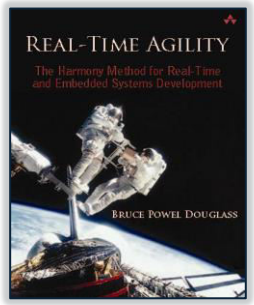
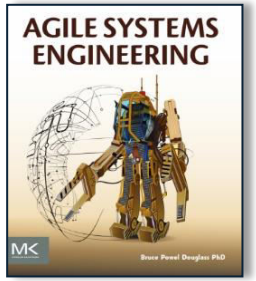
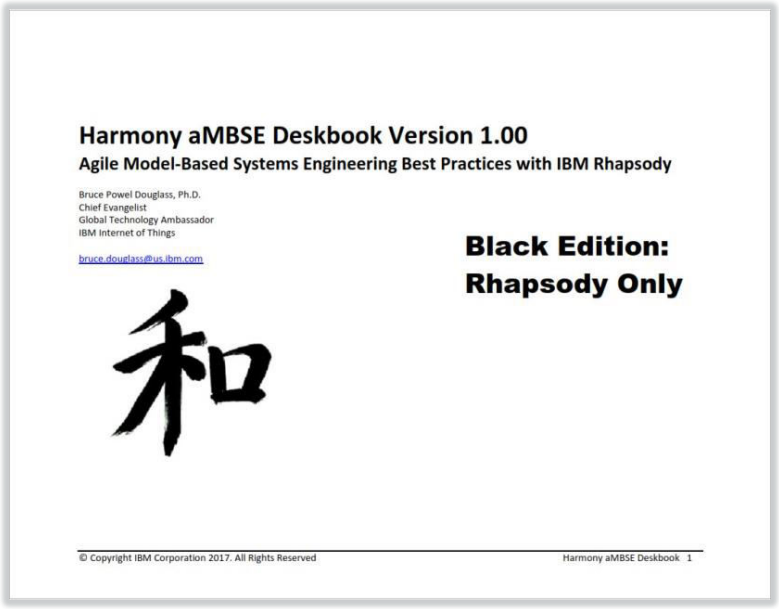
Improving System Requirements With Use Case Analysis

Bruce Powel Douglass, Ph.D.
Principal

A Priori Systems
www.bruce-douglass.com



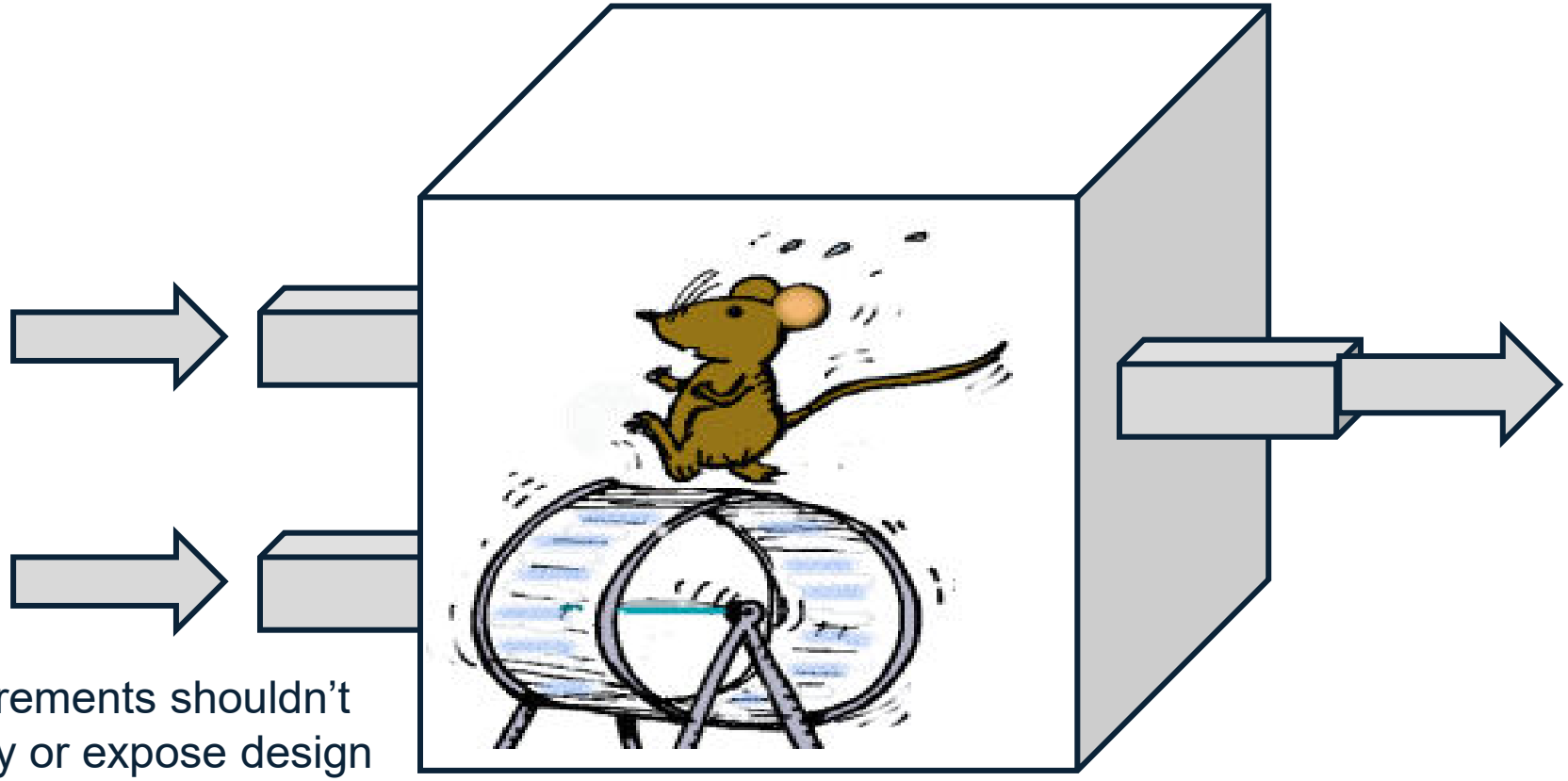




A Priori Systems
Real-Time Agile Systems and Software

Requirement

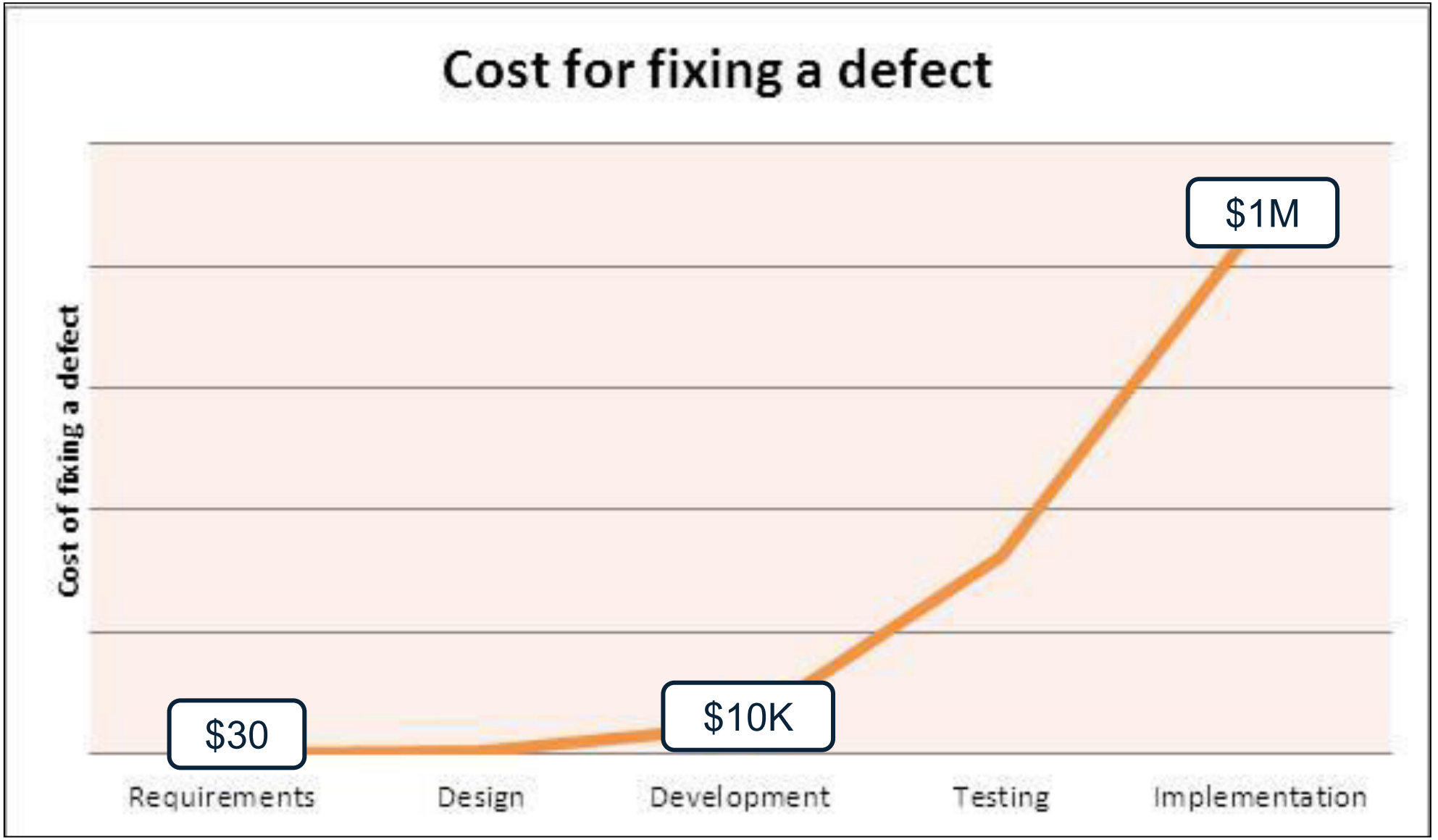
- A ***requirement*** is about ***what*** needs to happen and not about ***how*** it happens
- A ***functional requirement*** is a statement about input → output control or data transformation
- A ***quality of service requirement*** is a statement about how well that functionality must be performed.



