Model-Based Systems Engineering:
What is it and why does it matter?

Steven Jenkins
Principal Engineer
Systems Engineering Division

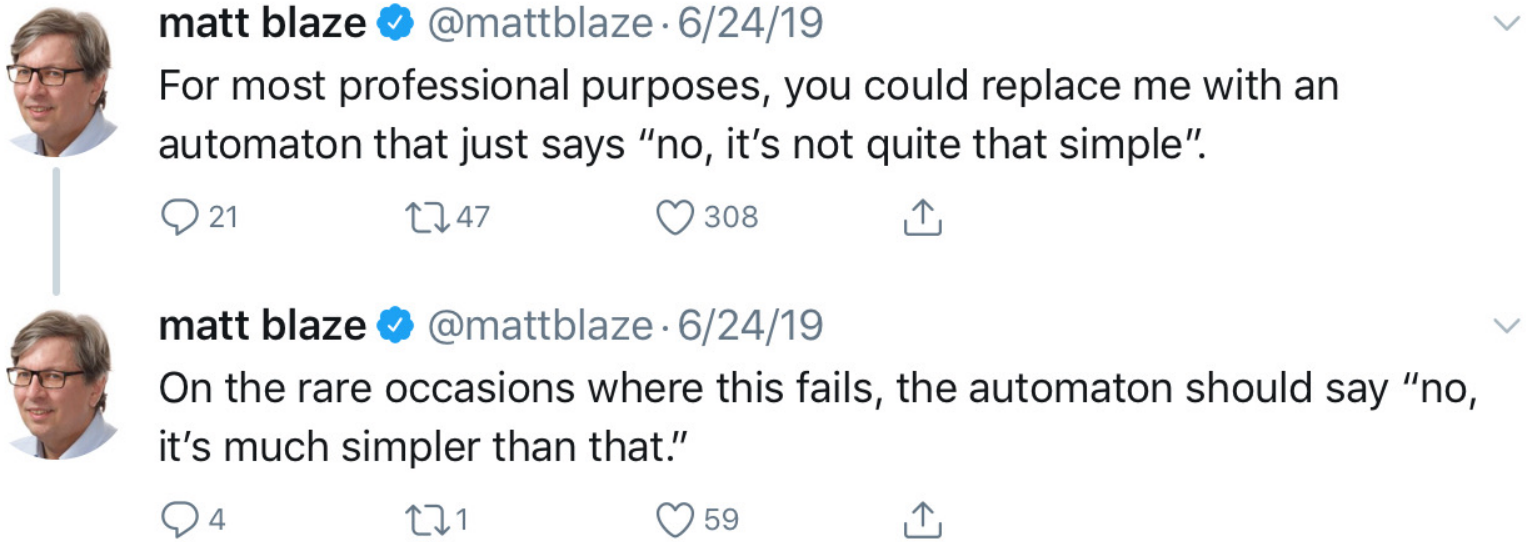
Jet Propulsion Laboratory
California Institute of Technology

Copyright © 2019 California Institute of Technology. U.S. Government sponsorship acknowledged. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government or the Jet Propulsion Laboratory, California Institute of Technology.

Preliminaries

- Who am I?
  - 30-year employee of JPL, most of that as a systems engineer
  - Experience on both engineering infrastructure (Deep Space Network, computer-aided engineering, information systems) and flight projects
  - Current focus on improving systems engineering practice through enhanced rigor and tools
  - Ph.D., Electrical Engineering (Control Systems), 1987, UCLA
  - No history with weapons systems, defense procurement, no security clearance
  - I use a lot of words, not so many pictures
- Why am I here?
  - I regularly give talks in which I scorn the current practices of SE
  - My goal, inspired by Ross Perot and the board of GM, is to be so annoying that people pay me $750M to go away
    - I'll go away for for less; make me an offer
  - Instead, people keep inviting me to speak

Preliminaries



- Today will be the latter

What is MBSE?

- There are a few definitions of MBSE out there
- I don't think they're particularly useful
- This is not a question you answer with definitions
  - What is MBSE?
  - System Engineering with Modeling
  - OK, what is modeling?
  - Building and using models
  - OK, what is a model?
  - and so on…
- I don't like the term MBSE and wish it would disappear
- We'll take the objective by a flanking maneuver

What should SE practice look like in the 21st century?

