Agenda

- System Complexity and Variability
- Product Lines
- Model-Based Systems Engineering
- Model Asset Reuse
- Model-Based Product Line Engineering
- Conclusions
- Questions?
System Complexity and Variability

- Automotive systems are complex
  - Thousands of requirements
  - Multiple systems containing multiple processors
  - Multiple variants and variability
  - Often driven by customer need

- Variability can affect multiple systems
  - Interconnections and interdependencies difficult to manage
  - Mutual exclusion of parts
  - Commonality between systems
  - Achieving economies of scale

- A new approach is needed
Product Lines

- In existence since the Industrial Revolution

- Common in automotive industry
  - Henry Ford was one of the first
  - Assembly lines with interchangeable parts
  - Often evolve over time rather than designed
  - Difficult to manage and control

- Typical Product Lines
  - ~80% of Products the Same with Predictable Component Costs
  - Commonality drives down costs
  - Focus on better Decision Making for the Variable ~20%
Changes in Systems Engineering Practice

Change from Document centric to Model centric

Old Approach


New Approach

Etc.
Modeling – A Way to Manage Complexity

- A Model is a set of entities and relations created as a result of an abstraction process and is used to:
  - neglect immaterial details, hence foster communication and system understanding
  - anticipate the relevant system behavior
  - test and recycle, because it is an operational design tool

- User can visualize entities and relations through diagrams which form the system’s views
  - Views are a set of information that describe a partial and particular system aspect: the human brain can only take into account a few things all together at the same time

- A Database is needed to archive all the Model information
SysML Taxonomy of Diagrams

Diagram

Structure

- Block Definition [1]
- Internal Block [2]
- Parametric

Requirements

- Package

Behavior

- Activity
- State Machine
- Sequence
- Use Case

[1] Modified UML Class Diagram
The Four Pillars of SysML

Structure

Parametrics

Definition

Interaction

State Machine

Activity/Function

Requirements

Behavior

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>d1</td>
<td>m1</td>
</tr>
<tr>
<td>tf</td>
<td>tl</td>
</tr>
<tr>
<td>bf</td>
<td></td>
</tr>
<tr>
<td>v</td>
<td>a</td>
</tr>
<tr>
<td>(v = dx/dt)</td>
<td>(a = dv/dt)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(F = ma)</td>
<td></td>
</tr>
<tr>
<td>(F = (tb<em>bf)</em>(1-tl))</td>
<td>(F = (tb<em>bf)</em>(1-tl))</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vehicle System Specification</td>
<td></td>
</tr>
<tr>
<td>Braking Subsystem Specification</td>
<td></td>
</tr>
</tbody>
</table>

© 2012 Atego. All rights reserved.
Cross Connecting Model Elements

Structure

- ibd [block] Anti-LockController [Internal Block Diagram]
  - satisfies <requirement> Anti-Lock Performance
- d1:TractionDetector
  - allocatedFrom <activity>DetectLossOfTraction
- m1:BrakeModulator
  - allocatedFrom <activity>ModulateBrakingForce
- c1:modulatorInterface
  - values DutyCycle: Percentage

Requirements

- req [package] VehicleSpecifications [Requirements Diagram - Braking Requirements]
  - Vehicle System Specification
    - <requirement> StoppingDistance
      - id="102"
      - text="The vehicle shall stop from 60 mph within 150 ft on a clean dry surface."
      - VerifiedBy <interaction>MinimumStoppingDistance
  - Braking Subsystem Specification
    - <requirement> Anti-LockPerformance
      - id="337"
      - text="Braking subsystem shall prevent wheel lockup under all braking conditions."
      - SatisfiedBy <block>Anti-LockController

Behavior

- act PreventLockup [Swimlane Diagram]
  - <allocate> :TractionDetector
  - <allocate> :BrakeModulator

Parametrics

- par [constraintBlock] StraightLineVehicleDynamics [Parametric Diagram]
  - v.chassis.tire.Friction:
  - v.brake.abs.m1.DutyCycle:
  - v.brake.rotor.BreakingForce:
  - v.Weight:
  - :Accelleration Equation
    - [F = ma]
  - :VelocityEquation
    - [v = dx/dt]
  - :DistanceEquation
    - [v = dx/dt]
Reuse Paradigms

