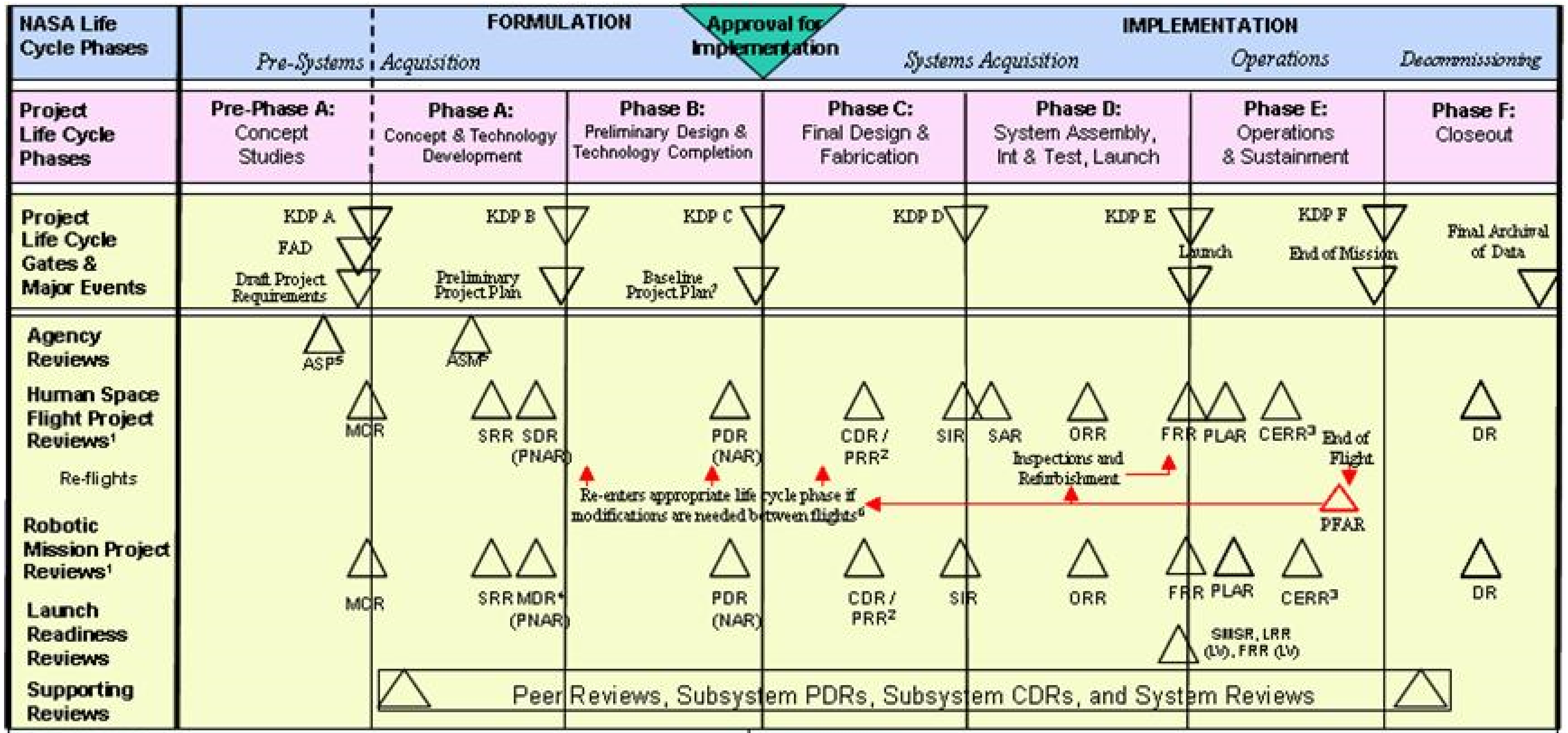
Two Systems Engineering Functions with SysML: The High and Low of MBSE Usability (Part 1)

Bjorn Cole

NASA Process (NPR 7120.5d)


NASA Life Cycle Phases
FORMULATION
Approval for Implementation
IMPLEMENTATION
Pre-Systems Acquisition
Systems Acquisition
Operations
Decommissioning
Project Life Cycle Phases
Pre-Phase A: Concept Studies
Phase A: Concept & Technology Development
Phase B: Preliminary Design & Technology Completion
Phase C: Final Design & Fabrication
Phase D: System Assembly, Int & Test, Launch
Phase E: Operations & Sustainment
Phase F: Closeout
Project Life Cycle Gates & Major Events
KDP A
FAD
Draft Project Requirements
KDP B
Preliminary Project Plan
KDP C
Baseline Project Plan7
KDP D
KDP E
Launch
KDP F
End of Mission
Final Archival of Data
Agency Reviews
ASP5
ASM5
Human Space Flight Project Reviews1
MCR
SRR
SDR (PNAR)
PDR (NAR)
CDR / PRR2
SIR
SAR
ORR
FRR
PLAR
CERR3
End of Flight
DR
Re-flights
Re-enters appropriate life cycle phase if modifications are needed between flights6
Inspections and Refurbishment
PFAR
Robotic Mission Project Reviews1
MCR
SRR
MDR4 (PNAR)
PDR (NAR)
CDR / PRR2
SIR
ORR
FRR
PLAR
CERR3
DR
Launch Readiness Reviews
SMSR, LRR (LV), FRR (LV)
Supporting Reviews
Peer Reviews, Subsystem PDRs, Subsystem CDRs, and System Reviews