Requirements shouldn't specify or expose design

A Priori Systems
Real-Time Agile Systems and Software

Poor requirements have a huge impact

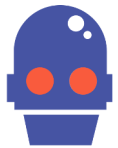


A Priori Systems
Real-Time Agile Systems and Software

The Three Ways of Verifying “Goodness”

1. Review / Inspection
2. Test
3. Formal methods

With textual requirements you can really only apply #1; with modeled use cases, you can apply all three



A Priori Systems
Real-Time Agile Systems and Software

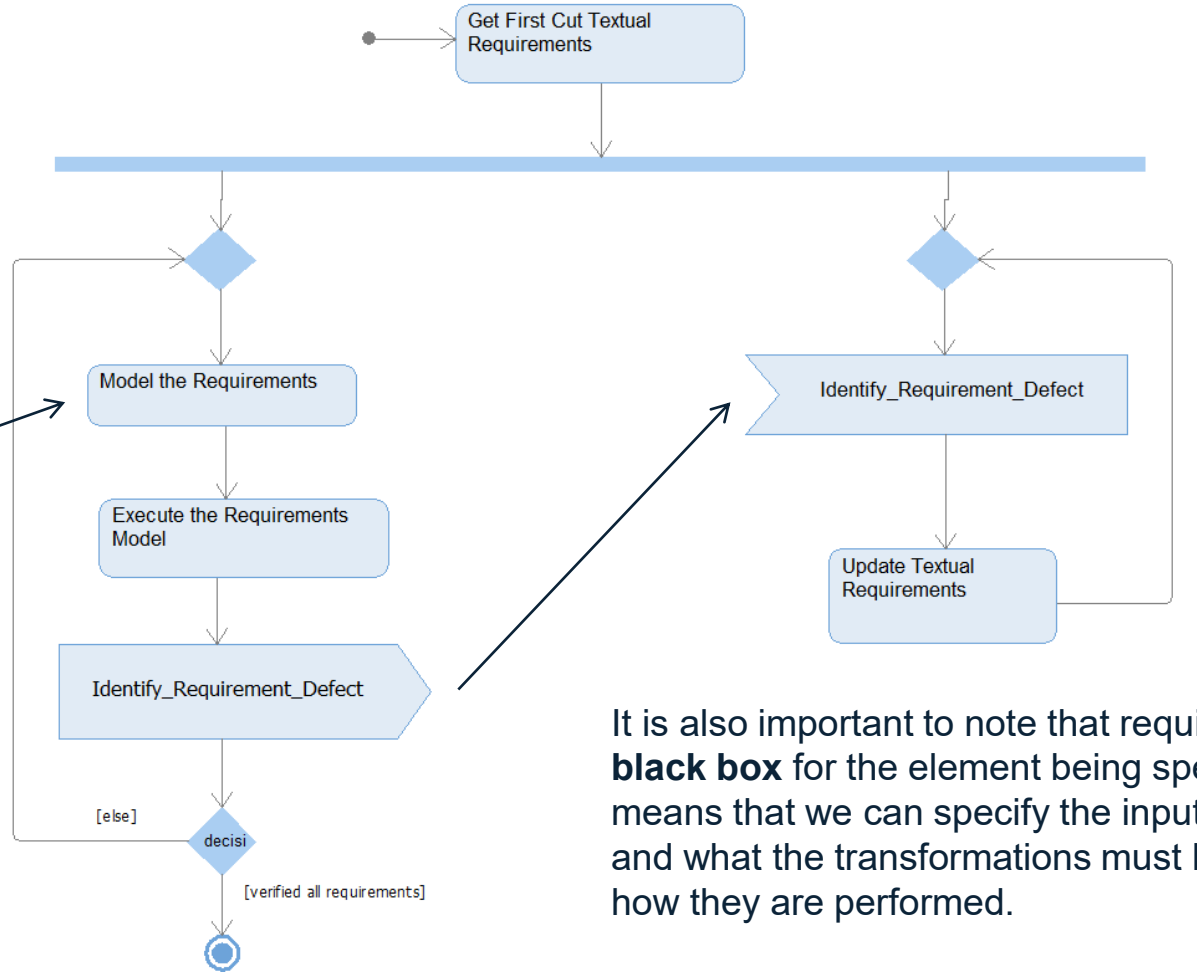
The Requirements Modeling Approach

- The approach we will take to perform verification and validation on our requirements before satisfying them with our design is to:

This is an important point:
When we model messages, activities, actions, states, transitions, etc., we are simply re-writing the requirements in a more precise language.

We are not modeling non-requirements or design.

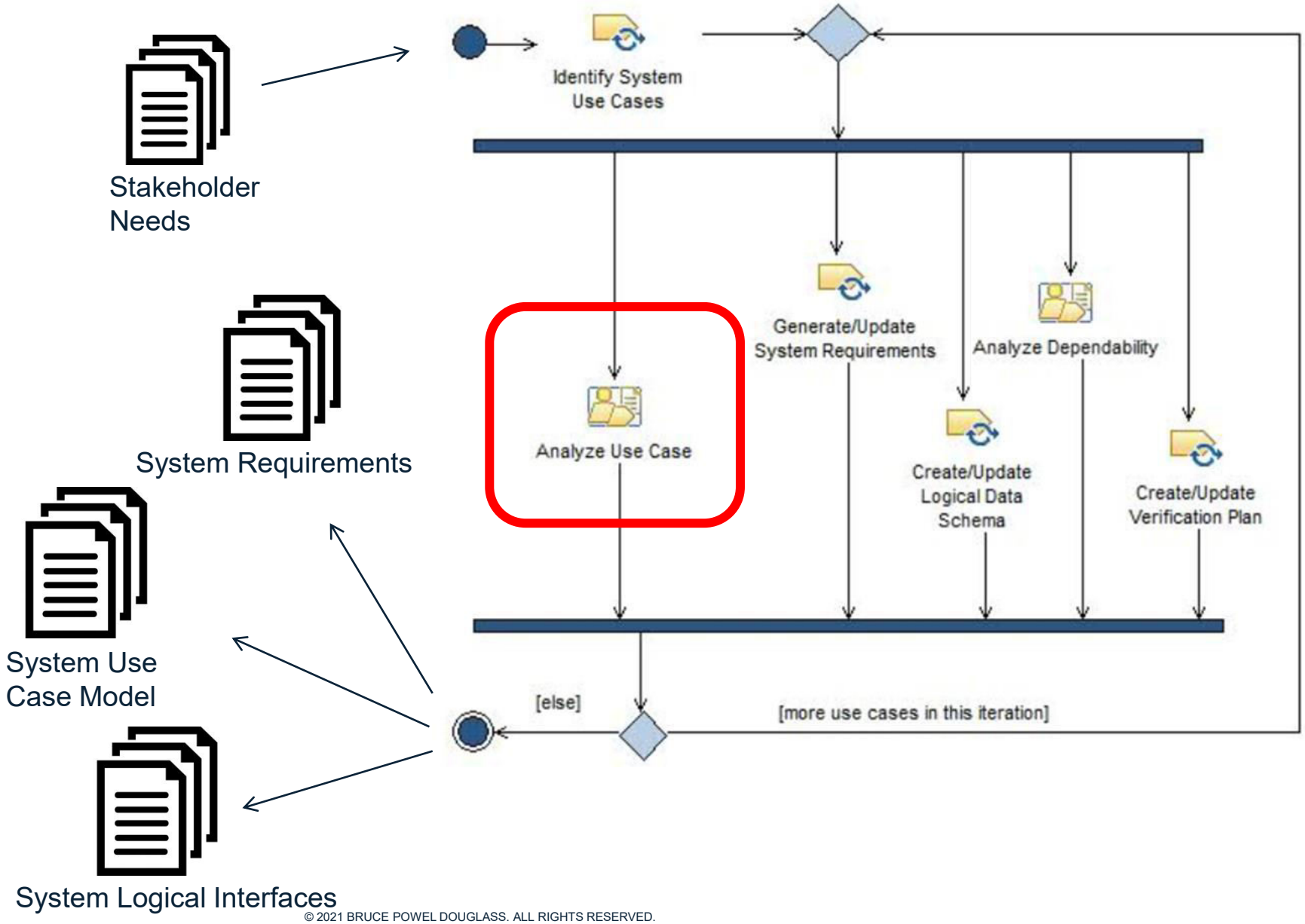
Put another way, our requirements model is just a precisely worded restatement of the requirements.



It is also important to note that requirements are **black box** for the element being specified. That means that we can specify the inputs, outputs and what the transformations must be, but not how they are performed.

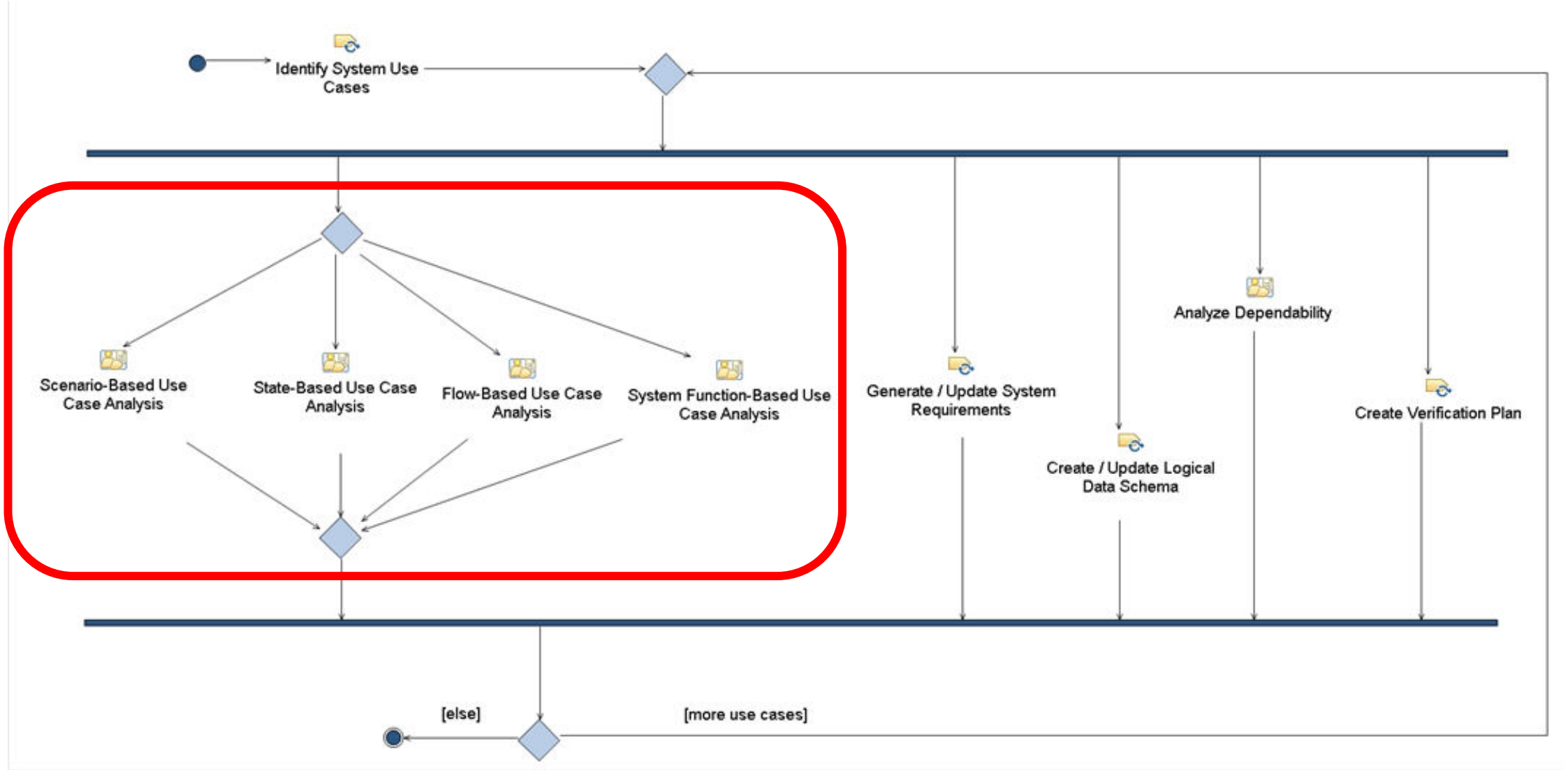
A Priori Systems
Real-Time Agile Systems and Software

