ENES 489P Hands-On Systems Engineering Projects

Modeling System Structure and System Behavior

Mark Austin

E-mail: austin@isr.umd.edu

Institute for Systems Research, University of Maryland, College Park

System Structure and System Behavior

Topics:

1. Abstractions for Modeling System Structure

   Components, Attributes and Relationships.

   Interconnect Design Methodology.

   Interface contracts.

2. Abstractions for Modeling System Behavior

   Simplfied procedure for modeling system behavior.

3. Viewpoints of Behavior

   Control flow, data flow, state machines.

3. Behavior of an Artillery Cannon

Abstractions for Modeling System Structure

Components, Attributes and Relationships

The structure of a system contains:

1. **Components**

   Components are the operating parts of a system consisting of input, process, and output.

   Each system component may assume a variety of values to describe a system state.

2. **Attributes**

   Attributes are the properties of the components in the system.

3. **Relationships**

   Relationships are the links between the components and attributes.

An important characteristic of a system is that ...

... its purpose is met by the properties of the system as "a whole," and not just by the union of the components.

Abstractions for Modeling System Structure

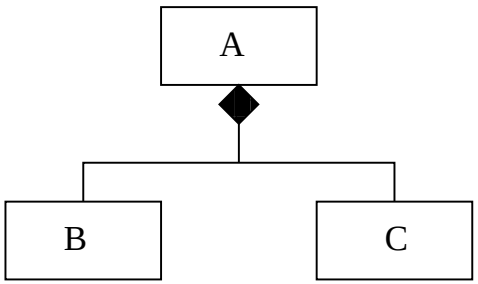
Interconnect Design Methodology

The interconnect design methodology assumes that ...

... complicated products and processes can be represented as hierarchies and networks of product and process blocks.

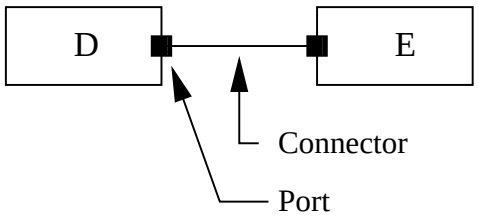
Hierarchy and Network Abstractions

Hierarchy Abstraction



Module A contains modules B and C

Network Abstraction



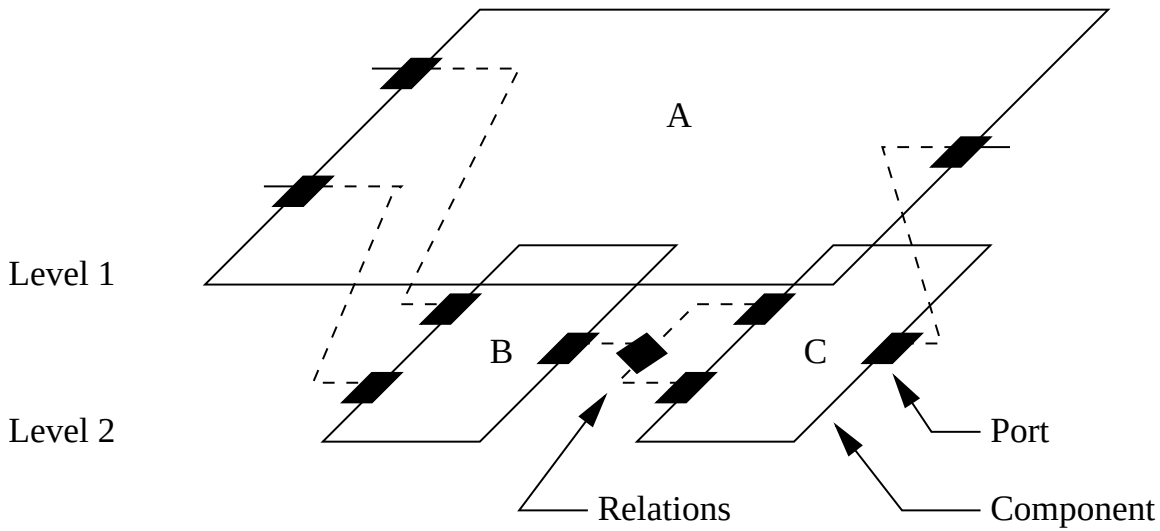
Module D is connected to module E

Note. Assigning properties and functions to blocks is relatively straightforward.

Abstractions for Modeling System Structure

Combining Hierarchy and Connectivity of Components

Simple system hierarchy decomposed into two levels, with a network of components (B and C) at level 2.



Abstractions for Modeling System Structure

System Assembly Elements

1. **Parts**

   Parts represent a set of instances that are aggregated within a containing classifier instance.

2. **Connectors**

   Connectors specify a link that enables communication between parts in a structure or with the surrounding environment.

3. **Ports**

   Ports specify an interaction point between a classifier and its environment and/or between a classifier and other internal ports.

4. **Interface Specification**

   An interface specification defines precisely what a client of that interface/component can expect in terms of supplied operations and services.

Abstractions for Modeling System Structure

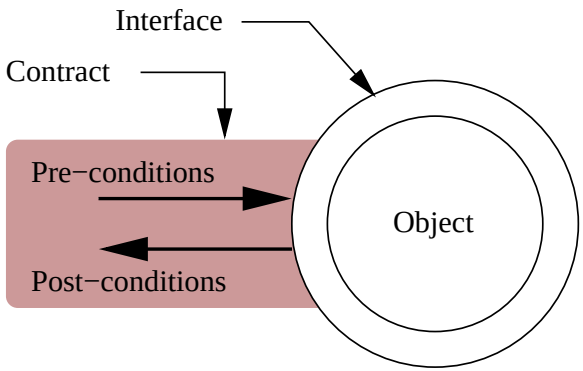
System Assembly Constraints

The system structure must satisfy the following constraints:

1. Within a hierarchy, each level is logically connected to the levels above and below it.
2. A port cannot be contained by more than one entity.
3. Links cannot cross levels in the hierarchy,
4. Port-to-port communications must have compatible data types (e.g., signal, energy, force).