- Let's start from scratch and think about what we should be doing from two angles:
  - Demand: What does the job require?
  - Supply: What does the world provide for such challenges?
- Let's think more about principles that endure over time and less about today's particulars
- Let's remember that systems engineering *is engineering*
  - Specialized, not separated
  - We have to meet the standards of the profession as a whole

The nature of engineering

- I don't want to try to define engineering
- But I would like to characterize it as *creativity tempered by scientific and mathematical rigor*
- Anyone can be creative; rigor is the difference between

The bridge is held up by wires	The bridge is of hybrid cable-stayed/suspension design
It's long	Total length 1834 m
It looks strong	Total load 17000 t
It looks safe	Cable factor of safety 4.1

A few more thoughts on rigor

- Rigor is not a cost to be traded against benefit
  - It's what you do to make the results trustworthy
- *Rigorous* does not mean *detailed*
  - It means knowing how much detail you need
- Rigor is not just for complex endeavors
  - It's a discipline for addressing complexity rationally
- Rigor is not just for large projects
  - It's a discipline for addressing scale rationally
- In the end, rigor is adherence to high standards to avoid deceiving yourself and others
- As such, it requires no justification

The nature of systems engineering

- Systems engineers
  - guide the collaborative design of complex systems
  - in such a way that a recommended set of design options
  - can be shown by analysis
  - to exhibit behavior that achieves mission objectives
  - subject to applicable constraints
  - with acceptably high confidence
- Two key facets:
  - Collaboration
    - Systems engineering by its nature focuses on endeavors requiring teams–sometimes numbering thousands or more–to achieve
  - Analysis
    - In the end we have to do the math, physics, chemistry, economics, politics, sociology, whatever it takes
- And behind it all, rigor

More effective collaboration and analysis

- Collaboration begins with communication, and communication begins with *language*
  - Vocabulary and grammar rules for concepts, properties, and relationships of interest
- The heart of engineering analysis is the use of *abstractions*
  - Geometry, analytic geometry, differential equations, etc.
  - Plus other not-so-traditional formalisms
- Both collaboration and analysis are greatly empowered by effective *automation*
  - Information storage, interchange, presentation, visualization
  - Computation
- Let's examine language, abstraction, and automation in established fields of engineering and systems engineering

Electrical engineering language

- EE has long employed regular language and notation for
  - Names of "elemental" objects: resistor, capacitor, inductor
  - Properties of elements: resistance, capacitance, inductance
  - Metrology: volt, ampere, ohm, henry, farad
  - Composition of elements into aggregates: L-network, Π-network, transmission line, amplifier
  - Properties of aggregates: impedance, loss, cutoff frequency
- Some EE languages are amenable to computer processing
  - VHDL, SPICE, Modelica, etc.
  - Around each language is a community that
    - Understands and improves the language
    - Uses the language to communicate designs, analyses, etc.
    - Creates economic value in the form of products and services
  - Human creativity is unleashed by common language

Systems engineering language

- Despite considerable progress in the SysML community, there is not yet widespread consensus on a unifying language for systems engineering
  - More on SysML later
- Part of the challenge is that the scope of systems engineering knowledge is for all practical purposes unbounded
  - Bigger than electrical, mechanical, thermal, telecom, etc.
- There is a wealth of published information (e.g., SEBoK) with definitions of terms and box-and-line drawings that describe *practices* in detail, but
- We still lack the facility for Collaborator A to send, e.g., a system breakdown structure to Collaborator B in such a way that B can unambiguously recognize it as a system breakdown and verify its well-formedness
  - Whereas VHDL, for example, has been used to exchange design data for years

Some helpful advice

A scientific discipline emerges with the — usually rather slow!— discovery of which aspects can be meaningfully "studied in isolation for the sake of their own consistency", in other words: with the discovery of useful and helpful concepts.