Requirements Workflow



A Priori Systems
Real-Time Agile Systems and Software

Harmony aMBSE: System Requirements Analysis Alternatives



A Priori Systems
Real-Time Agile Systems and Software

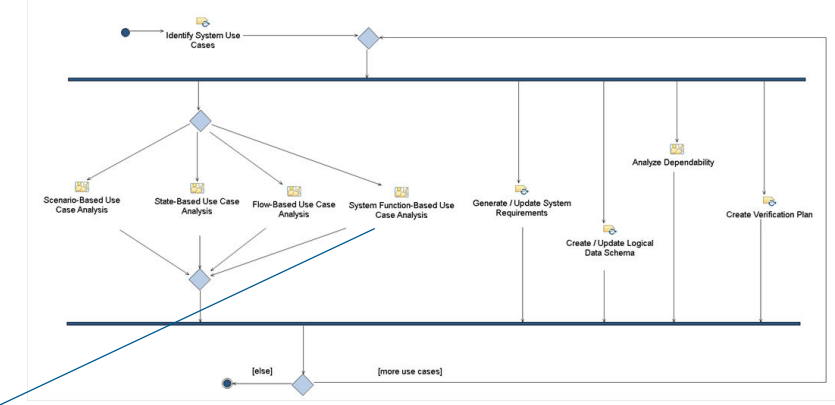
System-Function Based (most common)

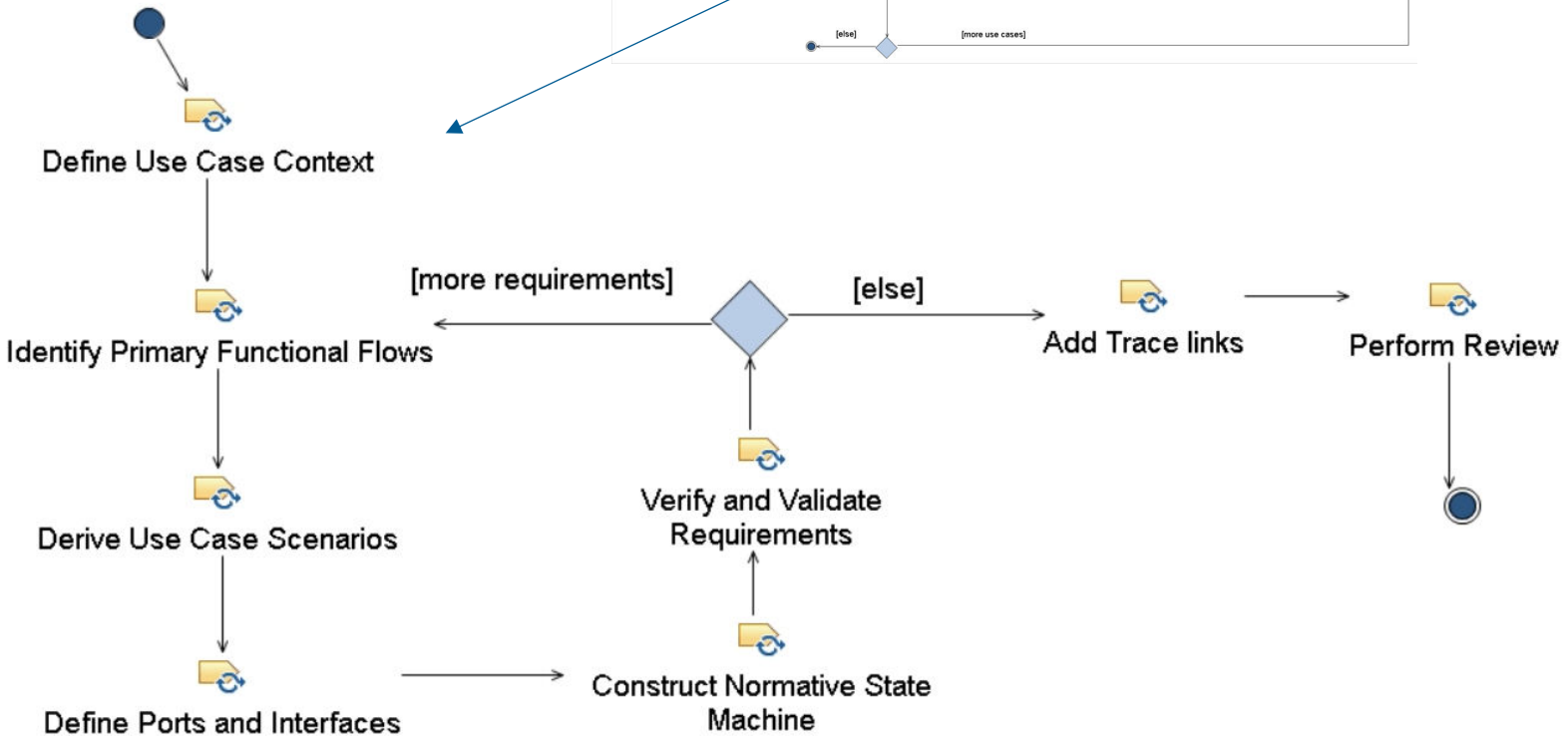
Properties

- Activity diagrams model incomplete but used to initiate the analysis
  - They only show primary control flows
  - May not be completely “well formed”
  - Actions map to system functions
- Activity diagram’s purpose is to identify and characterize system functions and their sequencing
- State machines hold the “source of truth” and scenarios are exemplars. Signals pass information that is defined in the data and flow schema

When to use

- The primary complexity or concern occurs in the set of system functions that implement the use case
- Continuous or value flows hold less concern
- System engineers are primary contributors to system understanding





A Priori Systems
Real-Time Agile Systems and Software

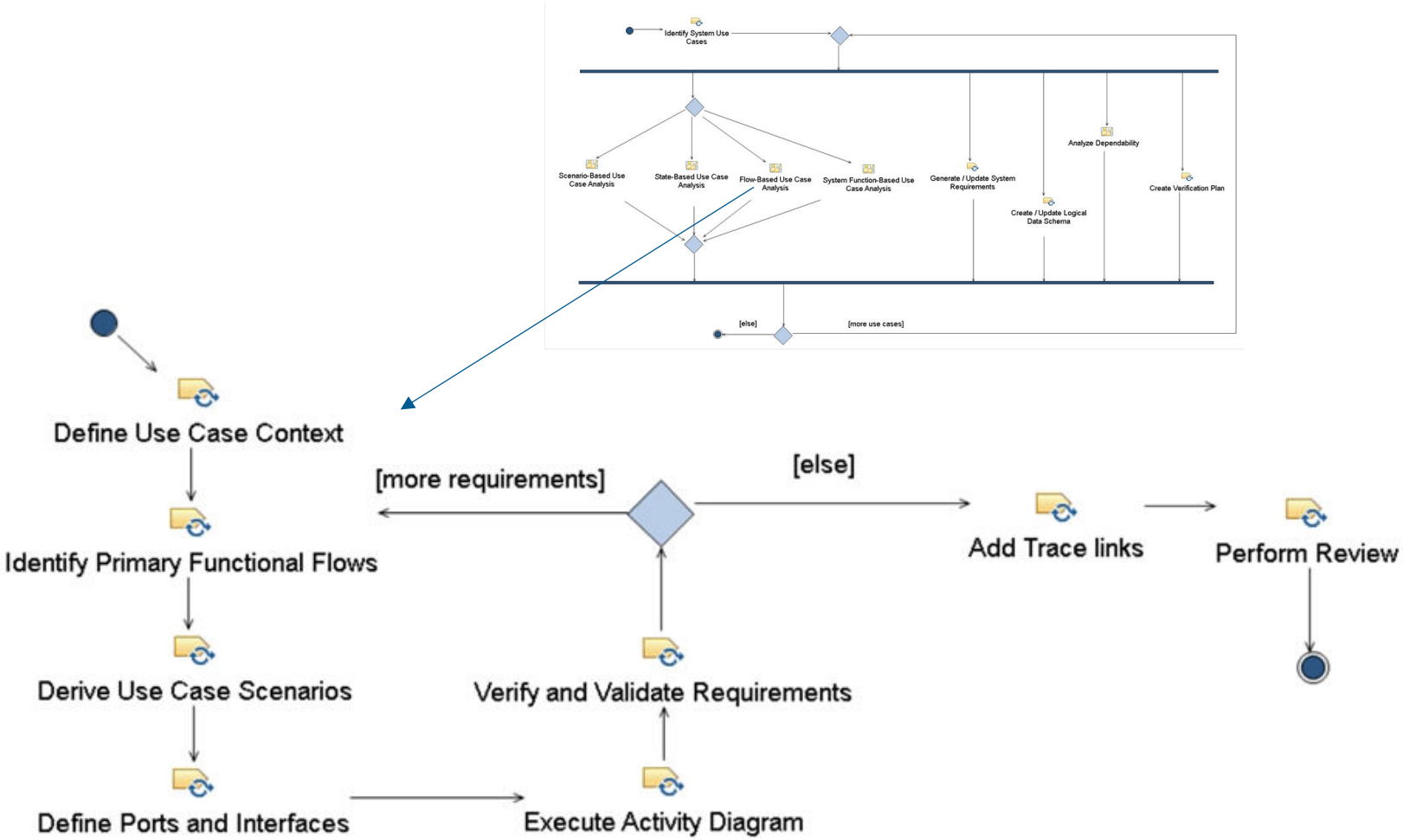
Flow Based

Properties

- Activity diagrams model becomes the source of truth
  - Must capture all primary, secondary, and exception control flows
  - Actions map to system functions
- Continuous and non-data flows are represented using pins and object flows
- Elements passed in object tokens are defined by data and flow schema

When to use

- The primary complexity or concern occurs in the input and output flows of the system when performing the use case
- Continuous or value flows are of significant concern, including energy, fluids, mass, and materiel
- System engineers are primary contributors to system understanding



A Priori Systems
Real-Time Agile Systems and Software

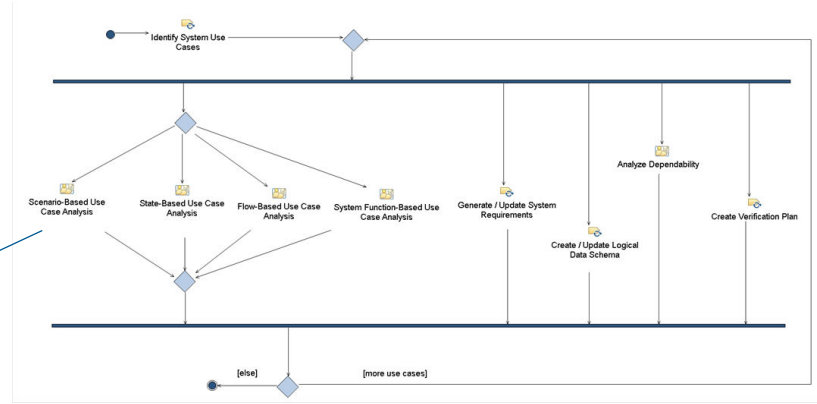
Scenario-based (aka “Interaction –Based”)

Properties

- Skips activity modeling
- Scenarios capture all requirements associated with the use case including
  - Quality of service
  - Edge/exception cases
- Scenarios are primarily used to elicit requirements
- State machine is the “source of truth” and represents all possible scenarios. Signals pass information that is defined in the data and flow schema