Abstractions for Modeling System Structure

Interface Contracts



Interface Contract Guidelines

Guidelines for the design of system interfaces are as follows:

1. Details of the system implementation should be hidden behind its interface.
2. Provide only as much functionality as needed (i.e., keep it simple).
3. Interface functions should not overlap (i.e., orthogonality condition).
4. Modules should only communicate with a customer via its interface.

Abstractions for Modeling System Structure

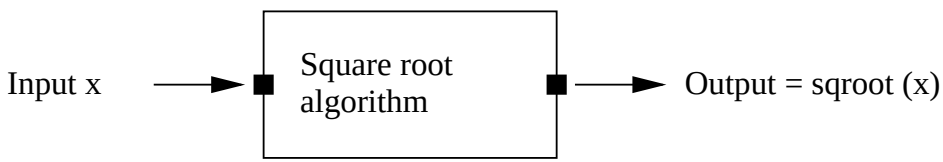
Examples of Simple Interface Contracts

Example 1: Division of floating point numbers

Contract: y != 0.

Example 2: Square root calculation

Contract: x >= 0.



Note. Actor models can setup and track the flow of data through the system graph.

Abstractions for Modeling System Behavior

Definition. For our purposes, system behavior defines:

> **... what a system will do in response to its external environment without referring to details on implementation (e.g., use of technologies).**

Understanding the behavior of a system as a whole requires (Kronloff, 1993):

1. A knowledge of the individual parts and their behavior,
2. The interfaces between the parts,
3. The traffic that passes along the interfaces, and
4. The system environment.

Benefits. Behavior models provide an executable description for what a system will do.

Link to Tradeoff Analysis. Then, we can systematically adjust the input parameter values to indentify tradeoffs in the critical measures of system effectiveness.

Abstractions for Modeling System Behavior

Elements of Behavior

The elements of behavior are as follows:

1. **Functions.**

   These are discrete tasks (or activities) that accept inputs and transform them into outputs.

   Functions may be decomposed into sub-functions.

   Individual functions are incapable of describing behavior.

2. **Inputs and Outputs.**

   Identify the required system inputs and outputs provided by the system.

3. **Control Operators.**

   Define the ordering of functions.

Note. Real-world behavior often emanates from the controled ordering of functions, which in turn, affect how inputs are transformed into outputs.

Creating a Behavior Model

Getting Started

1. **Develop Model of System Context**

   What is the context within which the system will operate?

2. **Operations Concept.**

   What is the required system functionality?

   What will the system do in response to external stimuli?

3. **Requirements.**

   What are the system inputs and outputs?

   What requirements are needed to ensure that the system will operate as planned?

Note. Usecase diagrams are a good way of capturing fragments of required system functionality.

Creating a Behavior Model

Create Behavior Model

1. **Identify Top-Level Functionality**

   What are the top-level functions?

   Define inputs and outputs for each top-level functions.

   In what order will execution of the top-level functions occur?

   Trace inputs to outputs through network of connected functions.

2. **Identify sub-tasks within each top-level function**

   Goal is to simplify models of funtionality by decomposing high-level functions into networks of lower-level functionality.

3. **Identify opportunities for concurrent behaviors**

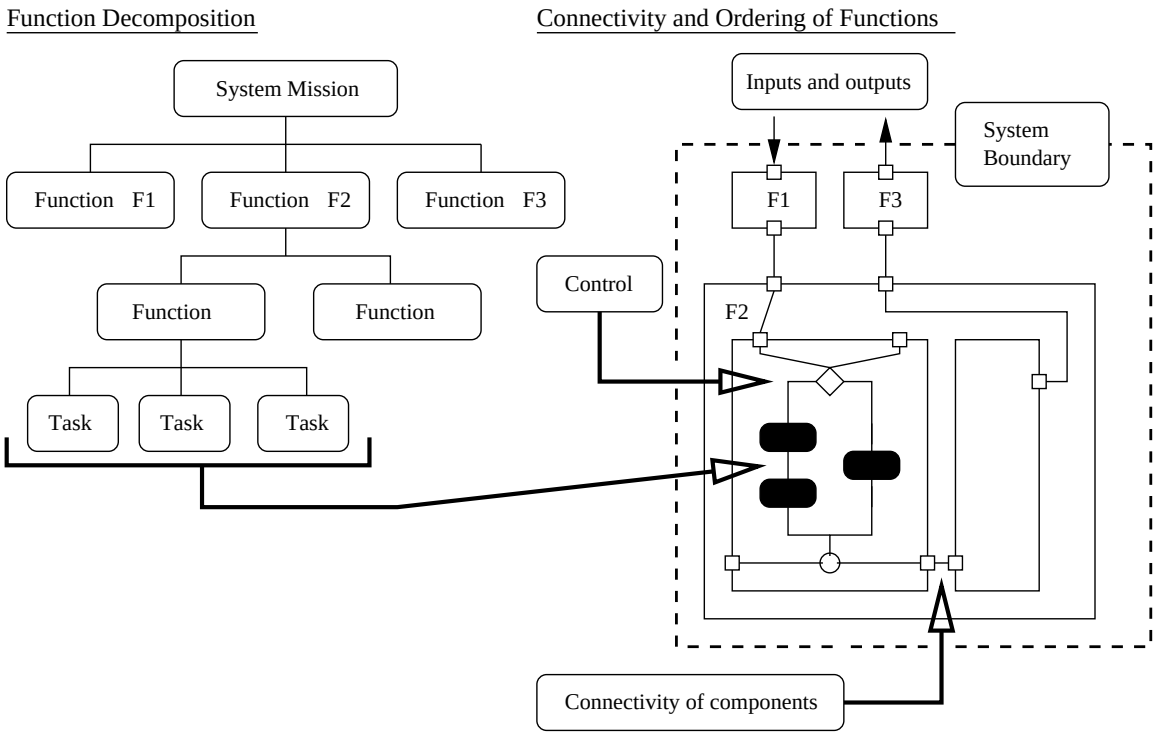
4. **Insert low-level functionalities**

Note. Several views of behavior may be required to obtain a complete picture of overall behavior.