E. W. Dijkstra
On the role of scientific thought
1974

A useful analog in physics is *energy*. No one can define precisely what energy *is* and we really don't care. What we care about are its properties, e.g., thermodynamics

Some useful and helpful concepts in systems engineering

- Things
  - Component
  - Function
  - Interface
  - Requirement
- Properties
  - identifier
  - mass
  - deviation
- Relationships
  - contains (Component contains Component)
  - performs (Component performs Function)
  - presents (Component presents Interface)

Electrical engineering abstractions

- Since the beginning
  - Calculus: differential equations, Fourier series, transfer functions
- More recently:
  - Measure theory: Hilbert space, functional analysis
  - Probability: Information theory, entropy, correlation, power spectrum
- Even more recently:
  - Abstract algebra: Galois fields
  - Probably lots of things I've never heard of

Language and abstractions are closely related

- What is a capacitor?
- We call things capacitors, but it's not a capacitor just because we call it that.
- A thing is a capacitor if and only if it exhibits *capacitance*
- And what is capacitance? A particular relationship between voltage and current described by a well-known *abstraction*:
  - $I = C\frac{dV}{dt}$
- Engineering language uses abstractions to link structure and behavior
  - An RC low-pass filter has transfer function $\frac{V_{out}}{V_{in}} = \frac{1}{j\omega RC+1}$
  - A Reed-Solomon (225, 223) 8-bit error-correcting code can correct up to 16 symbol errors per block

Terms that convey structure and behavior

- I-beam
- Truss
- Suspension bridge
- Full-wave rectifier
- Class C amplifier
- Butterworth filter
- Non-uniform rational B-splines curve
- Airfoil
- Heat pipe
- Stirling engine

What are the abstractions of systems engineering?

- As we observed before, systems engineering employs some more-or-less familiar language
- We don't, however, in general, associate these terms with abstractions in any systematic way
- Let's take a look a simple example: System Breakdown Structure

