Systems Thinking to Solve Complex Problems

In this module[1], you will learn a systems perspective that enables you to employ a holistic perspective to explore alternative solutions to engineering problems. A systems approach will naturally tap your curiosity and help you make connections between technologies, disciplines, and adjacent systems. A systems approach is key to avoiding many of the typical failure modes that occur when the prominent mindset is only at the component level. Systems thinking is essential to optimize value creation.

After defining terms and considering systems concepts, this module explores opportunities to solve problems by taking a systems perspective that promotes understanding connections within and adjacent to the system under development. The module outlines a systems engineering process designed to encompass all aspects of the problem, on multiple levels of the systems hierarchy.

At the end of the module, you will be expected to demonstrate how curiously exploring your system from various perspectives, including technical feasibility, value, risk, and societal impact enables you to make connections on the way toward creating value.

It should take you no more than 3 hours to complete all the activities within the module. It is suggested that you go through the module in sequence first, but after that, you may review the lessons in the module.

Introduction

The Applying Systems Thinking to Solve Complex Problems module illustrates the systems-approach when the problems are ill defined or multi-tiered in complexity. It can be used to apply some basic tools such as function mapping, decomposition and heuristic rules to make complex problems not so complex. The tools also help users avoid common mistakes such as mismatched interfaces between sub-systems.

The first part of module will be fairly straightforward, where key concepts and definitions will be reviewed. In the second part of the module, you will learn tool and techniques you can use to apply systems thinking to complex problems.

Learning Outcomes

After completing this module, you will be able to:

1. Define system, systems architecture, and system engineering

[1] Based on one of the KEEN-University of New Haven modules. licensed under Creative Commons (CC BY-NC-SA).

2. Decompose system hierarchy to at least four levels
3. Define any system from various perspectives, including technical feasibility, value, risk, and societal impact
4. Apply a heuristic architecting method to develop a system architecture

What is Systems Thinking?

"Systems thinking is a discipline for seeing wholes. It is a framework for seeing interrelationships rather than things, for seeing patterns of change rather than static "snapshots." It is a set of general principles— distilled over the course of the twentieth century, spanning fields as diverse as the physical and social sciences, engineering, and management....During the last thirty years, these tools have been applied to understand a wide range of corporate, urban, regional, economic, political, ecological, and even psychological systems. And systems thinking is a sensibility — for the subtle interconnectedness that gives living systems their unique character."

- Peter Senge, *The Fifth Discipline*

This course is intended to introduce you to the concept of Systems Thinking and show you how to apply it.

Your TTR Assignment

In this module, you will learn about systems thinking. You will have the opportunity to use what you have learned and apply the heuristic architecting method to develop a system architecture for a product your (hypothetical) company has been hired to design.
Right now, all you know is that TTR inc. (which stands for Tall Tree Research inc.) from Brea, California has hired your company to design a system that helps them study the effects of climate change by studying the tops of trees. The hypothetical project from this hypothetical company will be used to illustrate some concepts.

Module Lessons

Now that you've learned a little bit about the module, begin reviewing the lessons that follow. There are 5 lessons:

Lesson 1: Foundational Concepts
Lesson 2: Key Principles of Systems Engineering
Lesson 3: Architecture Development
Lesson 4: Multiple Views of a System
Lesson 5: Systems Verification & Validation

Lesson 1: Foundational Concepts

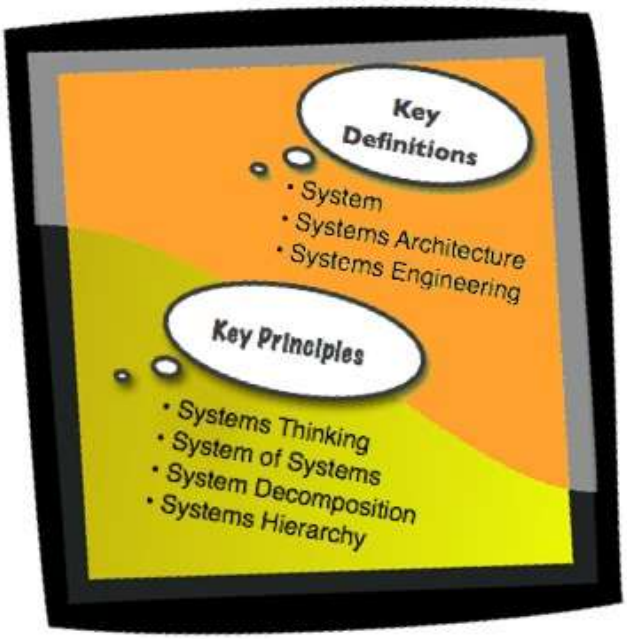
In this lesson you will explore foundational concepts of systems thinking, including key definitions of systems, systems architecture and systems engineering. You will also start to brainstorm some concepts for the TTR problem that was just introduced.

Introduction

Let's review the foundational concepts including key definitions of systems, systems architecture and systems engineering.

After we work through the foundational concepts, we will also discuss key principles related to systems: systems thinking, system of systems, system decomposition and system hierarchy.

Please review these items in the sequence presented, because they build on each other. We will look at the definitions first, and then the key principles.



