behavior of a reactive system is often defined by a set of interacting processes that operate in parallel.
- Often, reactive systems will need to operate in real time, and be subject to real-time constraints.
### Side-by-Side Comparison (Adapted from Wieringa, 2003)

<table>
<thead>
<tr>
<th>Transformational System</th>
<th>Reactive System</th>
</tr>
</thead>
<tbody>
<tr>
<td>May interact to capture more data.</td>
<td>Highly interactive.</td>
</tr>
<tr>
<td>Terminating process.</td>
<td>Non-terminating process.</td>
</tr>
<tr>
<td>Non-interrupt driven.</td>
<td>Interrupt driven.</td>
</tr>
<tr>
<td>Output not state dependent.</td>
<td>State-dependent response.</td>
</tr>
<tr>
<td>Output not defined in terms of the environment.</td>
<td>Environment-oriented response.</td>
</tr>
<tr>
<td>Sequential process.</td>
<td>Parallel processes.</td>
</tr>
<tr>
<td>Usually, no stringent time requirements.</td>
<td>Usually, stringent time requirements.</td>
</tr>
</tbody>
</table>
System Complexity
Measures of System Complexity

Definition

Complexity is ...

... a measure of the intricate intertwining or interconnectivity or elements within a system and between a system and its external environment.

Factors that Increase System Complexity

The complexity of human and engineering systems increases with ...

- The degree of connectivity among the elements (in a highly connected system, a decisions or action made by one element will affect many other elements in the system),

- The extent to which its properties evolve over time, and

- The extent to which the system behavior is governed by distributed control.
System Complexity and Systems Engineering

Within the systems engineering profession it is now generally agreed that ...

... increasing complexity is at the heart of the most difficult problems facing the architecting and engineering of modern systems.

Sources of complexity in engineered systems:

- Advances in technology make new types of engineering systems possible.
- People’s expectations have a tendency to expand over time.

An inability to deal with these complexities in a correct and time-efficient manner the biggest cause of cost-overruns in projects.
## Classification Metrics for Simple and Complex Systems

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Simple Systems</th>
<th>Complex Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of system elements.</td>
<td>Small</td>
<td>Large</td>
</tr>
<tr>
<td>Attributes of elements.</td>
<td>Predetermined</td>
<td>Not predetermined</td>
</tr>
<tr>
<td>Interaction between elements.</td>
<td>Highly organized</td>
<td>Loosely Organized</td>
</tr>
<tr>
<td>Behavior</td>
<td>Governed by well-defined laws.</td>
<td>Probabilistic</td>
</tr>
<tr>
<td>Evolution</td>
<td>Does not evolve</td>
<td>Evolves over time</td>
</tr>
<tr>
<td>Nature of sub-systems.</td>
<td>Do not pursue their own goal.</td>
<td>Are purposeful and generate their own goals.</td>
</tr>
<tr>
<td>Interaction with environment</td>
<td>Very little or none.</td>
<td>Interacts strongly</td>
</tr>
</tbody>
</table>
Implications of System Complexity on System Design and Management

- Long-term planning is impossible
- Dramatic change can occur unexpectedly
- Complex systems often exhibit patterns and short-term predictability
- Organizations can be tuned to be more innovative and adaptive.

Strategy

Rather than expend large amounts of resources on forecasting for unpredictable futures, it makes sense that ...

... systems be designed and managed to emphasize flexibility in response to the vagaries of surrounding environments.
Network-Theoretic Measures of Complexity

Measures of complexity for networks are associated with the degree of order in a network.

Complex networks are the backbone of complex systems...

- Every complex system is a network of interaction among numerous smaller elements
- Some networks are geometric or regular in 2-D or 3-D space
- Other networks contain long-range connections or are not spatial at all
- Understanding a complex system = break down into parts + reassemble
Diversity of Network Nodes. Node States and Dynamics

<table>
<thead>
<tr>
<th>Network</th>
<th>Node diversity</th>
<th>Node state/dynamics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internet</td>
<td>routers, PCs, switches</td>
<td>routing state/algorithm</td>
</tr>
<tr>
<td>brain</td>
<td>sensory, inter, motor neuron</td>
<td>electrical potentials</td>
</tr>
<tr>
<td>WWW</td>
<td>commercial, educational</td>
<td>popularity, num. of visits</td>
</tr>
<tr>
<td>Hollywood</td>
<td>traits, talent</td>
<td>celebrity level, contracts</td>
</tr>
<tr>
<td>gene regulation</td>
<td>protein type, DNA sites</td>
<td>boundness, concentration</td>
</tr>
<tr>
<td>ecology web</td>
<td>species traits (diet, reprod.)</td>
<td>fitness, density</td>
</tr>
</tbody>
</table>