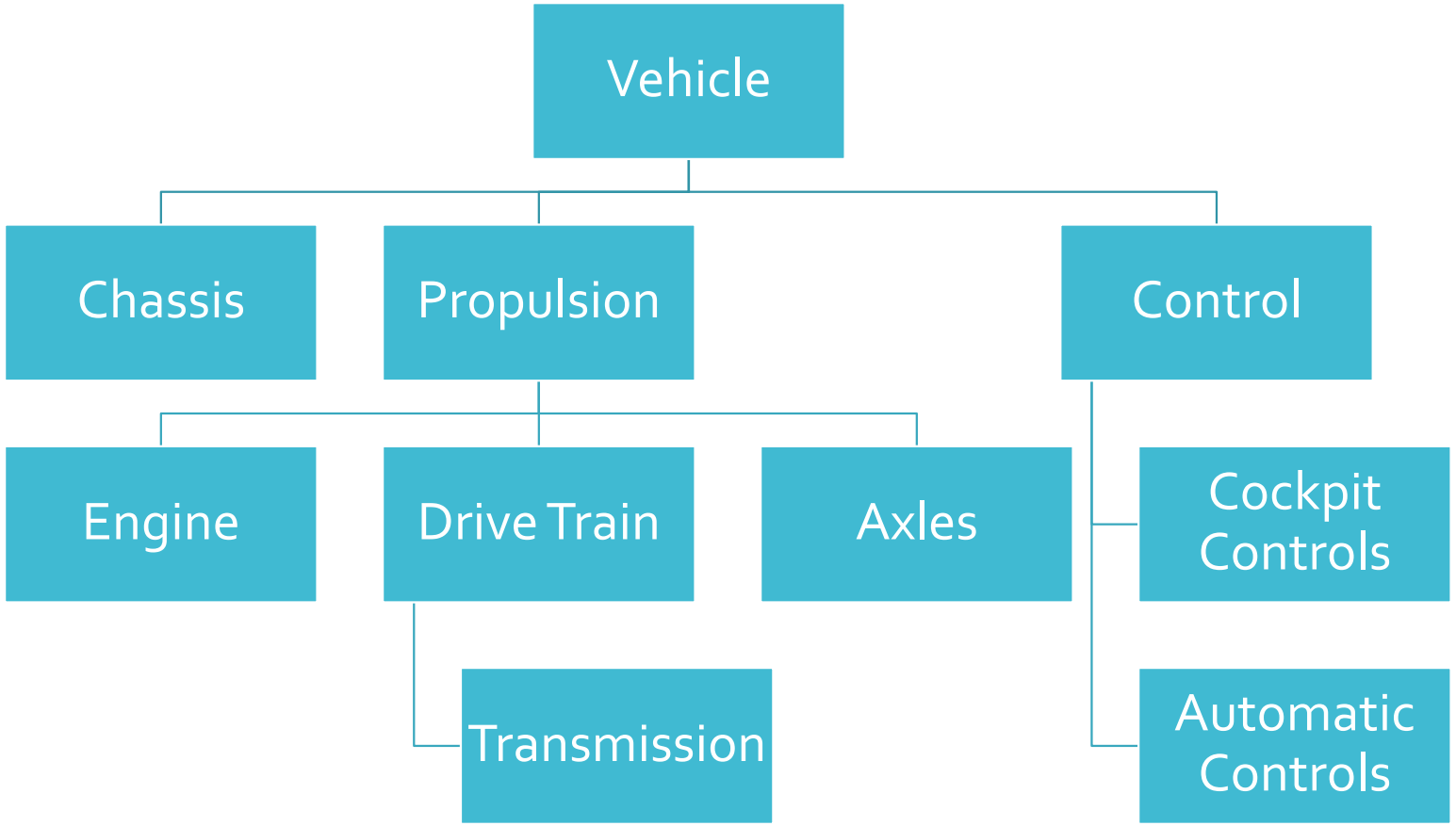
System Breakdown Structure

- IEEE 1220-2005 defines System Breakdown Structure as [*a*] *hierarchy of elements, related life cycle processes, and personnel used to assign development teams, conduct technical reviews, and to partition out the assigned work and associated resource allocations to each of the tasks necessary to accomplish the objectives of the project*
- Let's focus on the first four words….
- What is *a hierarchy of elements*?
  - In particular, what mathematical abstraction represents a hierarchy?
- We can't know for sure because the phrase itself and the word *hierarchy* don't have precise meaning
- But we can think about it

What are useful attributes of a system breakdown?

- Every element of the system appears at least once
  - That is, we don't omit anything
- Every element of the system appears at most once
  - That is, we don't over count mass, power, cost, whatever
- Every element except a single top-level element is a part of a single parent element
- The top element is a part of no parent element

It looks something like this



System breakdown as a graph

- Mathematical *graph theory* is the study of pairwise relations between objects
  - Such as the *is-a-part-of* relation in a system breakdown
- We can express the desired attributes of a system breakdown as properties of a graph
  - Specifically, a *rooted tree*

Desired Attribute	Graph Property
Every element of the system appears at least once	connected
Every element of the system appears at most once	acyclic
Every element except a single top-level element is a part of a parent element	directed
The top element is a part of no parent element	rooted

So What?

- Any graph can be represented by an *edge list*
  - In our case, a set of [child, parent] pairs
- We can agree on an *exchange syntax* for a System Breakdown Structure (e.g., a CSV file) such that
  - Two or more parties can exchange SBS without loss of information
  - Any party can readily verify that the SBS is well-formed
- That is we've created a fragment of language that describes an important concept based in mathematical formalism
- A side benefit—that turns out to be tremendously important—is that graph theory is an important foundation of computer science
- There is a great deal of high-quality, high-performance software for operating on graphs
  - E.g., calculating the mass of a system by rolling up leaf masses
- Which brings us to…

Engineering automation

- Computation is an essential component of modern electrical engineering
  - Semiconductor design and manufacturing
  - Digital signal processing design and implementation
  - Control system synthesis and implementation
  - Machine learning
- Likewise in mechanical engineering
  - CAD/CAM
  - Structural analysis
  - Fluid dynamics
- Heavy, widespread, routine use of computation for
  - Design capture
  - Inferring performance from design
  - Synthesizing design from requirements

What about systems engineering automation?

- Widespread use of various tools for design capture
  - Office automation tools: Word, Excel, etc.
  - Drawing tools: Visio, etc.
  - Modeling tools: MagicDraw, Enterprise Architect, etc.
- Somewhat less widespread use of analysis tools
  - Modeling tools
  - Mathematics tools: Matlab, Mathematica, etc.
  - Simulation tools: Simulink, etc.
- Less widespread use of design synthesis tools
- More importantly (my observation): SE as a discipline lacks doctrinal language and abstractions to make effective use of automation
- Drawing boxes and lines may be "modeling", but is it useful?