The Problem Statement

“The formulation of a problem is far more often essential than its solution, which may be merely a matter of mathematical or experimental skill.”

- Albert Einstein

Great thinkers recognized the importance of the problem statement. The next few sections introduce you to some fundamental concepts, which will provide the foundation looking at problems from a systems perspective.

Form and Function

Engineers are trained to look at the world in very specific terms. Let's do a quick exercise to learn about form and function.

What is the object you see above?

Engineers are also trained to predict how a product will behave (i.e. function).

Form Follows Function

Engineers, like many people, often confuse form & function.

How did your answers compare to those below?

FORM is what the item **IS:**

In this case, form is:

Rigid brace with elastic band and flexible, inelastic strap, and a stone.

FUNCTION is what the item **DOES**:

In this case, the function is:

- Steadies projectiles.
- Aims projectiles.
- Accelerate projectiles.

This item could also function as: a hammer, kindling wood, banding things together, whip, croquet wicket, brace, etc.

“Form follows function.”

- Frank Lloyd Wright

This concept is critical as we learn more about Systems Thinking and Systems Engineering.

Functions can be accomplished by alternative forms.

These functions could also be delivered by: a gun, a crossbow, a cannon, a trebuchet, a rocket launcher, a blow gun, a human, etc.

A Definition of Function

A common definition of "function" is useful for defining needs and opportunities.

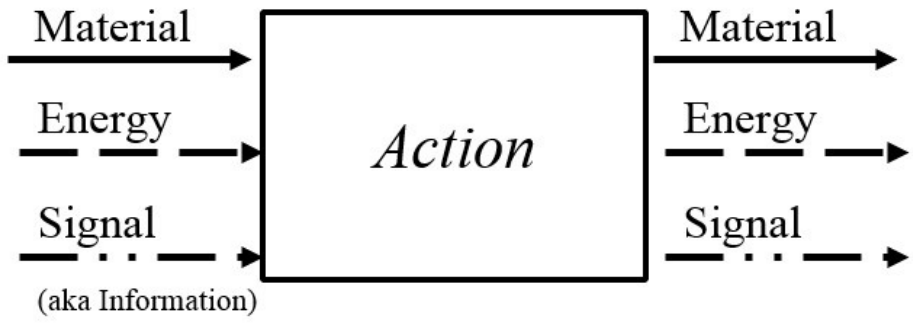
Function: Function is the logical transformation of a quantity (or set of quantities) from some initial condition (or state) to another condition (or state).

Also Known As: “Transfer Functions,” “Flow Charts” and “Parameter Diagrams”

Flows

The standard function element is used to convert from activities to functions.

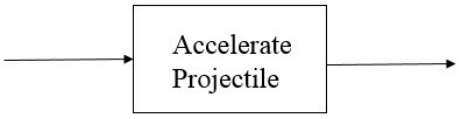
Three “Flows” are recognized as standard in the literature.

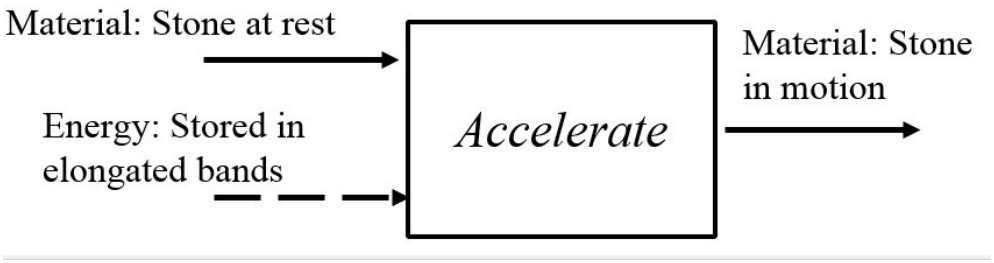


To date, there is limited agreement on standard “Actions.”

Accelerate

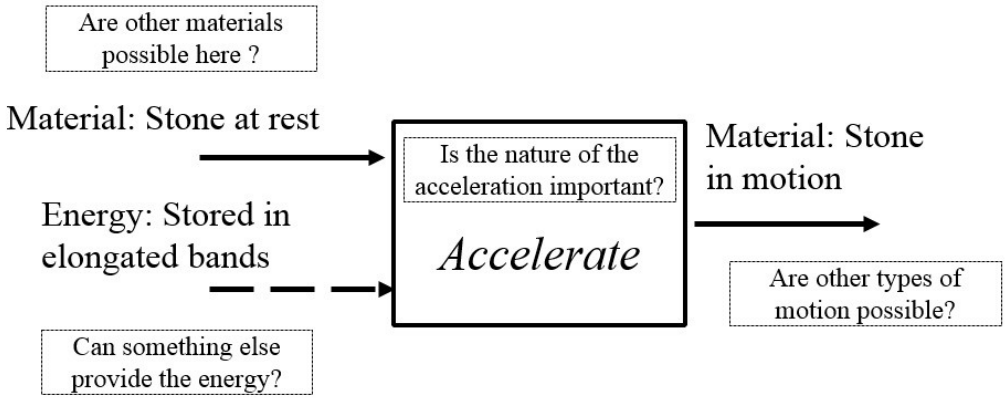
Below is an example of a standard element that represents one of the slingshot's functions.





