The New P-Diagram

The Use of SysML Activity Diagrams to Support Taguchi Methods & Robust Design

Kate Konczal (ARDEC) & Michael Vinarcik (Booz Allen Hamilton)

Prepared for the 18th Annual NDIA SE Conference

Bottom Line Up Front

- Taguchi Methods have been widely used to improve the robustness of engineered systems
- Parameter design classifies the inputs, outputs, and ideal functions of a system
    - P-Diagrams capture this information and provide a convenient framing mechanism
- Systems Modeling Language (SysML) activity diagrams can capture P-Diagrams
- A Model-Based Systems Engineering (MBSE) approach enables further analysis
    - Leverages traceability to support secondary work products (such as tables and dependency matrices)
    - Integrates a myriad of design characteristics from the system, subsystem, and components in one place: the system model
    - Enables information currency and consistency and makes this information available for continuous decision-making by the full spectrum of project stakeholders

Taguchi Methods

- Developed by Genichi Taguchi, Taguchi Methods focus on:
  - Understanding the loss function of a system
  - Designing products so they are insensitive to variation (“robust”)
  - Design of Experiments (DOE)
- Taguchi methods are widely used to deliver results
  - Automotive
  - Aerospace
  - Defense
  - Medical

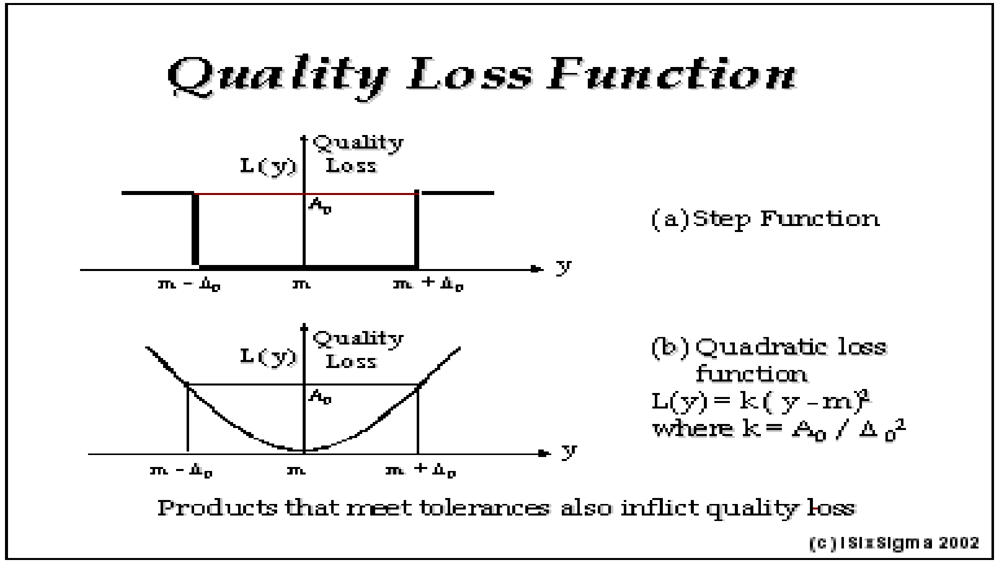


Image Source: http://www.isixsigma.com/methodology/robust-design-taguchi-method/introduction-robust-design-taguchi-method/

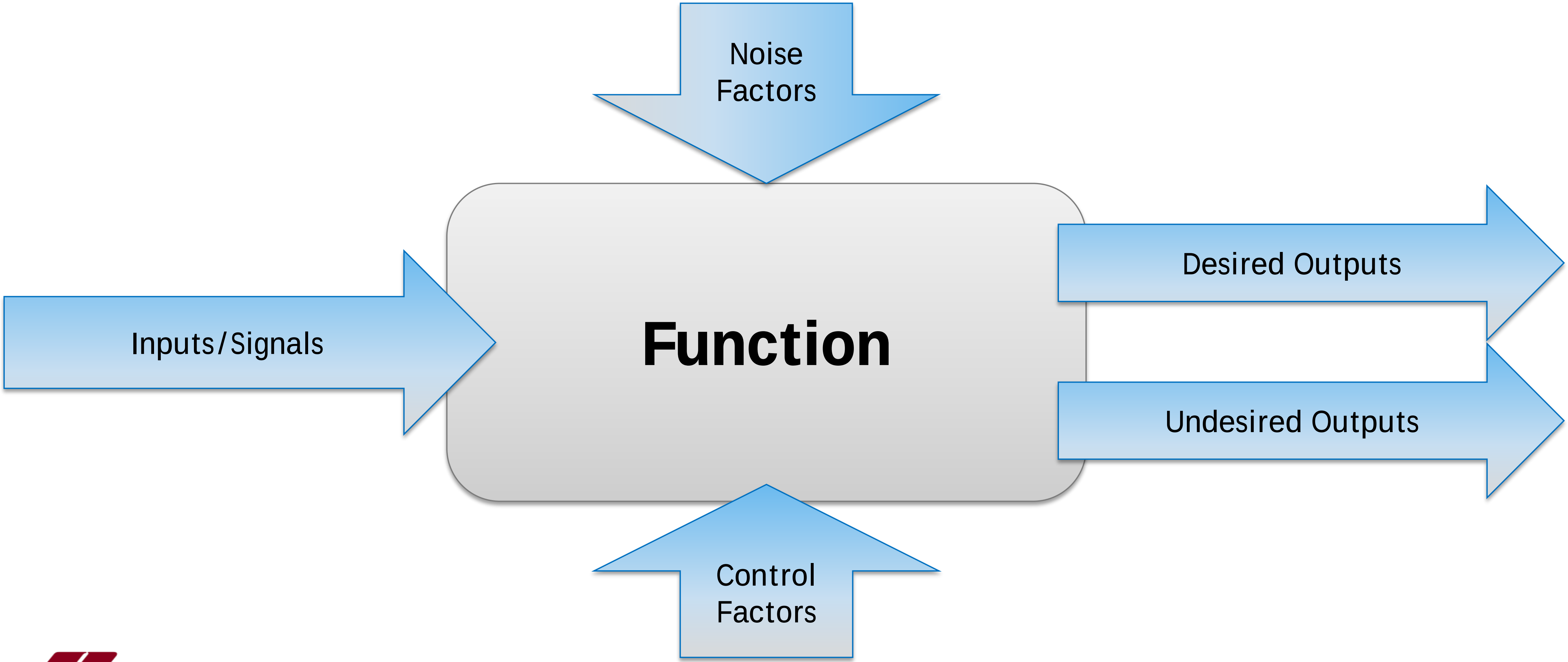
Parameter Design

- Parameter design is the cornerstone of Taguchi’s robust design philosophy

- Parameter design:
  - Identifies the element-of-interest’s transfer function that translates inputs into desired system response
  - Requires careful analysis of inputs, outputs, control factors, and noise factors
  - Facilitates understanding of design and manufacturing variables (control factors) that may be adjusted to maximize the system’s robustness

- Properly executed, Parameter Design provides:
  - Robust system design
  - Optimal performance
  - Rigorous understanding of the control factor trade space
  - Impact to understanding/maturing requirements

