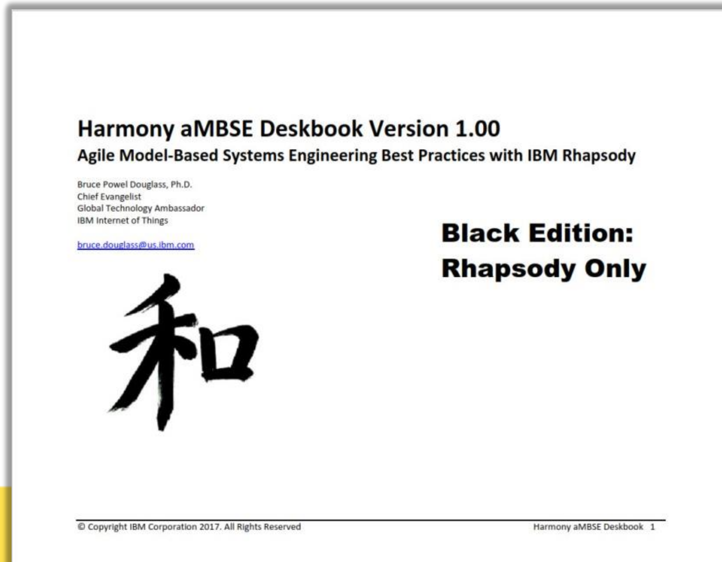
Improving System Requirements With Use Cases

Bruce Powel Douglass, Ph.D.
Chief Evangelist,
IBM Internet of Things (IoT)
Twitter: @IronmanBruce
www.bruce-douglass.com



Harmony aMBSE Deskbook Version 1.00

Agile Model-Based Systems Engineering Best Practices with IBM Rhapsody

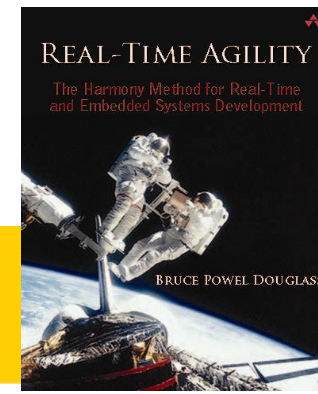
Bruce Powel Douglass, Ph.D.
Chief Evangelist
Global Technology Ambassador
IBM Internet of Things

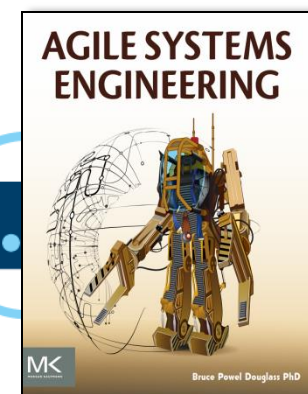
bruce.douglass@us.ibm.com

Black Edition: Rhapsody Only

和

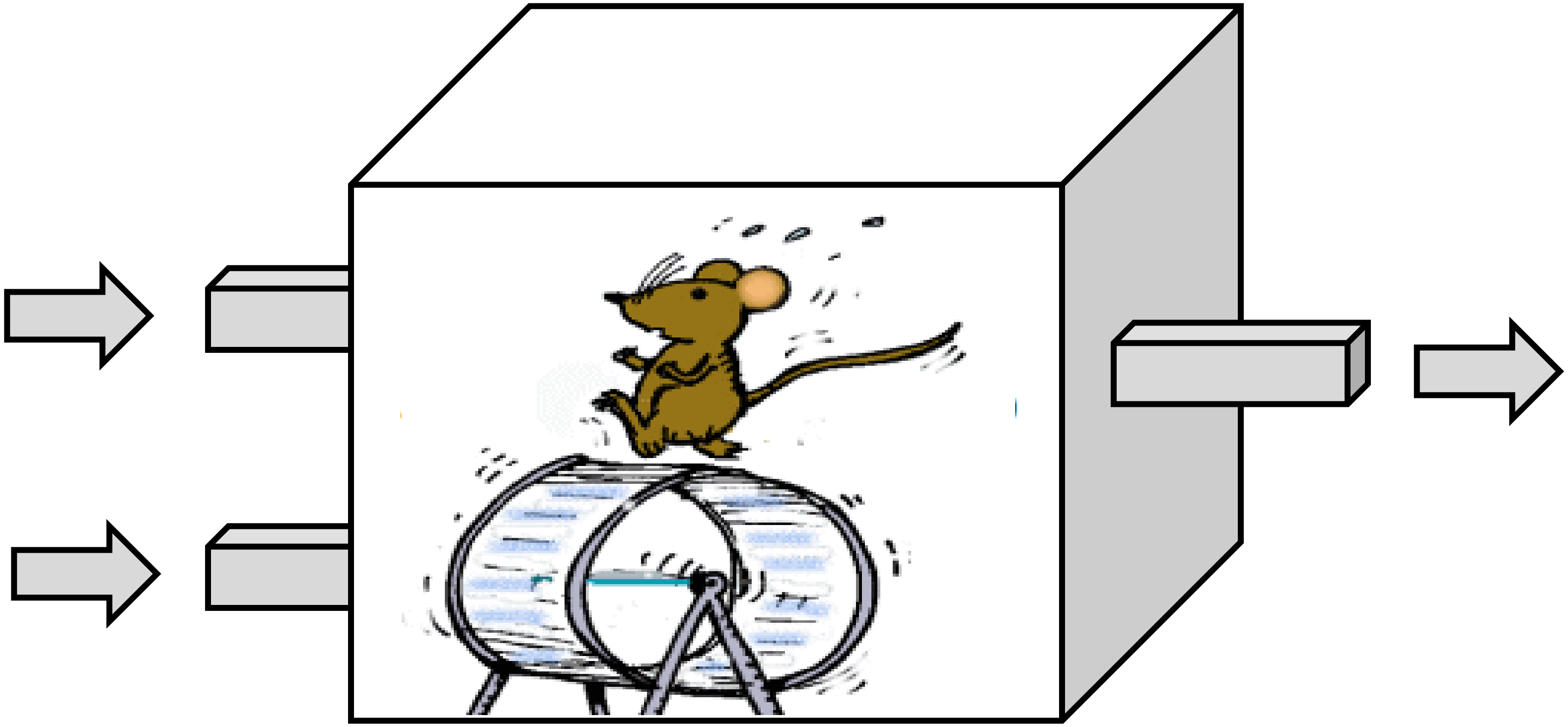
© Copyright IBM Corporation 2017. All Rights Reserved   Harmony aMBSE Deskbook 1





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.



Characteristics of Good Requirements

- Achievable
- Verifiable
- Unambiguous and Consistent
- Complete and Correct
- Identifies the Need (What) – Not the Solution (How)
- Appropriate for Level of Design
- Ranked / Prioritized
- Traceable

Requirement Verb Usage

- **SHALL** indicates a normative requirement that will be verified
  - Example: The system **shall** move the aircraft control surfaces within a range of 0 and 30 degrees in compliance to a pilot command.
- **SHOULD** indicates goals and non-mandatory, but recommended, provisions
  - Example: The flight data **should** be easily available to the pilot.
- **WILL** indicates a statement of fact which will not be verified, such as a factual statement about another system
  - Example: The system connects to a hose which **will** provide water.
- **MAY** indicates an optional provision without specifying a recommendation
  - Example: The system may use a standard display or a custom display

Other requirements recommendations

- Voice
  - Use active, rather than passive voice
    - (PASSIVE) Data shall be acquired by the system at a rate of 100 samples/sec
    - (ACTIVE) The system shall acquire data at a rate of 100 samples/sec.
- *Every functional requirement should have one or more quality of service qualifier requirements*
  - Example: **The system shall move the robot arm in compliance with user command.**
    - **Range**: The system shall move the robot are in compliance with user command in the range of – 30 degrees and + 45 degrees.
    - **Accuracy**: The system shall move the robot arm in compliance with the user command with an accuracy $\pm$ 0.1 degrees (accuracy is precision of the output)
    - **Fidelity**: The user shall be able to specify the robot arm position to within 0.05 degrees (fidelity is the precision of the input)
    - **Responsiveness**: The system shall move the robot arm to the specified position within 300ms.
    - **Exception**: The system shall reject movement commands that are outside of the allowable range and raise a Caution Alert.
    - **Exception**: The system shall raise a Warning Alert if the required accuracy or timing of the robot arm movement is not compliant upon completion of movrement.

Types of Requirements - examples

- Stakeholder requirements
  - The aircraft should be steady during flight
- Functional requirements
  - The system shall maintain airframe stability in all three rotational axes in the presence of steady winds.
- Functional quality of service (QoS)
  - The system shall maintain airframe stability in all three rotational axes within 0.5 degrees of arc in the presence of steady winds and within 2 degrees of arc in the presents of gusts up to 40 kph.
- System parametrics
  - The system shall weigh no more than 30 kg when fully loaded with hydraulic fluid.
  - The system shall have a maximum current draw of 0.5 amps at 240V.
  - The airframe shall be painted green and prominently display the corporate logo.
- Project QoS
  - The system design shall use existing hydraulic components from the FlightMagic system.
  - The system shall support the MagicCarpet series of vehicles and fit within the chassis housing attitude control component housing.
- Certification Requirements
  - The system shall be certifiable under DO-178B.

So now we’re done, right?

As tradition requires, I now do the engineer’s victory dance …

Why aren't we done?

Requirements are incomplete:
- all functionality?
- "edge cases"?
- built in test?
- safety?
- performance?
- security?

Requirements are complex:
- all combination of inputs values, sequences and timing?
- what happens for invalid inputs or conditions?