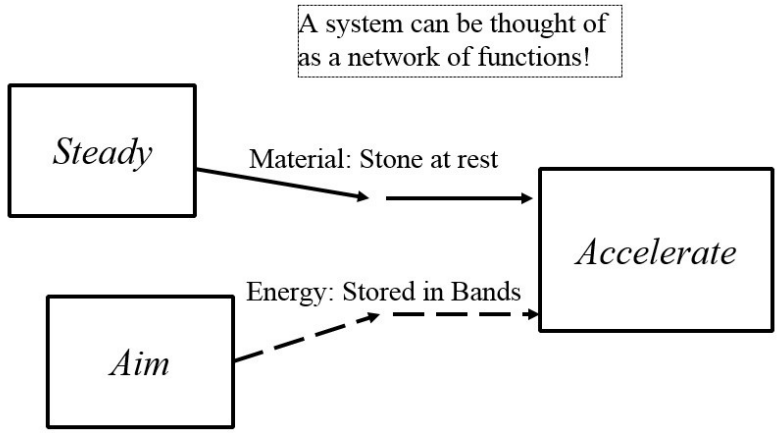
Exploring Alternatives

The function diagram is used to explore alternatives.



Interfaces

The function diagram allows us to consider the interface to mating sub-systems.



Introduction to Systems

Now that we have reviewed the key definitions of form and function, let's see how that plays a role in systems thinking, one of our key principles.

- Learning to formulate your problems in terms of basic functional requirements will help you formulate alternative forms.
- Function mapping helps define how sub-systems exchange materials, energy and signals.

In the next few sections, we will review some definitions of systems, systems interface and systems architecture.

Definition of a System

Let's take a look at the definition of System.

System: An organized set of parts/components that operate together to provide some useful function.

An example of a system would be an automobile. An automobile is an organized set of parts that operate together to provide mobility to individual(s) and some cargo. Note that it is the configuration of the parts in a system that provide the emergent function. If your automobile was disassembled the next time you went to use it, it would not function as a system, even though all the parts may be present. Also note that the 'parts/components' are often, but not always, physical; things like data, software, etc. are also considered parts or components of the system.

Definition of System Interface

Now that we have described form and function and system, let's take a look at the definition of System Interface.

System Interface: It is also worth noting that systems contain interfaces. An interface can be thought of as a shared boundary between parts of the system. The interface may be:

- physical (such as the tire being mounted to the wheel)
- material (the water pump takes in coolant and outputs it at higher pressure)
- information/signal (engaging the brake light switch turns on the brake lights)
- internal (the steering wheel interfaces to the steering column)
- external (the tires interface to the road).

Definition of Systems Architecture

Now that you have a better understanding of form, function system, and system interface, let's take a look at the definition of System Architecture.

Systems Architecture: Systems Architecture is the mapping of function to form via concept.

This is a deceptively simple definition, but let's elaborate a bit to better illustrate what it means.

Let's go back to the project you have been hired to do, designing a system to remove red leaves from the top of tall trees. TTR has given some clarification. They want to study red leaves near the top of the forest canopy. Now you know you have been asked to design a system to allow someone to get some of those leaves. **The function for this system would be to retrieve some red leaves from near the top of a tall tree.**

Note: We could decompose the function 'retrieve some red leaves from near the top of a tall tree' further. In this case, decomposing by the likely sequence, we might get: identify target tree, identify target red leaves near top of target tree, detach target leaves, obtain detached leaves. There are numerous unstated assumptions in the problem statement. For example, we are

assuming the tree exists already and we want its leaves asap, as opposed to we will work with a small tree that we can reach the top of today in order to retrieve leaves from it many years later.

Designing a System

As you proceed to design - or architect - such a system, you might first generate several possible concepts on how the system might accomplish the desired function(s).

Remember, at this point, you do not have a solution in mind. All you know is that you need to get leaves at the top of a tree. You SHOULD NOT jump to solutions too early, even though you may be inclined to do so.

It is helpful to think of the function in very simple terms. As we learned with the "accelerate projectile" example, function consists of a noun and a verb or verb phrase.

How could you do that? Let's brainstorm and come up with some ideas (or concepts) how we might design a system that gets leaves.

Brainstorming Concepts

For example, one concept or idea might involve flying above the trees in a helicopter, lowering someone down via a tether, and having them grab some leaves.

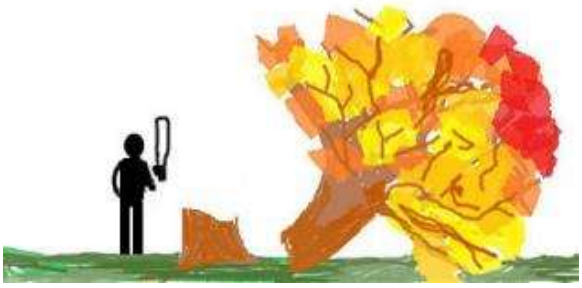
Concept: A concept is an idea that starts with a function that helps us map function (the noun and verb - system gets leaves) to form (a tangible solution).