Traditional P-Diagram



Noise Factors

Inputs/Signals

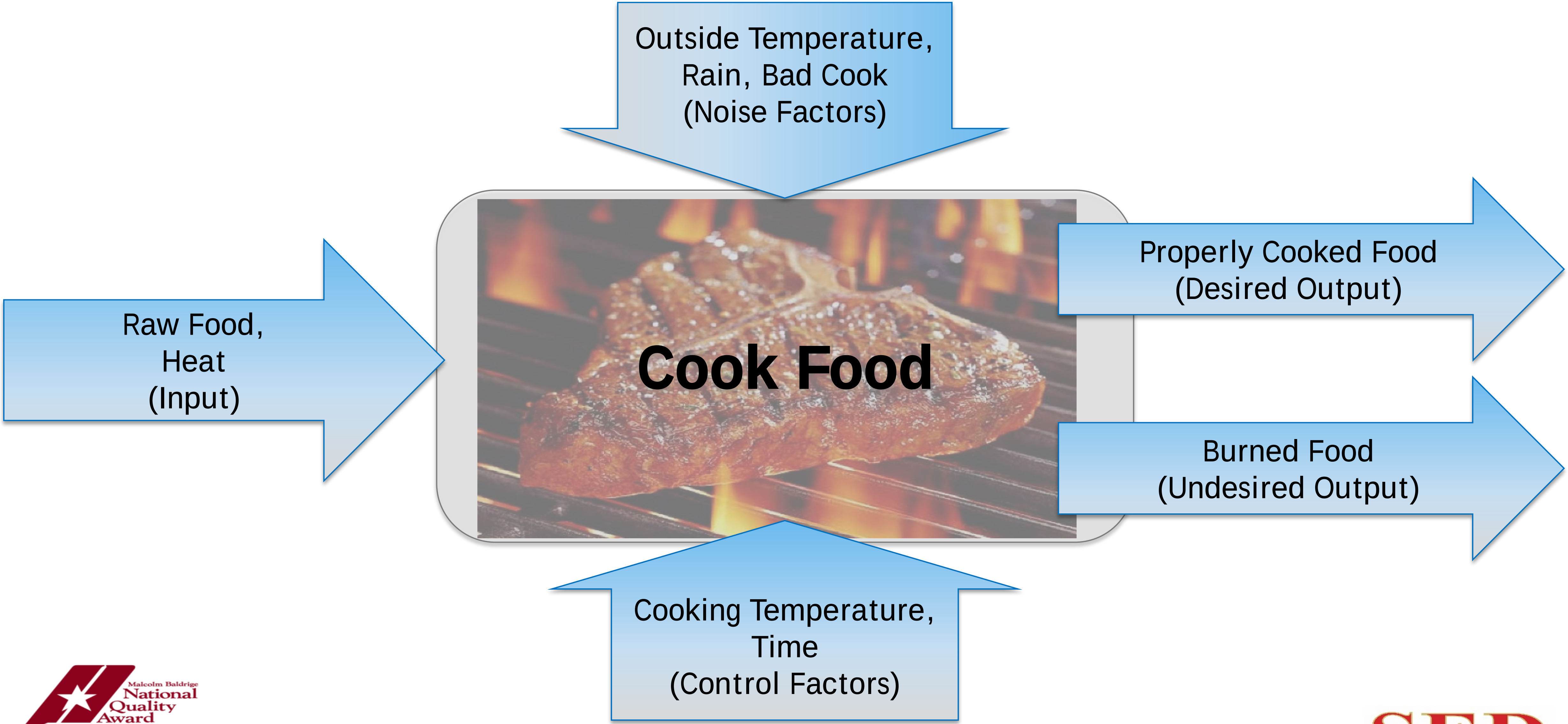
Function

Desired Outputs

Undesired Outputs

Control Factors

Traditional P-Diagram Example



Outside Temperature, Rain, Bad Cook (Noise Factors)

Raw Food, Heat (Input)

Cook Food

Properly Cooked Food (Desired Output)

Burned Food (Undesired Output)

Cooking Temperature, Time (Control Factors)

Limitations of P-Diagrams

- P-Diagrams, as developed traditionally, are inherently limited because they:
  - Typically are rendered as a “picture” in Visio or Excel
  - Are stored locally
  - Require a “human-in-the-loop” to interpret and share the content
  - May not be traced to other system elements

- Solution: Emerging system modeling techniques can be applied to expose the analysis and content inherent in developing a P-diagram to a wider audience of stakeholders

Engineers Love Models!

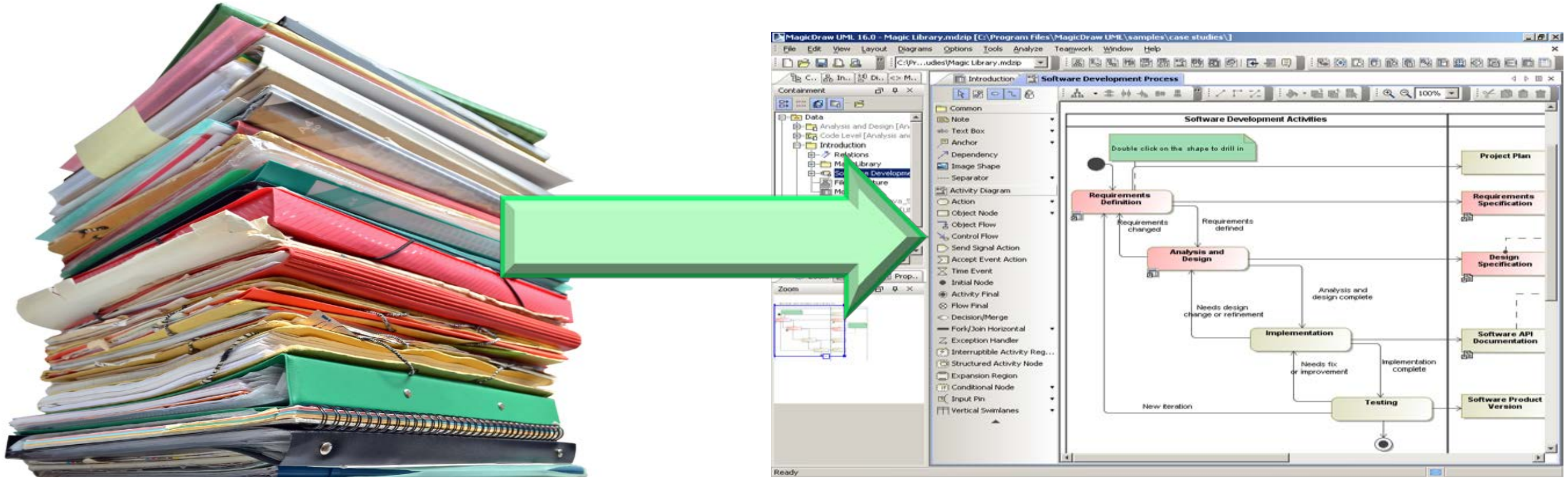
- What is a Model?
  - “A simplified or idealized description or conception of a particular system, situation, or process, often in mathematical terms, that is put forward as a basis for theoretical or empirical understanding, or for calculations, predictions, etc.; a conceptual or mental representation of something.”
    – Oxford English Dictionary

- Why do engineers love them?
  - Reality is often too complicated to “deal with” directly
  - Abstraction hides complexity and facilitates analysis

Models Provide Cognitive Leverage

Why Model Based Systems Engineering?