*nodes can be of different subtypes: *, , *

*nodes have variable states of activity: *, *, *, *, *
### Measures of System Complexity

#### Diversity of Network Edges, Edge States and Dynamics

<table>
<thead>
<tr>
<th>Network</th>
<th>Edge diversity</th>
<th>Edge state/dynamics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internet</td>
<td>bandwidth (DSL, cable)...</td>
<td>--</td>
</tr>
<tr>
<td>brain</td>
<td>excit., inhib. synapses ...</td>
<td>synap. weight, learning</td>
</tr>
<tr>
<td>WWW</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>Hollywood</td>
<td>theater movie, TV series ...</td>
<td>partnerships</td>
</tr>
<tr>
<td>gene regulation</td>
<td>enhancing, blocking ...</td>
<td>mutations, evolution</td>
</tr>
<tr>
<td>ecology web</td>
<td>predation, cooperation</td>
<td>evolution, selection</td>
</tr>
</tbody>
</table>

- edges can be of different subtypes: /, /, /...
- edges can also have variable weights: /, /, /, /
Measure of Network Complexity: Average Path Length

The path length between two nodes $A$ and $B$ is the smallest number of edges connecting them:

$$l(A, B) = \min l(A, A_1, \ldots, A_n, B)$$

The average path length of a network over all pairs of $N$ nodes is

$$L = \langle l(A, B) \rangle$$

$$= \frac{2}{N(N-1)} \sum_{A,B} l(A, B)$$

The network diameter is the maximal path length between two nodes:

$$D = \max l(A, B)$$

Property: $1 \leq L \leq D \leq N-1$
Measure of Network Complexity: Connectivity

- The degree of a node $A$ is the number of its connections (or neighbors), $k_A$.
- The average degree of a network is
  $\langle k \rangle = \frac{1}{N} \sum_A k_A$
- The degree distribution function $P(k)$ is the histogram (or probability) of the node degrees; it shows their spread around the average value.

The degree of $A$ is 5.

$0 \leq \langle k \rangle \leq N-1$
System Interfaces
Definition

An interface is ...

... the functional and physical characteristics required to exist at a common boundary between two or more systems, end products, enabling products or subsystems

These functional and physical characteristics can include physical, electrical, electronic, mechanical, hydraulic, pneumatic, optical, software, or human aspects.

Internal and External Interfaces

- Internal interfaces are those boundaries between products and subsystems subject to control by a developer.

- External interfaces are the boundaries between the system and those aspects of the surrounding environment which will interact with the system as it is begin used.
Importance of System Interface Design

The downside of neglecting the system interfaces is that ...

... they can easily become the “weak points” of the system lifecycle.

Incompatible interfaces can lead to ...

... bottlenecks in the system development, local failures, and in the case of software, system-wide failures.

Interfaces and SysML

A key benefit in the use of graphical languages such as SysML for specification of a system, is that ...

... it forces a designer to think about the components of a system, and how the components will interact to achieve the required system functionality.
Classification of Interface Types

Interfaces can be classified according to:

- The type of exchange at the interface (e.g., transfer of force or torque, transfer of energy, transfer of data, transfer of information),
- Whether or not the connection is physical (e.g., fingers touch the screen of an iPad),
- Whether or not the connection is reversible (e.g., a bolted connection is reversible; a welded connection may not be reversible), and
- Whether or not the exchange is directed (e.g., a one-way flow versus a synchronization).
Physical Connection

Two parts are in direct physical connection if they touch each other (e.g., a car wheel is in contact with a road surface; Lego blocks snap together).

Sampling of port types in Lego: (a) rail-port, (b) stud-port, (c) circular-hole-port, (d) TECHNIC-stud-port, (e) TECHNIC-tube-port, (f) axle-hole-port, (g) channel-port, (h) tube-port, (i) friction-pin-port, and (j) axle-port.
Physical Connection (Common Boundary Connection)

Perhaps nothing has done more to enable global transportation of goods than invention of the shipping container.

According to Wikipedia, there are now 17 million containers Worldwide.
Energy Flow Connection

An energy flow occurs when ...

... there is an exchange of work between two components.

The power (P) associated with the time-history of work (W) is:

\[
\text{Power} = \left[ \frac{dW}{dt} \right] \text{ Joules/sec.}
\]  

(2)

Power can take a variety of forms: Electrical power (e.g., AC Power 120V at 60 Hz), thermal power (e.g., dQ/dt) associated with conduction and convection, mechanical power (e.g., force times velocity, torque times angular velocity).