Normally, the concept is presented as a simple, rough sketch (such as the graphic to the right in this section and a few of the following sections) showing what major elements the system involves and how they come together to meet the intended function.

Think of a time when you were trying to explain something to someone and you pulled out the nearest piece of paper to draw a rough picture to help you. You may not have realized it, but you were starting with a function and trying to map it to a form by sharing your concept.

Another possible concept might involve flying a radio controlled quadrotor equipped with a vision system and an automated cutting/capture system to cut off and hold onto some leaves.

Another possible concept is to cut the tree down and then walk over and grab some leaves from near what was the top of the tree.



Another concept we will consider might involve using a slingshot to shoot a projectile high into the trees. The projectile would have some sharp edges to it, such that when it hits a bunch of leaves, many of the leaves would become detached and fall to the ground for easy retrieval.

Note that there are many more potential concepts to address the desired function (training a monkey or bird, climbing the tree, using a long pole saw, standing on tall stilts, etc). When architecting an actual system, you are encouraged to consider as many options as you can generate so as to maximize the likelihood that when you select a 'best' architecture it has a higher probability of actually being pretty good!

As we will discuss later, complex systems can involve so many options that it will not be possible to consider all combinations of ways to meet key functions. In these cases, we recommend a heuristic based architecting approach (to be described later) to help you generate system concepts likely to be successful.

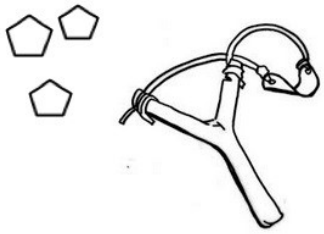
Definition of Systems Engineering

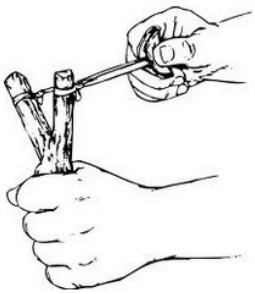
Now, let's explore the foundational concept of systems engineering as we explore ways to execute our task.

Systems Engineering: System Engineering is the process of executing a system's architecture.

In our slingshot example, the form would be the actual object (slingshot, projectile with sharp edges that could cut leaves from branches).

Suppose we want to execute the slingshot architecture for leaf retrieval described earlier. In other words, we want to use the form of the concept (the actual slingshot) to perform the function which is to get leaves.





Executing the Architecture

Some of the activities required to execute that architecture are listed below (Please note, this is not an exhaustive list - just an attempt to convey some of the key systems engineering activities involved). The left column shows activities that usually happen when you are executing a system architecture. The right column has examples as related to the slingshot.

General Systems Engineering Activities	Slingshot/Tree Example
Determine the System Level Requirements	Answer the following questions: How high are the highest leaves desired, how far laterally may you need to fire from? How many leaves per unit of time do you intend to retrieve? What, if any, damage to the leaves is acceptable as a result of retrieving them? What is an acceptable draw length and force, etc.?
Partition the system into subsystems	The handle, the Y, the elastic cord, the projectile, etc.
Cascade the system requirements through the decomposition	Answer the questions: What elastic properties are required for the elastic cords? What projectile mass is necessary to be effective?
Conduct trade studies	A heavier projectile might be more likely to detach the leaves, but also requires more draw force, and increases the possibility of accidental injury to any by standing people or animals.
Define and manage interfaces	For example, how does the elastic attach to the Y?
Develop test plans at various system levels	For example, one test may involve throwing various projectiles at lower leaves to see what works to detach the leaves.
Assembling, integrating and testing the final system	Put it all together, verifying proper function of the subsystems whenever possible, then ultimately test the top-level system to see how successful you are at getting leaves.

Complex Systems

Now we have defined system, but when engineers speak of systems, the term complex often arises. So what exactly is a complex system?

A **Complex System** is:

- Any system that will require multiple disciplines to execute (usually this translates into requiring multiple people).

- No individual normally has a complete grasp of all the details of the entire system.

Systems architecting and systems engineering principles have evolved primarily to allow us to deal with complex systems. We will discuss this idea in more detail later in the module, but first we want to review some key principles. Again, it is important to make sure everyone has a common understanding, as we build on these ideas.

Lesson 2: Key Principles of Systems Engineering

Introduction

Now we have working definitions of form, function, system, systems architecting, and systems engineering. Let's consider some key systems principles that we haven't talked about yet: systems thinking and systems of systems and systems decomposition.

Systems Thinking: Systems thinking is a way of describing the world in a more holistic manner based upon the model of a system, by allowing us to focus on the interconnectedness of the parts.

Watch the following youtube video that explains this concept in detail.

https://www.youtube.com/embed/AP7hMdnNrH4

There are many other YouTube videos on this topic. If you want to see more, search youtube for "systems thinking: an introduction" or "complexity science: systems thinking."

System of Systems

The system of systems principle is simple but profound. It merely states that every system is part of a larger system, and every system can be decomposed into smaller subsystems.

Let's take a look at an example with which we are all familiar to help us gain an understanding of this principle.

This is a picture of a highway system.

A subsystem of the highway is...a car.

