



National Aeronautics and  
Space Administration

**Jet Propulsion Laboratory**  
California Institute of Technology  
Pasadena, California

# What, If Anything, Is Systems Engineering?

**Steven Jenkins**  
**Systems Engineering and Formulation Division**  
**Jet Propulsion Laboratory**  
**California Institute of Technology**



Copyright 2017 California Institute of  
Technology. U.S. Government sponsorship  
acknowledged.



# What, If Anything, Is the Title About?

- **Wood 1957, *What, If Anything, Is a Rabbit?***
  - Do rabbits and their relatives form a distinct order, not closely related to rodents?
- **Gould 1983, *What, If Anything, Is a Zebra?***
  - Do the three species of zebras form a single evolutionary unit?
- **Jenkins 2017, *What, If Anything, Is Systems Engineering?***
  - What kinds of problems do systems engineers work on?
  - How do systems engineers work?



National Aeronautics and  
Space Administration

**Jet Propulsion Laboratory**  
California Institute of Technology  
Pasadena, California

## I. What Kinds of Problems Do Systems Engineers Work On?



# Systems Engineering Problems

- **There's no point in trying to develop a rigorous taxonomy of problem types**
  - Real problems don't fit neatly into categories
- **There are differences in emphasis, however**
- **I will illustrate some differences with four examples from one of the greatest (and least known) systems engineering successes in human history**
  - It saves millions of lives per year
  - It cost less than \$300M US total
  - It was achieved through a combination of science, engineering, politics, and psychology, assisted by humanitarianism, cleverness, guile,... and luck
- **The people who did it didn't consider themselves systems engineers, but I do**
- **What was it?**



# Eradication of Smallpox

- **Inoculation with smallpox, known to provide some benefits, practiced in 18<sup>th</sup> century and before**
- **Smallpox vaccine (from cowpox) reported 1797**
- **Many local and regional eradication efforts after that**
- **Eradication achieved in US and industrialized Europe by early 20<sup>th</sup> Century**
  - **Nevertheless: 300-500 million deaths in 20<sup>th</sup> Century**
- **Coordinated global effort begin in 1959 with adoption of WHA11.55 by the World Health Assembly**
- **Slow progress until 1966 with formation of the Smallpox Eradication Unit by the World Health Organization**
- **By 1975 smallpox is endemic only in the Horn of Africa**
- **Last naturally-occurring indigenous case worldwide was in Somalia in 1977**
- **Eradication officially declared in 1980**



# Problem I: Create a Machine

- **Smallpox vaccine is a liquid, administered via scratches in the skin**
- **We need a machine to deliver the proper dose**
  - **We need at least thousands of units**
  - **It must function reliably under third-world conditions**
  - **It must be operable by workers with minimal training**
  - **It must not infect patients with other diseases**
- **A variety of solutions employed, with varying success, until the 1960s**
- **In 1966, American microbiologist Benjamin Rubin developed the *bifurcated needle***
  - **one of the greatest machines in human history**



# The Bifurcated Needle

- **No moving parts**
- **\$0.005 each to produce**
- **Multiple-decade lifetime**
- **No credible intrinsic failure modes**
- **Meters dose by capillary action**
- **Very low vaccine waste**
- **Sterilized after each use**
  - low risk of infection
- **Delivered several hundred million doses**
- **Critical component of eventual mission success**





## More on Creating a Machine

- **Creating a machine is a canonical systems engineering problem**
  - but probably thought of as *the* canonical problem too often
- **Understanding what the machine (or system) has to do (i.e., knowing its stakeholders and their concerns) is the heart of the systems engineering process**
- **This understanding is often gained through an extended period of engagement with stakeholders, who**
  - may not be able to articulate their concerns well
  - may be unaware of other stakeholders and concerns
  - may not understand how their concerns conflict with others
- **The rest is *merely* engineering design**
  - The remaining design problem may be extraordinarily challenging (e.g., Apollo spacecraft)