- Systems Engineering “grew up” with the progressively more complicated and complex systems developed during the 20th Century.
  - Document-based
  - Empirical
- By the 1960’s, attempts were being made to inject mathematics into SE (e.g., Wymore’s “*A Mathematical Theory of Systems Engineering: The Elements”* in 1967)
- By the 1990’s, serious efforts were underway to apply modeling methodologies (Wymore’s “*Model-Based Systems Engineering,” 1993*)


MagicDraw UML 16.0 - Magic Library.mdzip [C:\Program Files\MagicDraw UML\samples\case studies\]
Software Development Process
Software Development Activities
Double click on the shape to drill in
Requirements Definition
Requirements changed
Requirements defined
Analysis and Design
Needs design change or refinement
Analysis and design complete
Implementation
Needs fix or improvement
Implementation complete
Testing
New iteration
Project Plan
Requirements Specification
Design Specification
Software API Documentation
Software Product Version
Ready

Parallels

- Other disciplines were harnessing increases in relatively inexpensive computing power to maximize productivity and conduct analyses that were impractical before:
  - Computer-Aided Design (CAD)
  - Computational Fluid Dynamics
  - Stamping simulations
  - Mold-flow analysis
  - And many others…

- Systems engineering focuses on dealing with the complexities of system-level **behaviors**, **structure**, **requirements**, and **relationships**
  - Better methods and tools were needed to capture and analyze them
  - The previously used manual process of administering technical information could now benefit from the efficiencies introduced by these improvements in technology

SysML: Systems Modeling Language

- In 2001, the International Council on Systems Engineering established a Model Driven Systems Design workgroup to customize UML for systems engineering

- By 2006, OMG adopted OMG SysML (the current version is 1.4, adopted in March 2014)

- SysML provides for the following diagram types, with numerous relationship available between model elements:
  - Behavioral Diagrams: Use case, Activity, Sequence, State Machine
  - Structural Diagrams: Block Definition, Internal Block, Package
  - Other Diagrams: Requirements, Package

The System Model

- Other system modeling languages exist, but SysML is the most widely-adopted and has a thriving tool ecosystem
- A well-constructed system model unambiguously represents a system's behavior, structure, and interrelationships between elements
- SysML fosters a "crispness" in the formulation of issues (according to David Miller, NASA Chief Technologist)
- Therefore, System Modeling is inherently compatible with Taguchi Methods
  - Functions/Operations are well-suited to capturing the content of a P-Diagram
- SysML tools allow the model content to be expressed as tables, matrices, and other derivative work products

Summary of Steps

1
- Create P-Diagram Function generalized block
  - Used as a template for your functions

2
- Create one operation for each function
  - These will be specialized P-Diagram Functions

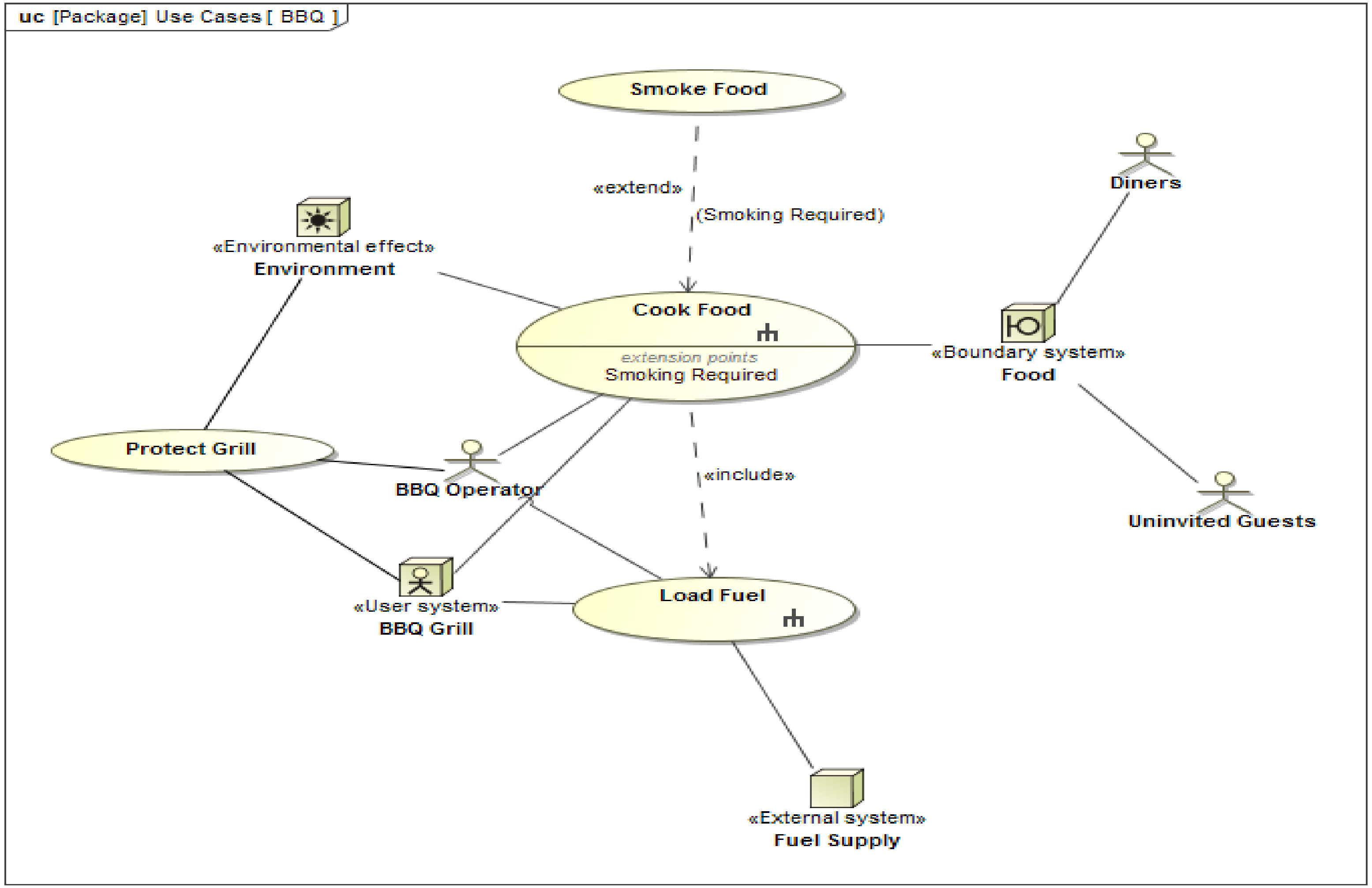
3
- Create a library of signals (parameters) to use for the inputs, outputs, controls, and noise factors

4
- Populate function (operation) pins with the signals
  - Display flow on an activity diagram

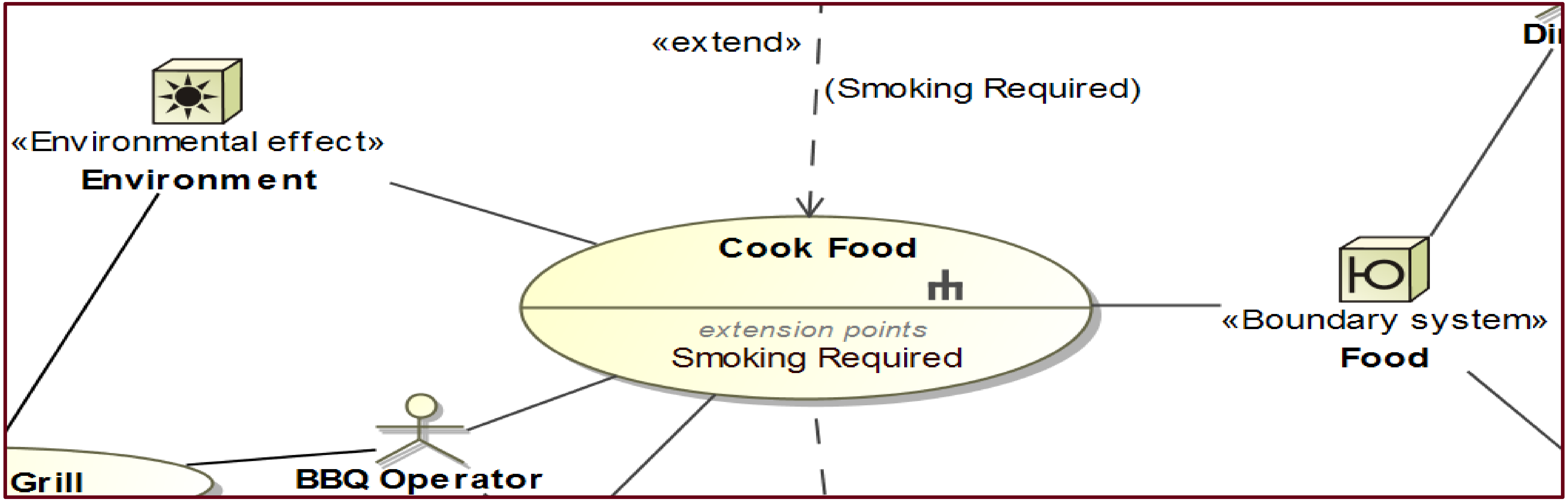
5
- Trace elements and generate secondary products

