Assurance Cases for Generic PCA Reference Implementation and Beyond

Insup Lee
PRECISE Center
Department of Computer and Information Science
School of Engineering and Applied Science
University of Pennsylvania

IFIP 10.4 Working Group Meeting
Rockport, MA
July 1, 2012

Penn Engineering

PRECISE
PENN RESEARCH IN EMBEDDED COMPUTING AND INTEGRATED SYSTEMS ENGINEERING

Trends in Medical Cyber-Physical Systems (MCPS)

Miniaturization

- Implantable devices
- Ingestible sensors

Interoperation

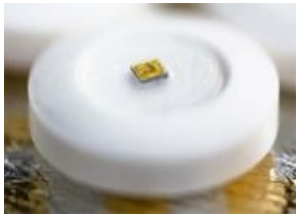
- Executable clinical scenarios
- Safety interlocks

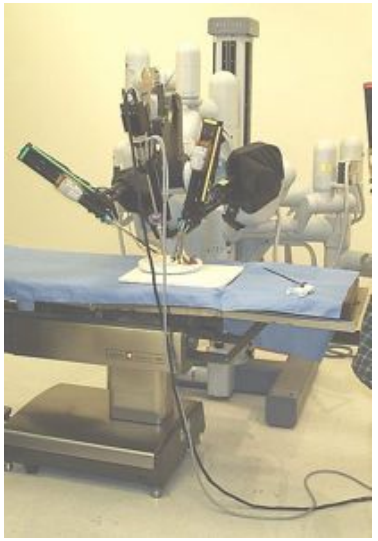
Teleoperation

- Tele-ICU
- Robotic surgery

Autonomy

- Smart alarms
- Context-sensitive decision support
- Physiological closed loop control



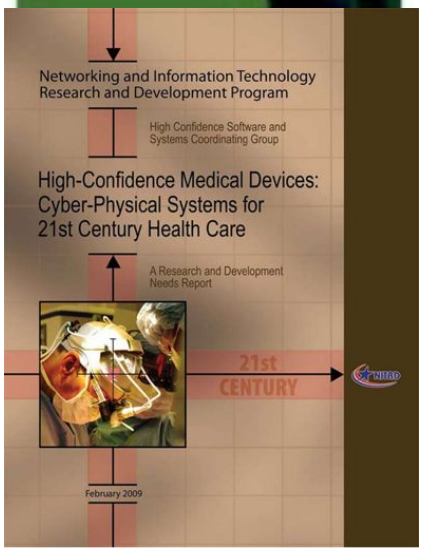


Penn Engineering

PRECISE

MCPS Research Challenges (partial list)

- High-confidence medical device software systems (HCMDSS)
  - Model-based and evidence-based development
  - Patient modeling and simulation
  - User-centered design
- Medical device integration and interoperation
- Adaptive patient-specific algorithms
- Incremental and compositional methods for certifiable assurance and safety



Penn Engineering

PRECISE

Safety-Assured Model-Based Development of GPCA Infusion Pump Software

BaekGyu Kim, Anaheed Ayoub, Oleg Sokolsky, Insup Lee,
Paul Jones, Yi Zhang, and Raoul Jetley

Penn Engineering

PRECISE
PENN RESEARCH IN EMBEDDED COMPUTING AND INTEGRATED SYSTEMS ENGINEERING

Infusion Pump Safety

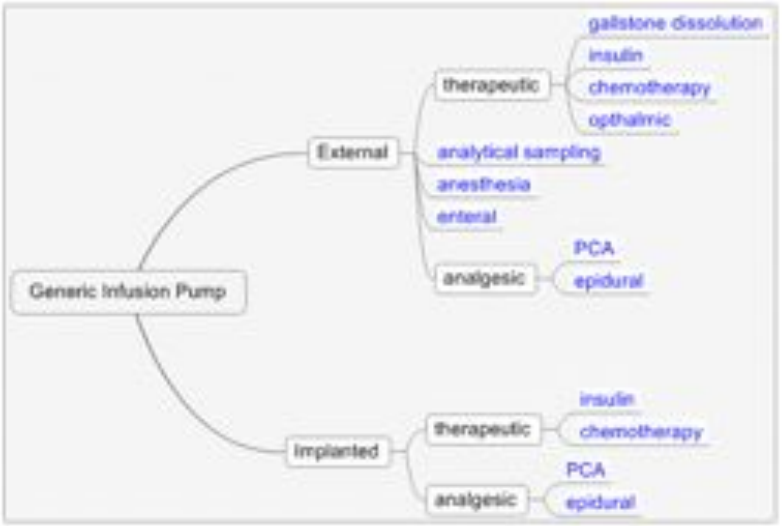
- During 2005 and 2009, FDA received approximately 56,000 reports of adverse events associated with the use of infusion pumps
  - 1% deaths, 34% serious injuries
  - 87 infusion pump recalls to address safety problems
- The most common types of problems
  - **Software Defect**
  - User Interface Issues
  - Mechanical or Electrical Failure

U.S. Food and Drug Administration, Center for Devices and Radiological Health. White Paper: Infusion Pump Improvement Initiative, April 2010

Generic Infusion Pump (GIP) Project

- The Goal of GIP Project
  - To develop a set of generic infusion pump (safety) models and reference specification that can be used as a reference standard to verify safety properties in different classes of infusion pumps

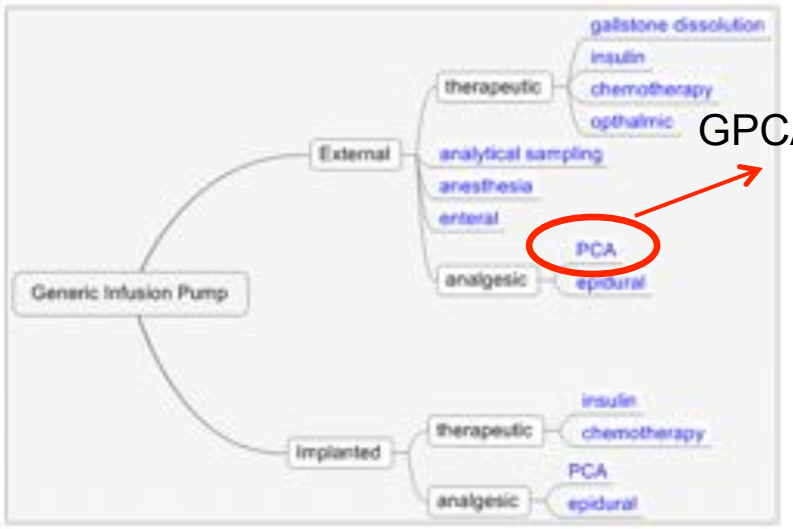
- GIP web site
  - provide a repository of medical device artifacts for use in projects that advance the science and practice of developing high-confidence medical devices, software, and systems, and
  - establish infusion pump safety reference models
  - Open contribution
  - http://rtg.cis.upenn.edu/gip.php3



GIP Class Diagram

Generic PCA (GPCA)

- Generic PCA (Patient Controlled Analgesic) Infusion pump
  - GPCA hazard analysis
  - GPCA safety requirements
  - GPCA reference model
- Goals
  - Demonstrate the use of model-based development techniques for engineering medical device software
  - Provide a base open-source reference model that can be extended and modified to develop specific implementations of PCA pump software
  - Provide an example assurance cases for medical device
  - Provide generic test suites (*)
  - Provide a reasonably complex medical design for researchers to use in developing, refining, and improving theories and methods needed to develop certifiably dependable medical devices
  - http://rtg.cis.upenn.edu/medical/gpca/gpca.html