An important note about automation

- Computers are valued for their scale and speed
  - Storing vast amounts of information
  - Performing extensive and expensive calculations
- But that presumes a feature that is taken for granted: reliability
- If computers made errors at human rates, scale and speed would not be virtues
- Automation is important for rigor because it gives us mechanisms to ensure that operations are performed precisely to specification without fatigue, inattention, subterfuge, etc.
- Automation is not just, or primarily, about going fast
- It's about making *vastly* fewer mistakes

Summary so far

- We talked about the importance of rigor in engineering
- We talked about three manifestations of rigor
  - Language
  - Abstraction
  - Automation
- We talked about the use of language, abstraction, and automation in mature fields of engineering and the lack thereof in systems engineering
- So where do we go from here?

Improving systems engineering language: semantic web

- There is a large, worldwide community devoted to the development of the Semantic Web

> The ultimate goal of the Web of data is to enable computers to do more useful work and to develop systems that can support trusted interactions over the network. The term "Semantic Web" refers to W3C's vision of the Web of linked data. Semantic Web technologies enable people to create data stores on the Web, build vocabularies, and write rules for handling data.

- Building vocabularies and writing rules *is* defining language
- Work is under way in the INCOSE Patterns Working Group to specify systems engineering patterns (e.g., decomposition, interconnection, specification) in OWL, the knowledge representation standard of the Semantic Web
  - Incorporating work done at JPL over the past decade
- The result will be standards-based language for SE

The Semantic Web standards are a clear win

- Strong grounding in formal mathematics and computer science
  - The expressivity of OWL is precisely specified
  - The computational complexity of machine reasoning over OWL is well-understood
  - OWL occupies a "sweet spot" enhancing cooperation of humans and computers in exploiting knowledge
- Large, active community
  - Theoreticians
  - Software developers (both open source and commercial)
  - Implementers
- Large body of experience in varied fields
  - Biology
  - Aerospace
  - Financial industry
  - Others

Improving systems engineering language: SysML v2

- Systems Modeling Language v1, a profile of the Unified Modeling Language, was primarily a *notation*, not a formal language as understood by computer scientist
- SysML v1 is a *very* expressive language
  - It's easy to say things in SysML v1 whose meaning is unspecified
- Interchange of SysML v1 models across tools is not simple
- The Object Management Group issued a Request for Proposal for SysML v2 December 2017
- SysML v2 will specifically address model interchange and formal semantics
  - Specifically, mapping to and from OML
- The INCOSE Patterns Working Group and SysML v2 Submission Team are collaborating

Improving systems engineering abstraction: graph theory

- This one is easy
- The Semantic Web is based on, among other things, graph theory
  - Information in the Semantic Web is organized into *knowledge graphs*
- Effective, principled use of OWL (and SysML v2) implies effective use of graph theory
  - Sometimes without knowing it
- Remember how language and abstraction are closely related?
- The more we express knowledge in OWL, the better we can store and retrieve it, exchange it, reason about it
  - That is, the more valuable it becomes

Improving systems engineering automation

- Again, it's easy
- In general, if we adopt *some* formal language and *some* formal abstractions, we've done most of the up-front work to prepare for automation
- If we adopt OWL and SysML v2 in particular, we inherit graph formalisms *and* numerous actual implementations of software for
  - Vocabulary development
  - Knowledge collection
  - Large-scale storage and retrieval
  - Machine reasoning
  - Document synthesis
  - Custom application development
- There is a great deal of value here for the taking

So, back to MBSE

- The word *modeling* is too vague to be useful
- But we can think of it as a shorthand for certain sound practices
  - Formal language
  - Mathematic abstraction
  - Automation
- If we engage these practices in principled ways, we can say we're doing model-based systems engineering
- If we don't, I'm not sure we can say we're *engineering*
  - at least, not in a credible way in 2019
- Systems engineering as a discipline has a long way to go but the path is clear and we're moving in the right direction
- It's up to all of us to push for progress