## Problem II: Create a Capability

- **It's not enough to have the means to vaccinate**
- **We need to deploy and maintain**
  - **trained workers (thousands)**
  - **needles (millions) and sterilization gear**
  - **vaccine doses (hundreds of millions)**
  - **educational materials (hundreds of millions)**
- **We need an operational infrastructure to transport**
  - **resources and directions to the field**
  - **status and requests from the field**
- **We have to coordinate with**
  - **local leaders, diplomats, health officials**
- **The operational capability must persist for decades**
- **A failure in the field (e.g., infection) can damage confidence in the entire program**



## More on Creating a Capability

- **For a long-lived program, the nature of the operational capability may change radically**
- **Most of the vaccine in the early years was provided by the United States and the Soviet Union**
- **By 1973 80% of all vaccine was produced in developing countries**
- **This is a good example of a higher-order effect**
- **Vaccine is vaccine, except**
  - **Contamination or ineffectiveness risk health, lives, and program success**
    - **arguing for centralized production**
  - **Local production may enhance program participation and strengthen local public health infrastructure in general**
    - **arguing for distributed production**
- **Proper systems engineering accounts for these secondary effects, sometimes exploiting them for gain**



## Problem III: Create a Strategy

- **It's not enough to have a capability that can be fielded**
- **We have to have a strategy for where and when to field**
  - **Where we can be most effective?**
  - **Where the need is greatest?**
  - **Those often don't overlap**
- **There are many immutable environmental constraints**
  - **Somalia was embroiled in civil war 1974-1991**
  - **Somalia and Ethiopia were at war 1977-1978**
  - **Poverty creates its own complications**
- **Information from the field may be unreliable**
  - **Developing nations trying to attract investment were reluctant to report outbreaks**
- **Costs and direct benefits are not co-located**
- **A great exercise in decision-making under uncertainty**
  - **Little of the uncertainty is statistical in nature....**



## More on Creating a Strategy

- **Smallpox has some specific properties that helped to make it the first (and so far, only) endemic human disease to be eradicated**
  - **It spreads only by direct human-to-human contact**
  - **Average incubation period is 12 days**
  - **Victims are not contagious during incubation**
  - **There is time to break the chain after infection**
- **These facts played into a major re-thinking of strategy in 1966**
  - **Since 1958 the strategy had been worldwide mass vaccination to achieve herd immunity**
  - **New strategy was based on surveillance and rapid-response containment**
  - **This required reworking much of the field capability**
- **Original strategy would probably not have succeeded**
- **New strategy led directly to mission success**



## Problem IV: Create a Culture

- **The world's position on smallpox eradication went through distinct phases:**
  - **1800: A Utopian vision**
  - **1958: We're in favor of it**
  - **1966: Let's put some muscle behind it and finish it off**
  - **1979: We seem to have done it**
  - **1980: We did it!**
- **This progression was not by accident; the culture for smallpox eradication was the result of sustained effort of key leaders, including D. A. Henderson**
  - **in my opinion, the most unsung hero in world history**
- **There were numerous close calls with total failure**
- **Solving such problems involves everything we know: science, psychology, economics, politics, culture, etc.**
- **Is it Systems Engineering? I don't see why not.**



## More on Creating a Culture

- **All known smallpox stocks are currently housed at the US Center for Disease Control and Prevention and the Russian State Research Center of Virology and Biotechnology**
- **The World Health Organization has recommended destruction of the remaining virus since 1986**
- **Resistance from the US and Russia has prevented destruction to the present day**
  - **Destruction would undoubtedly reduce risk of accidental reintroduction**
  - **Experts agree no public health purpose is served by retention of virus stocks**
  - **and yet, virus stocks are retained**
- **The Culture Creation challenge continues**
  - **This kind of problem does not yield to pure technical analysis and optimization**