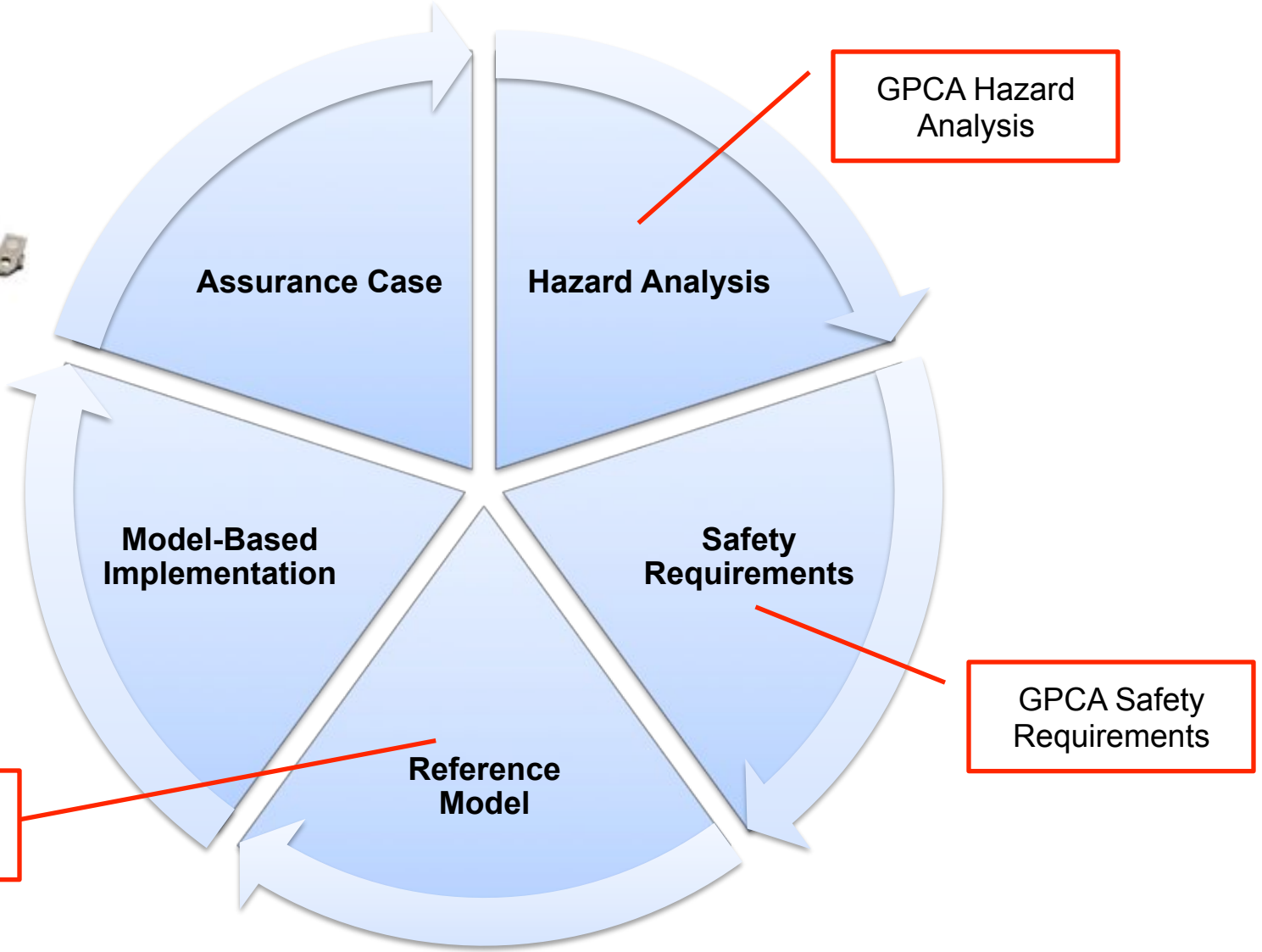
GIP Class Diagram

PCA Pump

Penn Engineering

PRECISE

Generic PCA (GPCA) Project

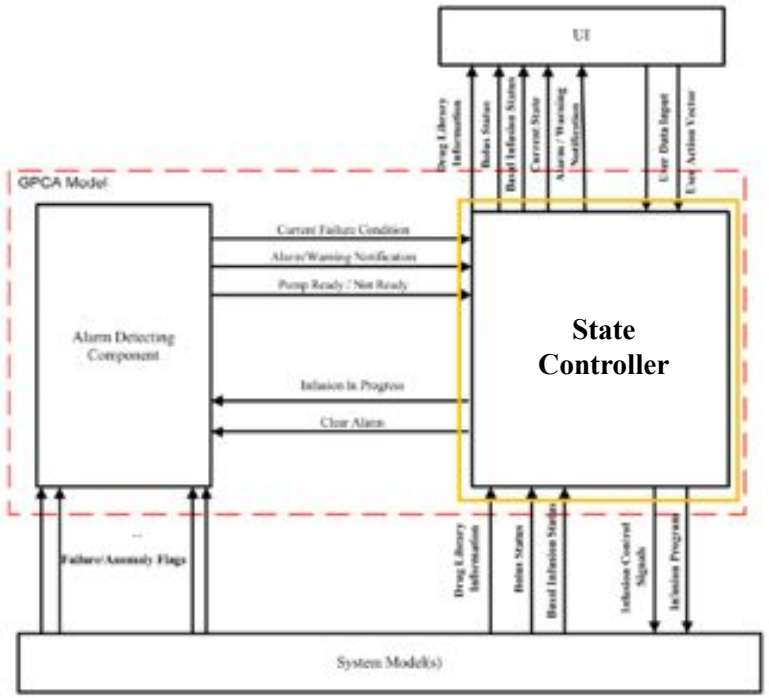


Penn Engineering

PRECISE

FDA's GPCA Model

- An abstract representation of software used in a typical PCA infusion pump.
- The model is built in Simulink and Stateflow.
- State Controller
  - Describes a drug administration process such as parameter setting and bolus request.
- Alarm Detecting Component
  - Check hardware conditions and process alarm on any hardware failure.
- GPCA Environment
  - User Interface
  - System model
    - The GPCA model interacts with pump hardware such as motor and sensors through the System Model.



The System Architecture of GPCA Model

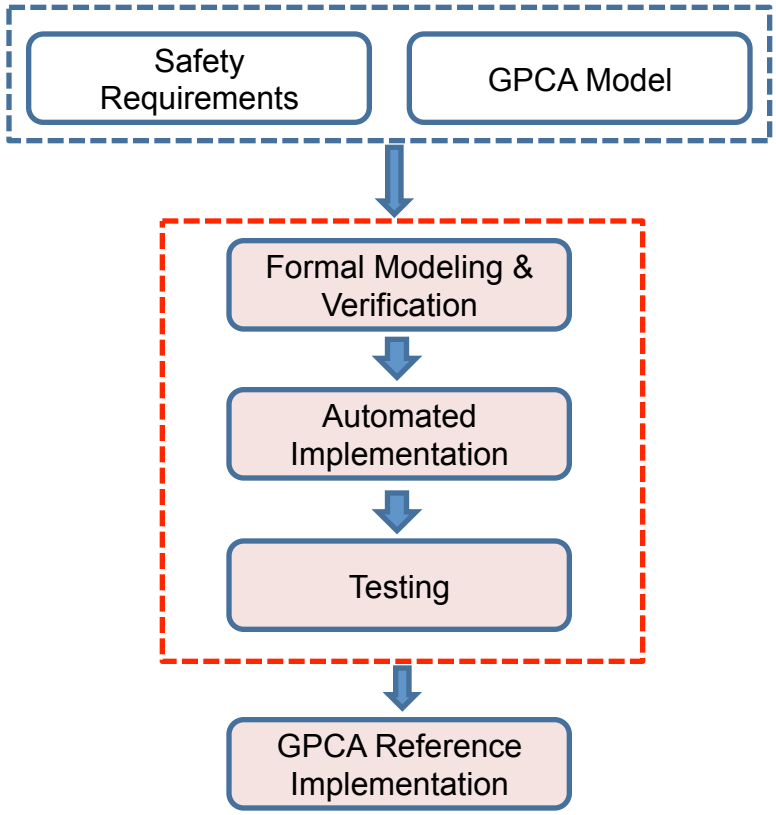
FDA's GPCA Safety Requirements

- A minimum set of generic safety requirements that can be used to evaluate and verify infusion pump software*
  - (e.g.) No normal bolus doses should be administered when the pump is alarming (in an error state).
  - (e.g.) If the calculated volume of the reservoir is *y ml*, and an infusion is in progress, an Empty Reservoir alarm shall be issued.
  - (e.g.) The pump shall issue an alert if paused for more than *t minutes*.