A subsystem of a car is...an engine.

A subsystem of an engine is....a valve train.

System Decomposition

We have looked two key principles: systems thinking and system of systems. Another key principle, system decomposition, helps us to define a system in functional terms in order to better understand the individual parts of the system.

System Decomposition: Defining a system in functional terms in order to better understand the individual parts of the system.

Two questions often arise when one first starts thinking about system decomposition:

(1) Can a single part be decomposed into subsystems?

(2) How far up or down the system decomposition should I go?

Let's discuss each of these questions in turn.

(1) Can a single part be decomposed into subsystems?

Regarding the question as to whether a single part can be further decomposed, the answer is a resounding YES. For example, a piston in an engine is s single piece of metal, but it could be decomposed into things like the dome, the valve clearances, the ring grooves, the wrist pin bearing surface, the skirt, etc.

(2) How far up or down the system decomposition (hierarchy) should I go?

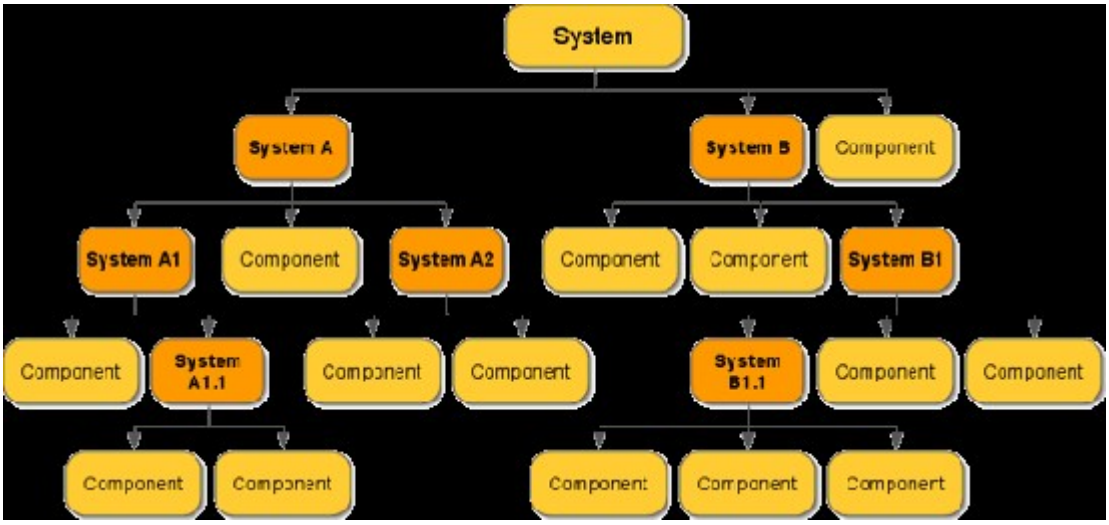
Regarding the question as to how far the decomposition needs to be carried down (or up), the answer is often not comforting: namely, as far as it needs to be to be most useful. Let's discuss an example to clarify this. Suppose I was designing a guardrail to keep vehicles out of the median. I may not need to decompose the system in this case much further than the vehicle and its mass properties. If, on the other hand, you are designing a speed bump, the vehicle would need to be decomposed at least as far as the suspension and undercarriage, but it probably does not matter too much whether the seats are cloth or leather.

So, essentially, you need to think about the purpose for your system decomposition, and decompose to the level that meets your intentions; going further only provides unnecessary complication, and not going far enough fails to provide sufficient resolution to be useful. In general, it is recommended that you consider working up the hierarchy at least one level above the system level you are working on.

Graphical Representation of a Systems Hierarchy

The most common way to represent the system hierarchy is with a block diagram like drawing similar to the one shown here.

Systems are broken down into other systems, until you can't break them down further, and you end up with the most basic unit, the system element.



To summarize, some of the key points in creating a system decomposition:

- There is no unique (or 'correct') decomposition, so it is recommended that you consider several possible alternatives to see which appears most useful for what you are trying to accomplish.
- You need to decompose to a level that supports what you are trying to accomplish.
- A good rule of thumb is to decompose to 7 + or - 2 subsystems at each level of decomposition.

Note that in the earlier highway/car/engine example, only one subsystem at each level was shown for simplicity; one way the automobile could be decomposed into subsystems would be: powertrain (includes the engine, transmission, driveline, etc.), body, chassis, electrical, interior).