FOOTNOTES

1. Flexibility is allowed in the timing, number, and content of reviews as long as the equivalent information is provided at each KDP and the approach is fully documented in the Project Plan. These reviews are conducted by the project for the independent SRB. See Section 2.5 and Table 2-6.
2. PRR needed for multiple (≥4) system copies. Timing is notional.
3. CERRs are established at the discretion of Program Offices.
4. For robotic missions, the SRR and the MDR may be combined.
5. The ASP and ASM are Agency reviews, not life-cycle reviews.
6. Includes recertification, as required.
7. Project Plans are baselined at KDP C and are reviewed and updated as required, to ensure project content, cost, and budget remain consistent.

ACRONYMS

ASP—Acquisition Strategy Planning Meeting
ASM—Acquisition Strategy Meeting
CDR—Critical Design Review
CERR—Critical Events Readiness Review
DR—Decommissioning Review
FAD—Formulation Authorization Document
FRR—Flight Readiness Review
KDP—Key Decision Point
LRR—Launch Readiness Review
MCR—Mission Concept Review
MDR—Mission Definition Review
NAR—Non-Advocate Review
ORR—Operational Readiness Review
PDR—Preliminary Design Review
PFAR—Post-Flight Assessment Review
PLAR—Post-Launch Assessment Review
PNAR—Preliminary Non-Advocate Review
PRR—Production Readiness Review
SAR—System Acceptance Review
SDR—System Definition Review
SIR—System Integration Review
SMSR—Safety and Mission Success Review
SRR—System Requirements Review

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

	SysML			Integration Tools		
	BDD	**IBD**	**Par**	**DB**	**Trans-formation**	**Spread - sheet**
Establish Resources						
Ease of Learning	Come packaged with notions of inheritance and abstraction that may require education		Depends on learning working style – can be both cumbersome (lots of blocks) or elucidating (which parameters interact)			
Efficiency of Use Routine	Levels of allocation easy to show in single diagram for generic components; capture values for standard practice on margins, etc.	Model the analysis environment to see where updates to values will come from	Can show “tree math” for different levels of hierarchy; many SE’s can be frustrated with “I just want a simple sum!”			Quick to build; usability fades as length of time using same spreadsheet grows
Efficiency of Use Non-Routine		Easy to show how components interact – some elements (structual loading, mechanical power transfer) not naturally represented as “flows”	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			Simple to develop, but often gets opaque once analysis is fnished
Error Frequency					Low once debugged	High; hard to track pedigree

Usability Matrix

	SysML			Integration Tools		
	Activity	**IBD**	**Par**	**DB**	**Trans-formation**	**Spread - sheet**
Track Resources						
Ease of Learning	Easily understood by Sys Engineers; may take a little extra time to master		Depends on learning working style – can be both cumbersome (lots of blocks) or elucidating (which parameters interact)		Requires a different mindset for use; is in common with XSL	
Efficiency of Use Routine	Model development environment to see where information will be developed and with which tools	Model the analysis environment to see where updates to values will come from	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!”	Concept is familiar; specifics of data entry / harvesting may be difficult	How does one provide a “tagged” handle of a property over to a disciplinarian for connection to his or her development effort?	Difficult to update, very manual process
Efficiency of Use Non-Routine			Previously captured sensitivity equations may be useful but hard to find / recall			
Error Frequency					Low once debugged	High; hard to track pedigree

Usability Matrix

	SysML			Integration Tools		
	BDD	IBD	Par	DB	Trans-formation	Spread - sheet
Barter Resources						
Ease of Learning			Depends on learning working style – can be both cumbersome (lots of blocks) or elucidating (which parameters interact)			
Efficiency of Use Routine		Show the paths for redirected surplus resources from one subsystem to others; examine current usage	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		Match resource use predicted by model into the same terms as the operating reports; generate reports and queries on current use	Nearly impossible unless the spreadsheet is in current use to understand how the system is modelled
Efficiency of Use Non-Routine		Identify damaged / reconfigured components; hard to mark up IBD				
Error Frequency					Low once debugged	High; hard to track pedigree

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?