* Raoul Jetley and Paul Jones. Safety Requirements based Analysis of Infusion Pump Software. Proceedings of the Workshop on Software and Systems for Medical Devices and Services, December 2007.

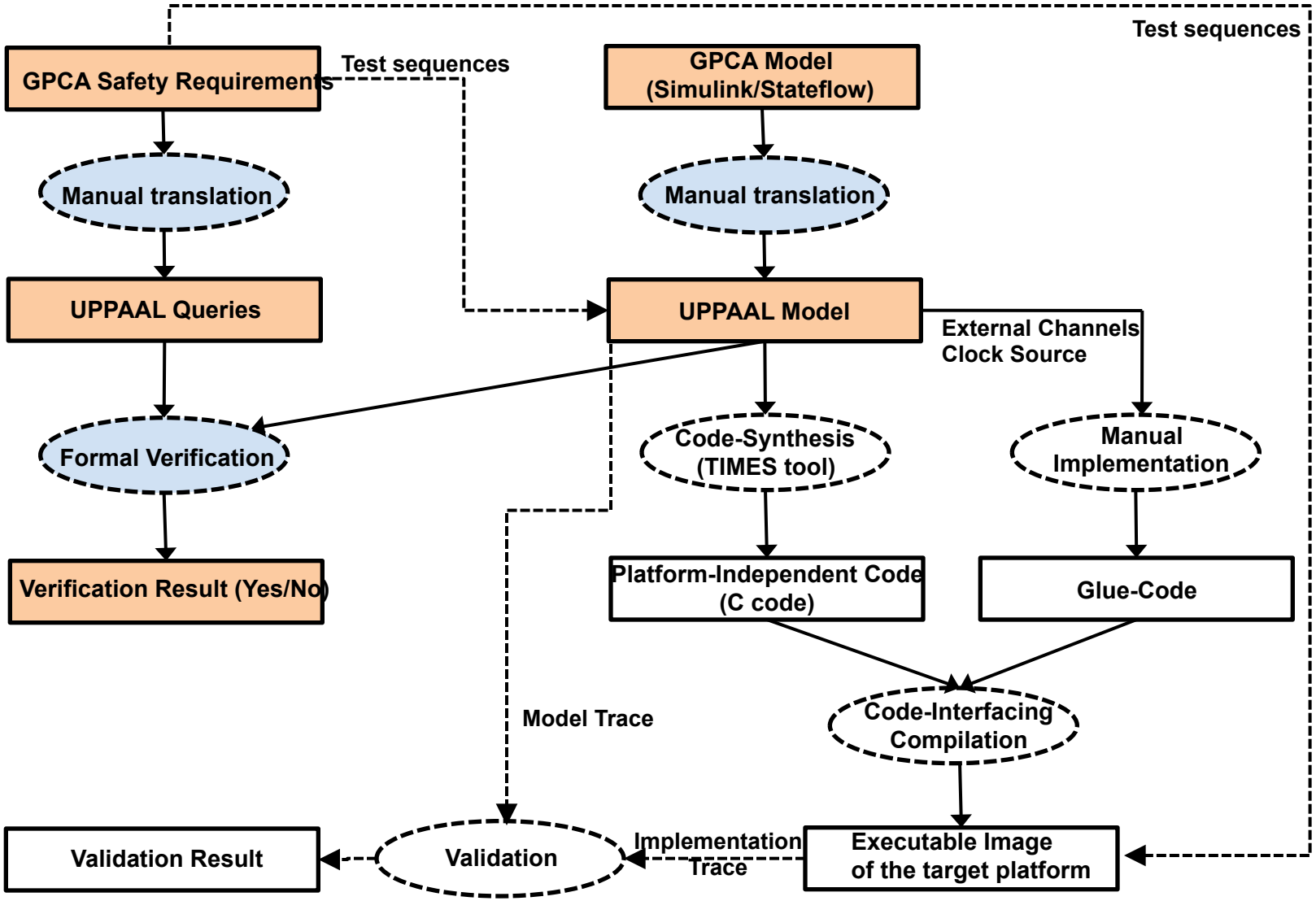
GPCA reference implementation

- FDA initiated
  - GPCA Safety Requirements
  - GPCA Model (Simulink/Stateflow)
- Goal: Develop a GPCA reference implementation
- Provide evidence that the implementation satisfies the safety requirements
  - Code synthesis
- Organize evidence for certification
  - Safety cases
  - Confidence cases
- All artifacts to be available as open source
  - [AADL case study by KSU]



Model-Based Development of GPCA Reference Implementation

Part 1: Formal Verification Part

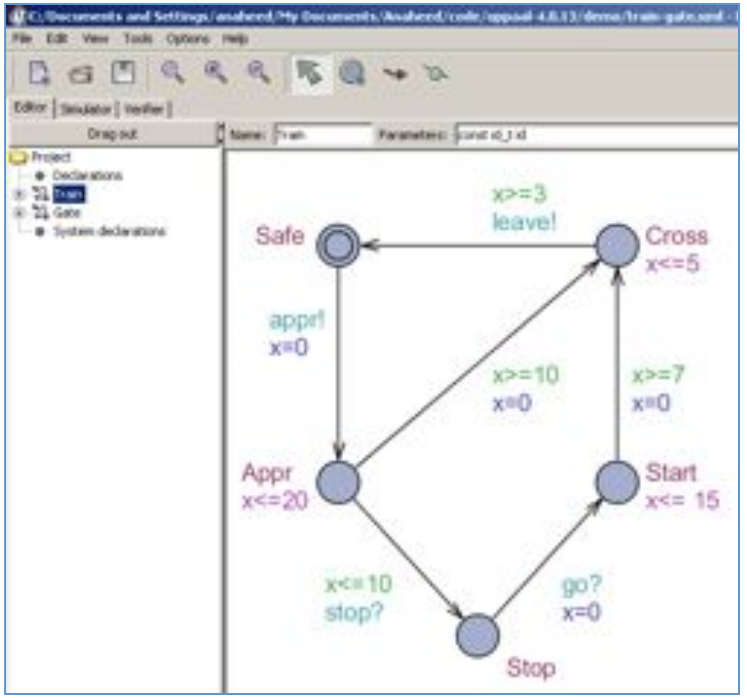


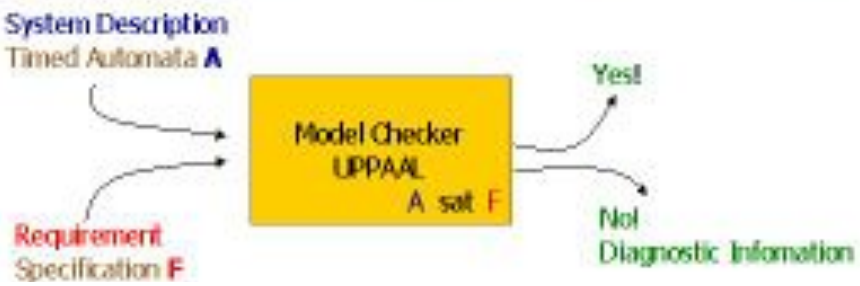
Penn Engineering

PRECISE

UPPAAL (**UPP**sala + **AAL**borg = UPPAAL)

- **UPPAAL** is a tool for Modeling, Validation, and Verification
- Major functionalities:
  - A description language: network of timed automata extended with variables.
  - A Simulator : validation tool which enables examination of possible executions of a system.
  - A Model-checker: for automatic verification of safety properties by reachability analysis of the symbolic state-space.