National Aeronautics and  
Space Administration

**Jet Propulsion Laboratory**  
California Institute of Technology  
Pasadena, California

## II. How Do Systems Engineers Work?

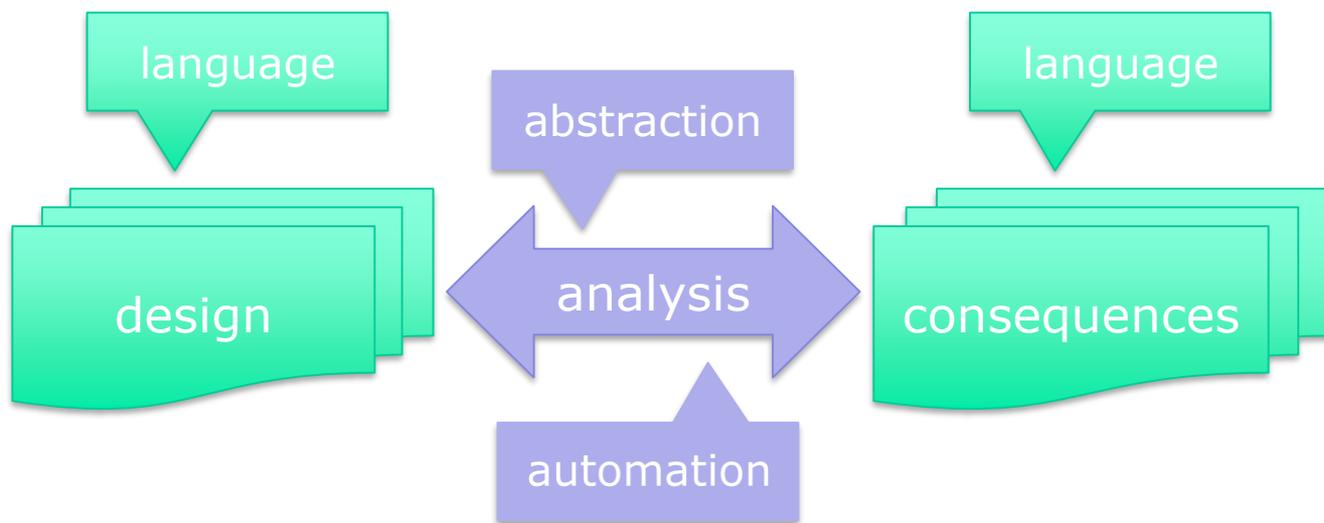


# How Do Systems Engineers Work?

- **Before talking about how Systems Engineers work, let's think about Engineers in general work**
- **Engineers do two complementary things:**
  - **they *describe* actual and imagined states of the world**
    - **actual states are facts**
    - **imagined states are designs and consequences**
  - **they *analyze* these descriptions**
    - **What are the consequences of a specified design?**
    - **What designs have a specified set of consequences?**
- **Engineering analysis is distinguished by its reliance on science and mathematics to achieve rigor**
- **Engineering rigor manifests in three dimensions:**
  - **language**
  - **abstraction**
  - **automation**



# Language, Abstraction, Automation





# Language

- **We can't analyze what we can't describe**
- **We can't describe precisely without precise language**
- **Electrical engineering, for example, defines precise descriptive terms like *resistor, capacitor, filter, rectifier, amplifier, etc.***
  - **and taxonomic relationships, e.g., *low-pass filter* specializes *filter***
- **From these terms we can compose circuit descriptions in an analysis language like SPICE netlist format**
  - **and then SPICE can tell us how the circuits behave**
- **Precise languages generally manifest three things:**
  - **vocabulary**
  - **syntax: rules for constructing sentences**
  - **semantics: meaning in the real world**
- **Engineering semantics are about *analysis***



# Abstraction

- **Abstractions are the key to analysis**
- **For example, an RC filter can be modeled by a linear ordinary differential equation**
  - **The equation is an abstraction in that it is a purely mathematical description of *idealized* behavior**
  - **We can perform operations on this abstraction; in fact we can *solve* it**
  - **The solution is a useful approximation of the *actual* behavior of the filter**
- **Mathematical analysis is a hallmark of engineering**
  - **Everything else is poetry or marketing or ....**
- **The scope of applicable math has enlarged over time**
  - **No longer just calculus, linear algebra and probability**
  - **Now formal logic, graph theory, abstract algebra, etc.**
  - **For example, error-correcting codes employ algebra proudly thought to be useless until the 20<sup>th</sup> century**