Abstractions for Modeling System Behavior

Functional Decomposition

System behavior defined through decomposition and ordering (control) of functions.



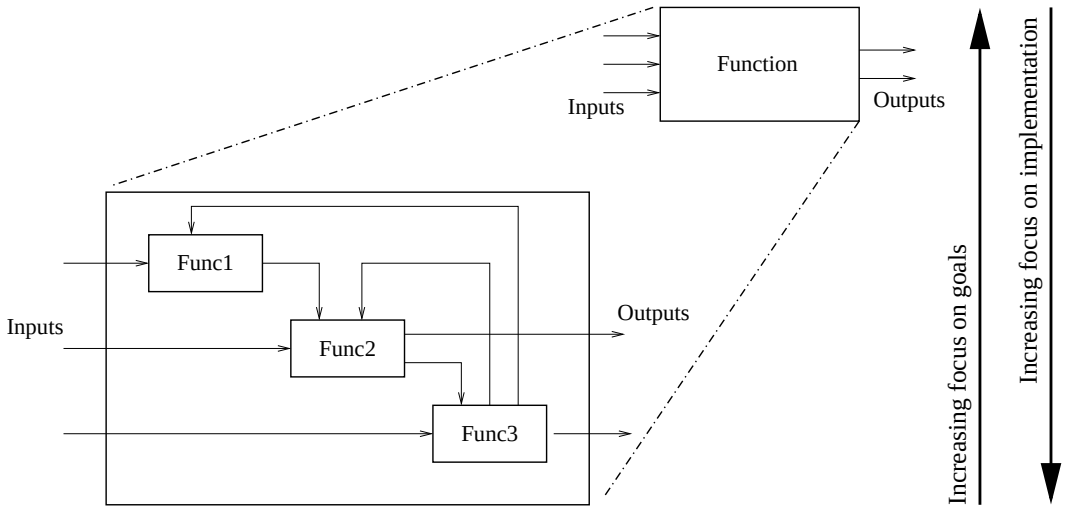
Note. The functional decomposition hierarchy says nothing about inputs and outputs.

Abstractions for Modeling System Behavior

Decomposition. Decomposition is the process of ...

... breaking the design at a given level of the hierarchy into components that can be designed and verified almost independently.

Decomposition of System Functionality



Note. Details of implementation are addressed in the lower levels of functional decomposition.

Abstractions for Modeling System Behavior

Evaluation Criteria for Functional Decompositions

Guidelines for the design of modules are as follows:

1. One module should have no more than seven subordinate modules.
1. There should be separation between the controller modules and the worker modules.
2. Every module must perform a task appropriate to its place in the hierarchy.
3. Every module should only receive as much information as it needs to perform its function.

The motivation for following these guidelines is modules that will be functional, easy to understand, testable, and reusable.

Abstractions for Modeling System Behavior

When should a product decomposition cease?

As a rule of thumb, the bottom of a product hierarchy is reached when ...

> **... all of the objects are physical modules, components, or resources.**

When should a process decomposition cease?

Guidelines for the decomposition of system processes are less clear.

It seems sensible that decomposition of processes should occur until ...

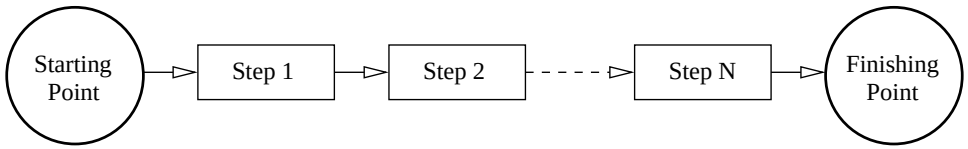
> **... a level of detail is reached where processes can be controlled in a repeatable manner.**

It seems equally important not to over decompose processes – this where management of processes becomes micro-management of processes.

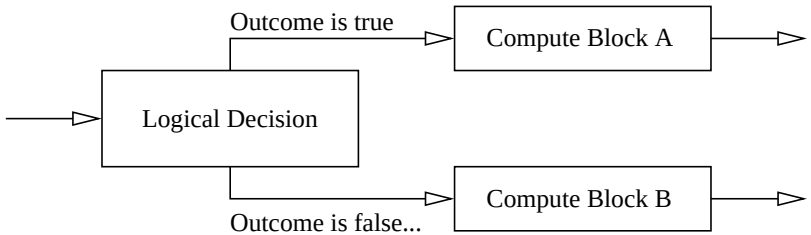
Abstractions for Modeling System Behavior

Ordering of Functions: Sequence, Selection, Iteration..

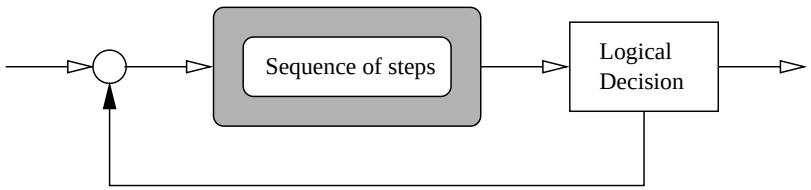
Sequence. Which functions must precede or succeed others?



Selection. Captures choices between functions



Iteration. Which functions can be repeated in a block?