Customers often don't know what they want

Requirements may not meet the actual need

Requirements are volatile

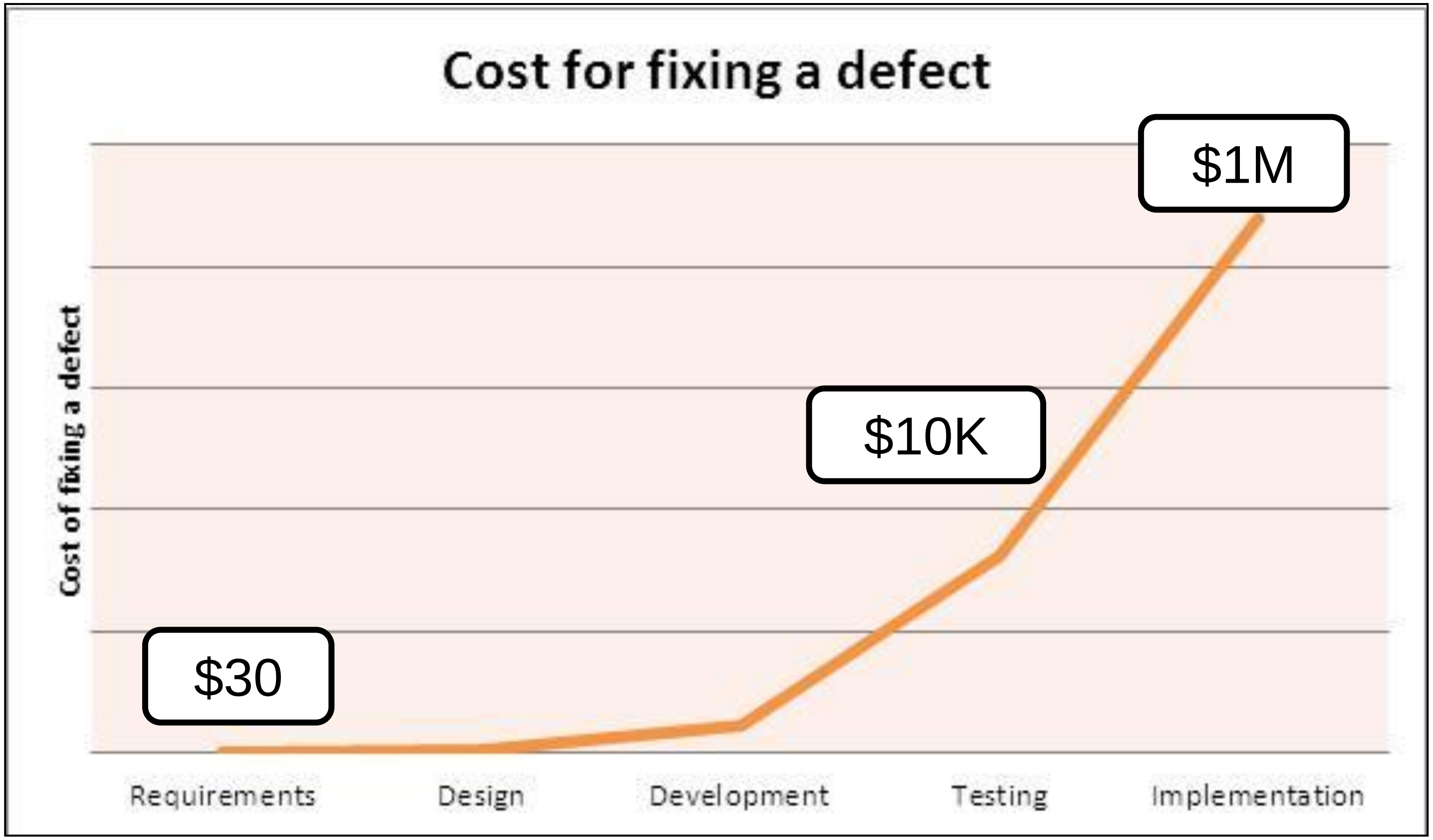
Requirements may be unachievable

Requirements may be untestable

Requirements may be ambiguous

Requirements may be inconsistent

Poor requirements have a huge impact



The Three Ways of Verifying "Goodness"

1 **Review / Inspection**

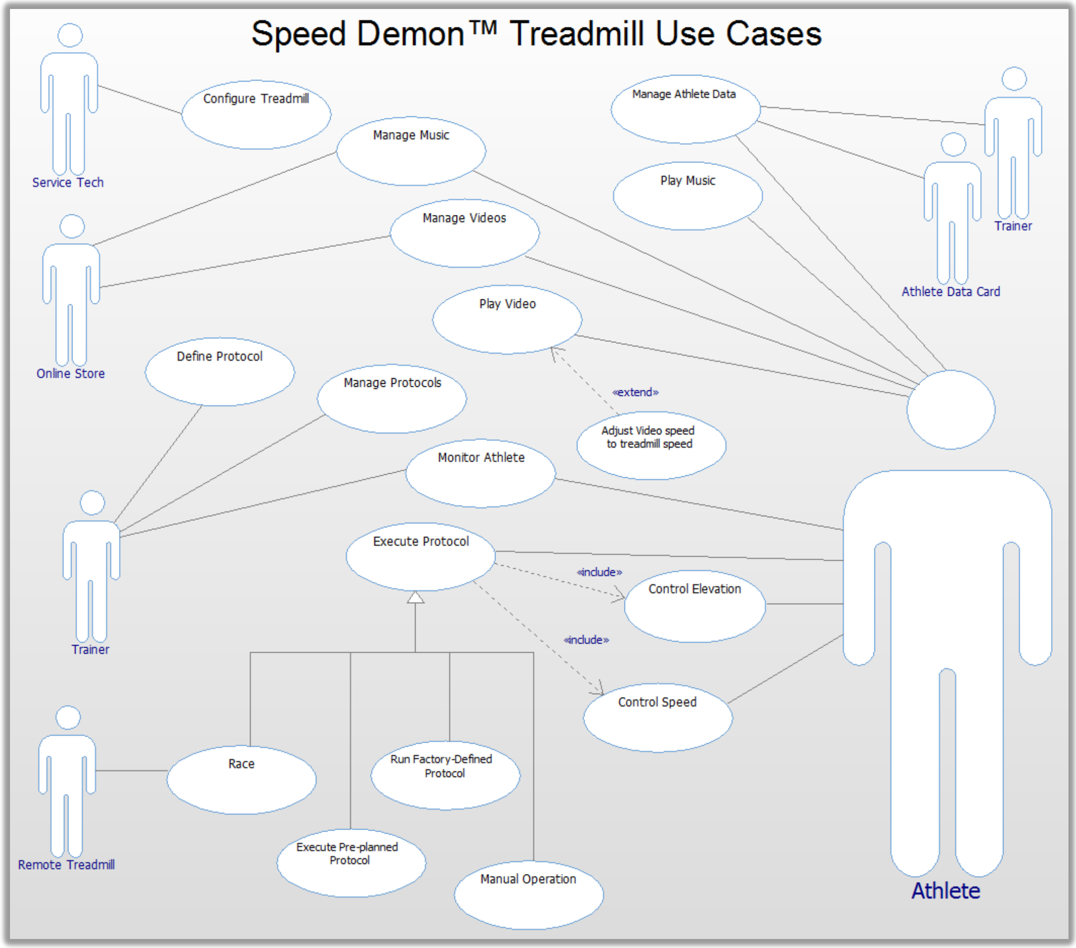
2 **Test**

3 **Formal methods**

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

What's A Use Case?

- is an operational capability of a system
  - why the user interacts with the system
- is an organizational unit for requirements
  - Normally 10-100 textual requirements
  - Normally a few to a few dozen use cases per system
- may group *stakeholder, system*, *subsystems* or *software* requirements
- returns a result visible to one or more actors
- does not reveal or imply internal structure of the system
- is independent of other use cases and may be concurrent with them
- May be constrained with various QoS parameters



Use cases group requirements into coherent sets

- There are a number of ways to think about what constitutes a use case
  - It is a named operational capability of a system
  - It is a collection of related specific usage scenarios of a system
  - It is a collection of requirements around a system usage
  - It is a coherent set of interactions of the system with a set of external elements (actors)
- Properties of good use cases
  - Coherence
  - Independence (from other use cases in terms of requirements (not necessarily in terms of implementation))
    - In the great majority of cases, a requirement is allocated to a single use case
  - Coverage – all functions and QoS requirements are allocated
  - Size
    - 10-100 requirements
    - 3-25 scenarios
  - Exceptions for large systems
    - Abstract use cases can be used to organize the use cases for large systems using
      - Generalization
      - Inclusion
      - Extension

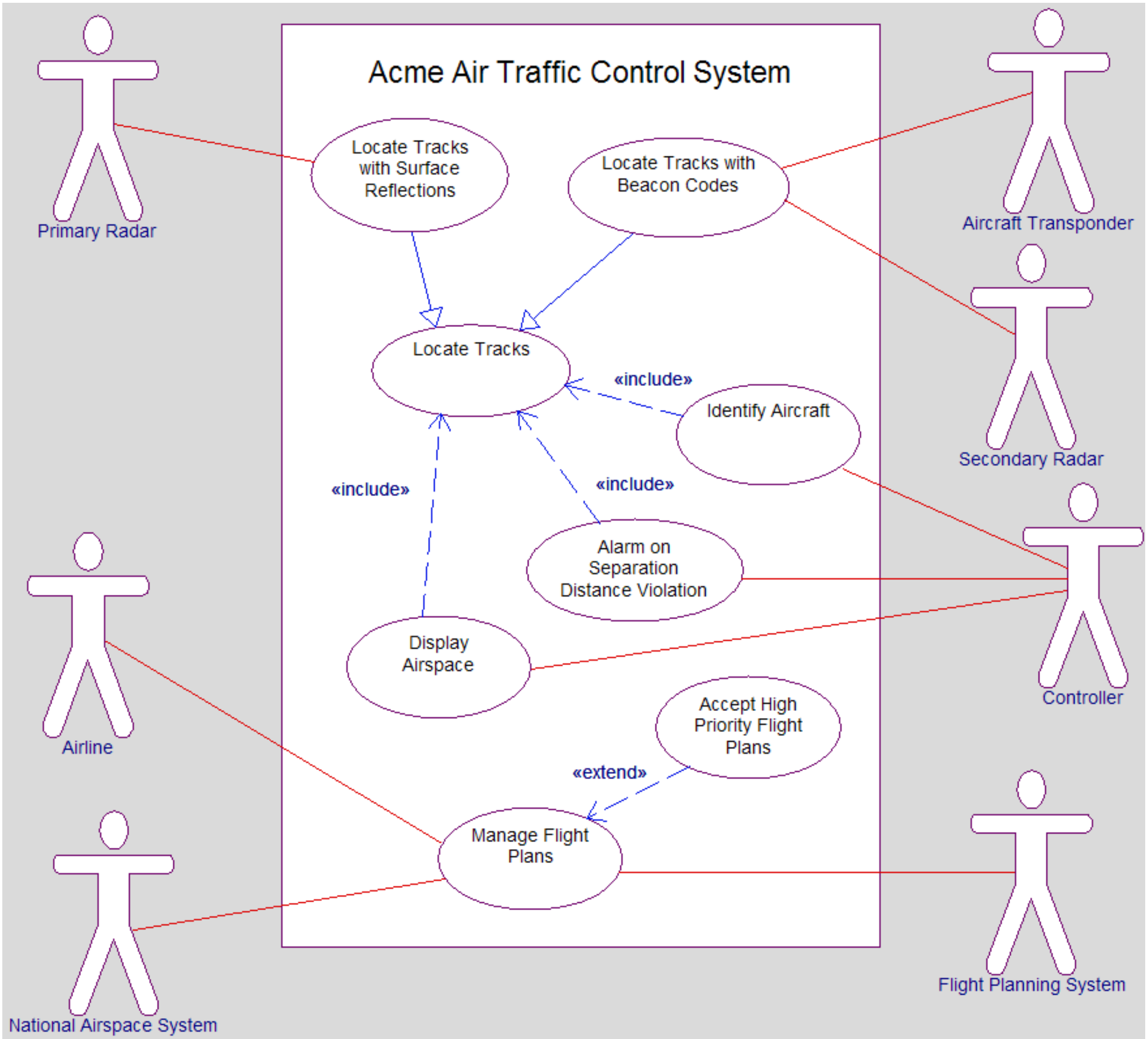
Categories of use cases uses

- **Stakeholder use cases** group stakeholder requirements
- **System use cases** group system requirements
- Note: there is normally a 1:1 relationship between stakeholder and system use cases, and they often have similar or identical names and have «trace» relations between them
- **Subsystem use cases** group subsystem requirements
  - Note: Subsystem use cases are logically included by system use cases via «include»
- **Software use cases** group software requirements and are logically included by system or subsystem use cases via «include»
- **Abstract use cases** contain no requirements themselves but are used to organize the taxonomy of use cases via relations
  - Generalization
  - Inclusion
  - Extension
- **Concrete use cases** have allocated requirements and may optionally also be used to organize the taxonomy of use cases
- **Leaf use cases** do not own any use case relations