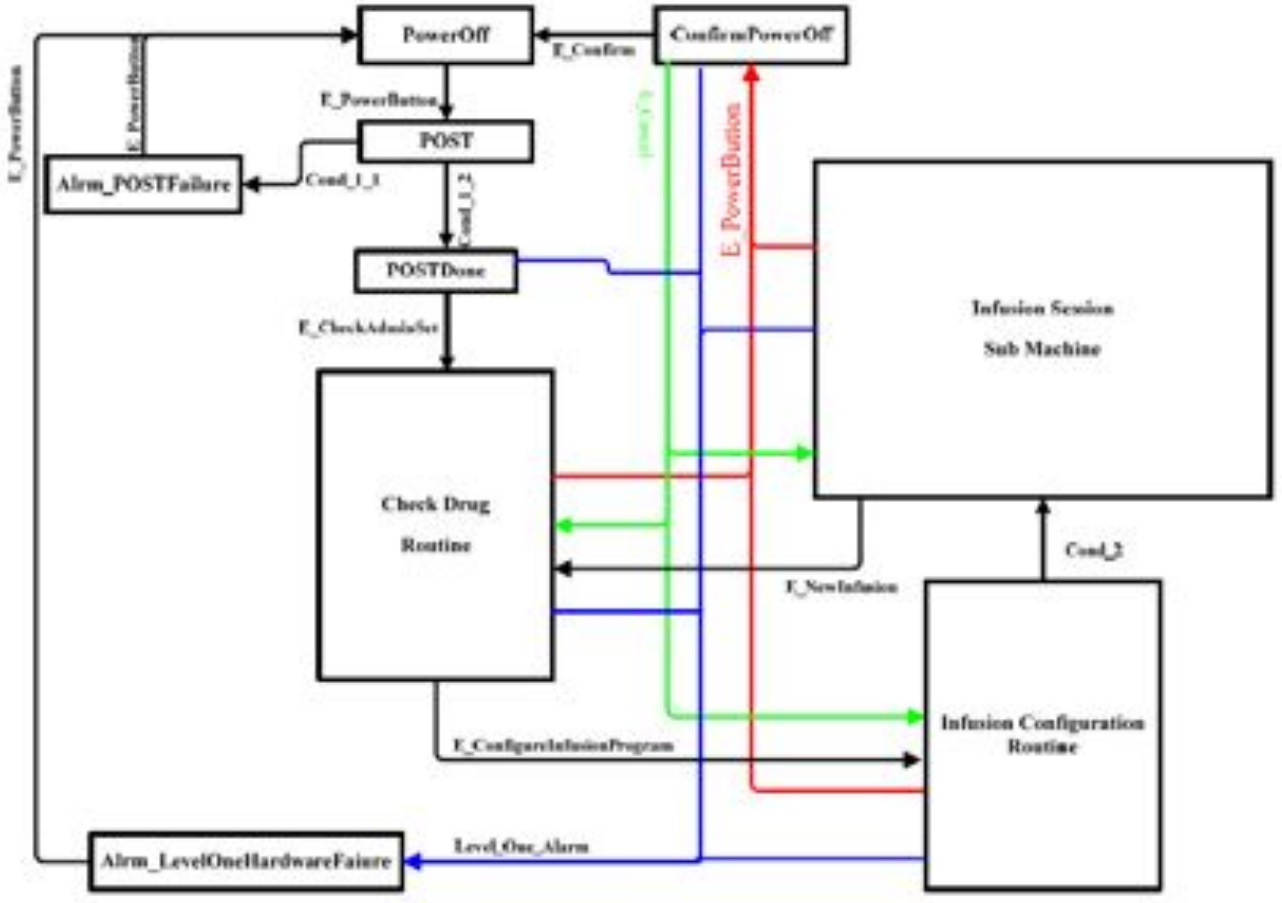
Penn Engineering

PRECISE

Formalization of the FDA's GPCA model

- Transform the GPCA model into a network of UPPAAL automata
  - Retain as much of the architecture of the GPCA model as possible following a rigorous manual translation process
  - Maintain one-to-one mapping between states, conditions, user actions, and transitions in the two models
    - State : Alarm-Empty-Reservoir
    - Condition : Cond-6-2 (An infusion error Empty Reservoir is detected during the ongoing infusion process.)
    - Action : E-RequestBolus (Request for a bolus dose by pressing a button)
  - Currently the UPPAAL model consists of approximately 50 states, 100 transitions, and 50 user actions and conditions