Abstractions for Modeling System Behavior

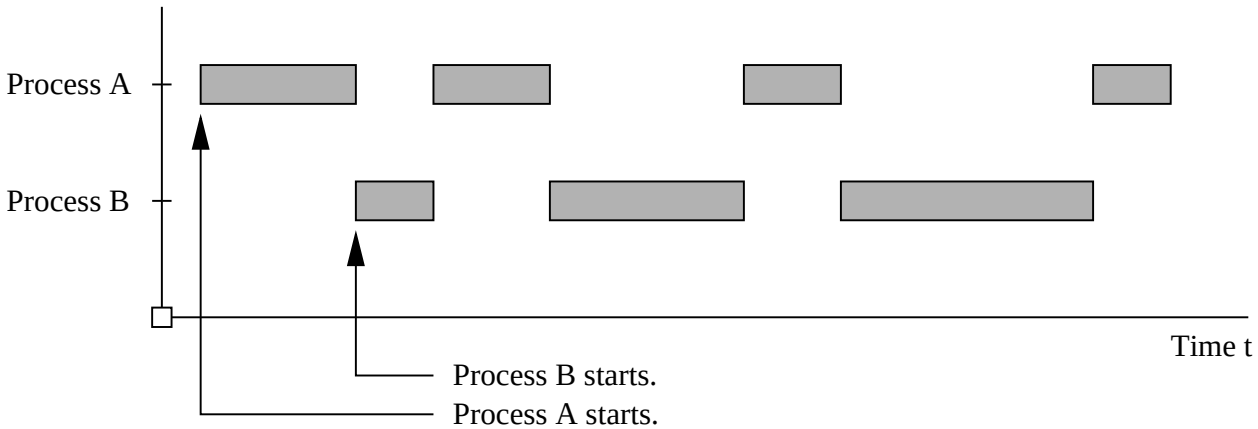
Ordering of Functions: Concurrency

Most real-world scenarios involve concurrent activities in one form or another.

The key challenge lies in the ...

> **... sequencing and coordination of activities to maximize a system's measures of effectiveness (e.g., production).**

Example 1. Running multiple threads of execution on one processor.



Control Flow Behavior

Control Flow Behavior Model

The control flow behavior model ...

... is most suitable for applications in which the exact sequence of steps is most important.

Control flow requires ...

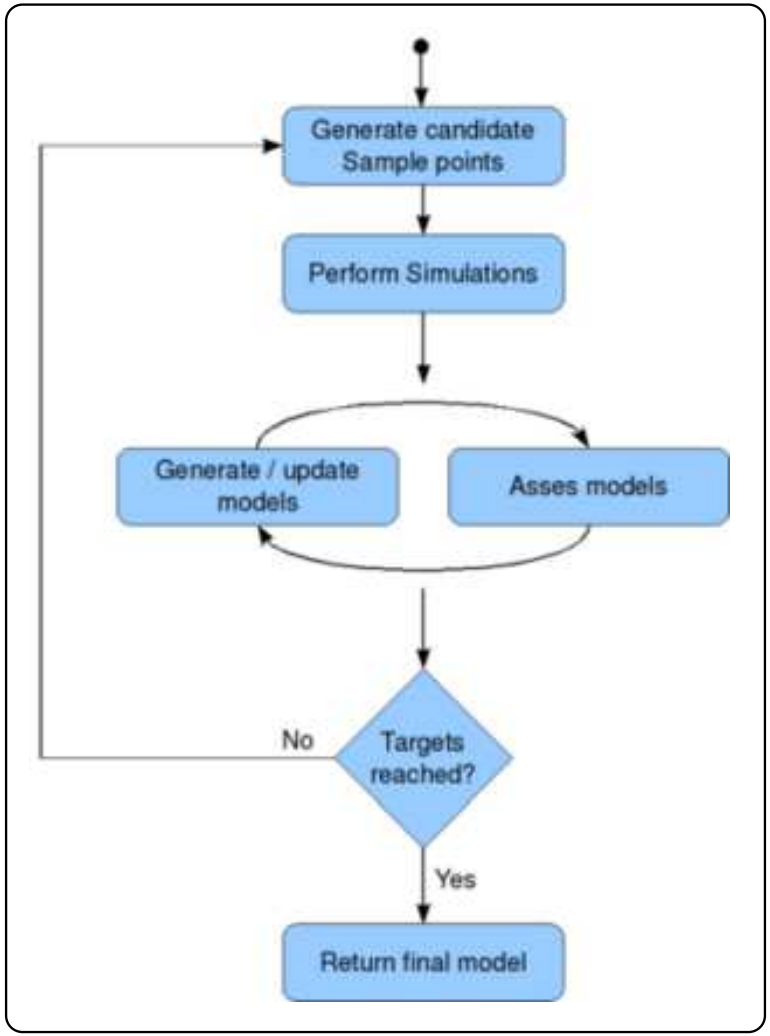
... one step to finish before another starts (e.g., postal delivery person).

It de-emphasizes ...

... the calculation of inputs by using whatever information happens to be available when a step starts.

Control Flow Behavior

Example. Control flow diagram for an iterative simulation process.



Data Flow Behavior

Data Flow Behavior Model

The data flow behavior model applies when data arrives in regular streams.

As such, the model ...

> **... focuses on functional dependencies between input and output – data flow takes each step when other steps provide its inputs.**