Use Case Recommendations

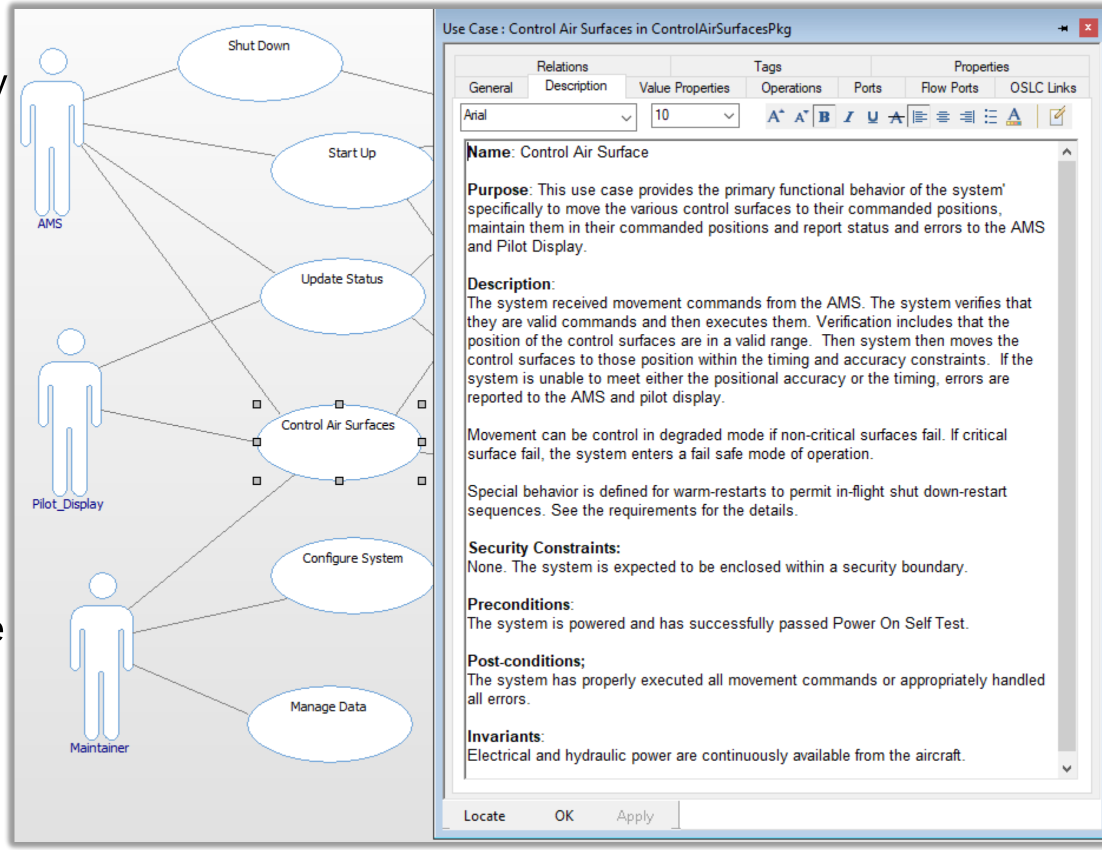
- Use short verb or verb-phrase names
  - Not nouns! “Move Control Surface” not “Surface Controller”
- Name from problem domain vocabulary
  - Not solution vocabulary! “Apply Braking” not “Apply Hydraulic Disk Pressure”
- Give each use case a short specification (more on this later)
- Aspects of use cases
  - Identify services (“system functions”) in/out
    - ex. **Heat water(set temp)**, **report water temp(measured temp)**
  - Identify data / flows in/out
    - ex. **Set temperature**, **measured temperature**, **alarm limit temperature**
  - Identify control/data/flow transformation
    - ex. **cold water in → hot water out**
  - Identify levels of fidelity (precision of the input) and accuracy (precision of the output) of the use case
    - ex. temperature set in **units of 0.5C**, accuracy managed to **0.1C**
  - Specify required performance, reliability, safety, security, etc
    - ex. **Water must be heated to set temperature within 30s**
- Actors
  - Identify their goals and objectives for the use case
  - Identify which services they need from or will provide to the system while executing the use case
  - Include data and flows in/out
  - What transformations are expected?

Use Case Syntax



For Every Use Case, a Description (Minispec) …

- Use Case Description Structure
  - Name
  - Purpose
    - Identifies the goals of the capability and its value to the stakeholders
  - Description
    - Summarizes the control and data transformations that the use case specifies
  - Preconditions
    - What is true prior to the execution of the capability?
  - Postconditions
    - What does the system guarantee to be true after the execution of the use case?
  - Invariants
    - What relevant conditions are assumed to be always true?
  - Constraints
    - Additional QoS requirements or other rules or limitations for the use case



Outcomes of Functional Analysis

- Primary
  - Demonstrably correct and complete set of textual requirements
  - Logical Interfaces between the system and the actors
- Secondary
  - Use Case Model
  - Use Case Execution Context
  - Executable Use Case Models (one form of a Digital Twin)
  - Specification Sequence Diagrams
  - Specification Activity Diagrams
  - Specification State Machine
  - Trace links
    - System Requirements → Stakeholder Requirements
    - Use Case → System Requirements
    - Use Case actions → System Requirements

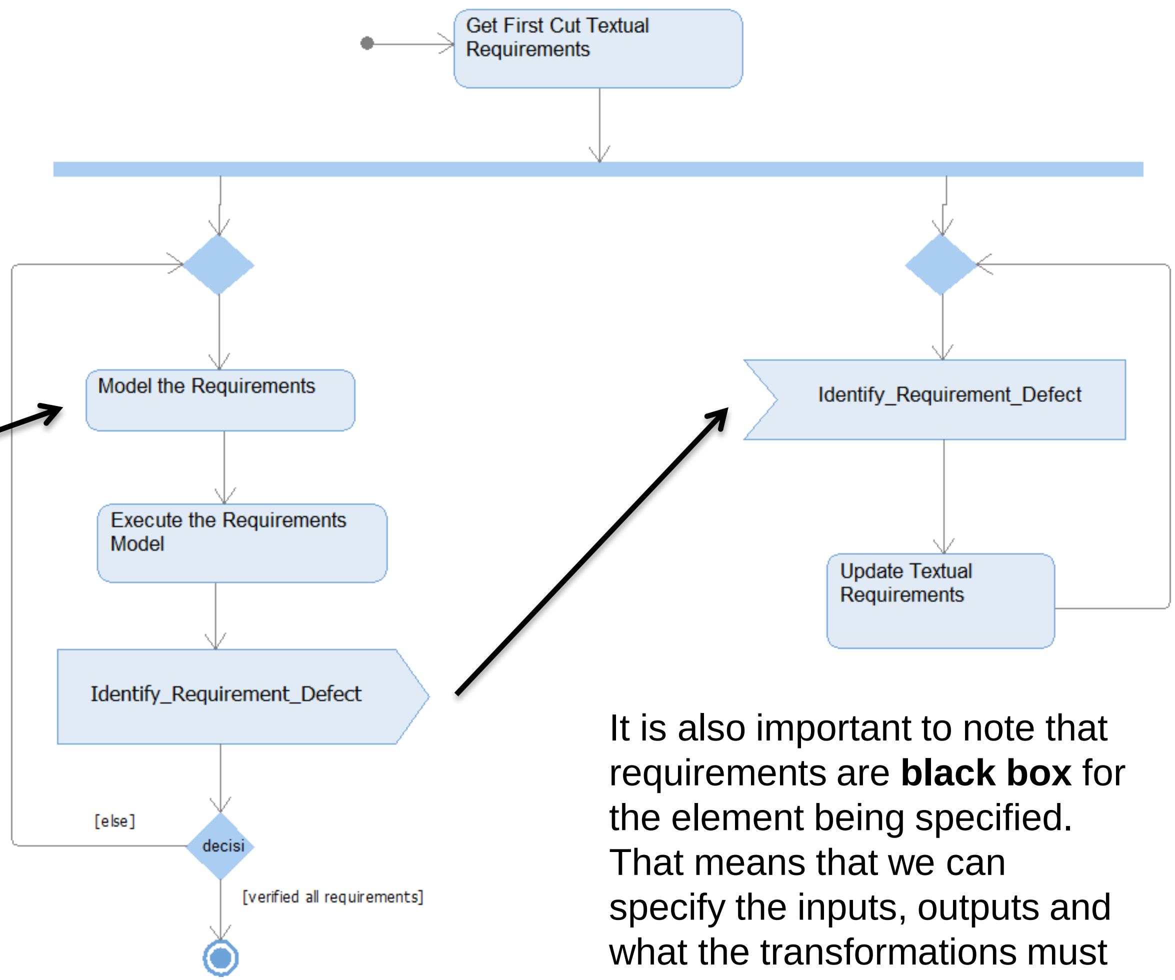
The Requirements Modeling Approach

- 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.