Lesson 3: Architecture Development

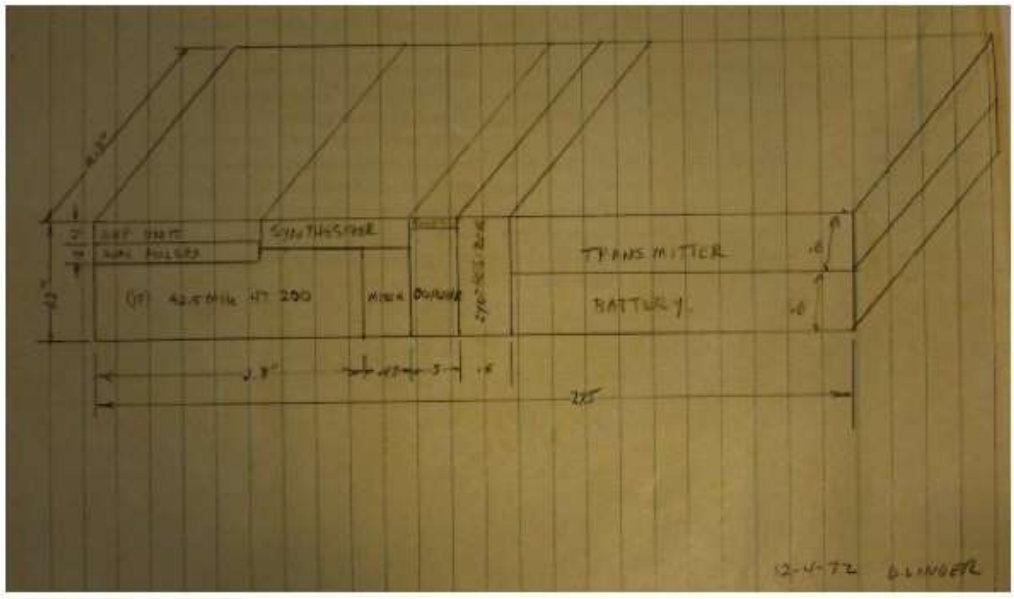
We define **systems architecting** as the mapping of (solution neutral) function to form via concept.

The fields of systems engineering and systems architecture are rapidly evolving to handle systems of unprecedented complexity. There are four recognized approaches to developing an architecture. This section will introduce those four methods (and the following section will go more deeply into heuristic based architecting).

The First Cell Phone

To start our discussion of developing systems architecture, let's take a look at a real example from Motorola.

On the day in 1973 that Motorola's Don Linder was assigned to oversee the design of a hand-held phone, he roughed out a sketch showing where and how big the components would need to be.



The architecture required a synthesizer, a supply unit, dial pulses, a transmitter, a battery etc.

The actual form of early cell phones greatly resembled his initial concept sketch.

The image below shows how the form of cell phones has evolved over time.

Architecture Development

Remember the Definitions of Systems Architecture and Systems Engineering

Architecture development involves the mapping of function to form through concept (systems architecture) and the successful execution of the systems architecture (systems engineering). To see some real examples of these concepts, watch the video in the link below by Aral Balkan,

Superheroes & Villains in Design. Pay particular attention to the section 32:33 - 36:33. If you are interested, feel free to watch the rest.

https://vimeo.com/70030549

Desired Function	Concept*	Form**
Heat Food		
Preserve Food		
Remove Waste		

Typically a sketch showing the major system elements and how they come together.
***Actual final product*

Great Architecture

Before we delve into architecting methods, let's look at characteristics of great architecture, some common pitfalls in architecting, and the fact great that architecture can lead to the development of standards.

Great Architecture is Identifiable By Common Traits

- Satisfies customer needs and wants
- Incorporates appropriate technology
- Meets strategic business needs
- Meets or exceeds present and future regulations
- Operable, maintainable, sustainable, reliable
- Can be evolved/modified as appropriate
- Can be designed and manufactured by envisioned team and with existing/planned capabilities
- Takes into account the system's position relative to the larger system and other systems in related products

Why do we care what great architecture looks like?

We reviewed traits of great architecture so that as we develop architectures, we can review a proposed architecture against these traits to make sure what we are proposing looks great.

You are not the first one to attempt system design. There are many lessons learned from those who came before you. Familiarize yourself with the following list of traits of poor architecture, and try to design systems that avoid these traits.

Great Architecture Avoids Common Pitfalls

- Ill defined purpose/priorities
- Poor definition of success
- Aimed at non-existent market
- Ignored sociopolitical factors
- Ignored opponent's response
- Overestimated technology advance
- Bad timing (sociopolitical, etc.)

A Good Architecture Can Lead to the Development of Standards

Great architecture often becomes the standards that others adapt. Here are some examples where a company's successful architecture has led to standards:

- Intel: x86
- Microsoft: Windows
- Adobe Acrobat: pdf
- Sun Microsystems: Java

Lesson 4: Multiple Views of a System

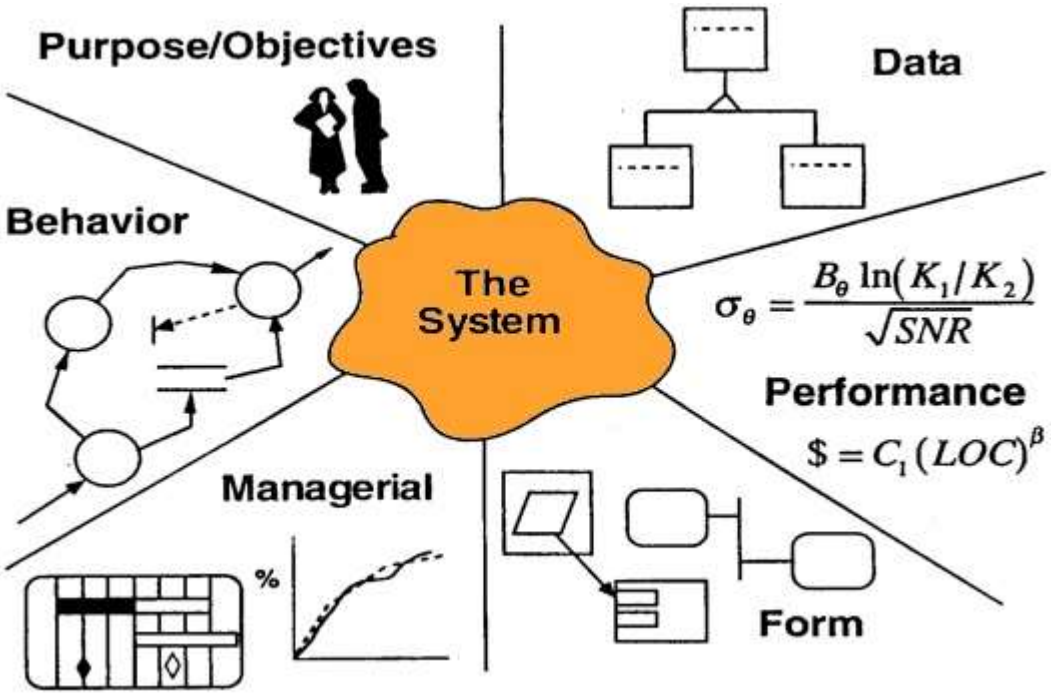
To start our thinking about principles of Systems Architecture consider the following key principles:

- Innovative architectures are driven by the functionality of the system.
- System robustness is largely determined by interface management.
- Architectures can be adapted to multiple products to enable reuse of core technology and engineering effort.

Six Views of a System

In addition to the types of systems, heuristics enable us to consider the systems from various views.

In the statements that follow, for each of the six views of the system shown here, a brief description is provided, as well as some guidelines to use when considering that view of the system.



View: Purpose / Objective

"What the client wants"

Ask early about how you will evaluate the success of your efforts.

Don't assume the original statement of the problem is the best, or even the right statement.

View: Data

"Manage information in the system and inter-relationships"

Work forward and backward.

Explore the situation from more than one point of view.

View: Performance Requirements

"How effectively the system does the job"

Discipline, discipline, discipline.

If it is a good design, make sure it stays sold.

High quality, reliable systems are produced by high quality architecting, engineering, design and manufacture, NOT by inspection, test, and rework.

View: Form

“What the system is”

If you can’t explain it in five minutes, either you don’t understand it, or it doesn’t work.

If the concepts in the mind of one person are very different from the mind of another, there is no common model of the topic, and no communication.

View: Managerial

"The process by which the system is constructed and managed”

Unless everyone who needs to know does know, somebody, somewhere will foul up.

Knowing a failure has occurred is more important than the actual failure.

View: Behavior/Function

“What the system does”

Be prepared for reality to add a few interfaces of its own.

Unbounded limits on element behavior may be a trap in unexpected scenarios.

Establishing a Framework for your System

Focus on Architecture decisions, and let descriptions flow from the decisions.

- The test of your framework, ask: “what decisions about changing our system(s) are contained in this document?”
-

Always use an iterative process to do Architecting.

- The test of your framework: “has the architecting process resulted in any clear decisions, in a reasonable amount of time?”

Do not overstaff early, the best alternatives come from small teams.

- The test of your framework: “Is the team mired in defining “as-is” or are they breaking new ground?”

Orient the project (and yourself) properly.

- The test of your framework: “Have you defined the context and effective attributes”

1. What is the system of interest?
2. What is the basis for the project?
3. What is in-scope and out-of-scope?
4. What will be done with the completed architecture?

Lesson 5: Systems Verification & Validation

What does it take to execute Systems Architecture?