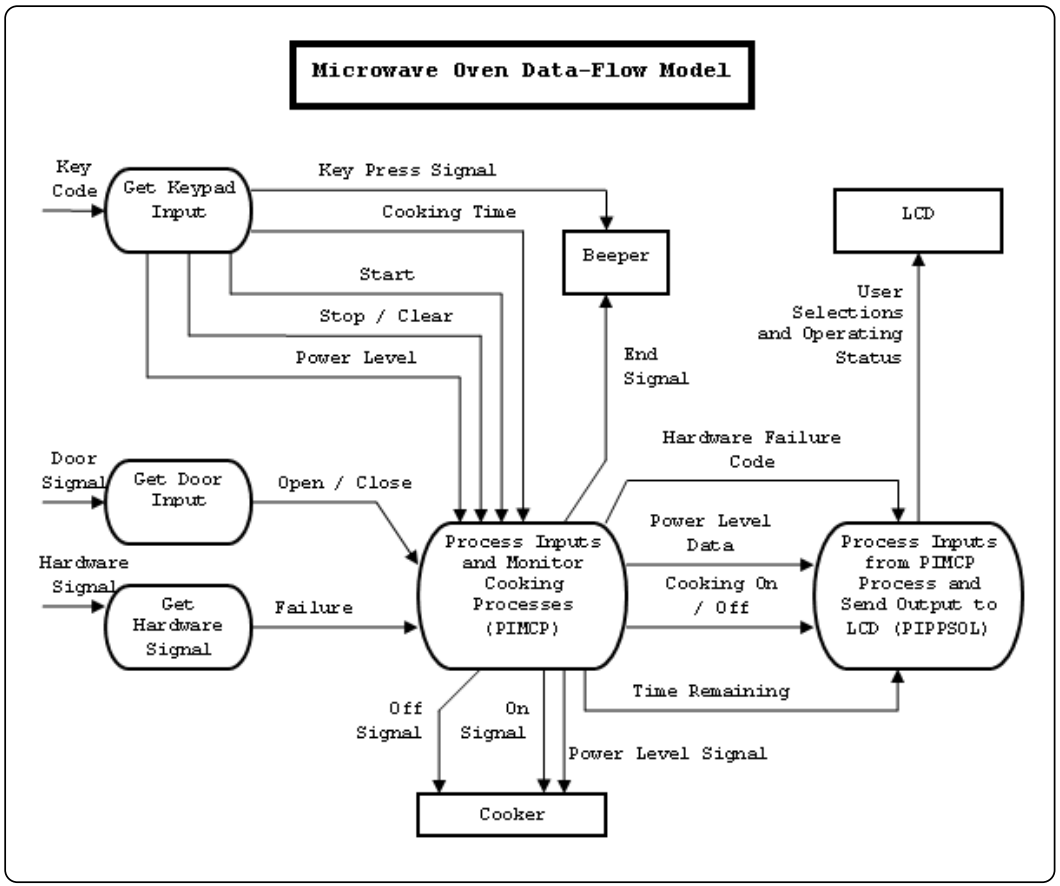
It also ...

> **... emphasizes the calculation of inputs by requiring the outputs of one step to be explicitly linked to inputs of another step or steps.**

The model de-emphasizes the sequencing of steps, because the time at which all the inputs to a step arrive is determined by however long the various input steps take to complete.

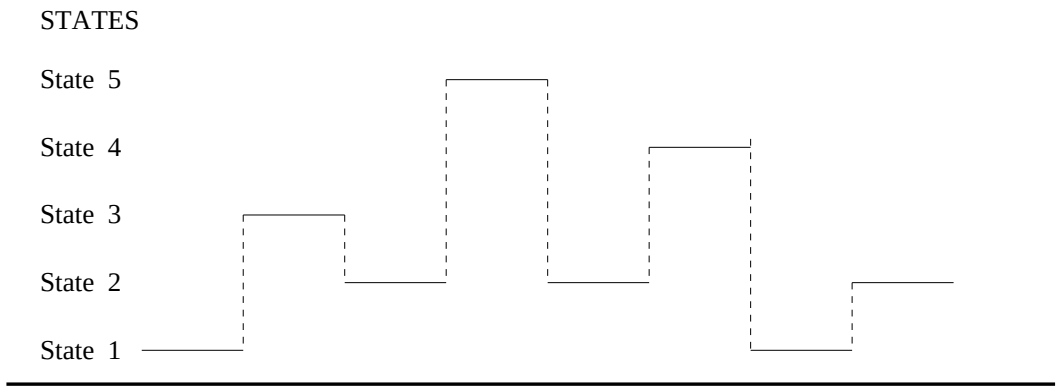
Data Flow Behavior

Example. Data flow diagram for operation of a microwave oven.



State Machine Behavior

State machine behavior can be viewed as a sequence of states versus time.



Key abstractions...

- **States** summarize the information associated with past inputs relevant to the current behavior of the system.
- **Transitions** take a system from one state to another. They fire one at a time.
- **Events** are an input (e.g., a kind-of stimulus or message) or interval of time.

State Machine Behavior

Recognition and Handling of Events

A state machine will only recognize those events defined in the model. All other events will be discarded.

Types of Events and associated Actions

Type of Event	Action
Signal event	The system receives a signal from an external agent.
Call event	A system operation is invoked.
Timing event	A timeout occurs.
Change event	A system property is changed by an external agent.

State Machine Behavior

State Machine Mechanisms

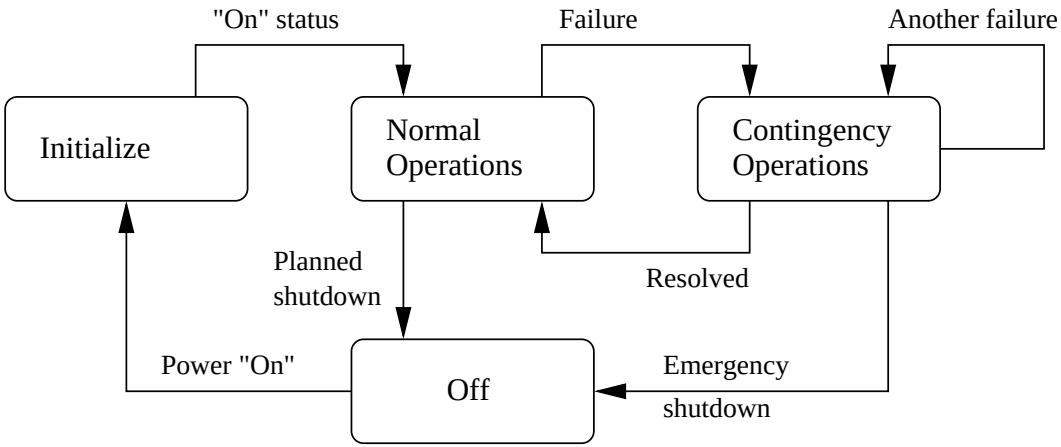
1. The machine begins at an initial state;
2. The machine waits for an event for an indefinite interval;
3. The event presents itself to the machine;
3. If the event is not accepted in the current state, it is ignored;
4. If the event is accepted in the current state, the designated transition is said to fire.