Use Case Diagrams

- Describe how the system is used
- Capture external systems and actors, goals, conditions, etc.
- Shows who participates in actions
- Are a useful starting point to capture system behaviors

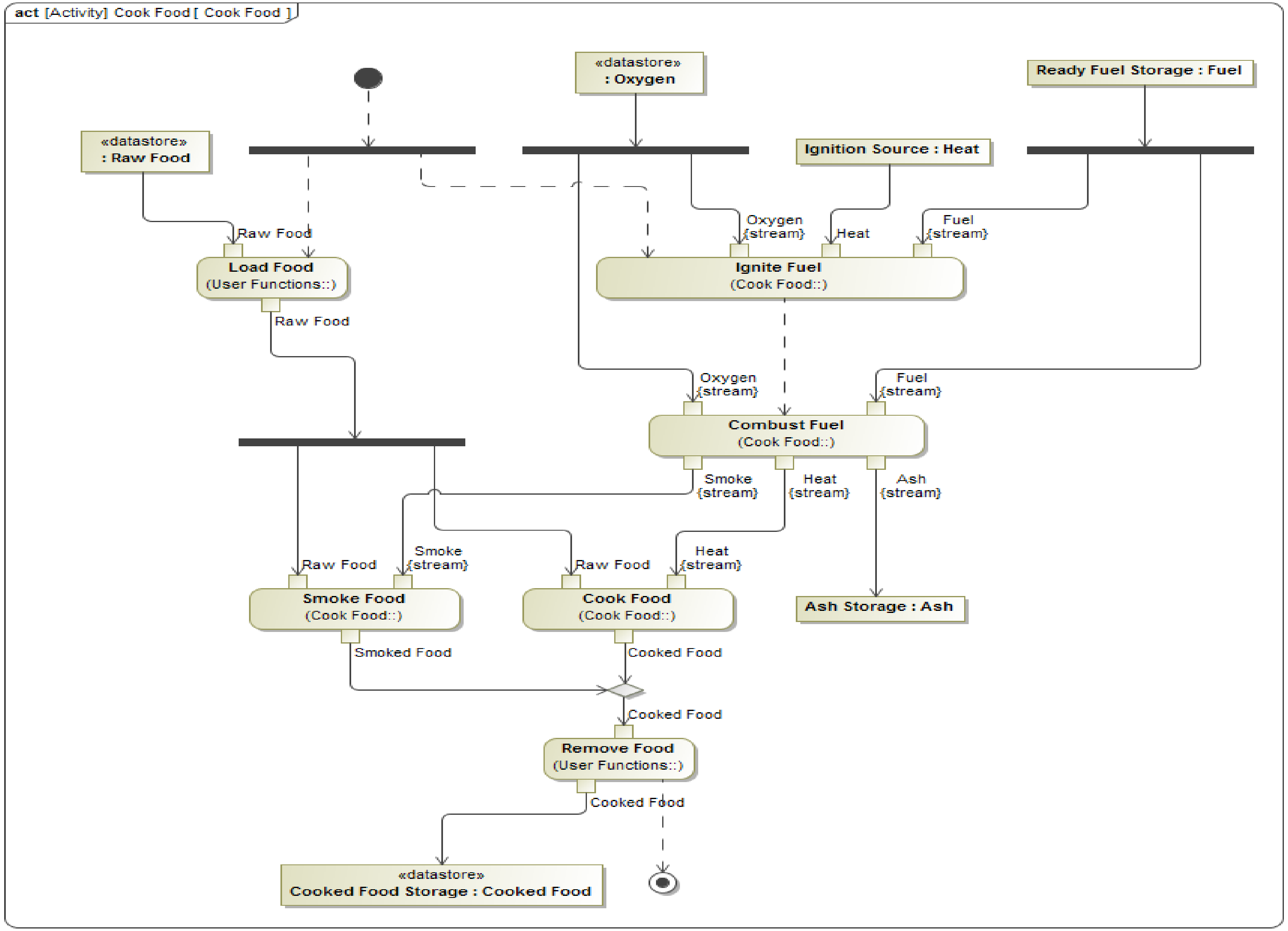


Use Case Diagram Detail

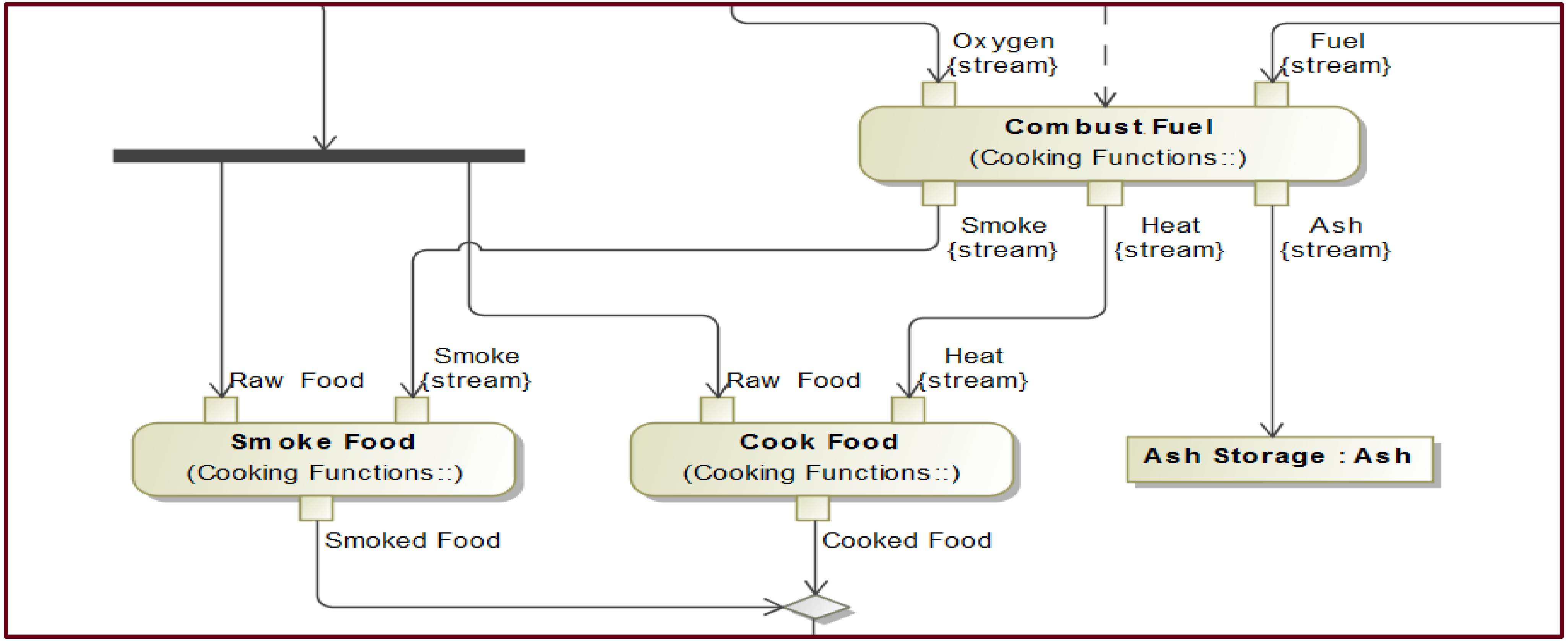


Activity Diagrams

- Show the flow of events
- Show the flow of signals
- Capture Decision Points
- Represent functional activities with inputs/outputs