Energy flows usually imply a physical connection, but not always (e.g., a conducting surface). Typically, energy flow will be directed from source to sink.
Mass Flow Connection

Mass flow occurs when there is an exchange of matter between two elements (or subsystems).

Flows of mass occur for fluids, gases and solids, typically imply a physical connection, and are directed from a source to a sink.
Data and Information Flow Connection

Data and information flows occur in user/operator and sensor/system interactions and device controls (that is, sensors, actuators, controllers).

An information flow is always directed.

Examples include: sensor data (i.e., How is my system doing?) and command data (i.e., Here is what I want my system to do).
Motivation

Software interfaces are the mechanism by which ...

... components describe what they do (or provide in terms of functionality and/or services).

Interface abstractions are appropriate for collections of objects that provides common functionality, ...

... but are otherwise unrelated.

Implementation

- An interface defines a set of methods without providing an implementation for them.

- An interface does not have a constructor – therefore, it cannot be instantiated as a concrete object.

- Any concrete class the implements the interface must provide implementations for all of the methods listed in the interface.
Example 1. Software Interface for Farm Workers

Class diagram for implementation and use of a farm workers interface.
Example 1. Software Interface for Farm Workers

Workers is simply an abstract class that defines an interface, i.e.,

```java
public interface Working {
    public abstract void hours();
}
```

In Java, the interface is implemented by using the keyword "implements" in the class declaration, e.g.,

```java
public class Farmer implements Working { ....
```

This declaration ...

... sets up a contract that guarantees the Farmer class will provide a concrete implementation for the method hours().
**Important Point**

Instead of writing code that looks like:

```java
Farmer mac = new Farmer (...);
WorkingDog max = new WorkingDog (...);
WorkingHorse silver = new WorkingHorse (...);
```

We can treat this group of objects as a set of Working entities, i.e.,

```java
Working mac = new Farmer (...);
Working max = new WorkingDog (...);
Working silver = new WorkingHorse (...);
```

Methods and algorithms can be defined in terms of all "Working" entities, independent of the lower-level details of implementation.
Motivation and Benefits

In Java, an interface represents ...

... what a class can do, but not how it will do it, which is the actual implementation.

Two key benefits:

- Information hiding – as long as the objects conform to the interface specification, then there is no need for the clients to know the exact type of the objects they use.

- Improved flexibility – system behavior can be changed by swapping the object used with another implementing the same interface.
Combining Abstract Classes and Interfaces

Now we can write:

Creating objects of type C, D and E.

```
B c1 = new C (...);
B d1 = new D (...);
B e1 = new E (...);
```

Executing methods ...

```
b1.method1();
c1.method2();
e1.method3();
```
Software design patterns promote **separation of concerns**.
Programming to a Software Interface

Composite Hierarchy Pattern

Example: Furniture Layout

Composite Tree Structure
Programming to a Software Interface

Example: Parametric Modeling of Building Floorplans

Example: Component-Based Modeling of Pipe Networks

\[
P_b [\text{Pa}] = P_a [\text{Pa}] - \rho \frac{\text{kg/m}^3}{\text{m}^3} \cdot g \frac{\text{m}}{\text{s}^2} \cdot \left( \Delta z [\text{m}] + f[\text{dimensionless}] \frac{L [\text{m}]}{D [\text{m}]} \frac{V^2 [\text{m}^2/\text{s}^2]}{2g [\text{m}/\text{s}^2]} \right)
\]

Source: Karam Rajab, MSSE Student, 2012-2013.
Strategies for Studying a System
Why Study a System

Generally speaking, we ...

... study systems in the hope of understanding cause-and-effect relationships appropriate to a desirable type of reasoning.

Strategies for Studying a System

- Experiment with actual system
- Experiment with model of system
  - Physical model
  - Mathematical model
    - Analytical solution
    - Simulation


Basic Observation

Most engineers do not make artifacts.

They design and analyze artifacts, and they do so by making models.

Models are what many engineers make.

Of course, there is a little more truth to this statement than many engineers would like to admit!
Abstraction

Concentrate on the essential features of one part of the problem, and abstract (or remove) from consideration details not immediately relevant.

Validation

Ensure that the model represents the real system to a certain degree of accuracy.
Features of Useful Models

- **Abstract**
  Good models emphasize important aspects of a problem and remove irrelevant ones.

- **Easy to Understand**
  Good models express information in a form end-users can easily understand.

