Two Systems Engineering Functions with SysML: The High and Low of MBSE Usability (Part 1)

Bjorn Cole
NASA Process (NPR 7120.5d)
Usability

• For tools that perform tasks, I will combine the criteria brought up by Ron Lyells with Achieving Usability Through Software (SEI paper referred to on message board)

• Architecture
  – Ease of learning
  – Efficiency of use
    • Expedites routine performance
    • Improves non-routine performance (helps problem solving and learning)
  – Error frequency / severity
    • Reduces impact of errors
    • Prevents errors
  – Memorability
  – Subjective Satisfaction (comes from above)
Task 1: Technical Resource Management

• Establish level of technical resources to be available (Adv Studies & Formulation)
  – Non-routine challenge: How to account for resource growth in interrelated systems (e.g., mass margins on Ares launch vehicle)

• Allocate resources to different parts of the system (Implementation)

• Track developing hardware and software against allocated resources (Implementation)

• Barter for resources (Operation)
What Services Might MBSE Provide?

• Establish / Allocate Resources
  • The place to lay out and solidify the system architecture; an analysis repository; storage of demarcations of allocated domains of further work

• Track Resources v. Allocations
  • A smart reporting center for updating estimated / measured values of resource use during development, which sorts information according to system model; may need to be away of not only system but development process

• Barter Resources
  • Describe to operators how resources are developed in system and how to shift them around; describe constraints on how resources are used or broken up
## Usability Matrix

<table>
<thead>
<tr>
<th>Establish Resources</th>
<th>BDD</th>
<th>IBD</th>
<th>Par</th>
<th>DB</th>
<th>Transformation</th>
<th>Spreadsheet</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Ease of Learning</strong></td>
<td>Come packaged with notions of inheritance and abstraction that may require education</td>
<td>Depends on learning working style – can be both cumbersome (lots of blocks) or elucidating (which parameters interact)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Efficiency of Use Routine</strong></td>
<td>Levels of allocation easy to show in single diagram for generic components; capture values for standard practice on margins, etc.</td>
<td>Model the analysis environment to see where updates to values will come from</td>
<td>Can show “tree math” for different levels of hierarchy; many SE’s can be frustrated with “I just want a simple sum!”</td>
<td></td>
<td>Quick to build; usability fades as length of time using same spreadsheet grows</td>
<td></td>
</tr>
<tr>
<td><strong>Efficiency of Use Non-Routine</strong></td>
<td>Easy to show how components interact – some elements (structural loading, mechanical power transfer) not naturally represented as “flows”</td>
<td>Specify sensitivities of one subsystem to another’s violation of allocation constraints; clear but not as simple as equation routines; hard to follow detailed calculations</td>
<td></td>
<td></td>
<td>Simple to develop, but often gets opaque once analysis is finished</td>
<td></td>
</tr>
<tr>
<td><strong>Error Frequency</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Low once debugged</td>
<td>High; hard to track pedigree</td>
</tr>
<tr>
<td>Activity</td>
<td>IBD</td>
<td>Par</td>
<td>DB</td>
<td>Transformation</td>
<td>Spreadsheet</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>-----</td>
<td>-----</td>
<td>----</td>
<td>---------------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td><strong>Track Resources</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Ease of Learning</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Easily understood by Sys Engineers; may take a little extra time to master</td>
<td></td>
<td>Depends on learning working style – can be both cumbersome (lots of blocks) or elucidating (which parameters interact)</td>
<td></td>
<td>Requires a different mindset for use; is in common with XSL</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Efficiency of Use Routine</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model development environment to see where information will be developed and with which tools</td>
<td>Model the analysis environment to see where updates to values will come from</td>
<td>Can show “tree math” for different levels of hierarchy; works better in abstract than concrete; many SE’s can be frustrated with “I just want a simple sum!”</td>
<td>Concept is familiar; specifics of data entry / harvesting may be difficult</td>
<td>How does one provide a “tagged” handle of a property over to a disciplinarian for connection to his or her development effort?</td>
<td>Difficult to update, very manual process</td>
<td></td>
</tr>
<tr>
<td><strong>Efficiency of Use Non-Routine</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Previously captured sensitivity equations may be useful but hard to find / recall</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Error Frequency</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Low once debugged</td>
<td>High; hard to track pedigree</td>
<td></td>
</tr>
</tbody>
</table>
## Usability Matrix