When to use

- Primary concern is the interaction between the system (running the use case) and the actors
- Non-technical stakeholders are primary contributors to system understanding



A Priori Systems
Real-Time Agile Systems and Software

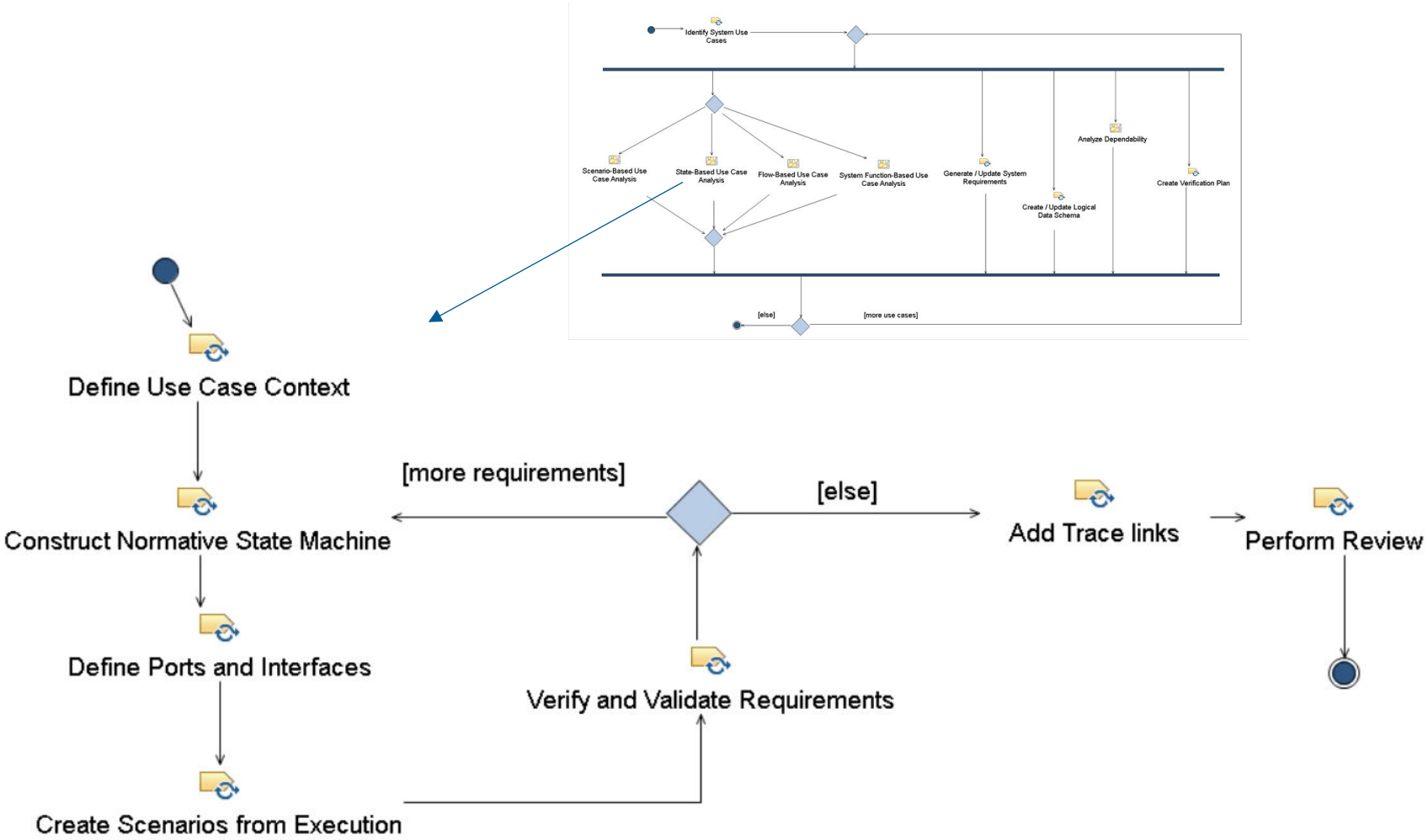
State-based

Properties

- Skips activity modeling
- Captures all requirements associated with the use case including
  - Quality of service
  - Edge/exception cases
- Scenarios are generated from state machines and are used to validate the requirements with the stakeholders.
- State machine is the “source of truth”. Signals pass information that is defined in the data and flow schema

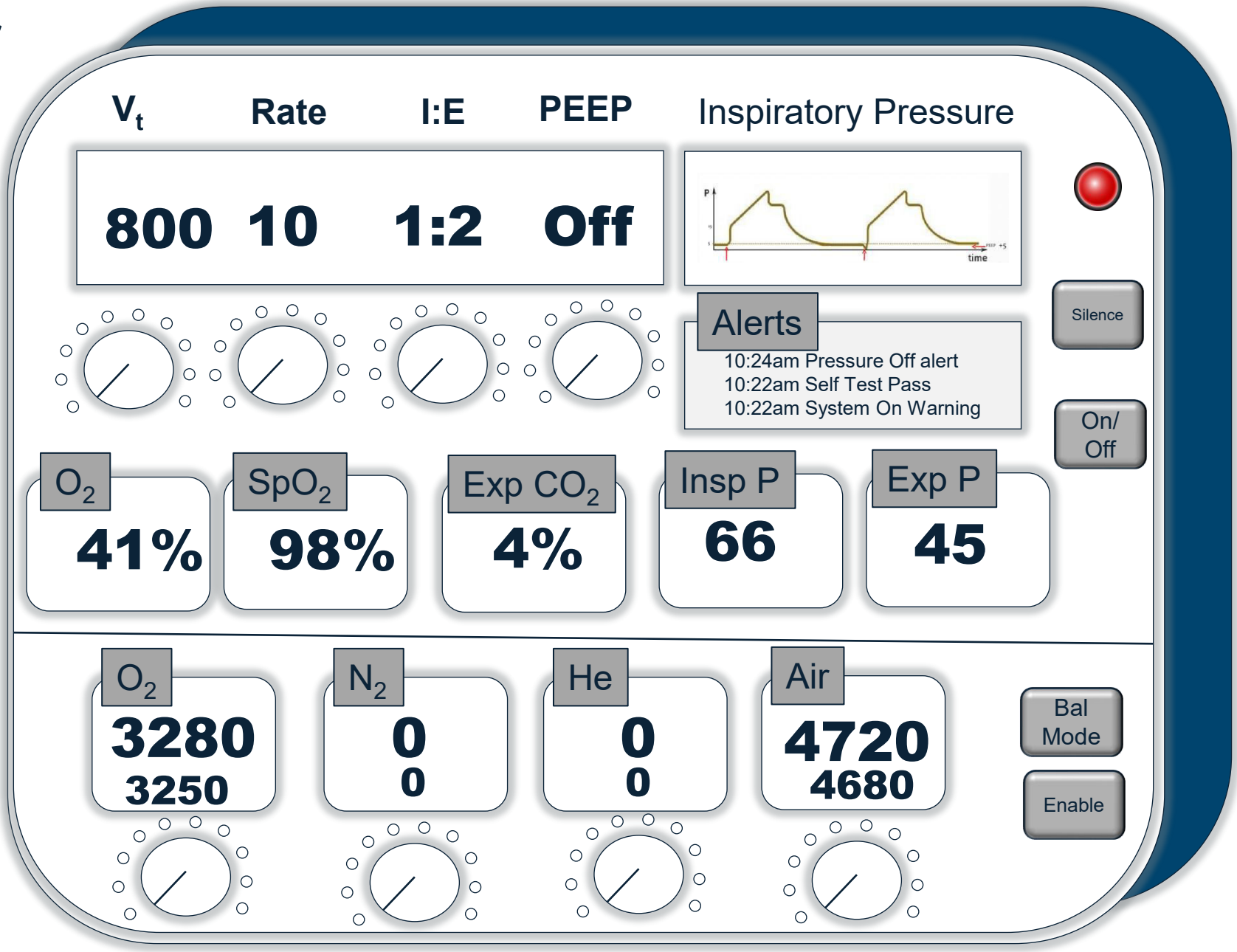
When to use

- Primary concern is the modes of operation and the differences in system behavior in those different modes
- Technical stakeholders are primary contributors to system understanding



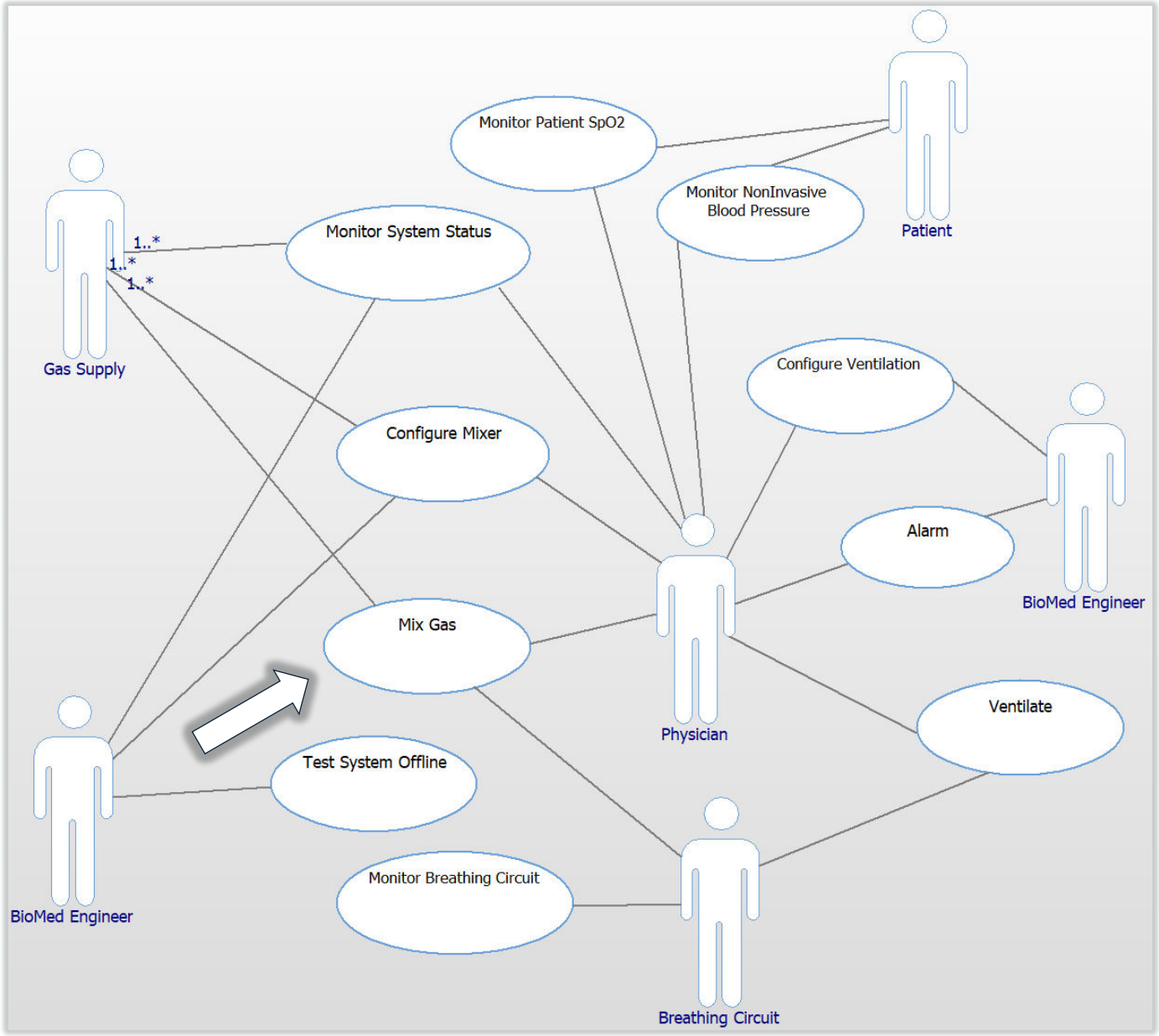
A Priori Systems
Real-Time Agile Systems and Software