   The associated action (if any) is produced and the state designated as the resultant state becomes the current state.

   The current and resultant states may be identical.
5. The cycle is repeated from step 2, unless the resultant state is the final state.

State Machine Behavior

Example. State machine behavior of a spacecraft computer system

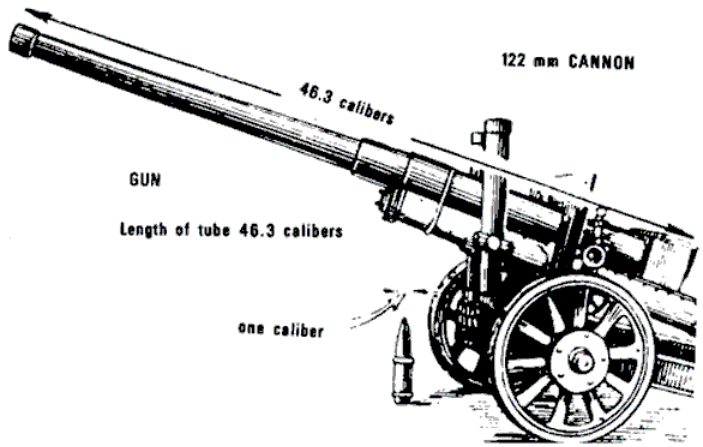


Points to note:

- The boxes in the state diagram show the valid states of the system, and the conditions needed to achieve each state.
- Support is provided for graceful shutdown in emergency situations.
- The remaining states relate to what the system needs to do under normal and contingency operating conditions.

Behavior of an Artillery Cannon

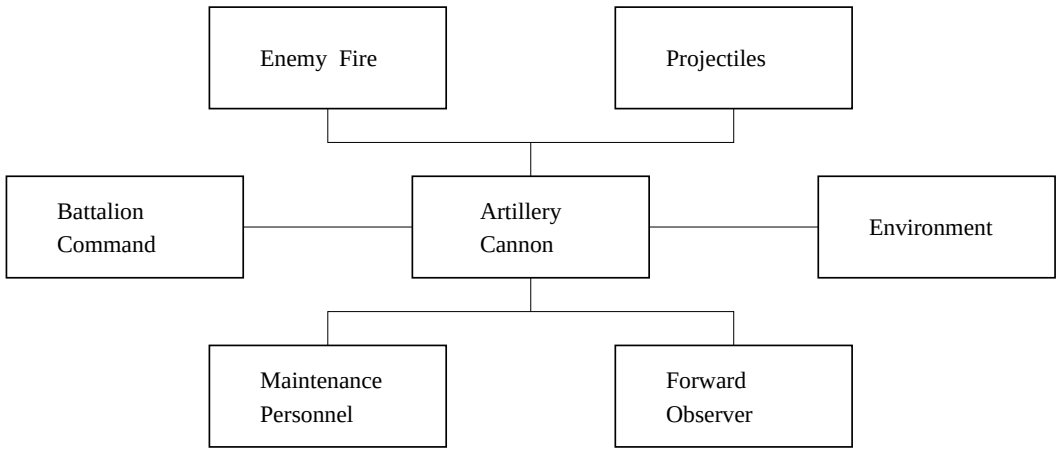
Problem Statement: Suppose that you have been asked to develop a system design an Artillery Cannon System (Source, Michael Krok at GE).



The systems design will include a behavior model, a structural model, and a combined model where behavior is mapped onto the structural configuration.

Behavior of an Artillery Cannon

Operations Context. The context will consist of a collection of systems, each having their own purpose, and interacting with other systems.



Operations Concept. The operations concepts is:

Upon receipt of Mission from Battalion Command, the system is to fire 10 rounds within two minutes, at the designated target, and without jeopordizing the safety of the crew.

Behavior of an Artillery Cannon

Preliminary Requirements.

System Input and Output Requirements

- Inputs to System

  Target location,

  Time to commence firing,

  Projectile type.

- Outputs from System

  Muzzle velocity for each projectile,

  Proper initial flight path,

  10 projectiles.

Behavior of an Artillery Cannon

Functional Requirements

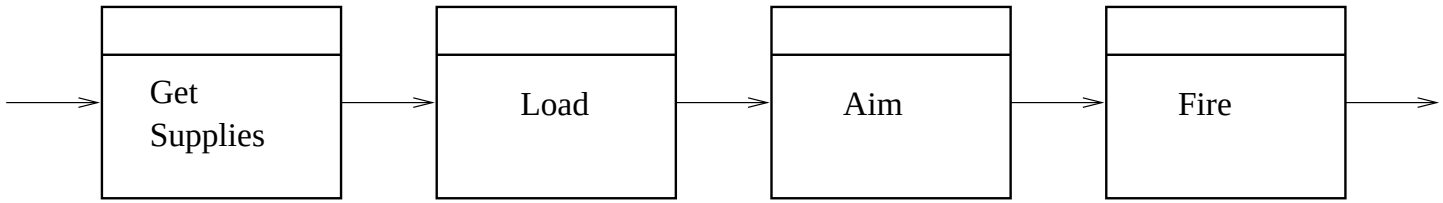
- The system shall store and manage both projectiles and propellant.
- The system shall obtain the proper projectiles and charge to be fired during each firing sequence.
- The system shall automatically open and close the breech.
- The system shall automatically load both projectiles and propellant into the cannon.
- The system shall control the firing sequence so that the proper launch conditions (i.e., muzzle velocity; azimuth; pointing angles) are achieved.

Temporal Performance Requirement

- The system shall be capable of firing 5 rounds per minute for 2 min + 3 seconds at the maximum charge possible for a given projectile.