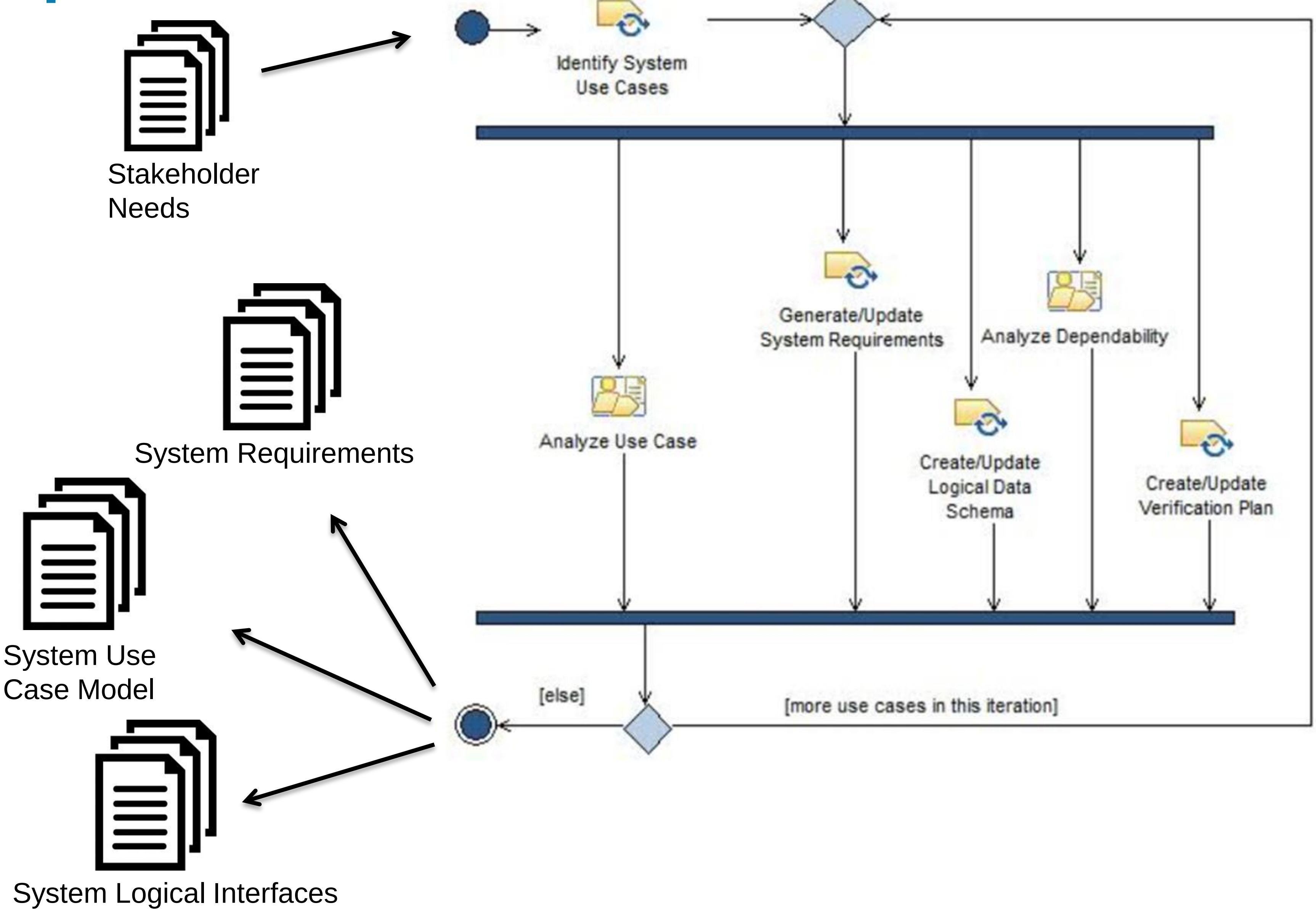
We are not modeling non-requirements or design.

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

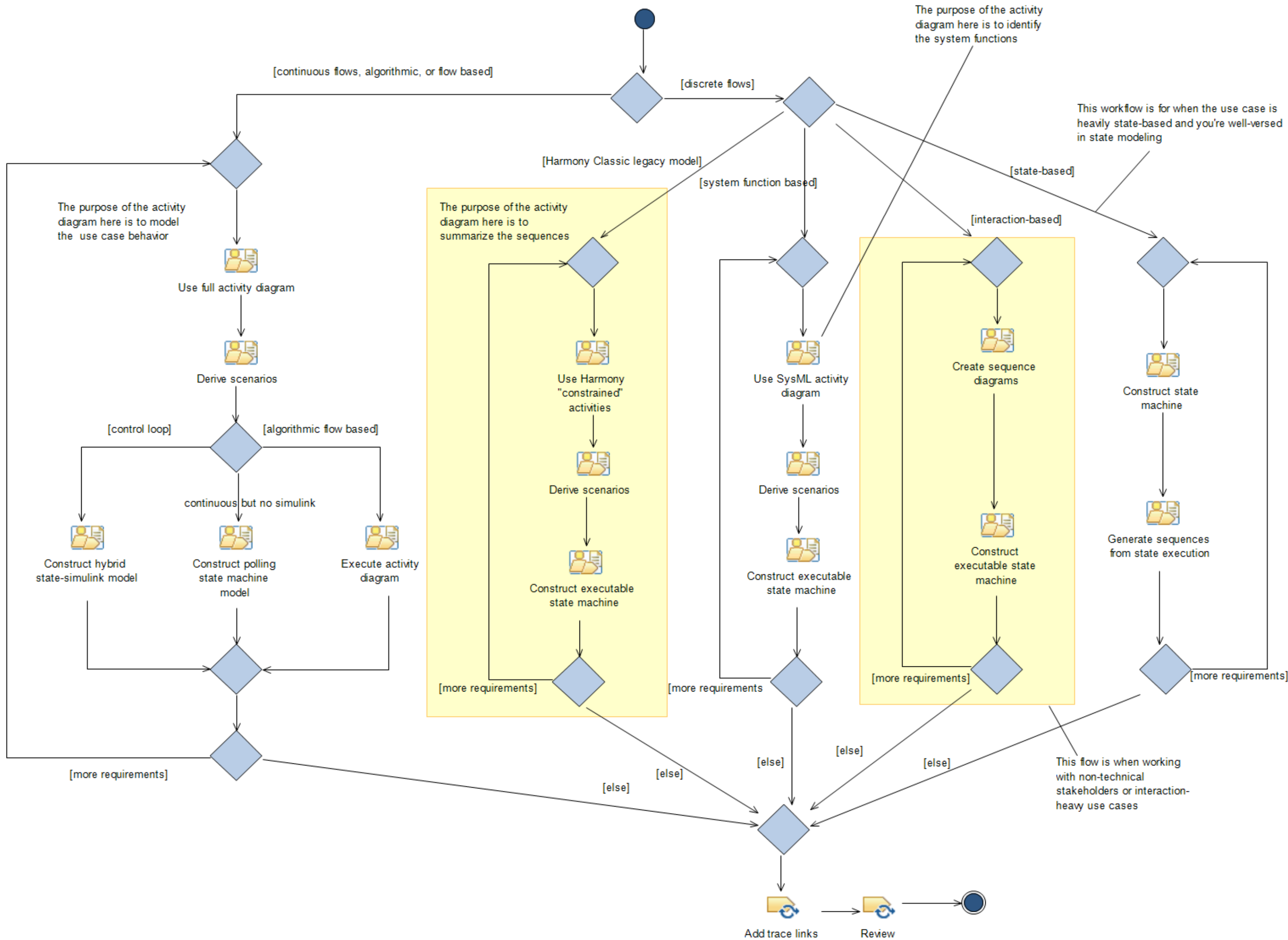


It is also important to note that requirements are **black box** for the element being specified. That means that we can specify the inputs, outputs and what the transformations must be, but not how they are performed.

Requirements Workflow



Harmony aMBSE: System Requirements Analysis Alternatives



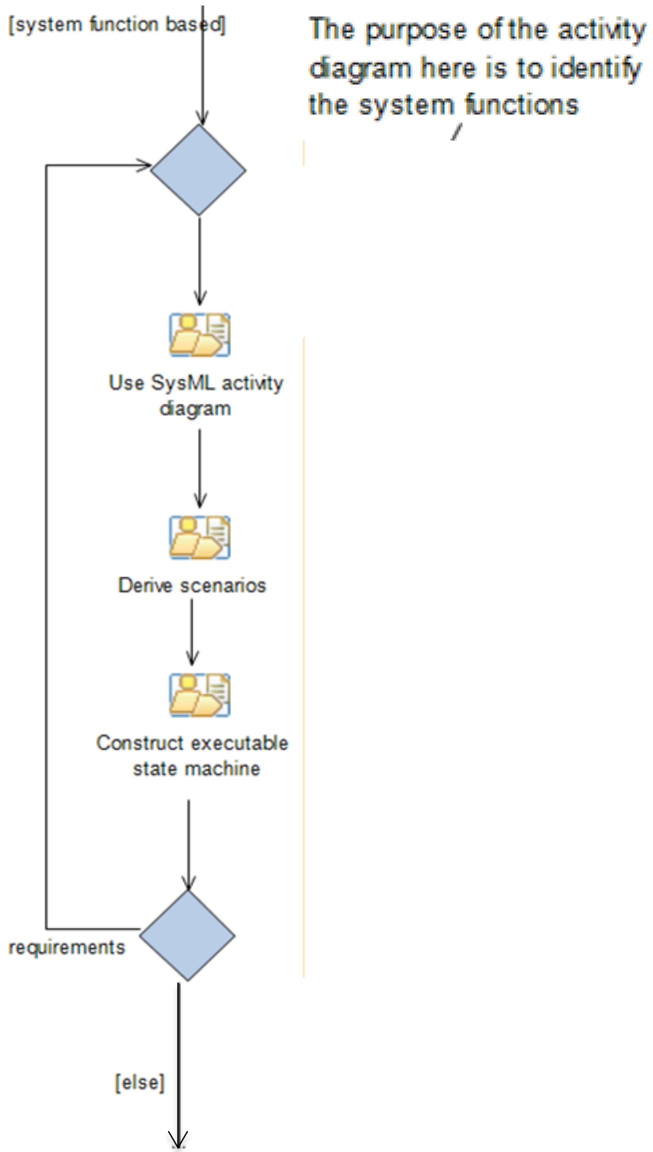
System-Function Based (most common)

Properties

- Activity diagrams model incomplete
  - 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
- Scenarios and state machines hold the “source of truth”

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



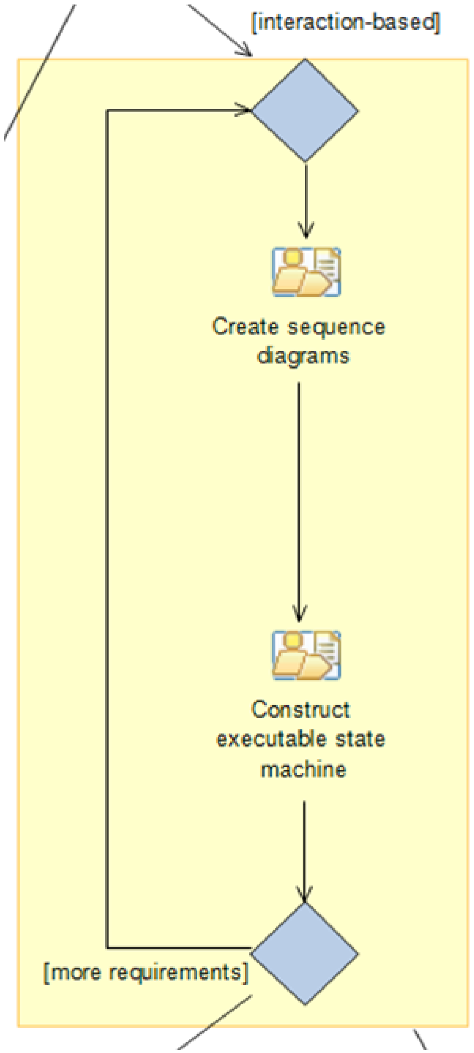
Scenario-based (aka "Interaction –Based")