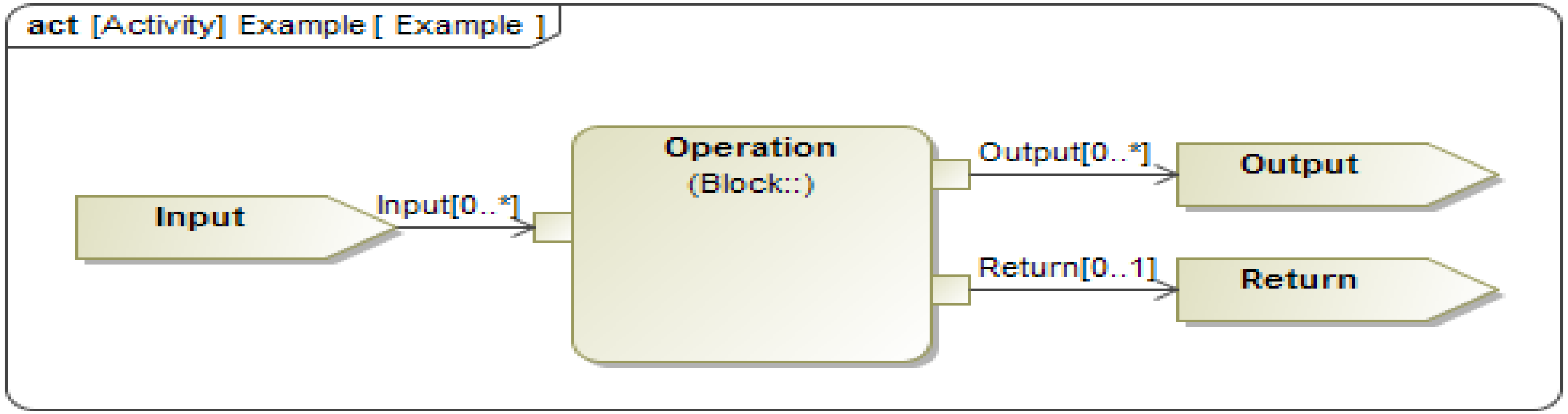


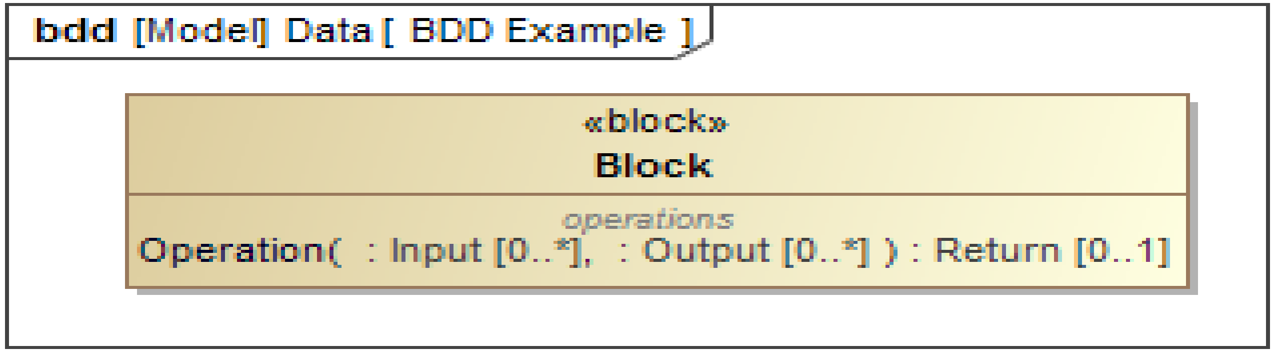
Activity Diagram Detail



Operations (Input/Output/Return)

- In SysML a *function* is represented by an *operation*
  - *Operations* must be owned by a *block*
  - *Operations* may have one or many *inputs* and *outputs* parameters
  - *Operations* may also have only one *return* parameter




bdd [Model] Data [BDD Example]
«block»
Block
operations
Operation(: Input [0..*], : Output [0..*]) : Return [0..1]

Using Signals for Inputs/Outputs

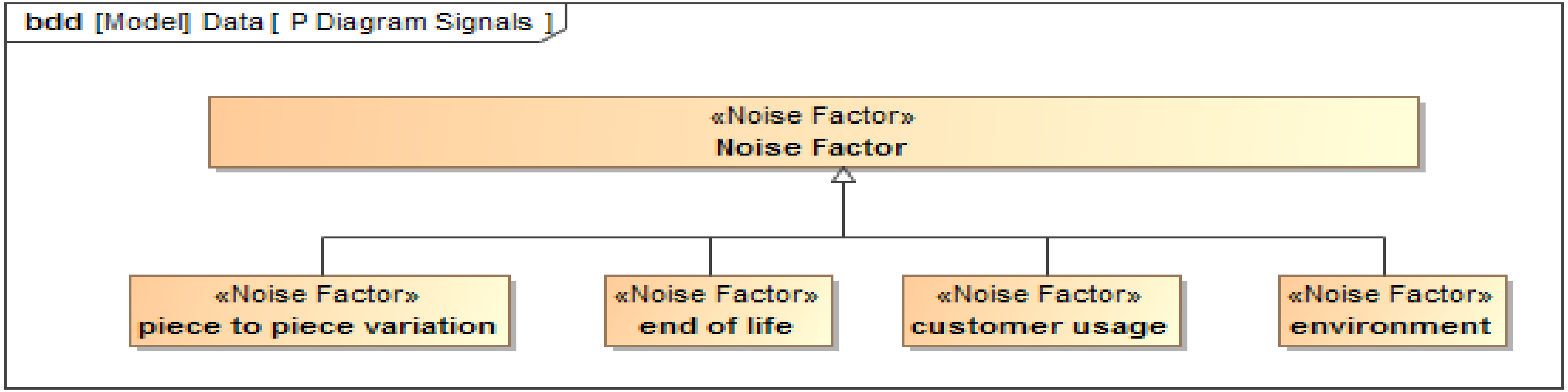
- In SysML, a *signal* is a model element used to type other model elements, such as:
  - *Ports*
  - *Input/output parameters*
  - *Information flows*

- By typing these elements, the modeling tool can check for consistency and ensure that compatible ports and parameters are connected

- Typing using signals allows for internal consistency and reduced manual inputs
  - For example, if you change the name of the signal, all the places it is used will also change

Stereotyping Signals

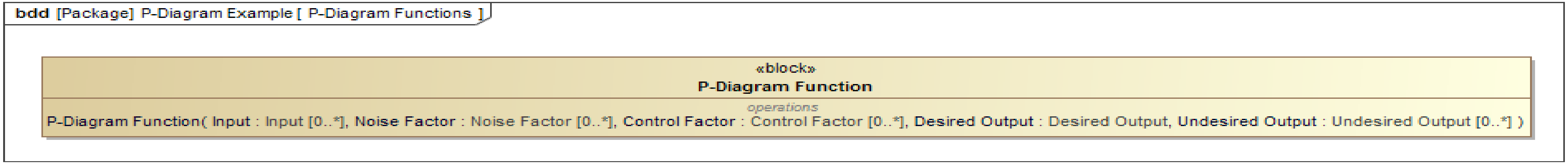
- SysML provides for the application of *stereotypes* to model elements
- Stereotyping signals is a useful method for classifying them
- Note that all subtypes (specializations) of the *Noise Factor* signal may satisfy requirements or ports that are typed with *Noise Factor*