- **Accurate**
  Good models faithfully represent the system that is being modeled.

- **Predictive**
  Good models can be used to derive conclusions about the system.

- **Inexpensive**
  Good models must are much cheaper to create than the real system itself.

- **Forgivable**
  The cost of making mistakes during trial-and-error experiments is considerably less with models than for the real system.
Informal, Semiformal, and Formal Models

Informal Representations
- Arbitrary graphics (i.e., back of the envelope drawing),
- Natural language (i.e., conversational), and (i.e., textual requirements).

Informal representations are imprecise, bulky and often unorganized.

Semiformal Representations
- Structured graphical representations such as UML and SysML,

Semiformal representations come with semantics in the form of a semi-formal language.

Formal Representations
- Richer semantic structure than their semi-formal counterparts.

Mathematically based approaches to development are important ...

... because of their potential for helping to ensure correct specification, design, and implementation of complex systems.
Formal Models for Engineering Design
Elements of Formal Modeling in Engineering Design

Formal models for engineering design should consist of the following components (Sangiovanni-Vincentelli, 1996):

- A set of explicit or implicit equations which involve input, output and possible internal (state) variables;
- A set of properties that the design must satisfy given as a set of equations over design variables (inputs, outputs, states);
- A set of performance indices which evaluate the quality of the design in terms of cost, reliability, speed, ... etc.... given as a set of equations involving design variables.
- A set of constraints on design variables and on performance indices specified as a set of inequalities.
Formal Approaches to Behavior Modeling and Decision Making

Appropriate formalisms depend on the design domain of interest.

- Physical aspects of behavior are often characterized by differential equations.
- Logical aspects of system design can be captured by binary and multi-valued logic variables and boolean equations.

Structural behavior corresponds to solution of differential equations.

Selection of the structural system can be written mathematically as a problem in multi–valued logic.
Structural Behavior

Time-dependent behavior corresponds to solutions of:

\[
[M] \frac{d^2 x}{d^2 t} + [C] \frac{dx}{dt} + [K] x = P(t). \tag{3}
\]

Here,

- \(M, C, \text{ and } K\) are \((n \times n)\) matrices,
- \(x\) is a \((n \times 1)\) vector of displacements,
- \(P(t)\) is a vector of external loads applied to the structural degrees of freedom.

Design Parameters

- Selection of the best structural system (e.g., braced system) from a list of options.
- Size of the beams, columns, and bracing (if required).
Design Constraints

- Maximum and minimum sizes of available structural elements.
- To prevent non-structural damage, peak floor accelerations $\leq 0.5g$.
- To prevent building collapse, peak displacement of the roof $\leq 0.02$ times the building height.

Design Objectives

- Maximize ease with which spaces may be adapted to different purposes.
- Minimize cost of materials.
- Minimize cost of construction and time-to-completion.
- Minimize long-term maintenance costs.

These problems can be cast in a multiobjective optimization-based design format.
Definition of Formal Models

Formal models are defined by meta models and ontologies:

- **Meta Model.** The rules for making a model (example UML/SysML).
- **Ontology.** A set of model building blocks (e.g., the entities of UML/SysML)

Structure of a Formal Model
Definition of a Meta-Model

A meta-model describes ...

... information about models.

The items of interest at this level are ...

... the elements, rules and meaning of the modeling constructs themselves.

Definition of a Meta-Meta-Model

A meta-meta-models describe ...

... information about meta-models (i.e., a language in which meta-models may be expressed).
Pathway from Meta-Meta-Models to Engineering Models

Source: (Wieringa, 1998)

- Engineering Model
- Engineering system ... a collection of interacting objects...
- Surrounding Environment

- Meta-model (e.g., Types of UML diagrams)
- Meta-meta-model (e.g., UML diagram elements — class, association, .. etc)
Example. Meta-model for a Finite State Machine

- Meta-Modeling Language
  - Meta-Meta-Model
  - Meta-Model
  - Domain Modeling Language
    - Model
    - Computer-Based System

Example: FSM Meta-Model

Example: FSM Model
Ontologies and Ontology-Enabled Computing
Definition of an Ontology

An ontology is (Hendler, 2001):

... a set of knowledge terms, including the vocabulary, the semantic interconnections,

and

... some simple rules of inference and logic for some particular topic or domain.