- Subroutines
- Modules
- Objects
- Components
- Software Product Lines
- Systems and Software Product Line Engineering

(Linda Northrop, SEI SSPL 2008-2012)
Model Asset Reuse

- The OMG Reusable Asset Specification (RAS)
  - Used for defining reusable assets, their interfaces, characteristics and supporting elements

- Three key dimensions describe reusable assets:
  - Granularity describes how many particular problems or solution alternatives a packaged asset addresses.
  - The visibility varies from black-box assets, whose internals cannot be seen and are not modifiable, to white box assets which are visible and modifiable.
  - The articulation describes the degree of completeness of the artifacts in providing the solution.

- Asset also include supporting documentation, requirements addressed, interfaces, etc.

- Provides a standards-based “model of models” approach instead of a “mega-model” approach.
Asset Reuse
Variability Management is needed…

In all phases of the development process:

- Requirements Engineering
  - What are common Requirements?

- System and Software Architecture
  - Different and/or Common Architectures?

- Implementation
  - How do the system and software differ in the products?

- Tests
  - Re-usable (common) test cases?
Product Line Modeling

- Object Variability Modeling (OVM)

- The concept of ‘Variability’ Modelling in OVM
  - Variation Points
  - Variants
  - Variability Constraints

- Integrates variability modeling with systems modeling

References:
- Klaus Pohl, Günter Böckle, Frank van der Linden, Software Product Line Engineering – Foundations, Principles, and Techniques, 2005
Variability Approaches

- Model Variability using inheritance

- Model Variability using OVM
  - Orthogonal Variability Modeling
Model-based Variability using OVM

- Allows both approaches
  - Separate views for variability
  - Link of variable elements with base model elements
  - Shared views showing variability and base model views

- Combination of both views within one model enables to
  - Separate the concerns
  - Keep base and variability models consistent

- Model-based variable Assets
  - As an option to combine product families with component-based design
Modeling Variability

- The selection of the Bluetooth connectivity requires the selection of the Bluetooth version.

![Diagram showing the selection of Bluetooth connectivity and versions.](image)
Modeling Variability

- Engine variability along with model dependencies

![Engine Variant Diagram]

```text
var 04 Engine Variant Diagram

- Engine
  - 1..1
  - Efficient
  - Fast

- «BlockProperty» «component» Diesel Engine
- «BlockProperty» «component» Gasoline Engine
```
Modeling Variability

- Types of Transmission

![Variety Diagram]

- AutomaticTransmission
  - proxyPorts
    - Bolt2
    - UJoint1
    - Spiggot
    - Bolt
    - Flange
    - Spline1

- SemiAutomaticTransmission
  - proxyPorts
    - Bolt2
    - UJoint1
    - Spiggot
    - Bolt
    - Flange
    - Spline1

- ManuallyShiftedTransmission
  - flowPorts
    - FlowPort in fp2 : Boolean
    - ProxyPorts
      - Spline1
      - Bolt2
      - UJoint1
      - Spiggot
      - Flange
      - Bolt

- Transmission type
  - Luxury
  - Medium Comfort
  - Regular

- Number-OfGears
  - 5Gears
  - 6Gears
Product Model Creation

- Auto-Creates Product Models
  - Variability Decisions Applied
  - Unnecessary Variation Points, Variants & Base Model Artefacts Removed

- New Product Model Branch, Original Product Line Model Retained

- Product Model suitable for Trade Studies, Simulation & Generation
# Decision Set Editor Example