P-Diagram Function

- A *P-Diagram Function* has been created as an operation

- Its inputs and output parameters are typed with the appropriate signals
  - The multiplicity for these parameters is set to [0..*]
    (they are optional, with no upper bound)



bdd [Package] P-Diagram Example [P-Diagram Functions]

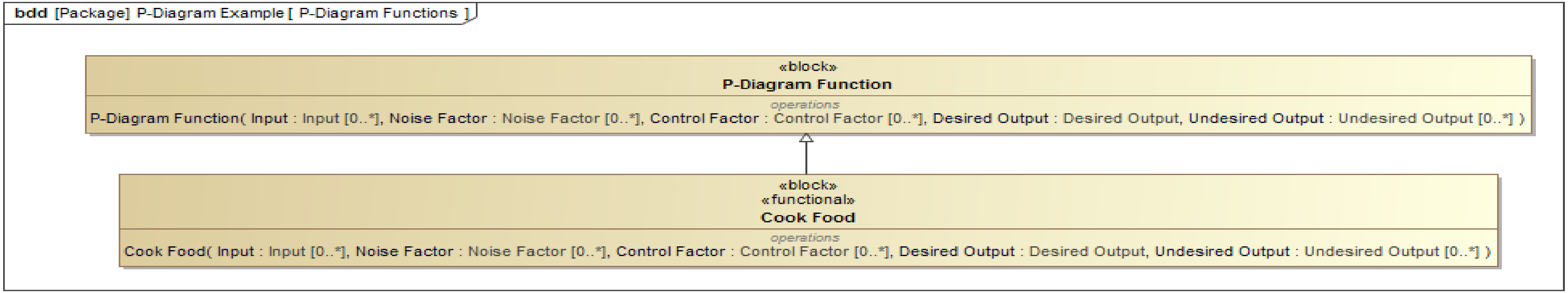
«block»
P-Diagram Function

operations

P-Diagram Function(Input : Input [0..*], Noise Factor : Noise Factor [0..*], Control Factor : Control Factor [0..*], Desired Output : Desired Output, Undesired Output : Undesired Output [0..*])

Specialization and Inheritance

- Now that the P-Diagram operation has been defined, any other operation may specialize it
  - A specialized block inherits the properties of the generalized block
  - The specialized block (*Cook Food*) can redefine *P-Diagram Function*



bdd [Package] P-Diagram Example [P-Diagram Functions]

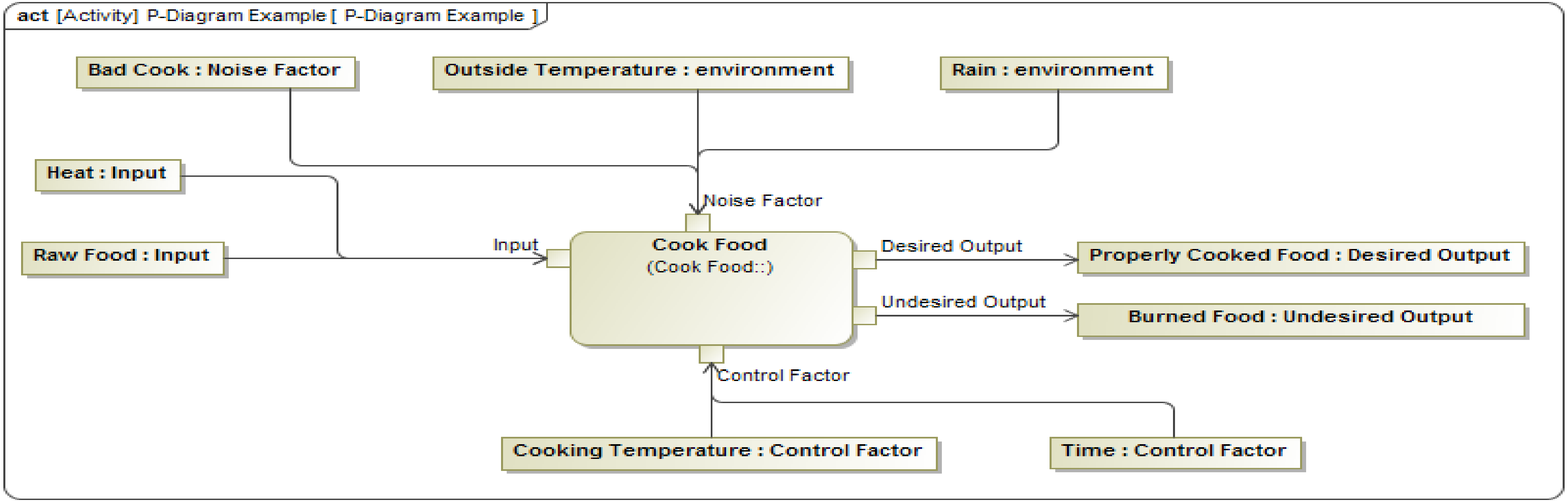
«block»
P-Diagram Function

operations
P-Diagram Function(Input : Input [0..*], Noise Factor : Noise Factor [0..*], Control Factor : Control Factor [0..*], Desired Output : Desired Output, Undesired Output : Undesired Output [0..*])

«block»
«functional»
Cook Food

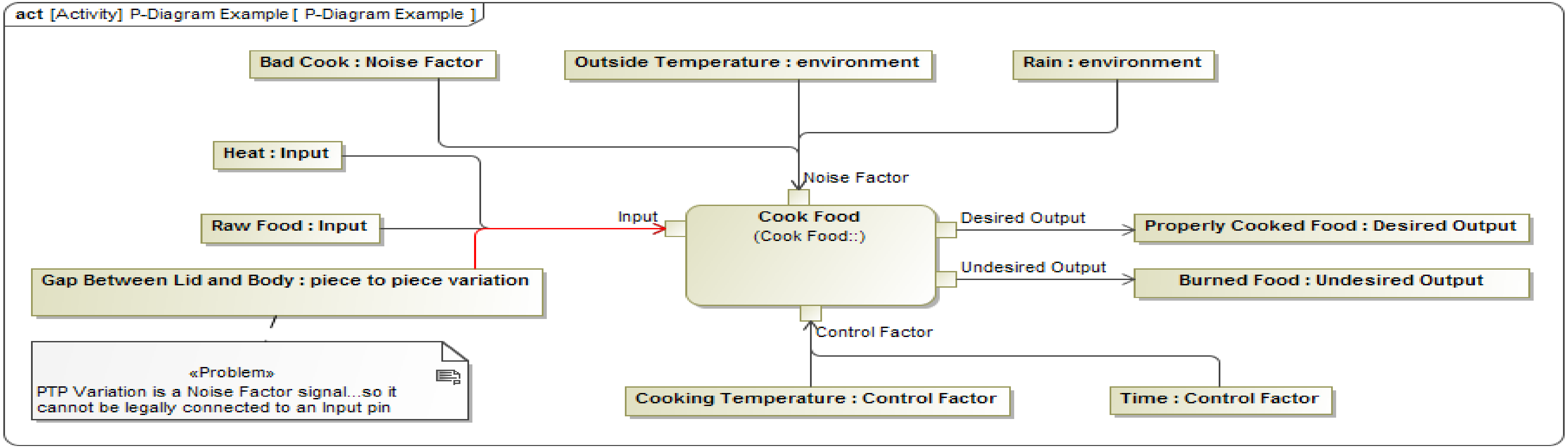
operations
Cook Food(Input : Input [0..*], Noise Factor : Noise Factor [0..*], Control Factor : Control Factor [0..*], Desired Output : Desired Output, Undesired Output : Undesired Output [0..*])