Goals of an Ontology

To provide for a formal conceptualization within a particular domain, an ontology needs to accomplish three things (Liang, 2004):

- Provide a semantic representation of each entity and its relationships to other entities;
- Provide constraints and rules that permit reasoning within the ontology; and
- Describe behavior associated with stated or inferred facts.
A Simple Example: Wine Ontology
Ontologies

A High-Level Sensor Ontology
Framework for Ontology-Enabled Traceability and Design Assessment

Textual Requirements → Individual Requirement → Instances → Data

Ontologies and Models:
- Classes
  - Relationships
  - Properties

Design Rules and Reasoner:
- Design Rules
- Reasoner

Engineering Model:
- System Structure
- System Behavior

Benefits of Rule-Based Approaches to Problem Solving

1. Rules that represent policies are easily communicated and understood,
2. Rules retain a higher level of independence than logic embedded in systems,
3. Rules separate knowledge from its implementation logic, and
4. Rules can be changed without changing source code or underlying model.

A rule-based approach to problem solving is particularly beneficial when ...

... the application logic is dynamic (i.e., where a change in a policy needs to be immediately reflected throughout the application) and rules are imposed on the system by external entities.

Both of these conditions apply to the design and management of engineering systems.
Goals of the World Wide Web

In his original vision for the World Wide Web, Tim Berners-Lee described two key objectives:

- To make the Web a collaborative medium, and
- To make the Web understandable and, thus, processable by machines.

Today’s Web provides a medium for presentation of data/content to humans ...

Machines are used primarily to retrieve and render information. Humans are expected to interpret and understand the meaning of the content.

Goals of the Semantic Web

Give information a well-defined meaning, thereby creating a pathway for machine-to-machine communication and automated services based on descriptions of semantics.
Technologies in the Semantic Web Layer Cake

![Layer Cake Diagram]

- User Interface & Applications
- Trust
- Proof
- Unifying Logic
- Ontology: OWL
- RDF-S
- Rule: RIF
- RDF
- XML
- URI/IRI
RDF Triple: Node A is a subject, predicate is a verb, and node B is an object.

English statements are transformed into RDF triples consisting of:

- A subject (this is the entity the statement is about),
- A predicate (this is the named attribute, or property, of the subject) and
- An object (the value of the named attribute).

A set of related statements constitute an RDF graph.
RDF graph of relationships important to Spiderman.
Ontology-to-Reasoning Tools Pathway

**Basic Reasoning Capability**

- **OWL**
- **RDF**
- **Jena**
- **Protoge**

**Reasoning with Ontologies and Rules**

- **Design Problem**
  - **OWL**
    - Ontology
    - Knowledge base
- **SWRL**
  - Rules
- **Facts in Jess**
- **Add inferred facts to knowledge base**
- **Jess Rule Engine**
  - Inferred knowledge base
  - Plugin
- **Ontology-enabled Application**
- **Save**

OWL format

---

**Protoge**

Depends on ..
Example 1: Simple Family Ontology

- Person
  - hasAge
  - hasWeight
  - hasBirthdate
  - attendsPreschool

- Male
  - Boy

- Female

- Child
Example 1: Simple Family Rules + Fact

- Rule 1: For a given date of birth, a built-in function `getAge()` computes a person’s age.
- Rule 2: A child is a person with age $< 18$.
- Rule 3: Children who are age 5 attend preschool.
- Fact 1: Chris is a boy. He was born October 1, 2007.

Example 1: Modeling Chris, a member (instance) of the Family
Example 1: Create Family Individuals

mark = male.createIndividual(ns + "Mark");
chris = boy.createIndividual(ns + "Christopher");
nina = female.createIndividual(ns + "Nina");

// Create statements "Chris has birthdate 2007-10-01" and "Chris has weight 35"

Literal birthdate01 = model.createTypedLiteral("2007-10-01", XSDDatatype.XSDdate );
Statement chrisBirthdate = model.createStatement( chris, hasBirthDate, birthdate01 );
model.add ( chrisBirthdate );

Literal weight35 = model.createTypedLiteral("35.0", XSDDatatype.XSDdouble );
Statement chrisWeighs35 = model.createStatement( chris, hasWeight, weight35 );
model.add ( chrisWeighs35 );

.....
Example 1: Facts in the Simple Family Model

```xml
<rdf:Description rdf:about="http://austin.org/family#Christopher">
  <j:hasWeight rdf:datatype="http://www.w3.org/2001/XMLSchema#double"> 35.0 </j:hasWeight>
  <j:hasBirthDate rdf:datatype="http://www.w3.org/2001/XMLSchema#date"> 2007-10-01 </j:hasBirthDate>
  <rdf:type rdf:resource="http://austin.org/family#Boy"/>
</rdf:Description>

........

<rdf:Description rdf:about="http://austin.org/family#Nina">
  <j:hasBirthDate rdf:datatype="http://www.w3.org/2001/XMLSchema#date"> 2010-02-03 </j:hasBirthDate>
  <rdf:type rdf:resource="http://austin.org/family#FemalePerson"/>
</rdf:Description>
```
Example 1: Jena Rules

@prefix af: <http://austin.org/family#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.