Example:
Medical Ventilator



A Priori Systems
Real-Time Agile Systems and Software

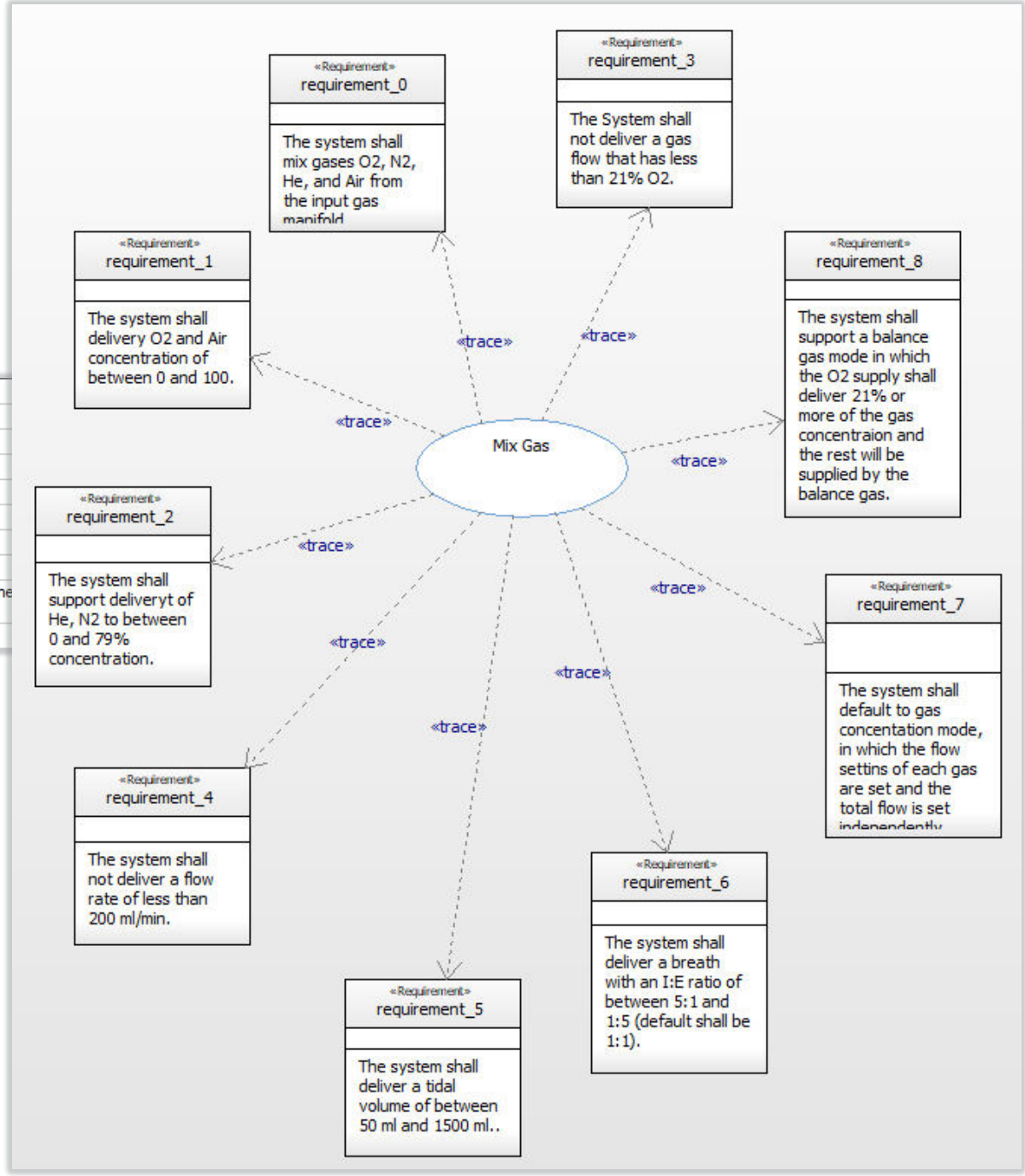
Use Cases



A Priori Systems
Real-Time Agile Systems and Software

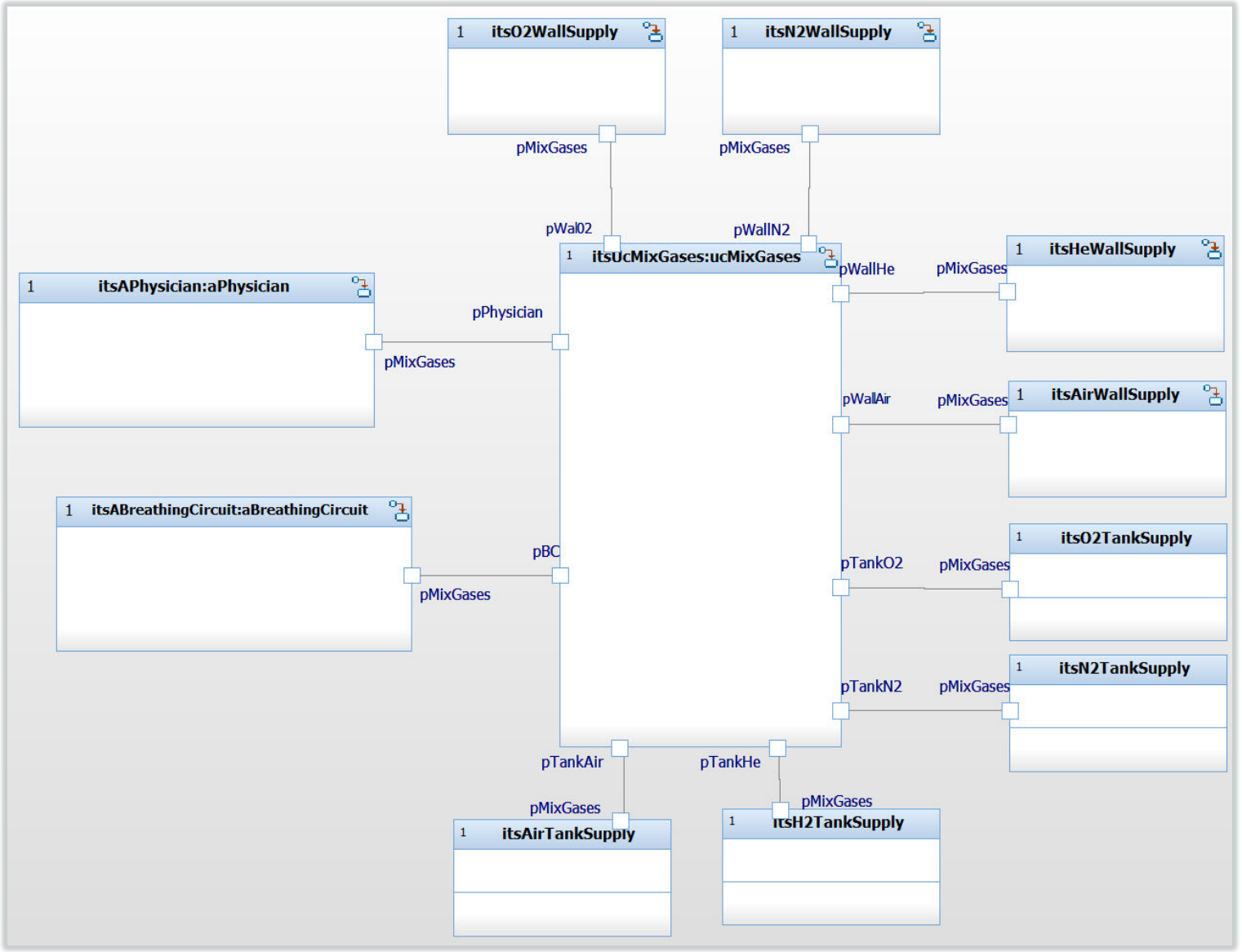
Requirements for Mix Gases Use Case

requirement_0	The system shall mix gases O2, N2, He, and Air from the input gas manifold
requirement_1	The system shalldelivery O2 and Air concentration of between 0 and 100.
requirement_2	The system shall support deliveryt of He, N2 to between 0 and 79% concentration.
requirement_3	The System shall not deliver a gas flow that has less than 21% O2.
requirement_4	The system shall not deliver a flow rate of less than 200 ml/min.
requirement_5	The system shall deliver a tidal volume of between 50 ml and 1500 ml..
requirement_6	The system shall deliver a breath with an I:E ratio of between 5:1 and 1:5 (default shall be 1:1).
requirement_7	The system shall default to gas concentation mode, in which the flow settins of each gas are set and the total flow is set independently.
requirement_8	The system shall support a balance gas mode in which the O2 supply shall deliver 21% or more of the gas concentraion and the rest will be supplied by the
requirement_9	The system shall alert if they command an O2 concentration, flow, I;E, tidal volume, or respiration rate out of range.