SysML Representation of P-Diagram



act [Activity] P-Diagram Example [P-Diagram Example]
Bad Cook : Noise Factor
Outside Temperature : environment
Rain : environment
Heat : Input
Raw Food : Input
Noise Factor
Input
Cook Food
(Cook Food::)
Desired Output
Properly Cooked Food : Desired Output
Undesired Output
Burned Food : Undesired Output
Control Factor
Cooking Temperature : Control Factor
Time : Control Factor

Error Checking

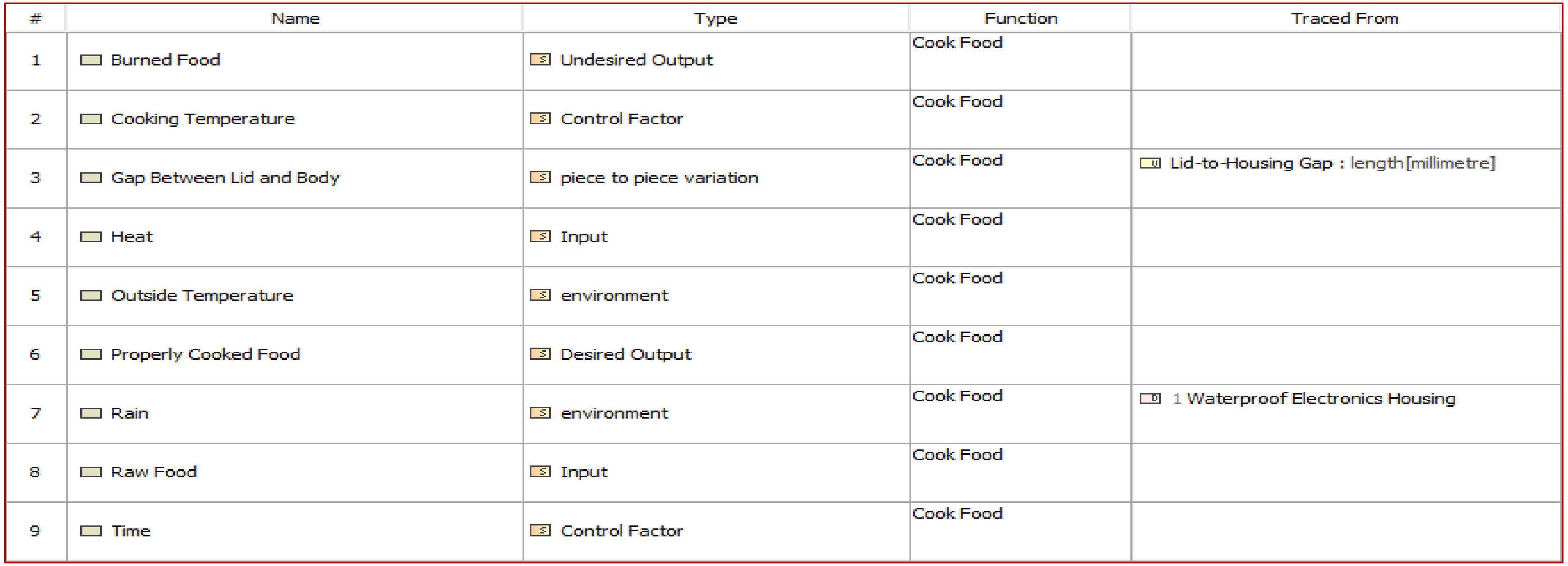

act [Activity] P-Diagram Example [P-Diagram Example]
Bad Cook : Noise Factor
Outside Temperature : environment
Rain : environment
Heat : Input
Noise Factor
Raw Food : Input
Input
Cook Food
(Cook Food::)
Desired Output
Properly Cooked Food : Desired Output
Undesired Output
Burned Food : Undesired Output
Gap Between Lid and Body : piece to piece variation
Control Factor
Cooking Temperature : Control Factor
Time : Control Factor

«Problem»
PTP Variation is a Noise Factor signal…so it cannot be legally connected to an Input pin

The tool checks for compatibility between signals and pins. Mismatches are shown as an error.

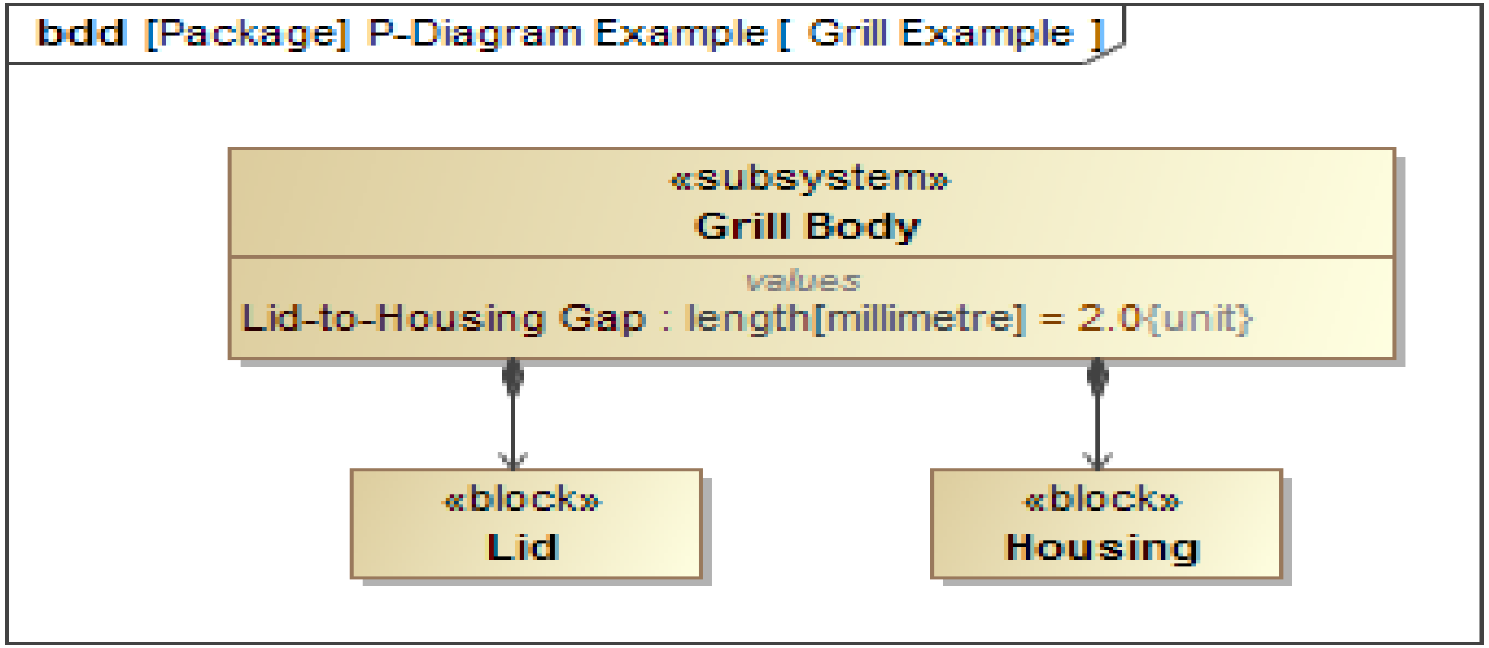
Secondary Work Product: Tables

- Tables may be used to conveniently summarize the inputs, outputs, noise factors, & control factors
- These model elements may then be traced to other elements, such as:
  - Value properties
  - Requirements