Formalization of the FDA's GPCA model

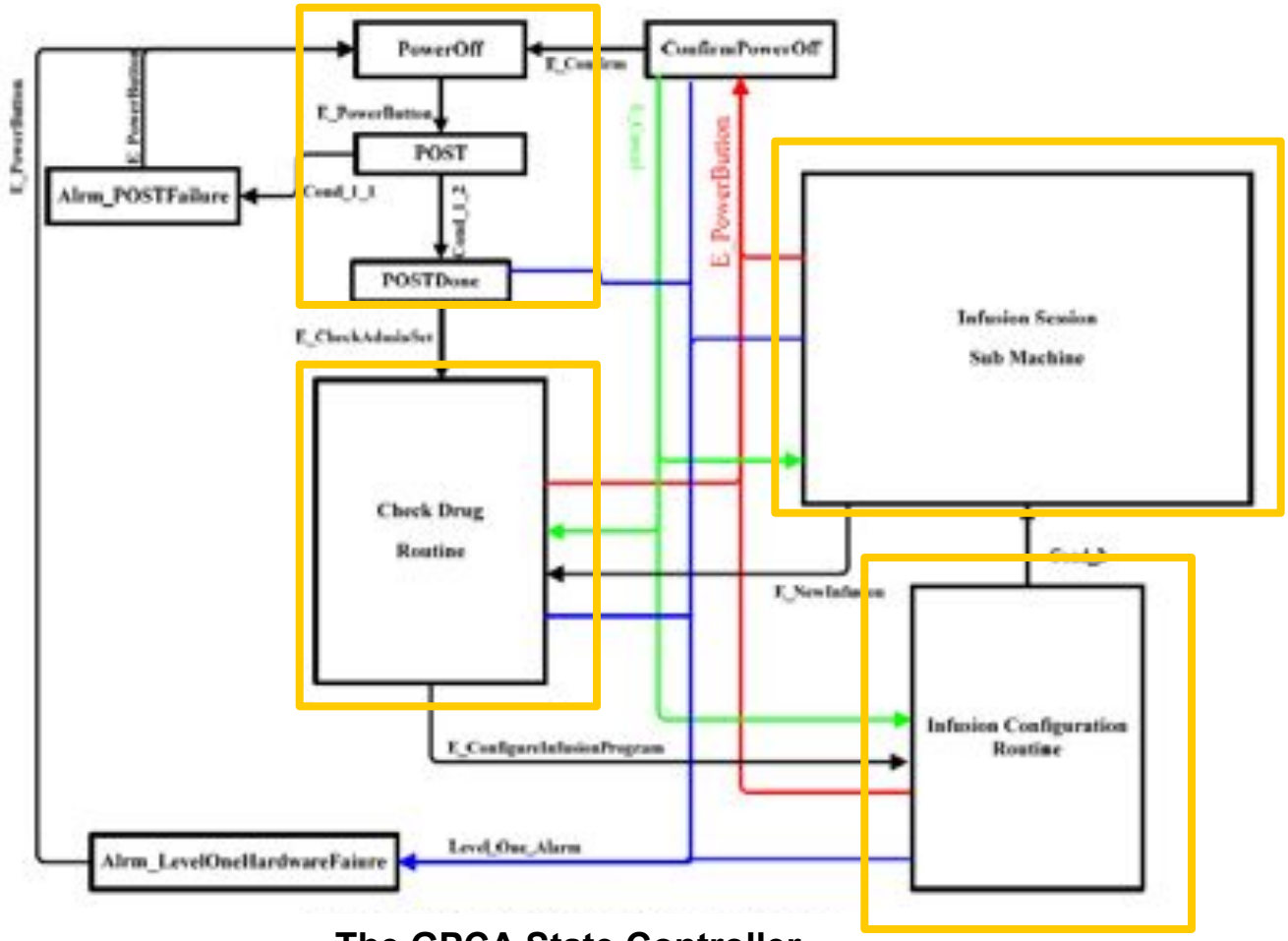


The GPCA State Controller

Penn Engineering

PRECISE

Formalization of the FDA's GPCA model

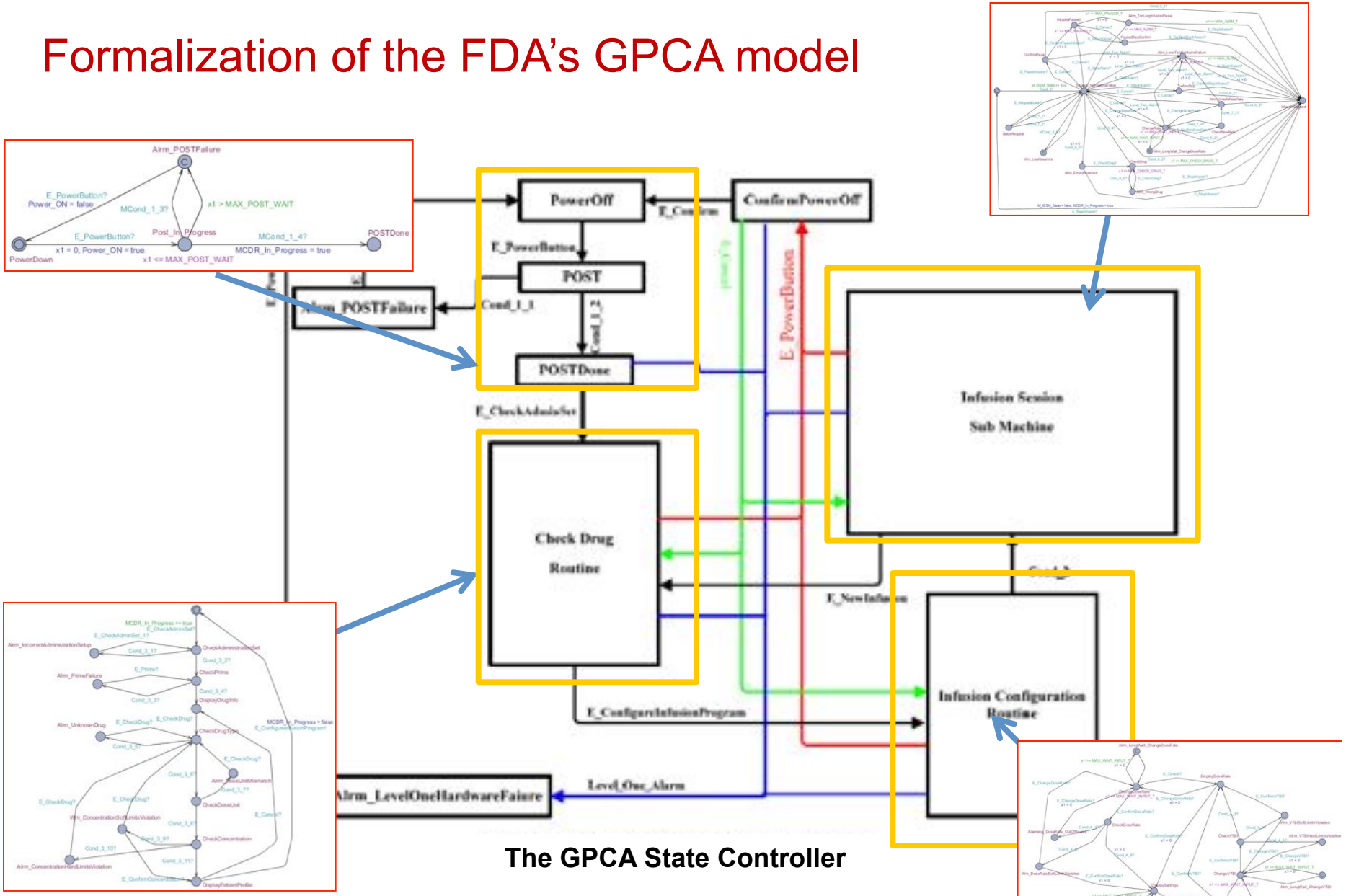


The GPCA State Controller

Penn Engineering

PRECISE

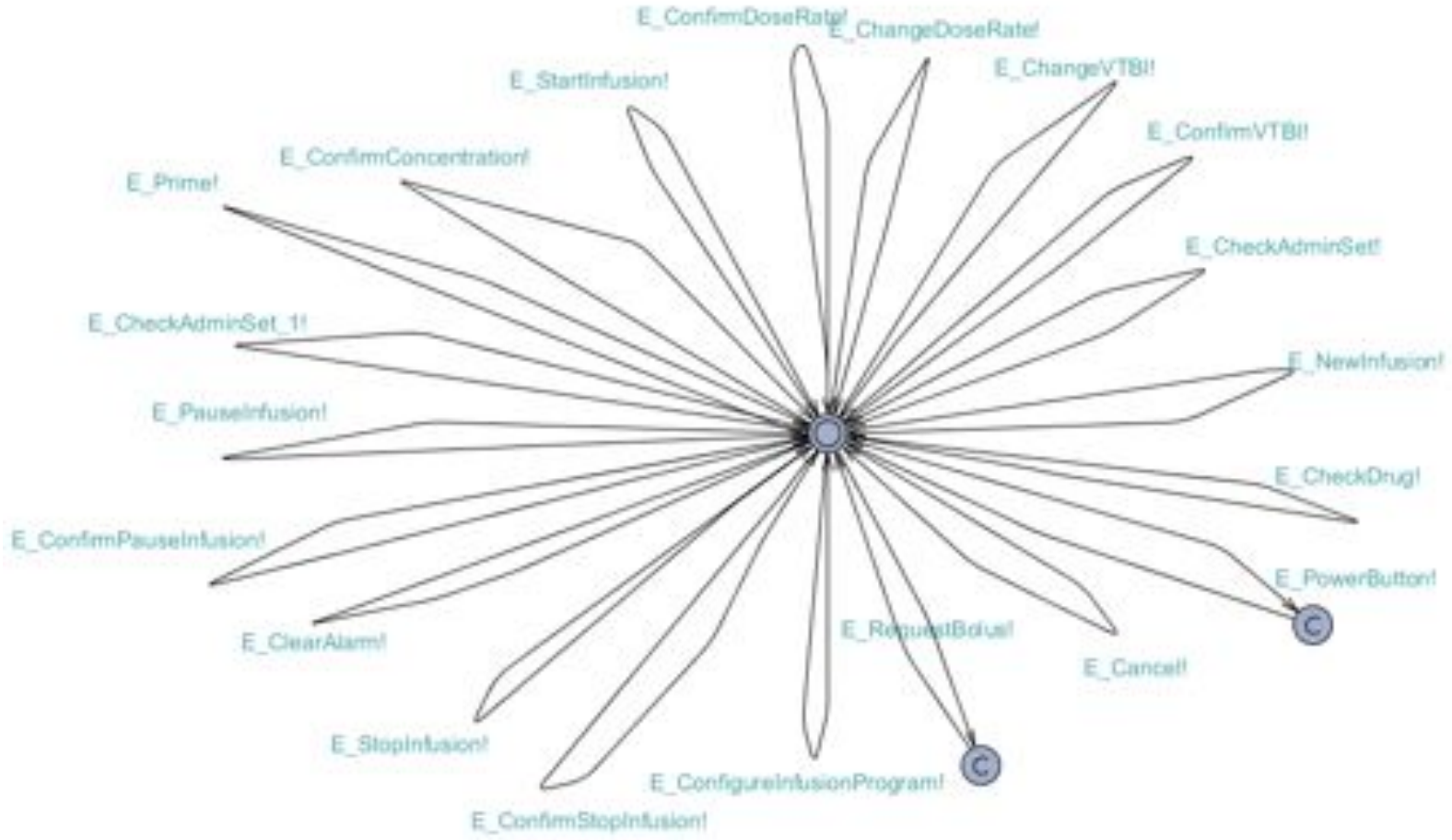
Formalization of the FDA's GPCA model



The GPCA State Controller

Penn Engineering

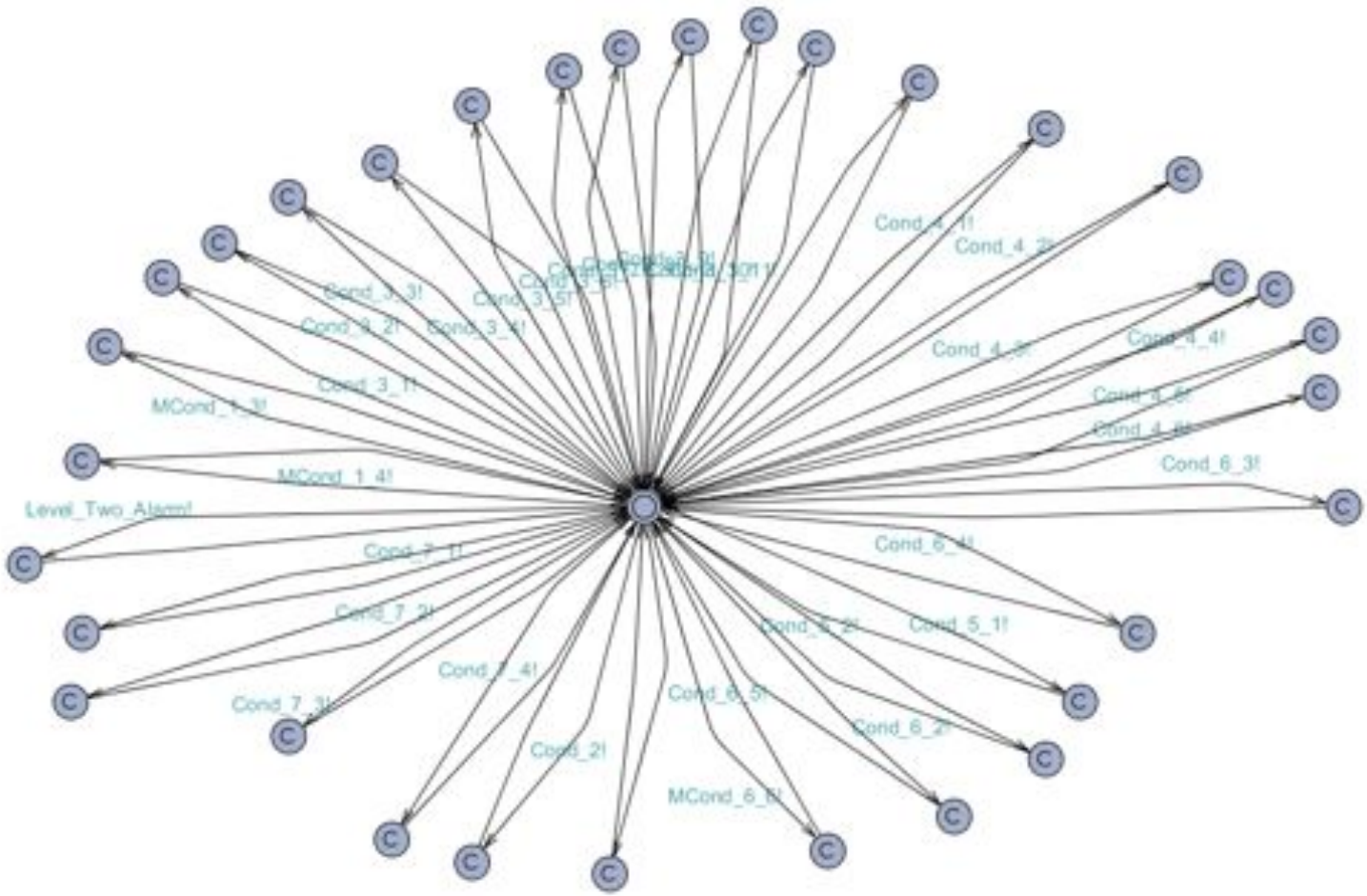
Environment: User Actions



Penn Engineering

PRECISE

Environment : Hardware Conditions



Cond-6-3 implies “An infusion error *Empty Reservoir* is detected during the ongoing infusion process”

Penn Engineering

PRECISE

Formalization of the Safety Requirements

- Safety requirements are translated into temporal logic formula using the UPPAAL query language.
- Example of Safety requirement formalization
  - No bolus dose shall be possible during the POST
    - A[] (! (POST.Post-In-Progress && ISSM.BolusRequest))
  - No normal bolus doses should be administered when the pump is alarming (in an error state).
    - A[](! (ISSM.BolusRequest && CDR.Alrm-UnknownDrug))
  - The pump shall issue an alert if paused for more than t minutes
    - (ISSM.InfusionPaused && x1 > MAX-PAUSED-T)
      -> ISSM.Alrm-TooLongInfusionPause
  - If the calculated volume of the reservoir is y ml, and an infusion is in progress, an Empty Reservoir alarm shall be issued.
    - (ISSM.Infusion-NormalOperation && Cond-6-3== true)
      -> ISSM.Alrm-EmptyReservoir

Formalization of the Safety Requirements

- Not all 97 safety requirements can be translated into temporal logic formula.
- Categorization of the safety requirements.

  Category 1) A safety requirement can be formalized and verified in the UPPAAL model. (~20 out of 97 requirements)
  - No bolus dose shall be possible during the POST
  - The pump shall issue an alert if paused for more than t minutes

  Category 2) A safety requirement can be formalized, but the GPCA model needs additional information to verify it. (~23 out of 97 requirements)
  - If the suspend occurs due to a fault condition, the pump shall be stopped immediately without completing the current pump stroke.