// Rule 01: Propogate class hierarchy relationships ....
[ rdfs01: (?x rdfs:subClassOf ?y), notEqual(?x,?y) -> [ (?a rdf:type ?y) <- (?a rdf:type ?x) ] ]

// Rule 02: Identify a person who is also a child ...
[ Child: (?x rdf:type af:Person) (?x af:hasAge ?y) lessThan(?y, 18) -> (?x rdf:type af:Child) ]

// Rule 03: See if a child attends preschool ...
[ Preschool: (?x rdf:type af:Child) (?x af:hasAge ?y)
  equal(?y, 5) -> (?x af:attendsPreSchool af:True) ]

// Rule 04: Compute and store the age of a person ....
[ GetAge: (?x rdf:type af:Person) (?x af:hasBirthDate ?y) getAge(?y,?z) -> (?x af:hasAge ?z) ]
Example 1: Query the Transformed Simple Family Model

Statements: Chris ...

Statement[ 1] Subject : http://austin.org/family#Christopher
Predicate: http://austin.org/family#hasAge
Object : "5.0"^^http://www.w3.org/2001/XMLSchema#double"

Statement[ 2] Subject : http://austin.org/family#Christopher
Predicate: http://www.w3.org/1999/02/22-rdf-syntax-ns#type
Object : http://austin.org/family#Child

Predicate: http://austin.org/family#attendsPreschool
Object : http://austin.org/family#True

Predicate: http://austin.org/family#hasWeight
Object : "35.0"^^http://www.w3.org/2001/XMLSchema#double"

Statement[ 5] Subject : http://austin.org/family#Christopher
Predicate: http://austin.org/family#hasBirthDate
Object : "2007-10-01"^^http://www.w3.org/2001/XMLSchema#date"

Predicate: http://www.w3.org/1999/02/22-rdf-syntax-ns#type
Object : http://austin.org/family#Boy

.... etc ....
Distributed System Behavior Modeling with Ontologies (2014)

In a decentralized system structure, no decision maker knows all of the information known to all of the other decision makers, ...

... yet as a group, they must cooperate to achieve system-wide objectives.

Communication/information exchange are important to decision makers because ...

... it establishes common knowledge which, in turn, enhances the ability of decision makers to make decisions appropriate to their understanding, of the system state, its goals and objectives.

While each of the participating disciplines will have a preference toward operating their domain as independently as possible, ...

... achieving target levels of performance and correctness of functionality nearly always requires that disciplines coordinate activities at key points in the system operation.

Source: Austin, Delgoshaei and Nguyen, CSER 2015.
System Architecture for Distributed System Behavior Modeling

<< abstract >>
AbstractOntologyModel

Semantic Model: Domain 1

Rule for domain 1

Import

Semantic Model: Domain 2

Rule for domain 2

<< abstract >>
AbstractOntologyInterface

Interface: Domain 1

Message passing

Listener

Message passing

Message input

Message input

Message input

Listens for ModelChange events
Model of Family-School Interaction

Family Domain

- Family Graph Model
- Reasoner
- family rules
- family – school interaction rules

School System Domain

- School System Graph Model
- Reasoner
- school system rules

Enrollment

Listen

Import
Abbreviated Family-School Rules

Abbreviated Family Rules

@prefix af: <http://austin.org/family#>.

// Rule 01: Propagate class hierarchy relationships ....
[ rdfs01: (?x rdfs:subClassOf ?y), notEqual(?x,?y) -> [ (?a rdf:type ?y) <- (?a rdf:type ?x) ] ]

// Rule 03: Compute and store the age of a person ....
[ Age01: (?x rdf:type af:Person) (?x af:hasBirthDate ?y) getAge(?y,?z) -> (?x af:hasAge ?z) ]

Abbreviated Family-School Interaction Rules

@prefix af: <http://austin.org/family#>.

// Rules 02: Children aged 6 through 18 attend regular school ....
[ School01: (?x rdf:type af:Person) (?x af:hasBirthDate ?a)
  getAge(?a,?b) ge(?b, 6) le(?b, 18) -> (?x af:attendsSchool af:True) ]

Abbreviated School Rules

@prefix af: <http://austin.org/school#>.