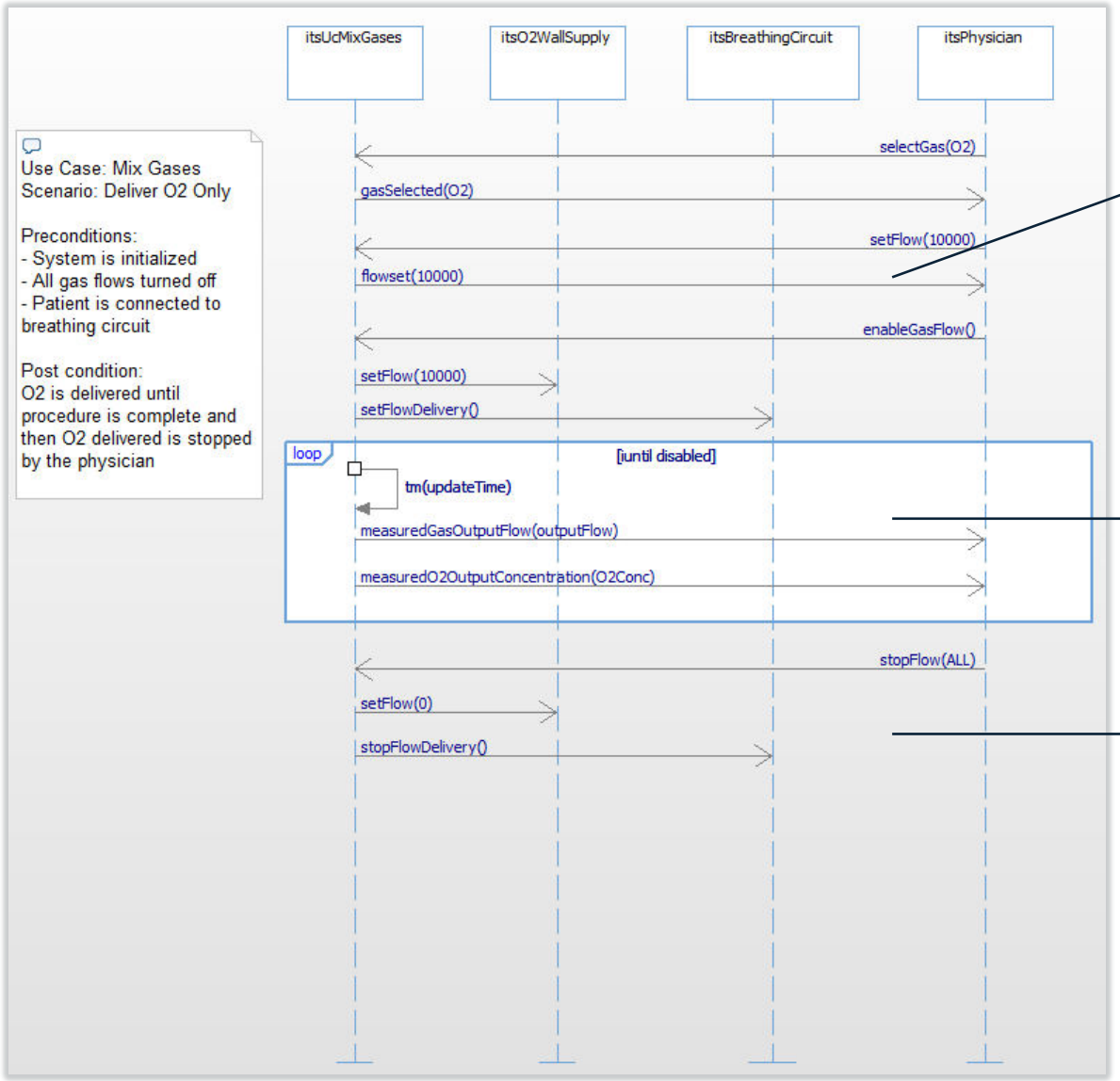
A Priori Systems
Real-Time Agile Systems and Software

Create Analysis Context (block diagram)



A Priori Systems
Real-Time Agile Systems and Software

Create Scenarios



Discovered Requirements

- The user shall select the desired gas.
- The system shall indicate to the user the currently selected gas.
- Once selected, the user shall be able to set a valid flow rate of the gas.
- The user shall set the desired flow with user action.
- The system shall acknowledge with the set value when the flow is set.

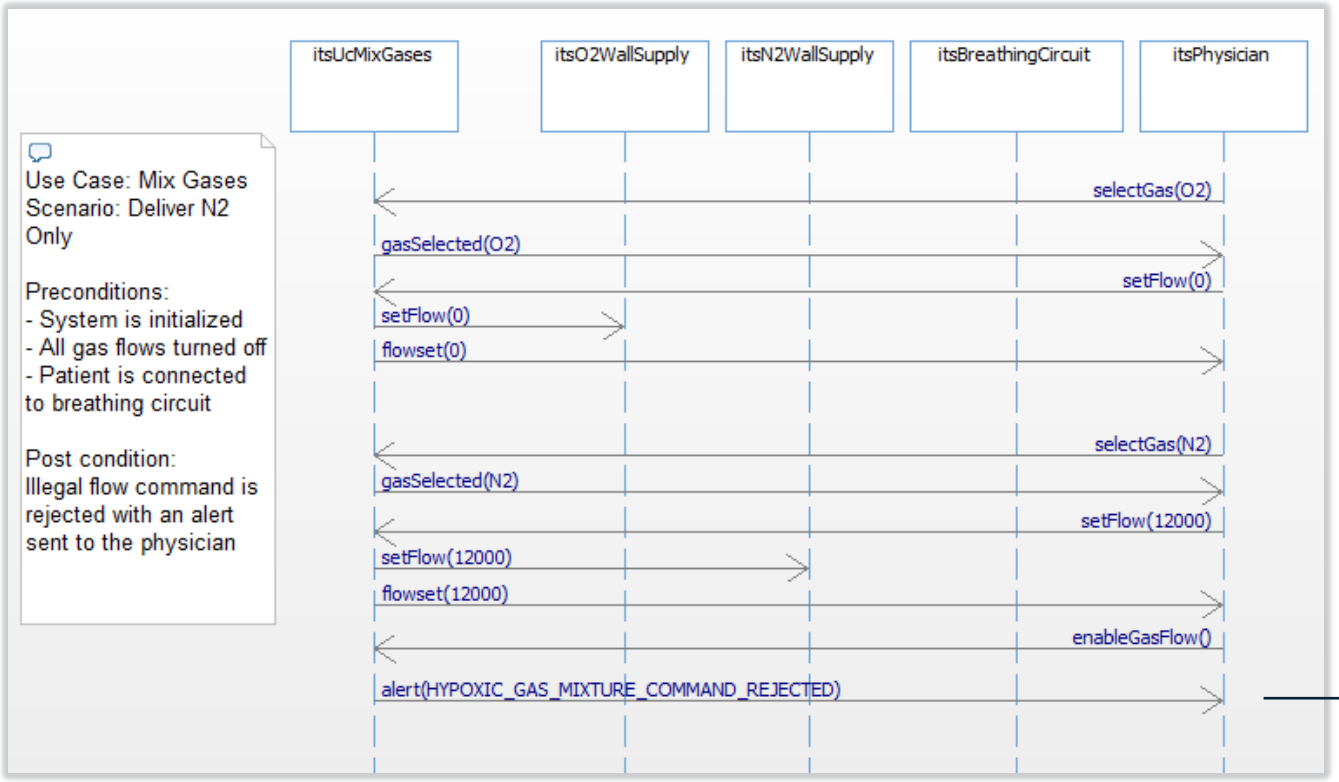
- The system shall report the measured total flow and measured oxygen concentration every 1.0s ± 0.25s

The use shall be able to command flow to stop.

- The system shall acknowledge the user command to stop flow.

A Priori Systems
Real-Time Agile Systems and Software

Create Scenarios

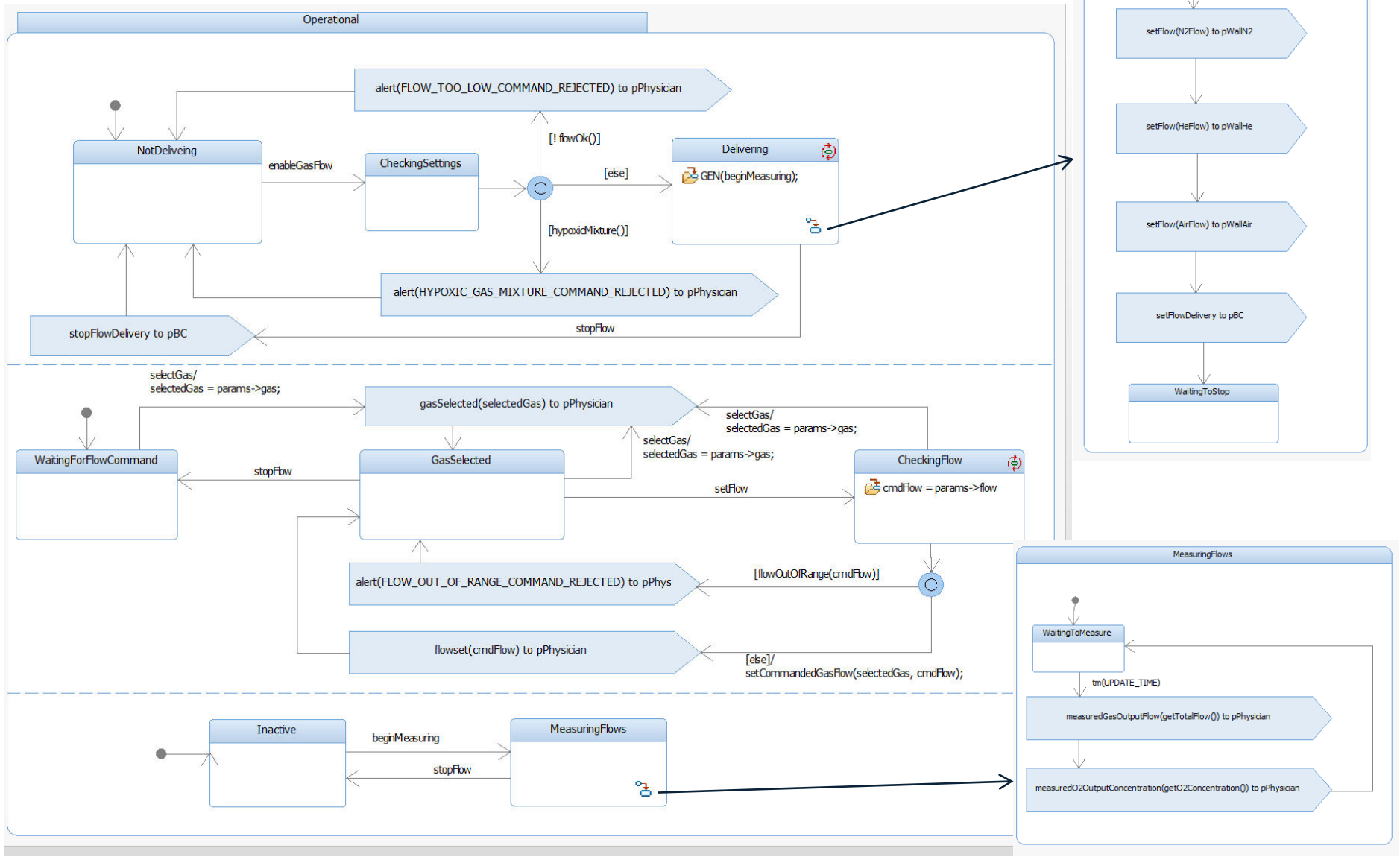


Discovered Requirements

- The system shall alert the user if they command a hypoxic gas mixture and reject the command.