# Language and Abstraction

- **What is a capacitor?**
  - Is it necessarily a discrete component?
- **A better definition: *something that exhibits capacitance***
- **And what is capacitance?**
  - A particular analytical relationship between voltage and current, represented by an ordinary differential equation
- **This is a key linkage: language concepts are fundamentally tied to analysis via abstraction**
- **This is generally true throughout engineering: our language allows us to describe things in such a way that analytical consequences directly follow**
  - “directly” may involve work but the implication is direct
- **Abstractions shape language and vice versa**
  - e.g., Modelica is all about differential-algebraic equations



# Automation

- **Automation is critical for engineering because it preserves rigor: scrupulous adherence to the highest standards for the conduct of work**
  - **Machines don't cut corners**
- **Automation has its own abstractions (e.g., algorithms, data structures)**
- **These abstractions can be mapped to the abstractions of engineering analysis**
  - **transitive closure maps to root cause analysis**
- **Automation is fundamental to modern engineering because**
  - **Well-designed languages are amenable to machine parsing**
  - **Many useful mathematical abstractions and related analyses are implemented in software libraries**
  - **Derivation of consequences of design can be automated**
  - **Design synthesis can be automated**



# Is Systems Engineering Really Engineering?

- **Systems engineers describe and analyze, but how well do we do it?**
- **Do we use precise language?**
- **Do we employ abstractions to empower analysis?**
- **Do we automate effectively?**
- **How can we do better?**



# Systems Engineering Language

- **It's fair to say that systems engineering employs distinct concepts: component, function, interface, requirement, risk, etc.**
- **It's also fair to say that we use some words frequently without being very clear about meaning**
  - *system vs subsystem* is subtle to pin down
- **We lack**
  - **agreement on names and definitions for these concepts**
    - I call it *component*, what do you call it?
  - **agreement on names and definitions for relationships**
    - **What's the name of the relationship between a component and a function? Between a component and an interface? Between a requirement and a component?**
  - **agreement on names and definitions for properties**
    - *estimated mass*, for example, is not really a property of a component, it's a constraint on the component's mass



# How Can We Do Better?

- **First and foremost, recognize that there is well-established field of theory, practice, and technology dedicated to precise representation of knowledge**
  - called (obviously) *Knowledge Representation*
- **Use the tools of Knowledge Representation and the Semantic Web to build communities of consensus around systems engineering language usage**
  - captured in formal ontologies
- **Incorporate this consensus into, not just tools and software, but human language**
  - We should *talk to each other* using our language
- **Incorporate this consensus into tools and software**
  - Particularly, SysML
- **Reject ambiguity from our practices**
  - Being precise about uncertainty is good
  - Being ambiguous about anything is not



# Systems Engineering Abstractions

- **It's fair to say that systems engineering doesn't yet recognize a fundamental set of abstractions**
  - **Unlike, say, control theory, which is strongly founded on functional analysis**
- **This is partly due to the fact that the scope of systems engineering is very broad (from machines to culture)**
- **The breadth of the subject matter can suggest to us that there is something fundamental to systems engineering about capturing a diverse set of facts and relating diverse concepts to each other**
- **What abstractions empower this activity?**