Properties

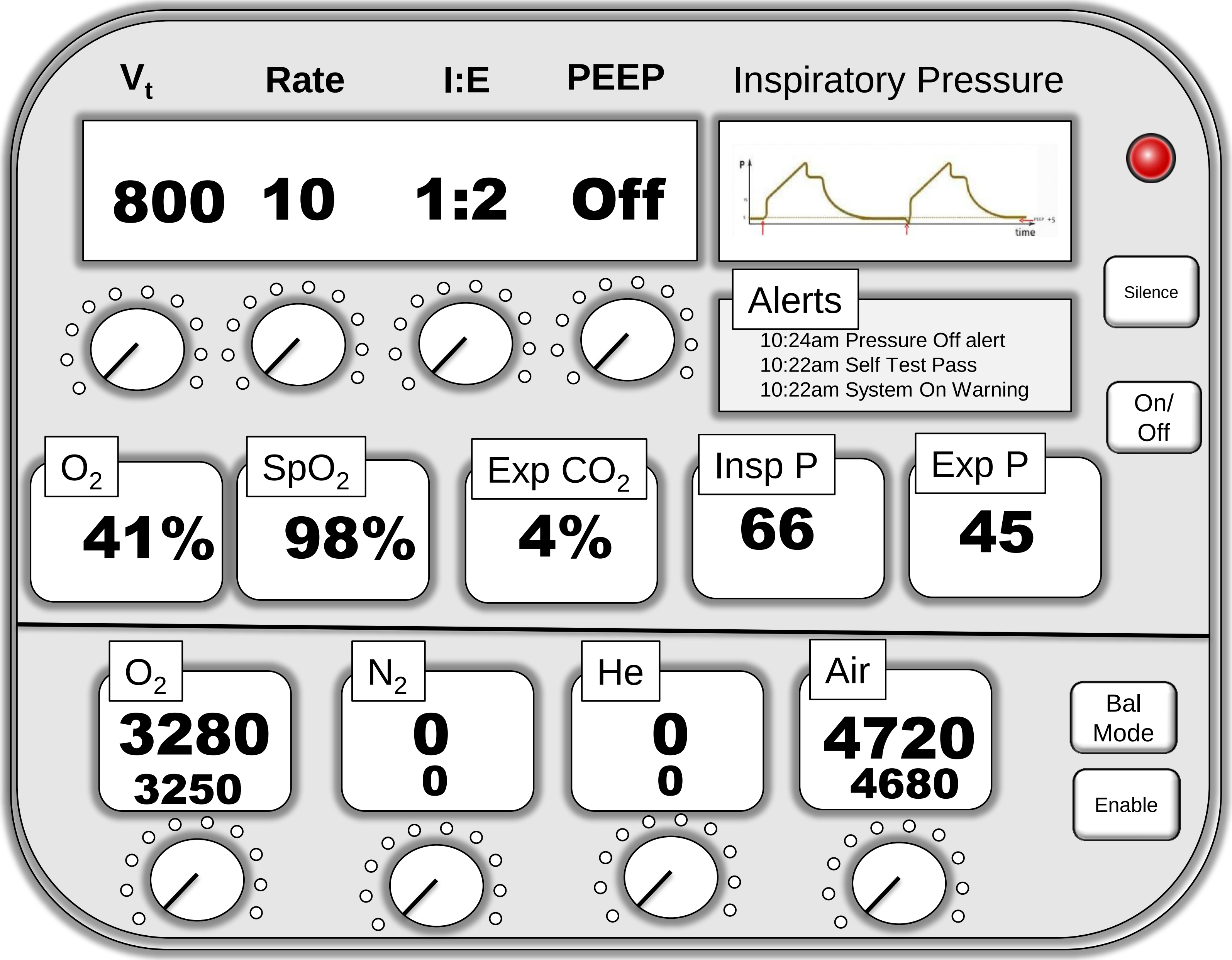
- Skips activity modeling
- Captures all requirements associated with the use case including
  - Quality of service
  - Edge/exception cases
- Scenaios are primarily used to elicit requirements
- State machine is the "source of truth"

When to use

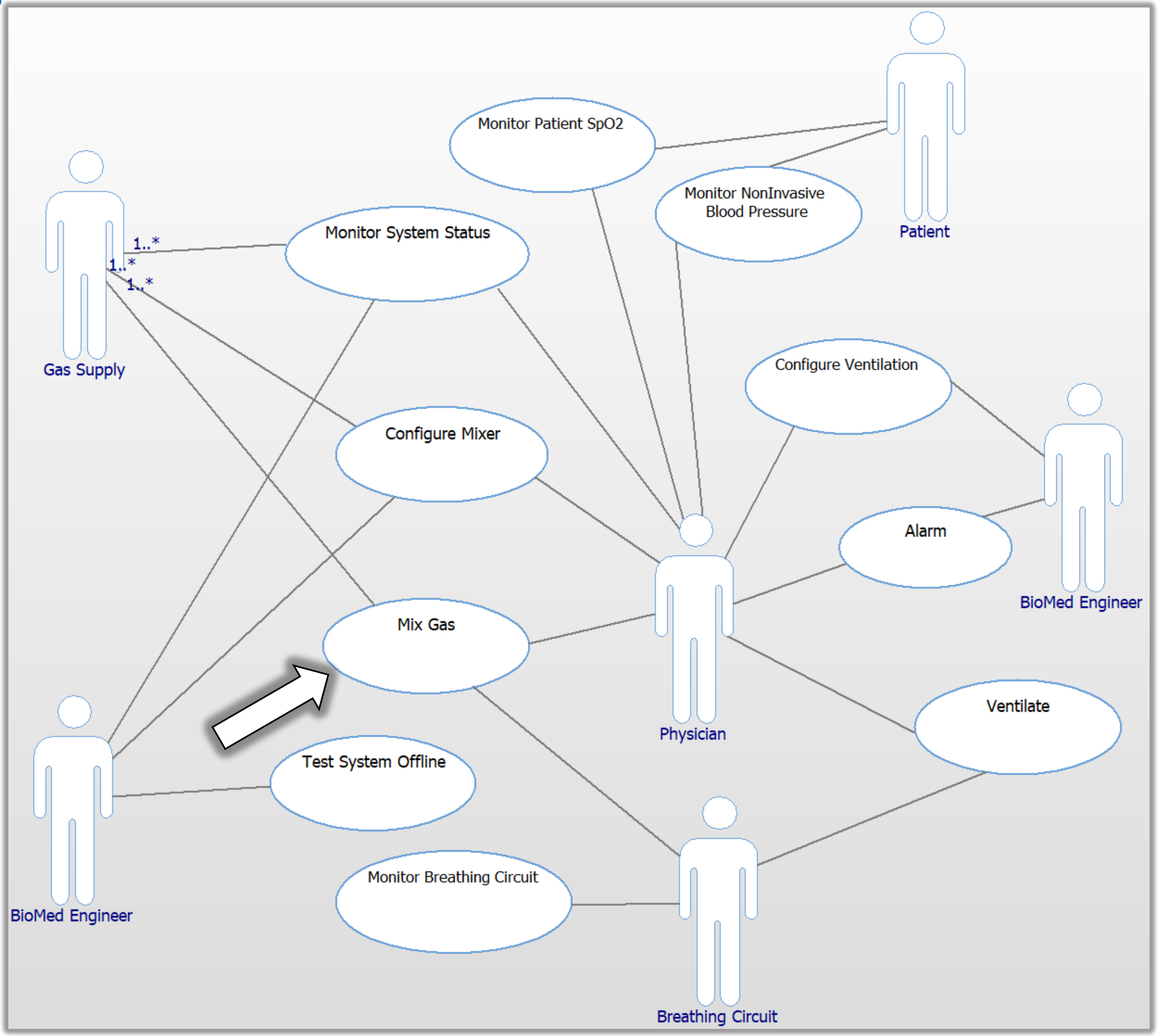
- Primary concern is the interaction between the system (running the use case) and the actors
- Non-technical stakeholders are primary contributors to system understanding