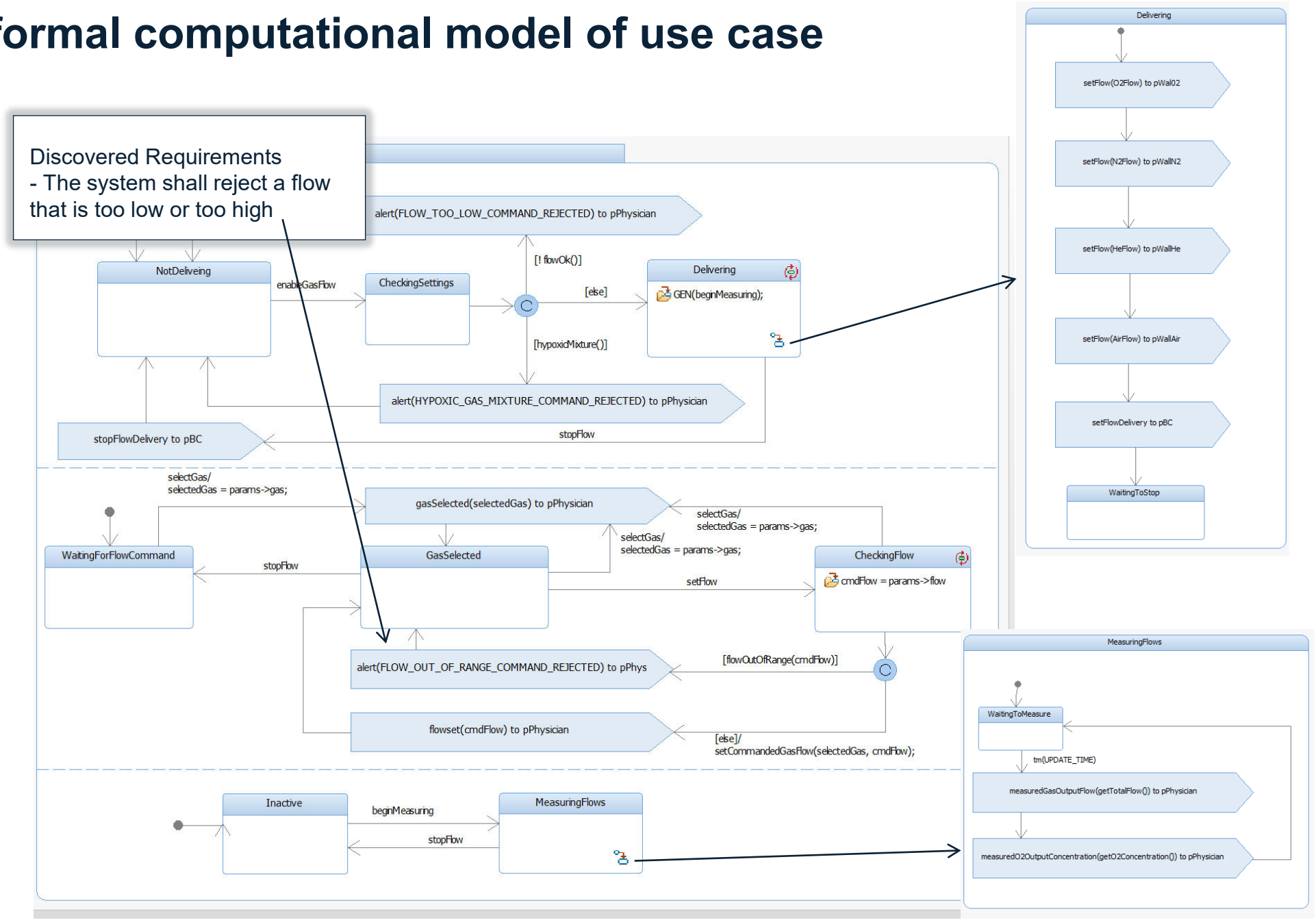
A Priori Systems
Real-Time Agile Systems and Software

Build up formal computational model of use case



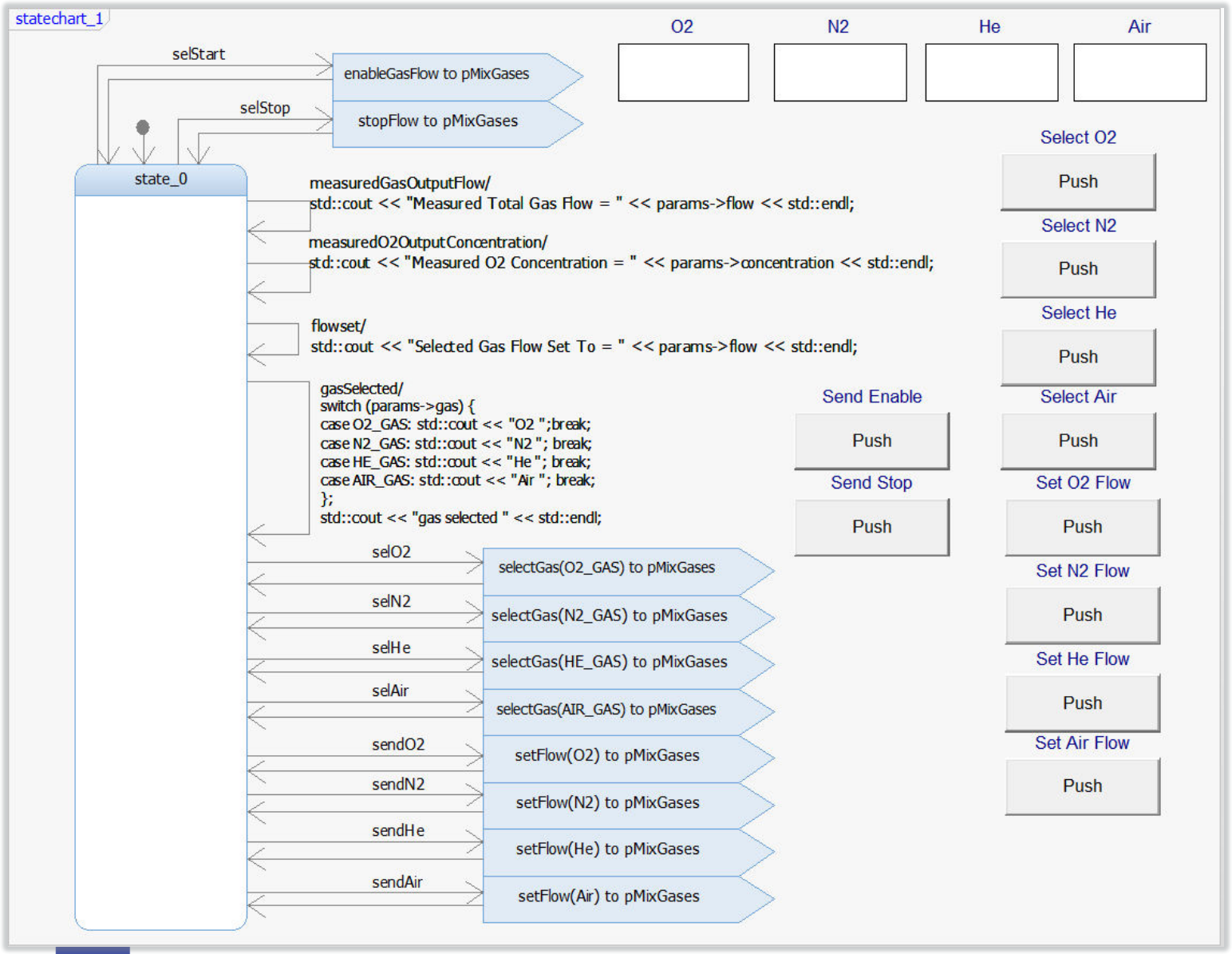
A Priori Systems
Real-Time Agile Systems and Software

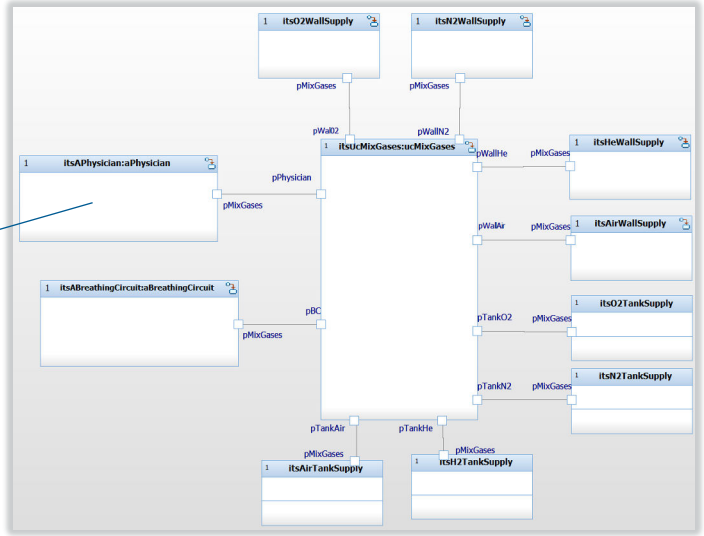
Build up formal computational model of use case