<table>
<thead>
<tr>
<th>Name</th>
<th>Decision</th>
<th>Status</th>
<th>Included By</th>
<th>Excluded By</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keyless Entry Option</td>
<td></td>
<td>Excluded</td>
<td></td>
<td>Variable sequence diagrams:Variability.Base</td>
<td>Variable sequence diagrams:Variability.Base</td>
</tr>
<tr>
<td>Line</td>
<td></td>
<td>Included</td>
<td>Variable sequence diagrams:Variability.Base</td>
<td></td>
<td>Variable sequence diagrams:Variability.Base</td>
</tr>
<tr>
<td>Base</td>
<td>Include</td>
<td>Included</td>
<td>Variable sequence diagrams:Variability.Base</td>
<td></td>
<td>Variable sequence diagrams:Variability.Base</td>
</tr>
<tr>
<td>Multimedia Supplier</td>
<td></td>
<td>Included</td>
<td>Variable sequence diagrams:Variability.The cheaper, better, but quite unknown Brand</td>
<td></td>
<td>Variable sequence diagrams:Variability.Multimedia Supplier.Alternative Choice1</td>
</tr>
<tr>
<td>The cheaper, better, but quite unknown Brand</td>
<td>Include</td>
<td>Included</td>
<td>Variable sequence diagrams:Variability.The cheaper, better, but quite unknown Brand</td>
<td></td>
<td>Variable sequence diagrams:Variability.Multimedia Supplier.Alternative Choice1</td>
</tr>
<tr>
<td>Battery Size</td>
<td></td>
<td>Included</td>
<td>Variable sequence diagrams:Variability.44Ah</td>
<td>Variable sequence diagrams:Variability.44Ah</td>
<td>Variable sequence diagrams:Variability.44Ah</td>
</tr>
</tbody>
</table>

Variants: 16/3 | Variation Points: 8/21 | Inconsistent: 0 | Undecided Mandatory Variation Points: 9
Variant Selector Example

There are three main car equipment lines, from which one is to be chosen: 1. Base for the price-sensitive customer 2. Dynamic for the more sporty customer 3. Luxus for the customer who wants to equip the car with all the luxury possible.

- Base
- Dynamic
- Luxus
Pruned Elements Example in SysML BDD

- Parts are deleted
  - ke : Keyless Entry in Locking System
Pruned Elements in SysML IBD

- The (unselected) Keyless Entry Variant is gone, including all Artifact Dependencies
- Keyless Entry Part is deleted from the Locking System
- All sub-parts, ports and connectors relating to the Keyless Entry Part are deleted as well
Pruned Elements in SysML IBD (cont’d)

- Within the key:Key Card Part, the relevant Transponder is pruned
- All sub-parts, ports and connectors relating to this Transponder sub-part are deleted as well
Cost Reduction & Time Improvements

- **SE (Non-Modelled Systems Engineering)**
  - 59% of Projects Delivered on Time

- **MBSE (Model Based Systems Engineering)**
  - 62% of Projects Delivered on Time

- **Compared to SE**
  - 55% Reduction in Total Development Cost per Project
  - 16% More Project Delivered on Time

- **MB-PLE (Model Based Product Line Engineering)**
  - 75% of Projects Delivered on Time

- **Compared to MBSE**
  - 17% Reduction in Total Development Cost per Project
  - 6% More Projects Delivered on Time

- **Compared to SE**
  - 62% Reduction in Total Development Cost per Project
  - 23% More Projects Delivered on Time

(EMF 2013 Independent Survey Results from 667 Systems engineering respondents)
Benefits Summary

- Improved productivity by as much as 10x
- Increased quality by as much as 10x
- Decreased cost by as much as 60%
- Decreased labor needs by as much as 87%
- Decreased time to market by as much as 98%
- Ability to move into new markets in months, not years

(Linda Northrop, SEI SSPL 2008-2012)

Weiss, D.M. & and Lei, C.T.R.,
Software Product-Line Engineering: A Family-Based Software Development Process
Conclusions

- Systems and models of systems have become increasingly complex.
- New ways are needed to organize models and the decisions made while creating them.
- SysML, Product Line Engineering, the Object Variability Modeling and the Reusable Asset Specification provide Model-Based Product Line Engineering (MB-PLE).
- Enables the asset reuse while making value-based decisions on system configuration.
- Provides a demonstrable ROI that will reduce development time and costs and help automotive engineers build better systems.
Questions and Answers

Speaker

Thanks for your attention!