Medical Ventilator



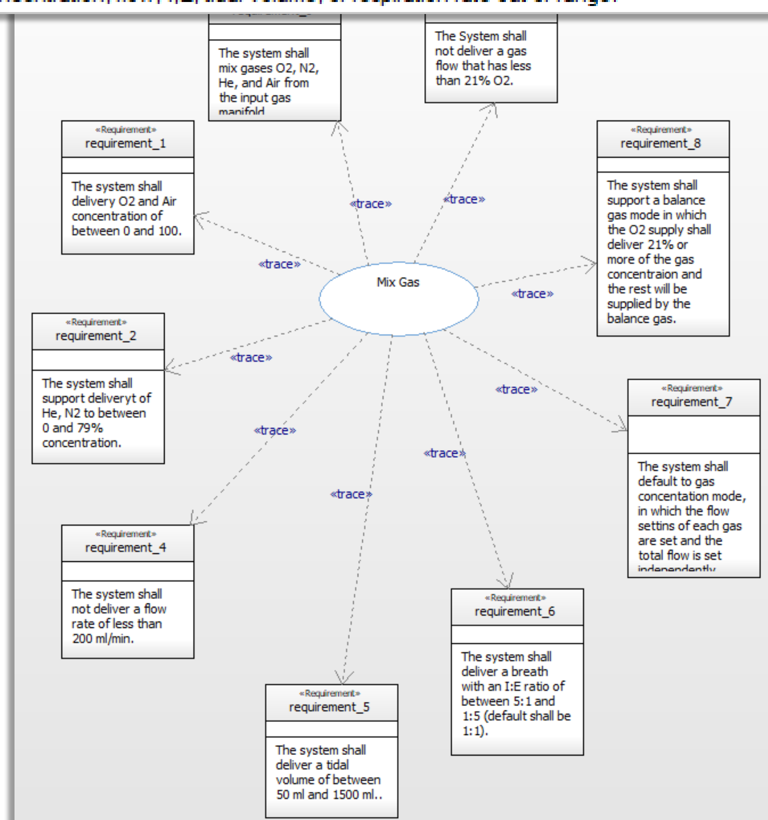
Use Cases



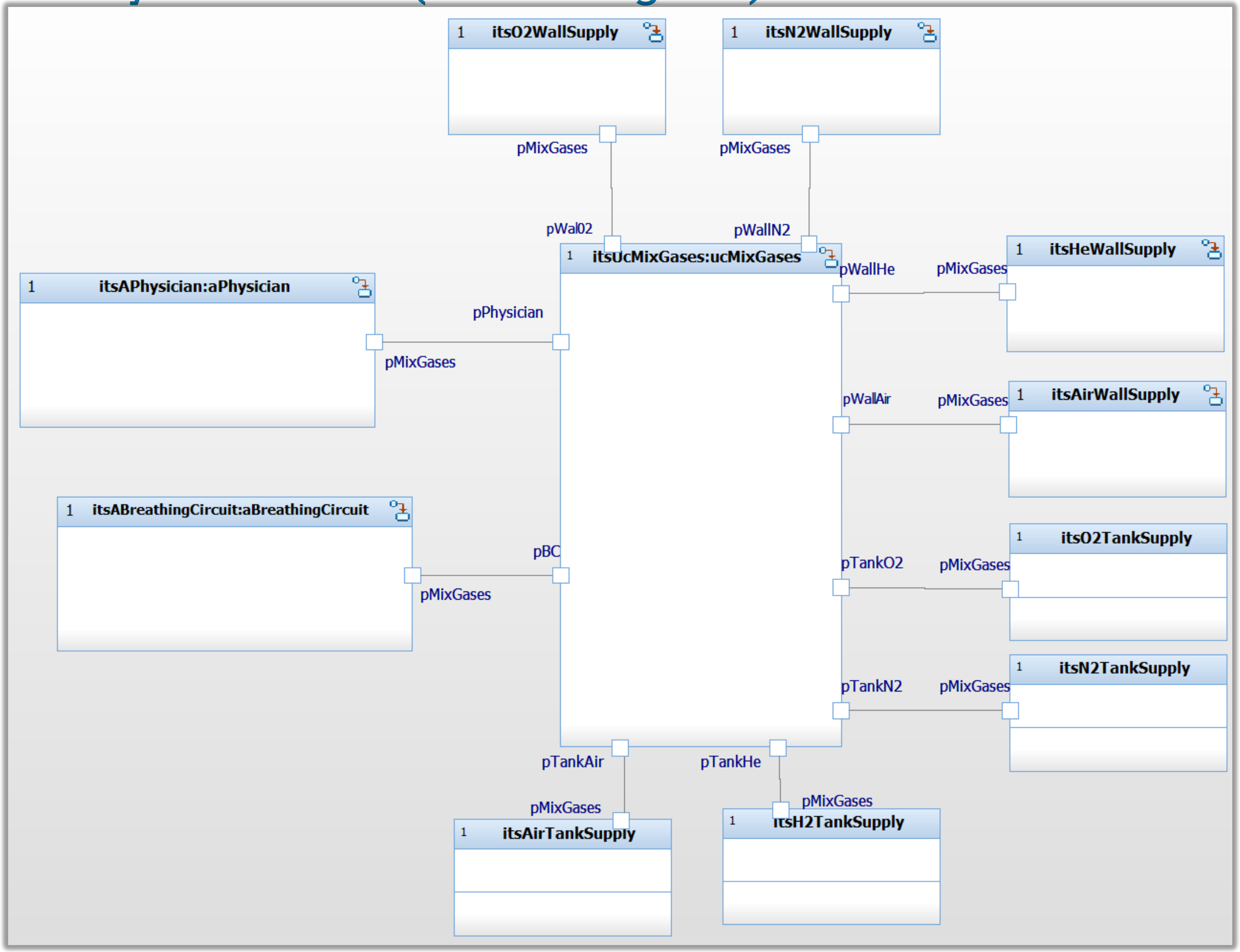
Monitor Patient SpO2
Patient
Monitor NonInvasive Blood Pressure
Monitor System Status
1..*
1..*
1..*
Gas Supply
Configure Ventilation
Configure Mixer
BioMed Engineer
Alarm
Mix Gas
Physician
Ventilate
Test System Offline
Monitor Breathing Circuit
BioMed Engineer
Breathing Circuit

Requirements for Mix Gases Use Case

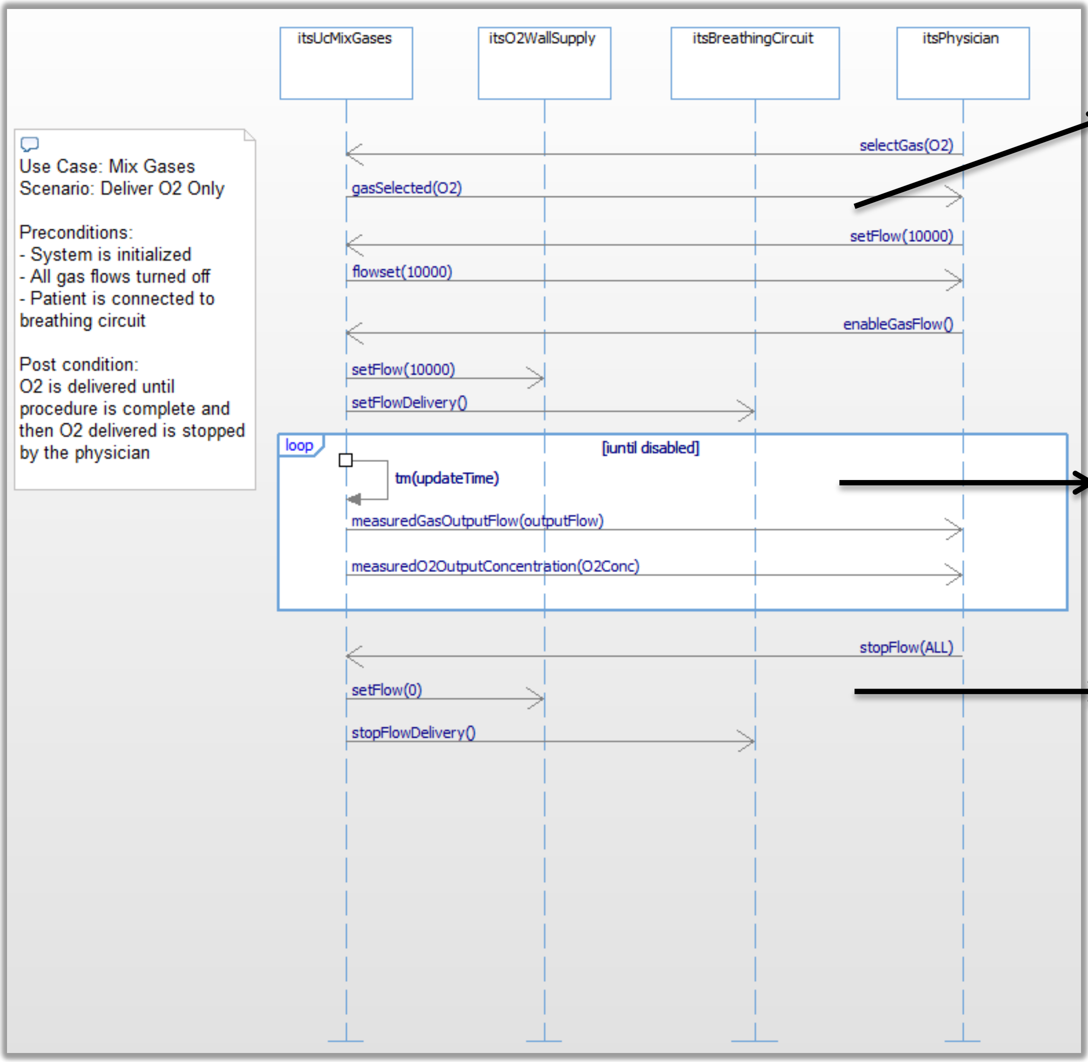
requirement_0	The system shall mix gases O2, N2, He, and Air from the input gas manifold
requirement_1	The system shalldelivery O2 and Air concentration of between 0 and 100.
requirement_2	The system shall support deliveryt of He, N2 to between 0 and 79% concentration.
requirement_3	The System shall 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 balance gas.
requirement_9	The system shall alert if they command an O2 concentration, flow, I;E, tidal volume, or respiration rate out of range.



Create Analysis Context (block diagram)



Create Scenarios



Discovered Requirements

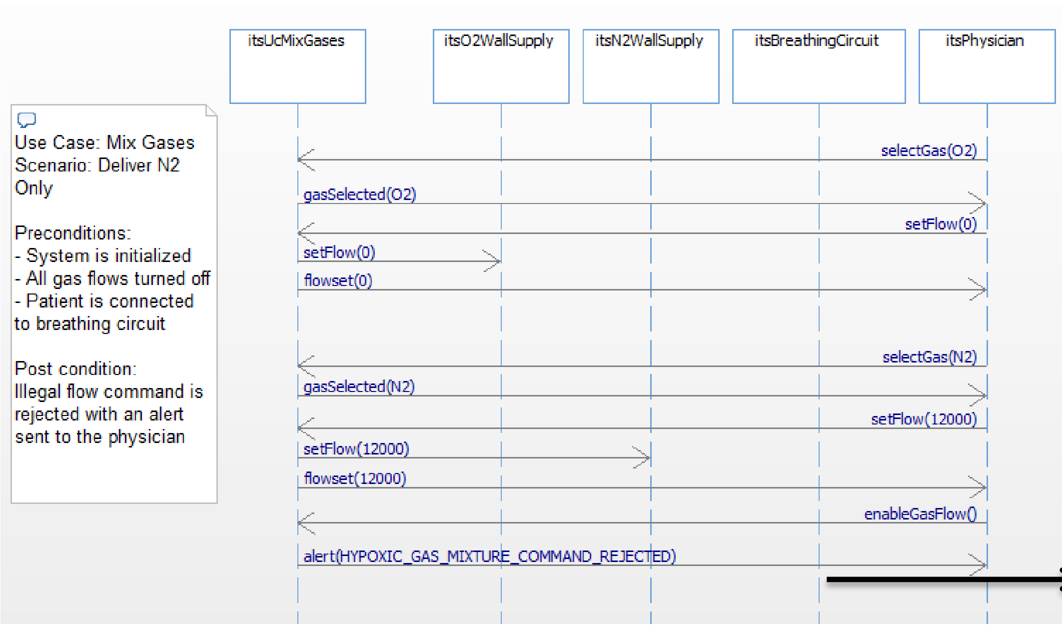
The user shall select the desired gas.
The system shall indicate to the user the currently selected gas.
Once selected, the user shall be able to set a valid flow rate of the gas.
The user shall set the desired flow with user action.
The system shall acknowledge with the set value when the flow is set.

The system shall report the measured total flow and measured oxygen concentration every 1.0s $\pm$ 0.25s

The use shall be able to command flow to stop.
The system shall acknowledge the user command to stop flow.