Behavior of an Artillery Cannon

Develop Top-level Firing Behavior of Cannon

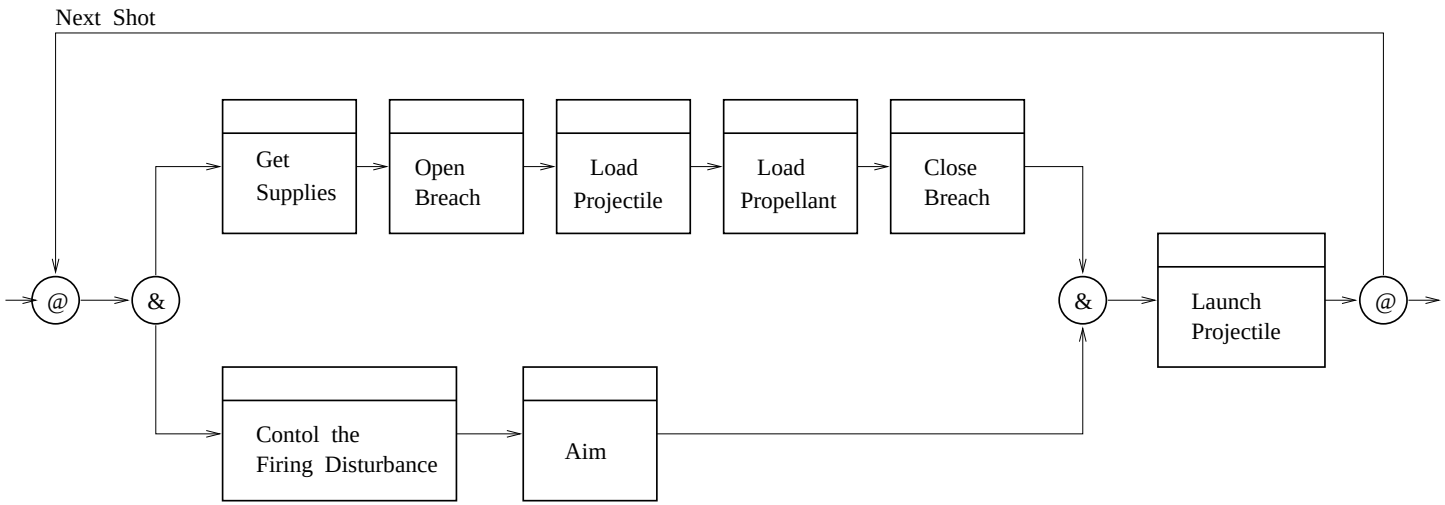


First find the intrinsic behavior (i.e., what must be done by the system, either automatically or manually). That is:

One shot = get supplies, load, aim, fire!

Behavior of an Artillery Cannon

Develop Sub-Tasks within each Top-level Function



Identify the sub-tasks within each top-level function, e.g.,

- Task “Load” is divided into four distinct functions whose ordering is dictated by logic and physical considerations (e.g., a path must be cleared before occupying space). The implied sequence of tasks is open breach, load projectile, load propellant, close breach.

Behavior of an Artillery Cannon

Consider Emergent Behavior

This includes:

- Factors that alter the behavior, but do not involve system structure.
- Deriving appropriate behavior from intrinsic behavior.

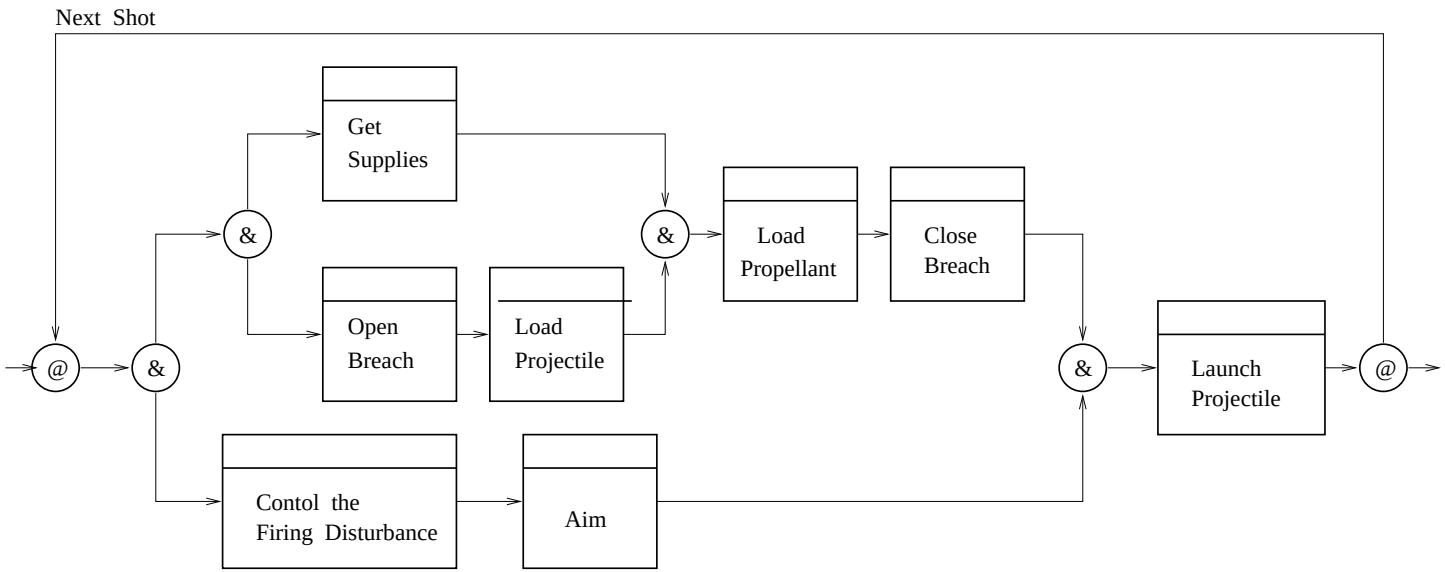
  One of the creative portions of systems engineering

  Formal methods and approach cannot replace bad decision making, nor replace experience and creativity.