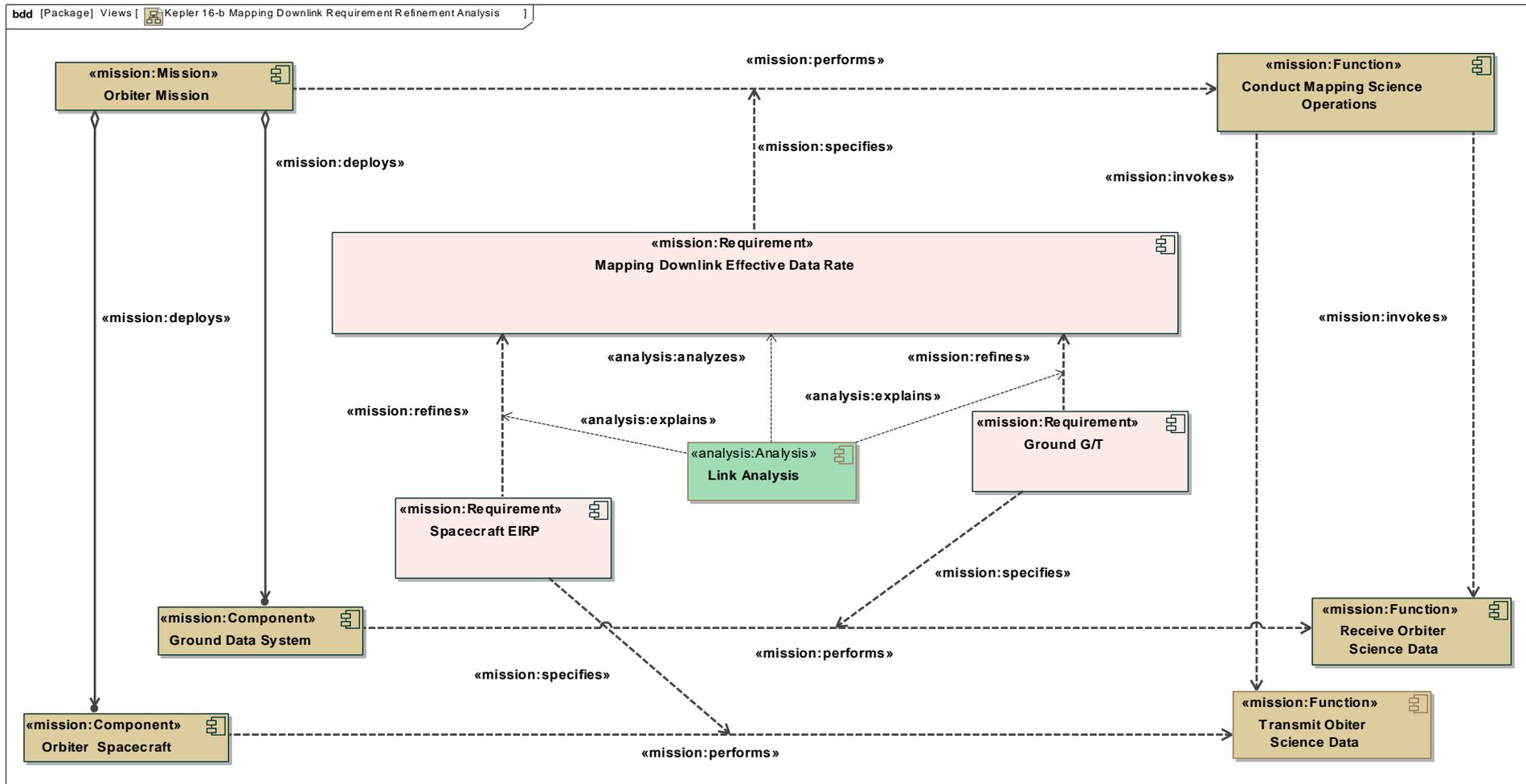


# How Can We Do Better?

- **Recognize that graph theory is the mathematical study of graphs, which represent pairwise relations between objects**
- **Knowledge representation theory makes heavy use of graphs**
- **We can use graph theory to structure and organize the facts (language assertions) about the objects of our design and analysis**
  - **We can reason about whether the resulting graph is well-formed according to the rules of our language**
  - **We can reason about all kinds and degrees of relatedness**
    - **e.g., What requirements does this requirement directly refine?**
    - **Indirectly?**
- **Well-known graph algorithms have direct application**
  - **connected components, transitive closure, topological sort**



# Example: Knowledge as a Graph





# Systems Engineering Automation

- **In the lifetime of the Systems Engineering discipline, computing has gone from a scarce, precious resource to a commodity sold in bulk**
- **It doesn't seem that the discipline has taken this fact into account**
- **How are our methods today different from those of 1970?**
- **Fortunately, well-designed languages and abstractions lend themselves directly to automation**
  - machine parsing
  - graph analysis
  - logical reasoning
  - query answering
  - search
  - planning and scheduling



# Summary

- **Systems Engineering has as its subject matter everything required to accomplish something difficult**
  - from designing machines to changing culture
- **Systems Engineers, like all engineers**
  - describe states of the world
  - associate design with consequences through analysis
- **Rigor in description and analysis makes use of**
  - precise language with rules and meaning
  - mathematical abstractions
  - automation
- **Graph theory is fundamentally empowering for all three**