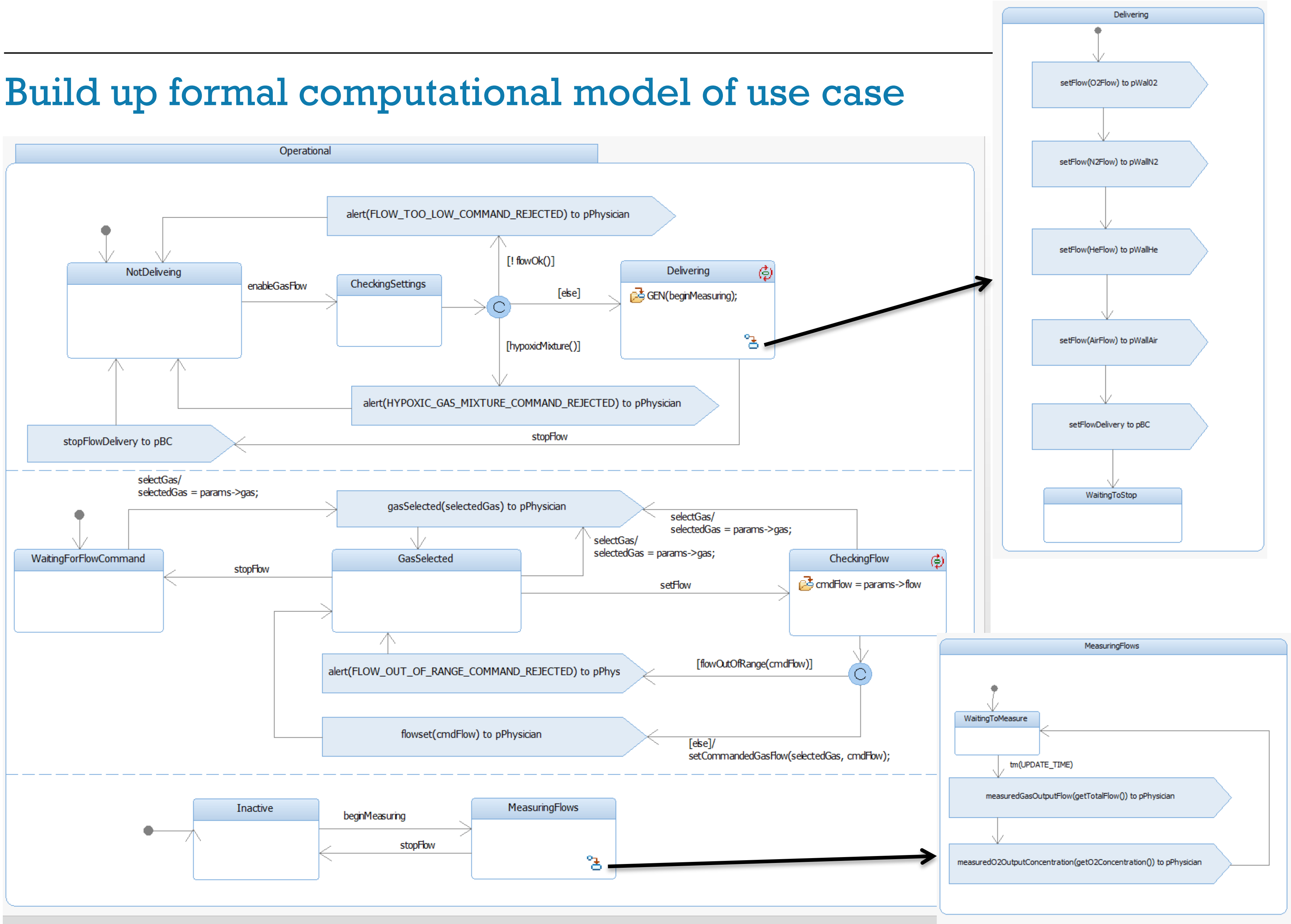
Create Scenarios

Discovered Requirements



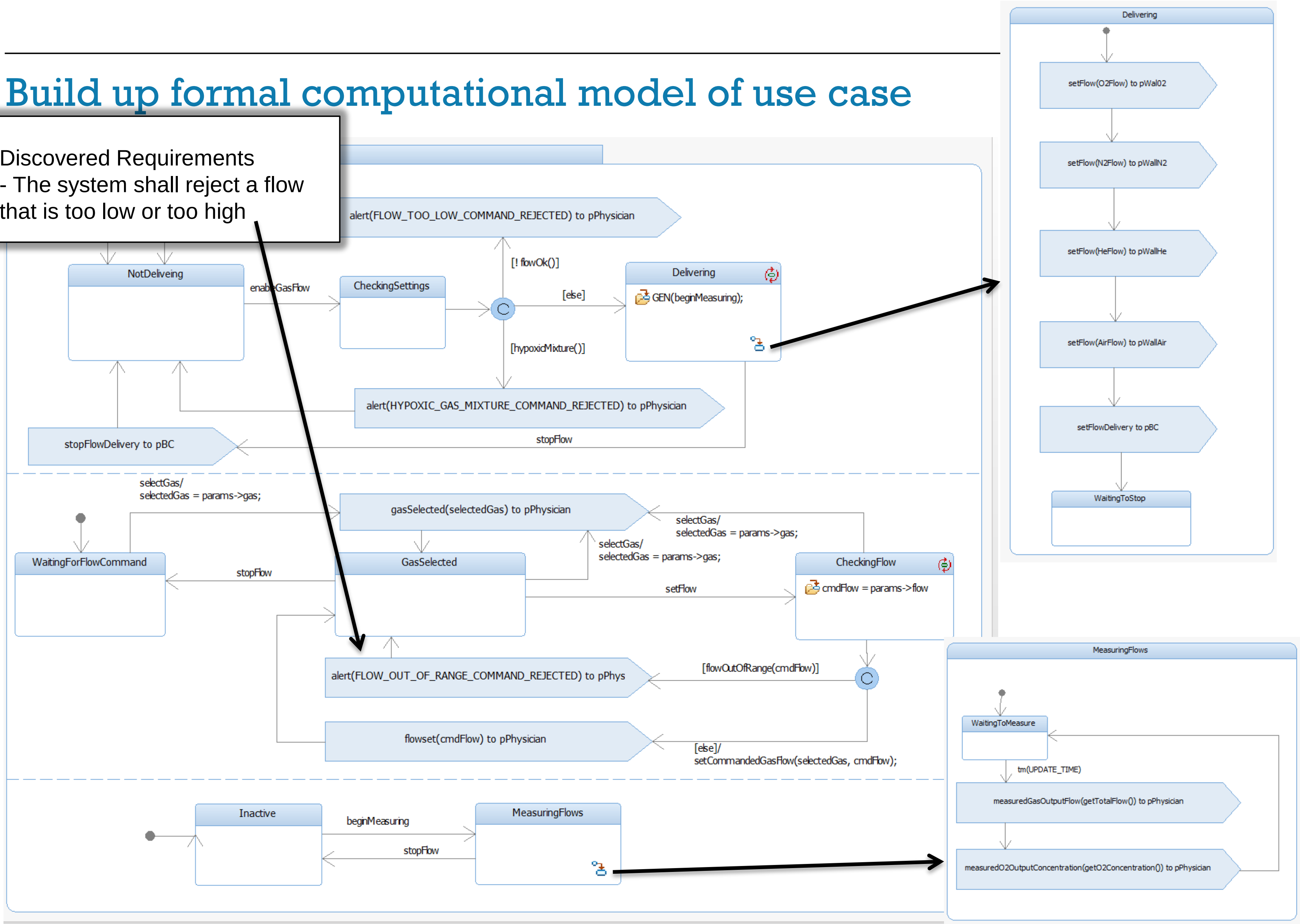
The system shall alert the user if they command a hypoxic gas mixture and reject the command.

Build up formal computational model of use case

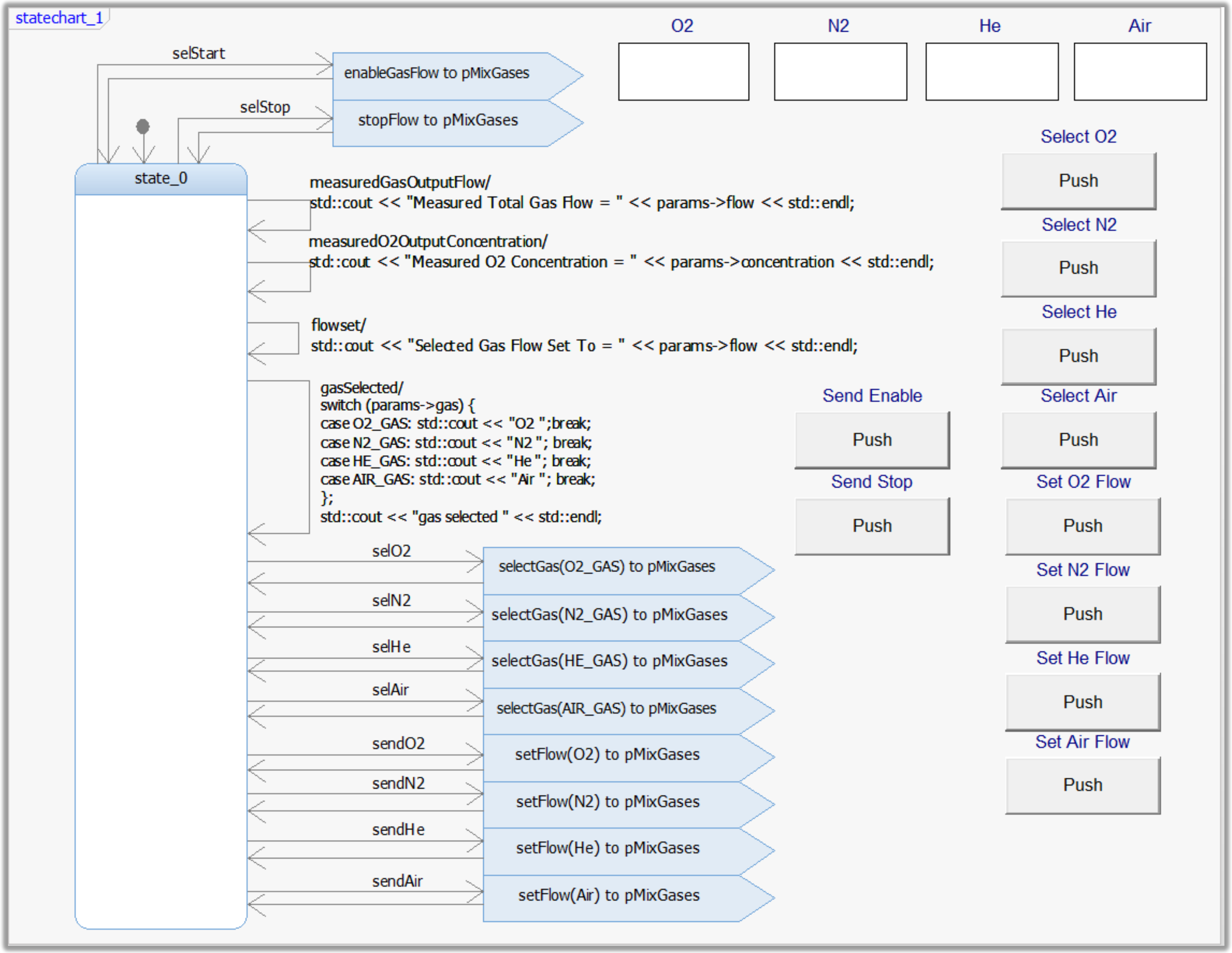


Build up formal computational model of use case

Discovered Requirements
- The system shall reject a flow that is too low or too high