  Best practices – usage of executable models to help evaluate decisions.

Behavior of an Artillery Cannon

Indentify Opportunities for Concurrent Behavior



Decompose tasks (e.g., get supplies, control firing disturbance, load propellant) in order to expose opportunities for further concurrency while accommodating safety considerations.

Behavior of an Artillery Cannon

Check Timeline

We can insert information on the expected mean time (and variance) for each task, and estimate the total expected time to launch a projectile, e.g.,

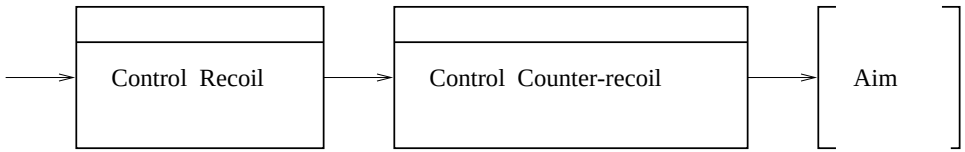
- Get Supplies = 6 seconds $\pm$ 0.3 seconds 1-σ.
- Open Breech = 1.5 seconds $\pm$ 0.1 seconds 1-σ.
- Close Breech = 1.5 seconds $\pm$ 0.1 seconds 1-σ.
- Load Projectile = 2.0 seconds $\pm$ 0.25 seconds 1-σ.
- Load Propellant = 2.0 seconds $\pm$ 0.25 seconds 1-σ.
- Control Firing Distance = 1.0 seconds $\pm$ 0.05 seconds 1-σ.
- Launch Projectile = 0.2 seconds $\pm$ 0.05 seconds 1-σ.
- Aim = 8.0 seconds $\pm$ 0.25 seconds 1-σ.

The expected total duration of this timeline is 14.2 seconds.

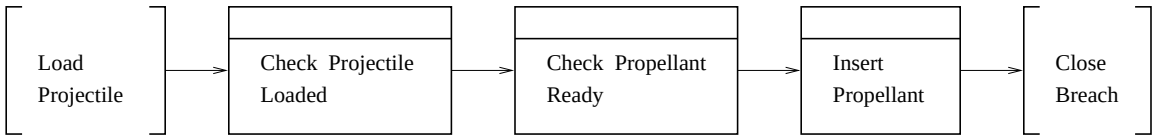
Behavior of an Artillery Cannon

Develop Low-Level Behaviors

Develop Low-level Behavior for Controlling the Firing Disturbance



Develop Low-level Behavior for Loading Propellant



We need to assure that the path is clear and the propellant is ready to be loaded before attempting to load. These checks require 0.1 seconds each. Inserting the propellant takes 1.8 seconds.

Behavior of an Artillery Cannon

Complete Description of System Behavior

A complete description of system behavior would also include:

- Insert lower-level component functions into FFBD.
- Need to define system inputs/output, together with information on expected mean value, range, duration, ..etc.

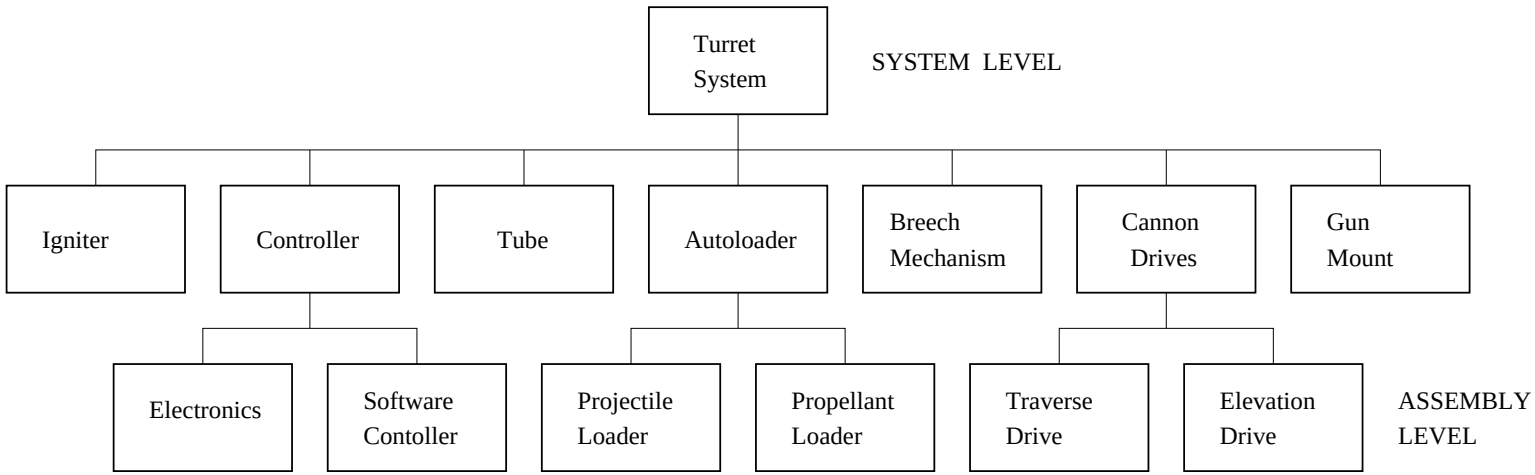
  Pathways of system input/output may be drawn on top of a detailed FFBD.
- Need to validate behavior by either manual means or automated/executed with a tool.
- A structural model is needed to visualize subsystem interfaces.
- Create N-squared diagram (or interaction diagram or design structure matrix) to show system function dependencies.

The next step is to allocate these functions to hardware and software configurations.

Behavior of an Artillery Cannon

System Architecture Model (Hierarchical Structure)

Structure diagram shows decomposition of turret system into main subsystems and mechanical/electrical assemblies.



An object diagram would show interconnectivity of system elements.

Behavior of an Artillery Cannon

Complete Description of System Behavior. FFBD mapped onto system architecture ....