// Rules 01: Elementary school rules ...
[ Elementary01: (?x rdf:type af:Student) (?x af:hasBirthDate ?a)
  getAge(?a,?b) le(?b, 18) -> (?x af:isElementarySchool:Student af:True) ]
Weather System Model \(\rightarrow\) Washington DC Metro System

**Metro Domain:**

- **Metro Graph:**
  - **Station:** Prince George, College Park, Green Belt
  - hasNext, hasNext
  - isClosed F, isTerminal F

**Weather Domain:**

- **Weather Graph:**
  - **Site:** College Park
  - hasWeatherCondition: snow
  - hasMeasurement: 0
  - hasTemperature: 30

**Weather Report:**

- College Park snow

**Reasoner**

**Metro Rules**

**Metro-Weather Rules**

**Metro Interface Model**

**Weather Interface Model**

**Import:** Metro\-Weather Rules

**College Park 25 inch snow**

**Weather Interface Model**

**Weather Graph:**

- **Site:** College Park
  - hasWeatherCondition: snow
  - hasMeasurement: 25
  - hasTemperature: 28

**Reasoner**

**Weather Rules**

**Import:** Weather Rules
Application: Synthesis of System Architectures

Suppose that we could obtain complete descriptions of products on the Web.

www.panasonic.com
Product specifications.

www.jbl.com
Product specifications.

www.sony.com
Product specifications.

External Description of Product Specifications

Partially assembled system

Textual description of requirements.
Graphical description of relationship among requirements.

Plasma Screen Display

Power

This would enable new types of editors for synthesis of system designs.
Stereo Example. Architectural Synthesis

Synthesis of a stereo connectivity model from electronic components.

Amplifier – Speaker – Cable Connectivity

Port–Connection–Cable Model
Stereo Example. Ontology Definition with Protege
Generation of Design Alternatives through Configuration Modeling

- System-Level Architecture
- Library of Components
- Environmental Model
- User Requirements
- Configurator
- System Design Alternatives

Add new components to library.
Select components

**Electronic Component Ontology**

<table>
<thead>
<tr>
<th>AMP_6</th>
</tr>
</thead>
<tbody>
<tr>
<td>hasPrice (400)</td>
</tr>
</tbody>
</table>

**Fact in Working Memory**

| ( assert ( hasPrice AMP_6 400 ) ) |

**Formal Definition of a Cheap System**

\[
\text{AMP}(\text{\(?a\)}) \land \text{hasPrice}(\text{\(?a\)}, \text{\(?p1\)}) \land \text{DVD}(\text{\(?d\)}) \land \text{hasPrice}(\text{\(?d\)}, \text{\(?p2\)}) \land \\
\text{swrlb:add}(\text{\(?s\)}, \text{\(?p1\)}, \text{\(?p2\)}) \land \text{swrlb:lessThan}(\text{\(?s\)}, 600.0) \rightarrow \text{hasCheapSystem}(\text{\(?a\)}, \text{\(?d\)})
\]

**Problem Definition in English**

- I have 3 amps and 3 dvds.
- My system should include a dvd and a amp and a price has to be less than $600.
- Do I have such a system?

Jess spits out 3 combinations of suitable dvd/amp/price. Natasha: It’s very cool =)
Synthesis of Component Architectures with RDF Graph Models (2012)

**Motivation:** Working with Jess/JessML is difficult.

**Research Objective:** Can we design a simpler approach using only RDF graph models + a sequence of graph transformation to synthesize and organize design solutions?

Trade-Off Curves

Multiple Viewpoint Modeling
Viewpoints

Models are based on a point of view (i.e., frame of reference) that can guide the identification of relevant issues and information.

Multiple Viewpoints

For the design of real-world systems, multiple viewpoints and representations occur because:

- Multiple perspectives of a complex system (e.g., mechanical engineering; electrical engineering) are needed simply to understand it completely.
- Large systems have stakeholders with multidisciplinary interests (e.g., engineers; architects; builders; customers).
- The development process for engineering systems is simplified if discipline-specific views/models of the system can be established.
Multiple Views of a System

While complexity associated with each disciplinary contribution is likely to lie in the details of that discipline, for multidisciplinary projects

... capturing the functional concerns of those viewpoints and their interaction ... is most important.
Multiple Viewpoints

Modeling the Interaction of Viewpoints

Viewpoint models should be linked together via appropriate interfaces, thereby allowing for ...

... cause-and-effect studies among viewpoints (e.g., a mechanical engineer might want to know how changes in his/her work will affect other disciplines).