A Priori Systems
Real-Time Agile Systems and Software

Instrument the context for execution

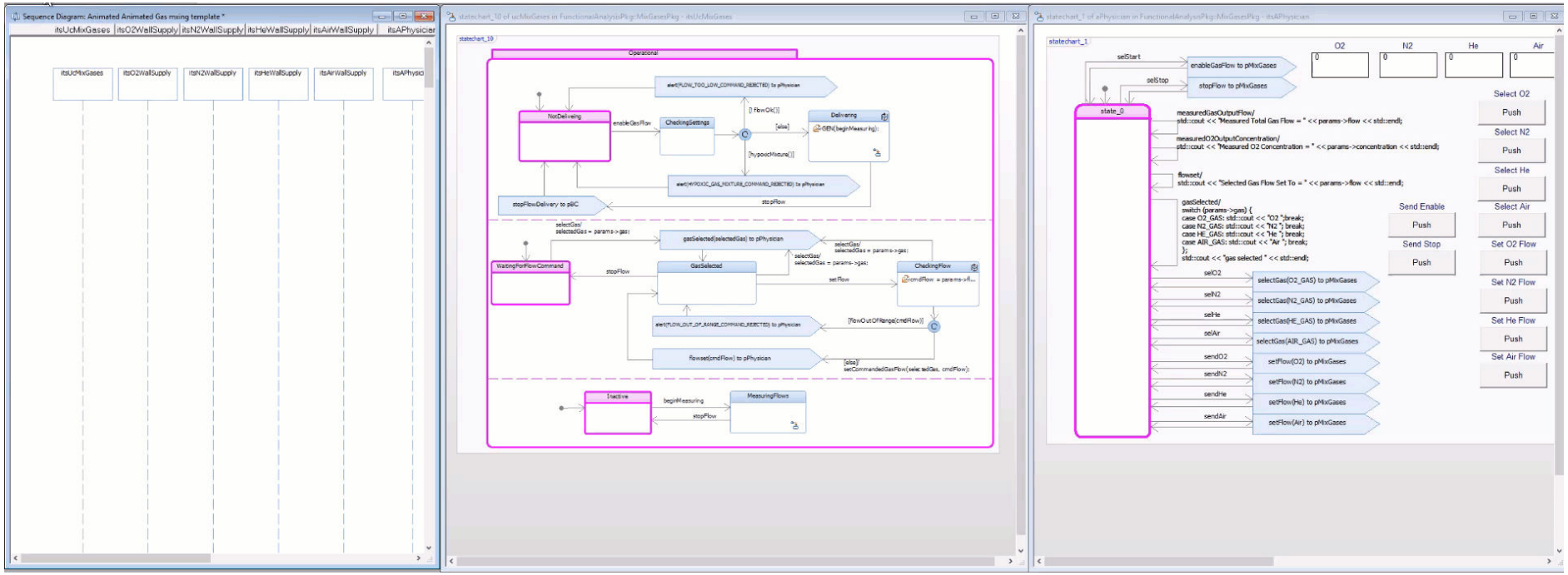




Physician Actor

A Priori Systems
Real-Time Agile Systems and Software

Running the model



Physician actions

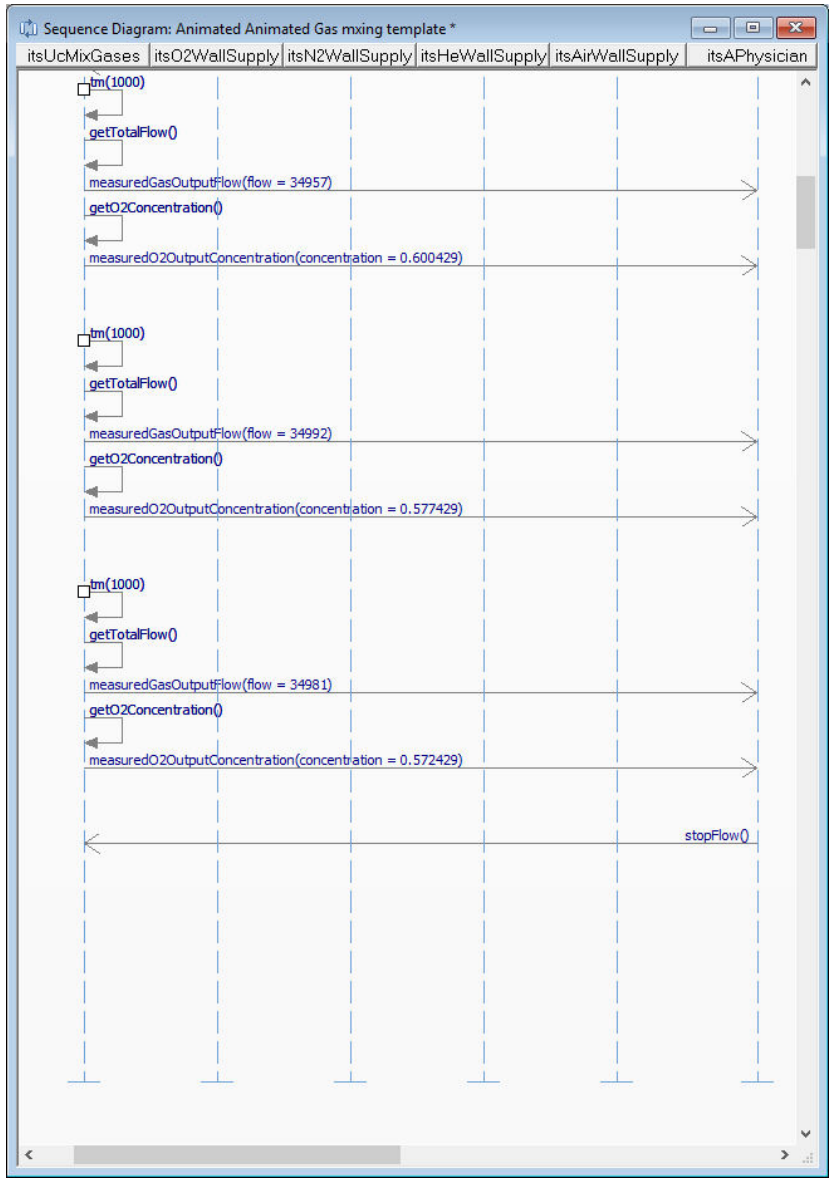
1. Select the O2 gas
2. Set the flow to 20000 ml/min
3. Push the Set O2 flow button
4. Select the N2 gas
5. Set the flow to 10,000 ml/min
6. Push the Set N2 Flow button

Physician actions

7. Select the He gas
8. Set the flow to 5000 ml/min
9. Push the Set H2 Flow button
10. Enable mixing

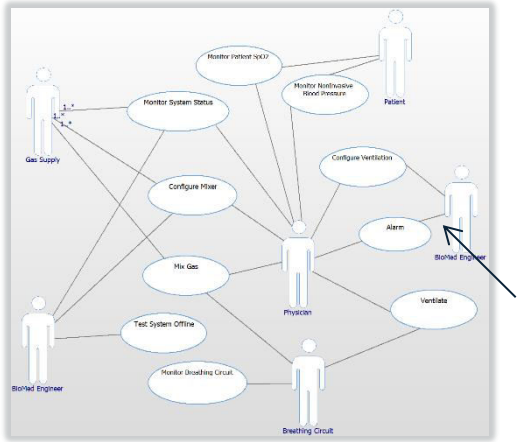
System determines mixture is not hypoxic and with valid ranges and so starts delivering.

Output (Animated Sequence Diagram)



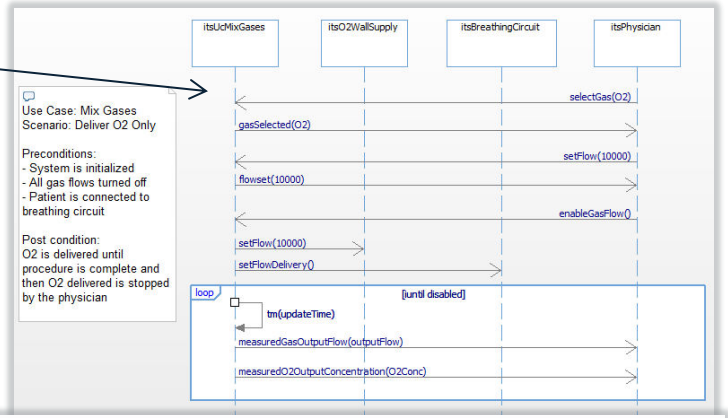
A Priori Systems
Real-Time Agile Systems and Software

Logical Interfaces are a natural outcome of use case analysis



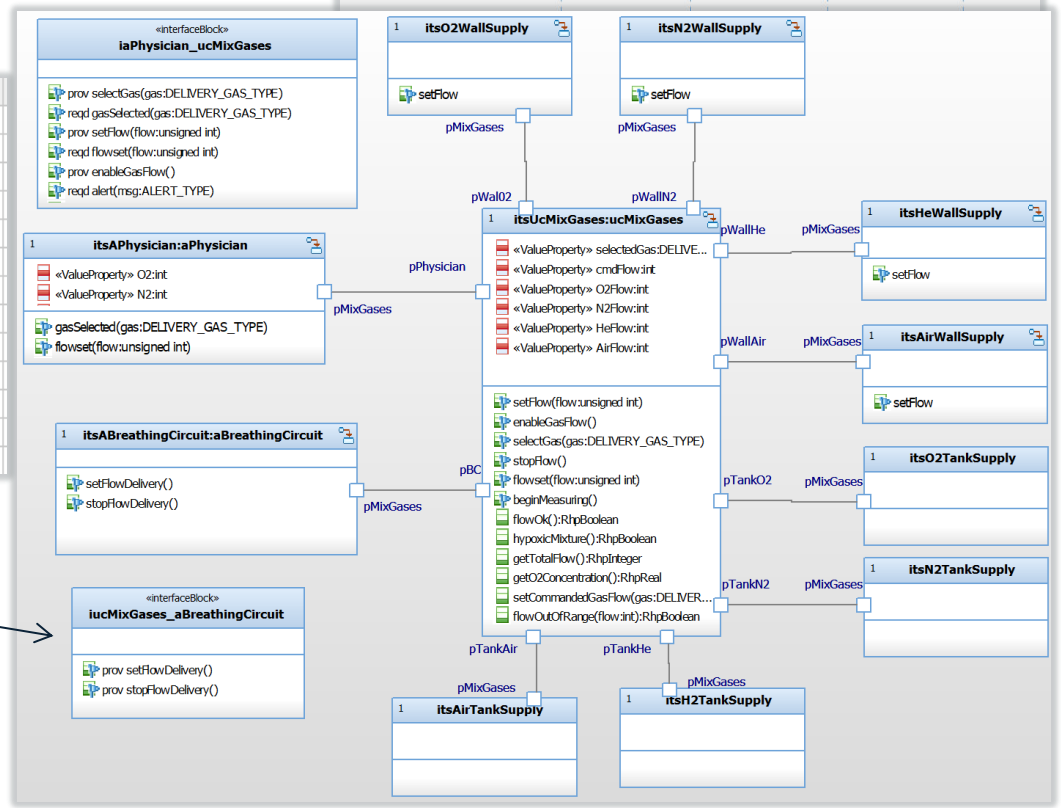
Use case-actor associations indicate the existence of interfaces

Interfaces can be detailed with behavioral views



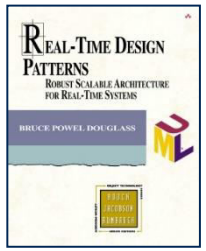
InterfacesPkg			
	iaPhysician_ucMixGases	alert	msg
	iaPhysician_ucMixGases	enableGasFlow	
	iaPhysician_ucMixGases	flowset	flow
	iaPhysician_ucMixGases	gasSelected	gas
	iaPhysician_ucMixGases	measuredGasOutputFlow	flow
	iaPhysician_ucMixGases	measuredO2OutputConcentration	concentration
	iaPhysician_ucMixGases	selectGas	gas
	iaPhysician_ucMixGases	setFlow	flow
	iaPhysician_ucMixGases	stopFlow	
	iucMixGases_N2WallSupply	setFlow	flow
	iucMixGases_O2WallSupply	setFlow	flow
	iucMixGases_aBreathingCircuit	setFlowDelivery	
	iucMixGases_aBreathingCircuit	stopFlowDelivery	

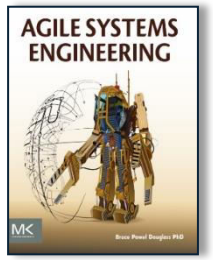
Resulting interfaces

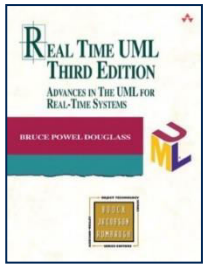


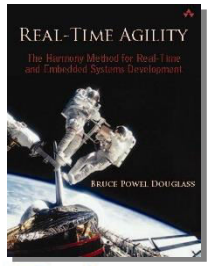
A Priori Systems
Real-Time Agile Systems and Software

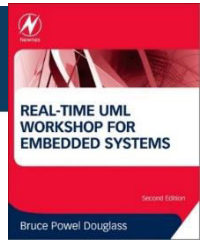
For more information

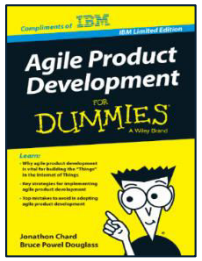


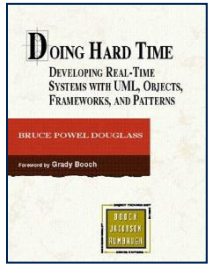


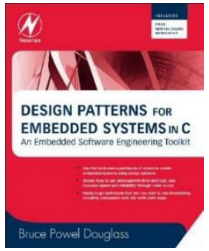


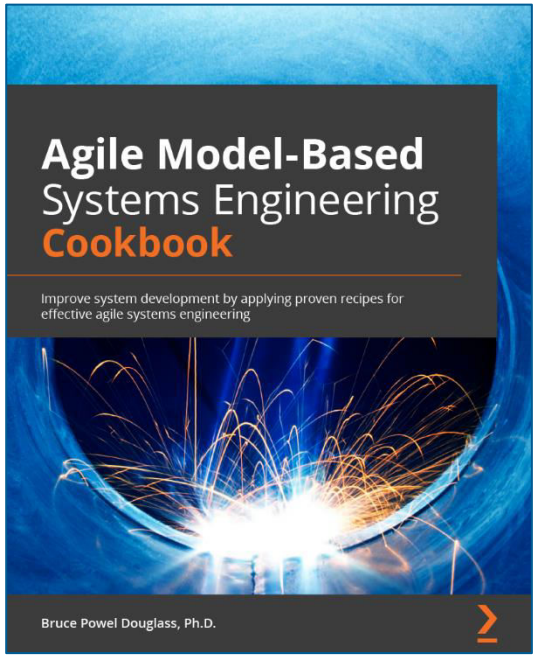












DE Platform | MITRE | DE Platform