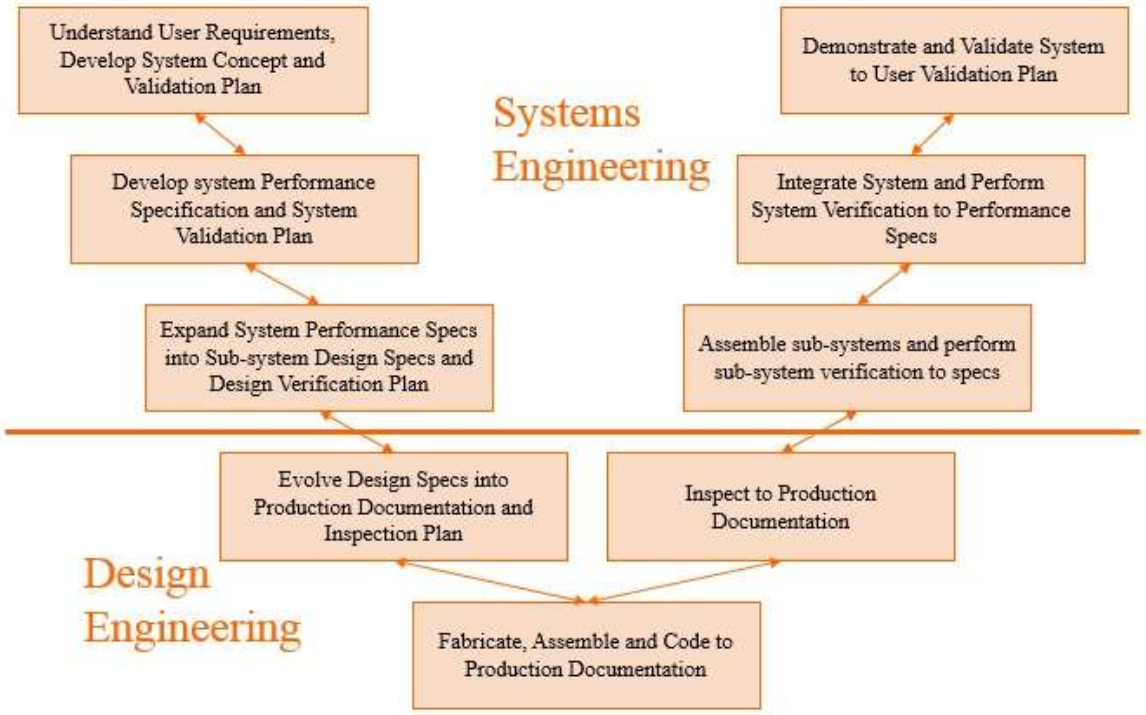
Now that we have learned how to develop the architecture for a complex system, let’s turn to executing that architecture. Recall from the intro that our definition of systems engineering is the process of executing a systems architecture. A robust systems engineering process is necessary to minimize mistakes, rework, and component or system level failures.

The V Model

The Systems Engineering V

Forsberg and Mooz were the first to model the SE processes as a V.

For complex, multidisciplinary systems, the V model provides a nice framework for the key systems engineering activities it takes to realize a successful system. In the context of this section of the module, the assumption is that the basic top-level system architecture has already been developed, and we are now in execution mode. The next few sections will review this concept in detail.



Left Side of the V

- On the left side of the V, decomposition and definition activities resolve the system architecture and create the details of the design.
- Integration and verification flow up and to the right as successively higher levels of subsystems are verified, culminating at the system level.
- It is important to distinguish between verification and validation; verification ensures the system was built right (meets requirements), whereas validation ensures the right system was built (meets customers' needs).

Right Side of the V

- Ascending the right side of the V is the process of integration and verification.
- At each level, there is a direct correspondence between activities on the left and right sides of the V.
- This is deliberate - the method of verification must be determined as the requirements are developed and documented at each level.

- This minimizes the chances that requirements are specified in a way that cannot be measured or verified.

Using the V to Implement a System

If you need to implement a complex system, we suggest you organize your development according to the V. This will minimize unnecessary rework, errors in requirements development and cascading. This will force verification to occur at various levels of integration (rather than build the top level only to discover major issues at lower system levels).

Bicycle Example

To help us better understand the concept of the Systems V, let's take a look the systems that compose a bicycle, which is a system with which we are all familiar.
The subsystems of a bike are:

- frame
- fork
- stem
- handlebars
- drivetrain (includes shifters and derailleurs)
- brakes
- wheels/tires
- seat assembly (includes seat post)
- reflectors

Top Level Requirements

Some top level requirements include comfort, stability, ability of the bike to be ridden on a variety of terrains/grades, safety, durability.

Here are three more top level requirements.

- Affordability
- Availability of Spare Parts
- Attractive Appearance

Cascading (Derived) Requirements

f we take comfort as a system level requirement, that would cascade onto several of the subsystems. Some of this cascade could be given, such as: comfort drives frame geometry (since riding position affects comfort), frame material (carbon fiber provides better ride quality than aluminum, which provides better ride than steel).

How does comfort cascade onto other subsystems? Possibilities include:

- handlebars and stem (for riding position)
- seat (for obvious reasons)
- tires (wider tires generally weigh more and develop more rolling resistance but feel more comfortable)

Additional Requirements

The seat could then be further analyzed from the cascaded requirement for comfort resulting in requirements on the shape of the seat and the material that the seat is covered with.
Additional requirements related to the seat stem from the top level of comfort requirement. An obvious one would be the properties of the material/padding used within the seat.

Now that we have applied the concepts of decomposition to a bicycle, let's see how we might apply that to the Systems V that we talked about.

Recommended Process

Recommended Process for Heuristic Architecting

- Critically examine the problem statement to make sure you are solving the right problem before you worry about the right solution.

- Apply the techniques covered in the previous section on functional decomposition to make sure you truly understand your system's key functions (in a solution neutral way).

- Browse the list of heuristics and see which ones 'jump out at you' as likely describing great architecture in the domain of your problem (we know engineers like step by step, mindless procedures, but you cannot remove the thinking from this step!)

- Try to refine the selected heuristics so that they more directly and clearly guide development of your system.

- Try to refine and descriptive heuristics to make them prescriptive (remember that prescriptive heuristics tend to be more useful).

- As you generate architectural concepts, keep the governing heuristics in mind to reduce the exorbitant number of possible architectures you'd need to consider in a brute-force search of the design space.

- When faced with several alternative architectures, use the heuristics and traits of good architecting to evaluate the relative goodness of the alternatives.