Usability Thoughts

R. D. Welling to the MBSE Usability Team 12 July 2010
Usability uses

• Usability for development control
• Usability for introduction, instruction in MBSE
Usability Questions

• What are we using?
• How are we using it?
• How easy is it to use, whatever it is?
• How do we measure usability?
MBSE tool uses

• Which is the primary goal of a modeling tool, model validity or usability?
• If the goal is validity, then does usability mean the ease of building valid models?
• Does usability mean something different for experts and novices? Yes!
• The proof of the above is that very few who understand and use SysML!
Usability for development control

• Mechanical and electrical drawings are models
• Classic engineering status/control mechanism: drawing completion
• Quantity: Number of “approved” drawings
• Quality: Approval asks, “Is this drawing correct?”
• System models also must answer correctness questions
Model navigability

• Finding one’s way around an existing or in-process model

• Navigating to determine
  – Model validity: Is the model syntactically correct?
  – Model completeness: How much work do I have left?
Example: Usability for control

- Sandy Friendenthal has talked about SysML-Lite as a possible path for improving usability
- “SysML-Lite” happened in 2006 mapping a SLATE tool schema to the SysML metamodel
- Diagrams used SLATE Visio API
- Briefings began with this question: “How do we gain and maintain control of the developing system?”
Usability for control

• SLATE/SysML-Lite used basic SysML constructs: Activities, “Logical” Blocks (almost everything in SLATE is a block)
• Quantity: Weekly parsing of the developing model
• Syntactical testing: Valid relations between object types
• Quality: Defect types to direct development
“SysML Lite” c. 2006 in SLATE

<table>
<thead>
<tr>
<th>Supplier</th>
<th>Property</th>
<th>Client</th>
<th>Triple</th>
<th>Defect Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activity</td>
<td>allocatedTo</td>
<td>LogicalBlock</td>
<td>allocatedTo(Activity,LogicalBlock)</td>
<td>AllocationDefect</td>
</tr>
<tr>
<td>Activity</td>
<td>allocatedTo</td>
<td>SoftwarePartition</td>
<td>allocatedTo(Activity,SoftwarePartition)</td>
<td>AllocationDefect</td>
</tr>
<tr>
<td>Activity</td>
<td>ownedBy</td>
<td>Activity</td>
<td>ownedBy(Activity,Activity)</td>
<td>HierarchyDefect</td>
</tr>
<tr>
<td>Action</td>
<td>ownedBy</td>
<td>Activity</td>
<td>ownedBy(Activity,Activity)</td>
<td>HierarchyDefect</td>
</tr>
<tr>
<td>Activity</td>
<td>identifies</td>
<td>Configuration</td>
<td>identifies(Activity,Configuration)</td>
<td>AllocationDefect</td>
</tr>
<tr>
<td>Insertion</td>
<td>implements</td>
<td>Activity</td>
<td>implements(Insertion,Activity)</td>
<td>AllocationDefect</td>
</tr>
<tr>
<td>SOWBlock</td>
<td>directs</td>
<td>Insertion</td>
<td>directs(SOWBlock,Insertion)</td>
<td>AllocationDefect</td>
</tr>
<tr>
<td>SOWBlock</td>
<td>allocatedTo</td>
<td>AB3_Team</td>
<td>allocatedTo(SOWBlock,AB3_Team)</td>
<td>AllocationDefect</td>
</tr>
<tr>
<td>Activity</td>
<td>flowsDownTo</td>
<td>Activity</td>
<td>flowsDownTo(Activity,Activity)</td>
<td>TraceDefect</td>
</tr>
<tr>
<td>Activity</td>
<td>hasInput</td>
<td>ParameterNode</td>
<td>hasInput(Activity,ParameterNode)</td>
<td>TraceDefect</td>
</tr>
<tr>
<td>Activity</td>
<td>hasOutput</td>
<td>ParameterNode</td>
<td>hasOutput(Activity,ParameterNode)</td>
<td>TraceDefect</td>
</tr>
<tr>
<td>DataEntity</td>
<td>isKindOf</td>
<td>ParameterNode</td>
<td>isKindOf(DataEntity,ParameterNode)</td>
<td>TraceDefect</td>
</tr>
<tr>
<td>DataEntity</td>
<td>allocatedTo</td>
<td>LogicalMessage</td>
<td>allocatedTo(DataEntity,LogicalMessage)</td>
<td>AllocationDefect</td>
</tr>
<tr>
<td>ParameterNode</td>
<td>allocatedTo</td>
<td>LogicalPort</td>
<td>allocatedTo(ParameterNode,LogicalPort)</td>
<td>AllocationDefect</td>
</tr>
<tr>
<td>LogicalBlock</td>
<td>composedOf</td>
<td>Part</td>
<td>composedOf(LogicalBlock,Part)</td>
<td>AllocationDefect</td>
</tr>
<tr>
<td>LogicalModule</td>
<td>composedOf</td>
<td>Code</td>
<td>composedOf(LogicalModule,Code)</td>
<td>AllocationDefect</td>
</tr>
<tr>
<td>LogicalPort</td>
<td>nestedPort</td>
<td>LogicalPort</td>
<td>nestedPort(LogicalPort,LogicalPort)</td>
<td>AllocationDefect</td>
</tr>
<tr>
<td>LogicalPort</td>
<td>matchedPort</td>
<td>LogicalPort</td>
<td>matchedPort(LogicalPort,LogicalPort)</td>
<td>AllocationDefect</td>
</tr>
<tr>
<td>LogicalPort</td>
<td>nestedPort</td>
<td>RealPort</td>
<td>nestedPort(LogicalPort&lt;RealPort&gt;)</td>
<td>AllocationDefect</td>
</tr>
</tbody>
</table>