Schematic for Viewpoint Interactions

- Viewpoint 1
- Entity 1
- Interaction Mechanism
- Entity 2
- Viewpoint 2
Multiple Viewpoints

Multiplicty of Contextual Interpretations

This task is complicated by the fact that ...

... each discipline will interpret design objects relevant to their set of concerns.

Viewpoint
Functionality

Objects

Same object has multiple representations
Same object serves multiple purposes ...

- p. 117/129
Problem Statement

Object: Wall in a house

Stakeholder Viewpoint / Concerns

Architect

- Surrounds a space (e.g., a room)
- Wall contains a door and window
- Window provides ambient light
- Door enables access to space...

Structural Engineer

- Wall is a load bearing object
- Wall transfers load from upper floor levels to the ground
- The window and doorway are holes in the wall
- Holes in the wall make it weaker...
- Analysis concepts include internal forces, bending moments, displacements

Now suppose that ...

- The architect decides that interior room is too dark, and that the problem can be solved by installing a second window.
- Before installation of a second window can be approved, the architect will need to check with the structural engineer.
Multiple Contextual Interpretations of a House Wall

- Architectural Requirements
  - Space
  - Comfort
  - Access
  - Aesthetics
  - Strength

- Structural Engineering Requirements

- Wall Primitives
  - width
  - height
  - geometry
  - material
  - door
  - window

- Wall Attributes

- Wall Implementation

- Material Properties
Example. A Simple Wall in a House

Linking Architectural and Structural Engineering Ontologies

Simplified Architectural Engineering Ontology

Wall System

Space

Functional Purpose

Architectural Design Concerns

Structural Engineering Design Concerns

Simplified Structural Engineering Ontology

Transfer of loads to foundation

Wall System

Complies with

3, *

Material

Portal

Geometry

Opening

Window

Door

Material

Strength

Geometry

Aesthetics

Occupant

Comfort

Access

Material

Affects

Affects

Affects

Affects

Same as

Requires

Requires

Requires

Requires

Same as
Systems Engineering View of Modeling
Systems engineers are ..

... the keepers of the processes, methods and tools needed to establish and maintain a shared vision of the system problem definition and solution.

This process is complicated by the

- technical,
- social,
- regulatory, and financial aspects

of a complex design being too broad and detailed for a single individual to master. Hence, by necessity,

... system development is a team activity involving multiple stakeholders, their design concerns, viewpoints, and ways of doing things.
System Engineering View of Modeling

Pathway from Functions to Representations in Multiple-Stakeholder Design

Use cases and scenarios

Preliminary Requirements

Integrate and Organize Requirements into Layers for Team-based Development

Models of System Behavior, System Structure, System Architecture .... etc.....
IEEE 1471 Standard. Development of Architecture Descriptions

- System Described by Architecture
- Stakeholder identifies Architecture Description
- Concern identifies Viewpoint
- Viewpoint selects 1...* Model
- Model Conforms to
- View Covers
- View of Architecture Description
- Crosscutting and Basic
- Reliability, Security, Development Pathway
- Requirements, Functional, Engineering Implementation
- Established procedures, methods, and abstractions for...
IEEE 1471 Standard. Development of Architecture Descriptions

Chain of Many-to-Many Relationships

query

Graph Databases

Established procedures, methods, and abstractions for ...

Basic

Crosscutting

Requirements

Reliability

Functional

Security

Engineering Implementation

Development Pathway

Stakeholder

Concern

Viewpoint

Architecture Description

View

Model

System

Architecture

Described by...

identifies

selects

Covers

Conforms to...

1...*

1...*

1...*

1...*

1...*

1...*

1...*

1...*

1...*

1...*

1...*

1...*
Networks of Processes and Models in Systems Engineering

Customers / Users

Organization

Requirements

Engineering System

Model of Organization

Model of Requirements

Model of Engineering System

REAL WORLD SPACE

MODELING SPACE

Data

Sol’ns

Organizational Level

Project Level
System Engineering View of Modeling

Systems Engineering Modeling and Process Requirements

- Customer/Users
- Organization
- Requirements
  - Legal agreement
- Engineering System
- Strategy
  - Business processes
  - Resources
  - Staff
- Capture
  - Representation
  - Organization
  - Evaluation
  - Allocation/Flowdown
  - Traceability
  - Validation/Verification
  - Management
- Behavior
  - Cost
  - Assembly
  - Maintenance
  - Retirement

REAL WORLD SPACE

MODELING SPACE

Data

Sol’ns
References


- Delgoshaei P., Software Patterns for Traceability of Requirements to State Machine Behavior, M.S. Thesis in Systems Engineering, University of Maryland, December 2012.