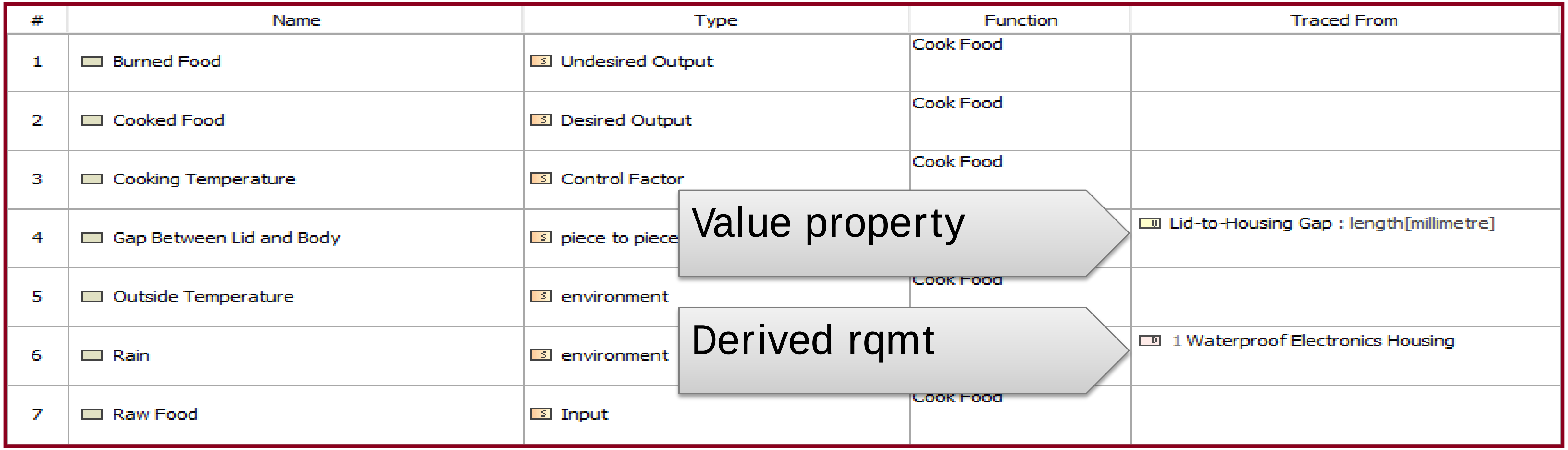
#	Name	Type	Function	Traced From
1	Burned Food	Undesired Output	Cook Food	
2	Cooking Temperature	Control Factor	Cook Food	
3	Gap Between Lid and Body	piece to piece variation	Cook Food	Lid-to-Housing Gap : length[millimetre]
4	Heat	Input	Cook Food	
5	Outside Temperature	environment	Cook Food	
6	Properly Cooked Food	Desired Output	Cook Food	
7	Rain	environment	Cook Food	1 Waterproof Electronics Housing
8	Raw Food	Input	Cook Food	
9	Time	Control Factor	Cook Food	

Value Property Example


bdd [Package] P-Diagram Example [Grill Example]
«subsystem»
Grill Body
values
Lid-to-Housing Gap : length[millimetre] = 2.0{unit}
«block»
Lid
«block»
Housing

A value property can be a control factor or a noise factor. These typically include a value and unit type.

Factors with Traces

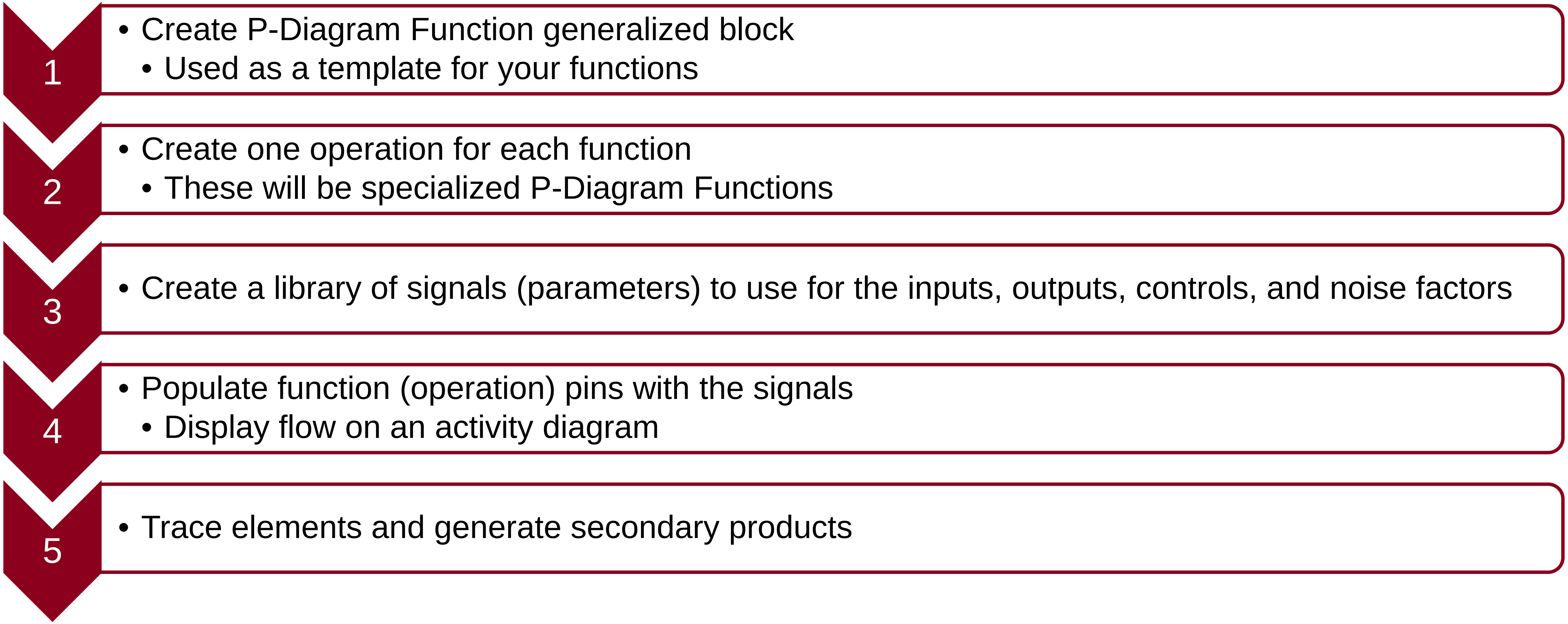


#	Name	Type	Function	Traced From
1	▭ Burned Food	▭ Undesired Output	Cook Food	
2	▭ Cooked Food	▭ Desired Output	Cook Food	
3	▭ Cooking Temperature	▭ Control Factor	Cook Food	
4	▭ Gap Between Lid and Body	▭ piece to piece		▭ Lid-to-Housing Gap : length[millimetre]
5	▭ Outside Temperature	▭ environment	Cook Food	
6	▭ Rain	▭ environment		▭ 1 Waterproof Electronics Housing
7	▭ Raw Food	▭ Input	Cook Food	

Value property

Derived rqmt

Summary of Steps



1

- Create P-Diagram Function generalized block
  - Used as a template for your functions

2

- Create one operation for each function
  - These will be specialized P-Diagram Functions

3

- Create a library of signals (parameters) to use for the inputs, outputs, controls, and noise factors

4

- Populate function (operation) pins with the signals
  - Display flow on an activity diagram

5

- Trace elements and generate secondary products

Conclusion

- System models are the most useful when they serve as a **"single repository of truth"**
- As much relevant information as possible should be integrated into the model
  - Maximize the benefit of the model
  - Get the most out of secondary work products, such as tables and matrices
  - Expose, analyze, and control the relationships between system elements
- Tools and methods unburden, but do not replace, good engineering judgment
- This presentation describes the methods to capture P-Diagram information in a manner that facilitates information integration and traceability using SysML