<table>
<thead>
<tr>
<th>BDD</th>
<th>IBD</th>
<th>Par</th>
<th>DB</th>
<th>Transformation</th>
<th>Spreadsheet</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Barter Resources</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Ease of Learning</strong></td>
<td></td>
<td></td>
<td>Depends on learning working style – can be both cumbersome (lots of blocks) or elucidating (which parameters interact)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Efficiency of Use Routine</strong></td>
<td>Show the paths for redirected surplus resources from one subsystem to others; examine current usage</td>
<td>See parts of system dynamical models laid out, see where to tune parameters; hard to see the integrated system of equations at once across many blocks and values</td>
<td>Match resource use predicted by model into the same terms as the operating reports; generate reports and queries on current use</td>
<td>Nearly impossible unless the spreadsheet is in current use to understand how the system is modelled</td>
<td></td>
</tr>
<tr>
<td><strong>Efficiency of Use Non-Routine</strong></td>
<td>Identify damaged / reconfigured components; hard to mark up IBD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Error Frequency</strong></td>
<td></td>
<td></td>
<td></td>
<td>Low once debugged</td>
<td>High; hard to track pedigree</td>
</tr>
</tbody>
</table>
MBSE and Resources Summary

• Mostly good for the high-level problem
  – Prepare analyses of budget
  – Define how different budgets are “rolled up” to higher hierarchy levels
  – Show where the system fights for resources or how bloat in one subsystem will impact others
  – A model of your development process will help understand where you need to harvest information from to keep an up-to-date picture
  – Model of system helps operators understand it better

• Not good for the details
  – Lists, matrices much easier to read and update than blocks
  – Too many steps for defining sums of hierarchies
  – Parametrics good for describing two levels (aggregator and aggregated) of interaction, but takes sophistication by modeler to know how this turns into a query-based or automated calculation
  – Best use may be to hand a “handle” over to developers, but how to make it easy to connect?

• Parametrics are often awkward
  – Good for simple equations, but equation systems can quickly sprawl
MBSE and Resources Cures

• Language cure for parametrics
  – This problem deals with hierarchy trees all of the time – special notation or pattern for “do this for every parent-child pair you see” would be useful
  – Notation for how to proceed with collections as inputs to constraint equations would also be helpful (work as a matrix, element-by-element, row * column, etc.)

• Tool- and training-based cures
  – Need to connect SE concepts of allocation, trees, etc. to their computer science counterparts if laying out an information capture environment (which is what tracking is)
  – Render tables in SysML tools (at least a couple do this already)
  – Synchronization tools definitely needed to bring information from disciplinarians back to SE’s on a regular basis, and to update discipline work with the new parameters of the system at large
  – Render parametric systems from multiple blocks in a hierarchy, allow for selective display (think Simulink); some tools are moving in this direction

• Process-based cures
  – Develop a process for specifying (highly-targeted) transformations between analysis tool outputs and tracked values from development workflow model
  – Another tactic would be applying tag labels in output files for information crawlers to harvest
Task 2: Interfaces (for another time)

• Subdivide the system (Adv Studies & Formulation)
  – Shows where the interfaces will arise and hints at difficulties
• Elaborate Interfaces (Implementation)
• Control Interfaces & Verify Conformance (Implementation)
What to do from here

• Should look at Usability Matrices as a starting point for gathering more input and developing recommendations
• Determine other tasks of interest to apply this work to
• Suggestions?