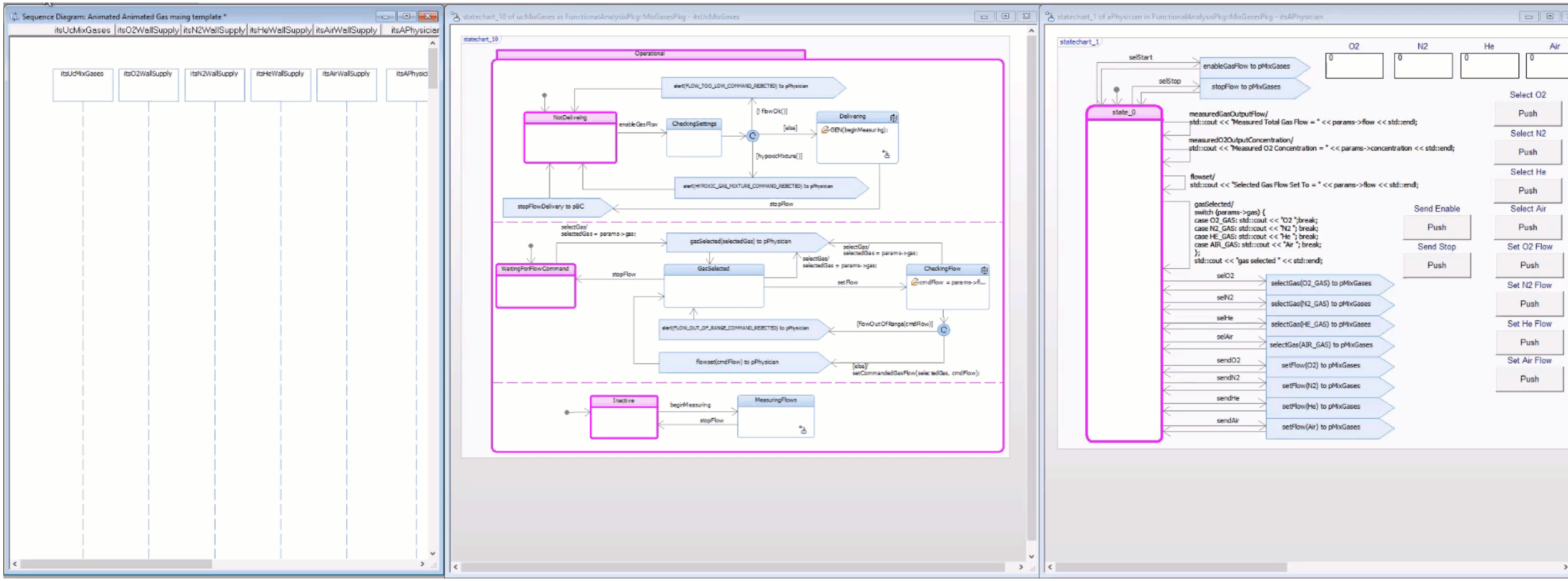


Instrument the context for execution



Physician Actor

Running the model



Physician actions

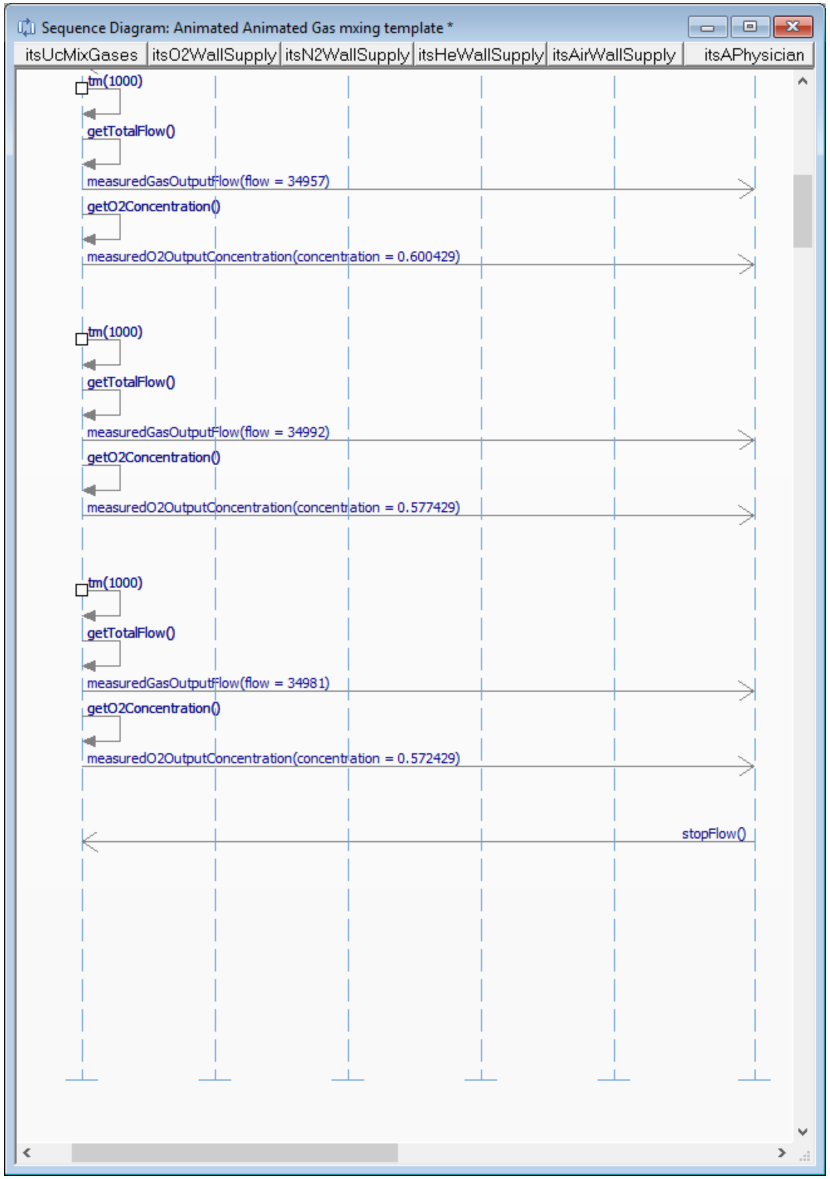
1. Select the O2 gas
2. Set the flow to 20000 l/min
3. Push the Set O2 flow button
4. Select the N2 gas
5. Set the flow to 10,000 l/min
6. Push the Set N2 Flow button

Physician actions

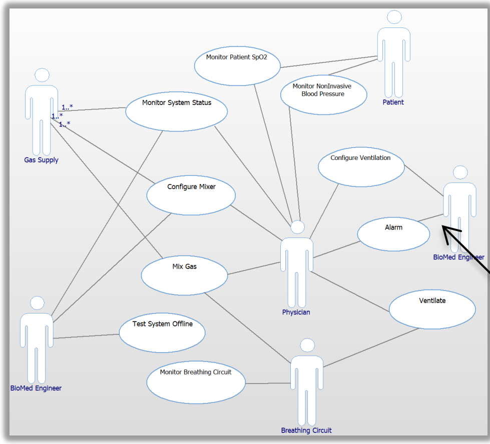
7. Select the He gas
8. Set the flow to 5000 l/min
9. Push the Set H2 Flow button
10. Enable mixing

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

Output (Animated Sequence Diagram)

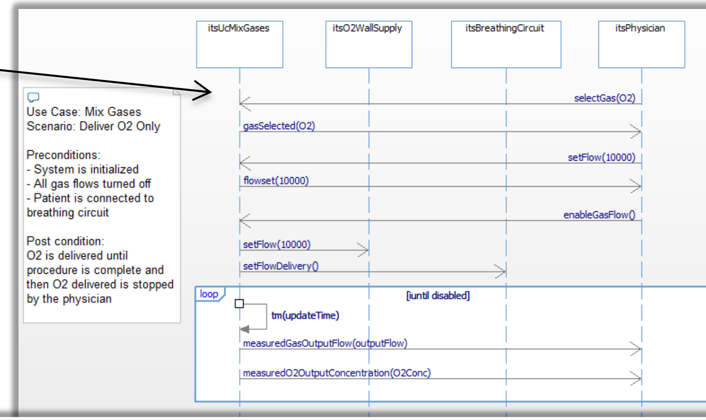


Logical Interfaces are a natural outcome of use case analysis

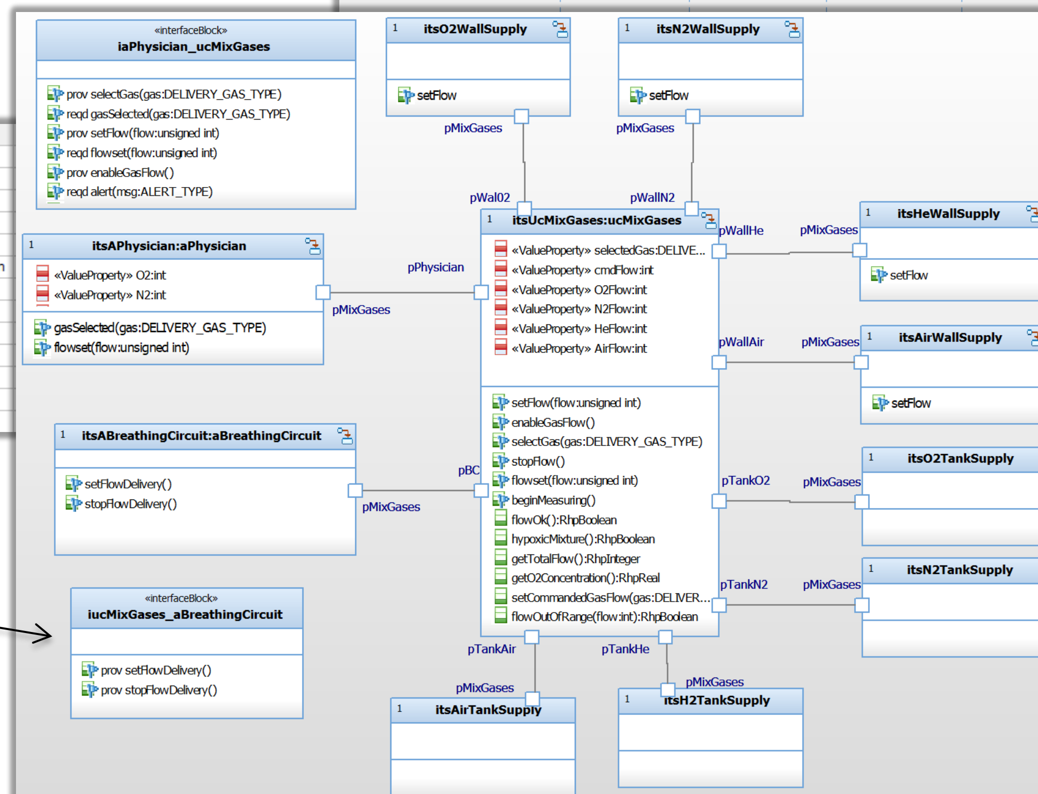


Use case-actor associations indicate the existence of interfaces

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

Download Papers, Presentations, Models, & Profiles for Free

www.bruce-douglass.com