Formalization of the Safety Requirements

- Not all safety requirements can be translated into temporal logic formula.
- Categorization of the safety requirements.

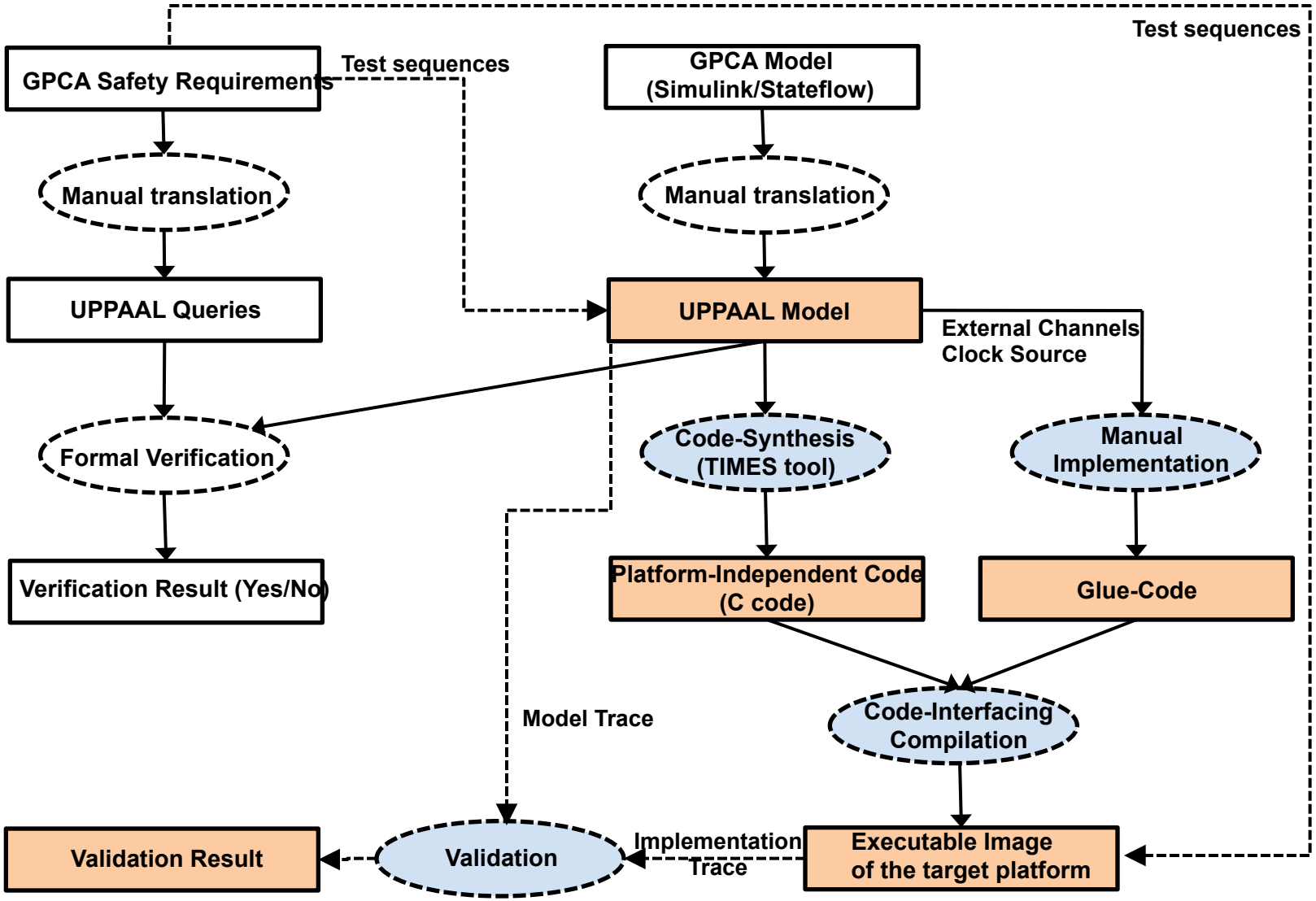
  Category 1) A safety requirement can be formalized and verified in the UPPAAL model. (~20 out of 97 requirements)
  - No bolus dose shall be possible during the POST
  - The pump shall issue an alert if paused for more than t minutes

  Category 2) A safety requirement can be formalized, but the GPCA model needs additional information to verify it. (~23 out of 97 requirements)
  - If the suspend occurs due to a fault condition, the pump shall be stopped immediately without completing the current pump stroke.

  Category 3) A safety requirement cannot be formalized, but can be validated at the implementation level. (~31 out of 97 requirements)
  - The flow rate for the bolus dose shall be programmable.

  Category 4) A safety requirement cannot be formalized because the statement is too vague or related to the environment of the GPCA model. (~23 out of 97 requirements)
  - Flow discontinuity at low flows should be minimal (“minimal” is not clear).
  - A key that is depressed shall not be identified as a distinct key press for a period of t seconds (related to UI).

Part 2: Implementation



Penn Engineering

PRECISE

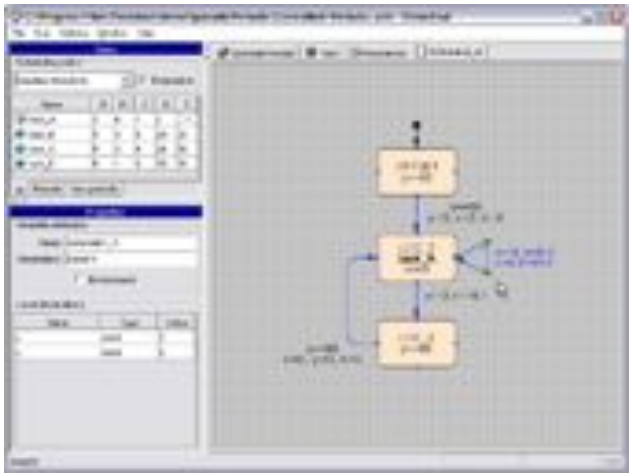
Code Synthesis

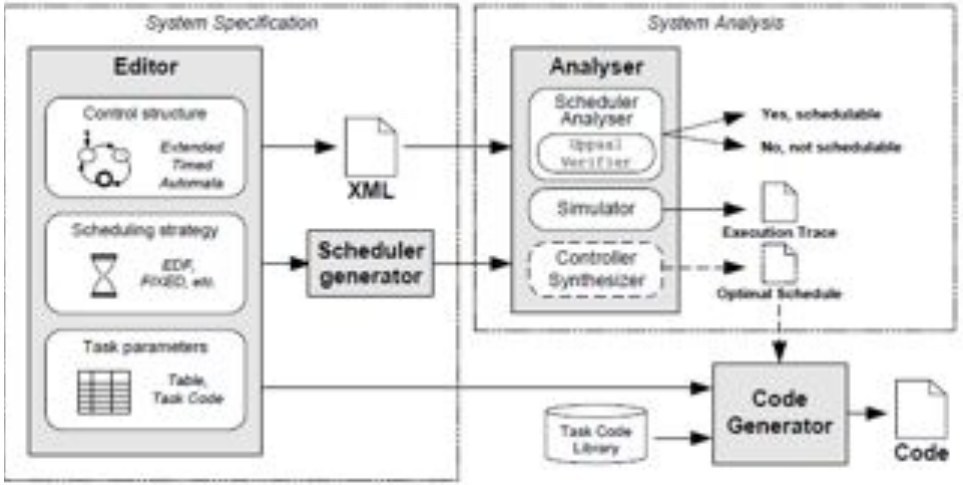
- Advantages of automated implementation
  - An automated implementation improves the quality of embedded software by preserving the properties of model verification.
- Practical obstacles in automated implementation
  - There is a gap between abstract model and implementation