SysML Object types
Associations (connectors?)
What object types get to be clients?
RDF-style triples implemented in Tcl
Control and feedback

RD Welling 7/5/10
Issues for original SysML-Lite

• Weekly parsing for status and direction (control) was almost agile
• The system model as a “principal artifact” was and is a foreign concept
• How do we get engineers to communicate thoughts in an abstract language outside their domain?
• Means of improving interpretation: trees, graphs, temperature charts
Sandy’s note (extract)

• Are there ways we can significantly reduce the time to develop a basic level of competency [for the new user who is just learning SysML]

• We have talked about defining a SysML Lite... the new SysML certification includes a basic feature set that could be a starter.

• ...we are thinking of including only a subset of the diagrams in SysML Lite such as bdd, ibd, activity, requirements, and use case.
Engineers and models

- Engineers are not familiar with the formalism of generalized systems
- Modelica, e.g., uses objects exhibiting specific behaviors (pumps, relays, etc.)
- VLSI tools, e.g., result in mechanical layouts
- Generalizing the many domain-specific concepts sets a gap in understanding
- Is MBSE usability concerned with closing the gap between specific and abstract knowledge?
White board model

Naïve Modeling Domain = White Board Engineering (WBE)
Same system in SysML
System has an internal arrangement

Item flows? Reference properties? Yikes!
Levels of Abstraction

- More intuitive
- Lower abstraction
- Limited information potential
- Domain specific (white board domain)

- Less intuitive
- Greater abstraction
- Rich information potential
- Domain independent (any system)
Transformations

• Making a tool usable amounts to changing the way that its authors are quite comfortable in using it.
• SysML is complex even for the experts.
• Non-expert users need only “SysML Lite”
• SysML Lite needs to conform to the SysML metamodel
• Enhancing usability is a matter data transformation: from the authors to the users.
Usability for introduction, instruction in MBSE

- Method for new users to “wade” into MBSE
- “Naïve” or intuitive SysML (White Board Engineering)
- Automatic or aided (prompted) transformation to valid SysML
SysML-Lite as a DSL

- The UML2 extension mechanism allows for profiles of domain specific language (DSL)
- SysML is an extension of UML to avoid language proliferation
- A SysML-Lite profile would provide a controlled method for defining
  - Valid SysML-Lite
  - The transformation between SysML and SysML-Lite
SysML-Lite Profile

After J. Bankston, A. Shah, ISYE 8813 Domain Specific Modeling of an ECAD System, 2006

*White Board Engineering