TIMES

(Tool for Modeling and Implementation of Embedded Systems)

- **TIMES** is a tool set for modeling, schedulability analysis, synthesis of executable code:
  - Modeling – timed automata extended with tasks
  - Analysis – simulator and model checker of UPPAAL
  - **Code synthesis –** from timed automata model to C-code for either Brick OS or platform-independent

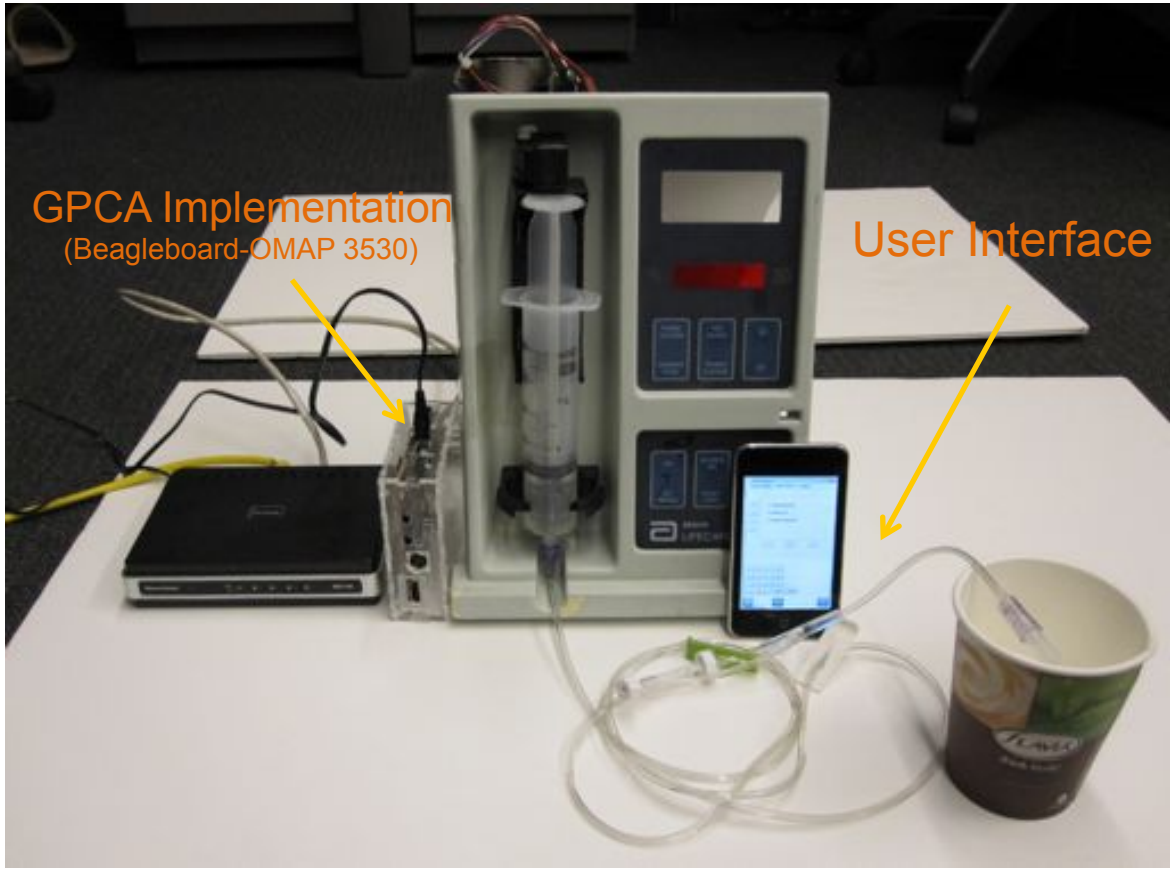


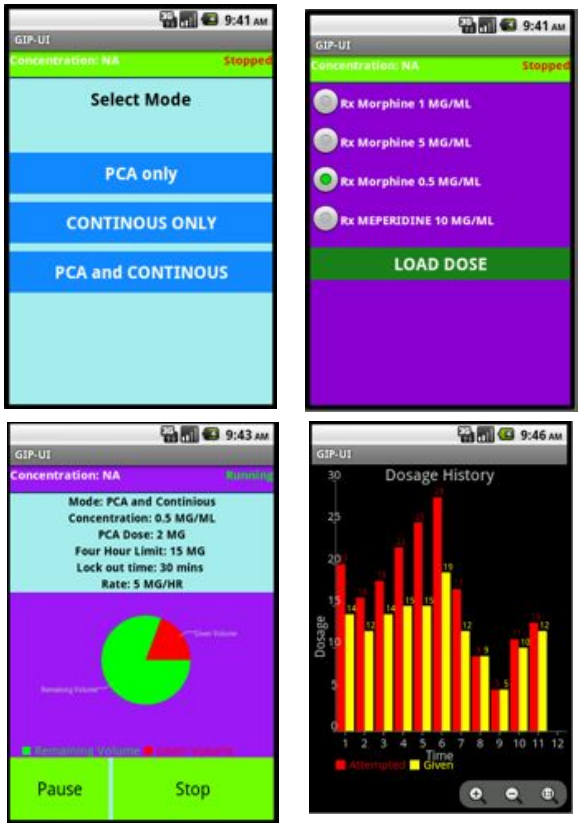


Penn Engineering

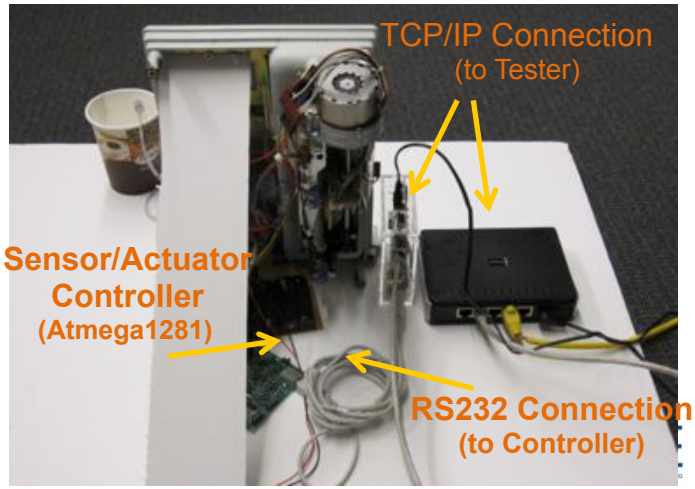
PRECISE

GPCA Implementation Testbed



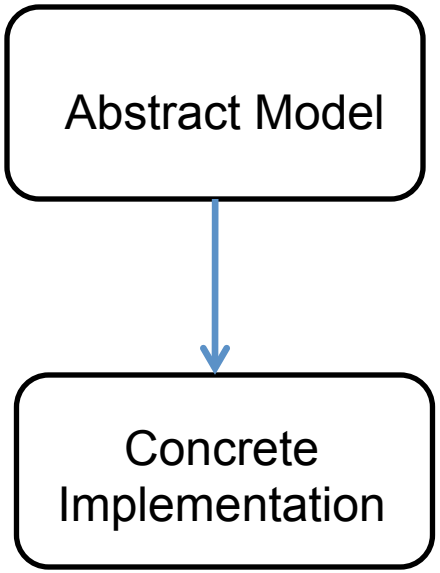


•We note that the Android UI design is motivated from CADD –Solis Ambulatory Infusion System. The functionalities are instantiated from the GPCA model.



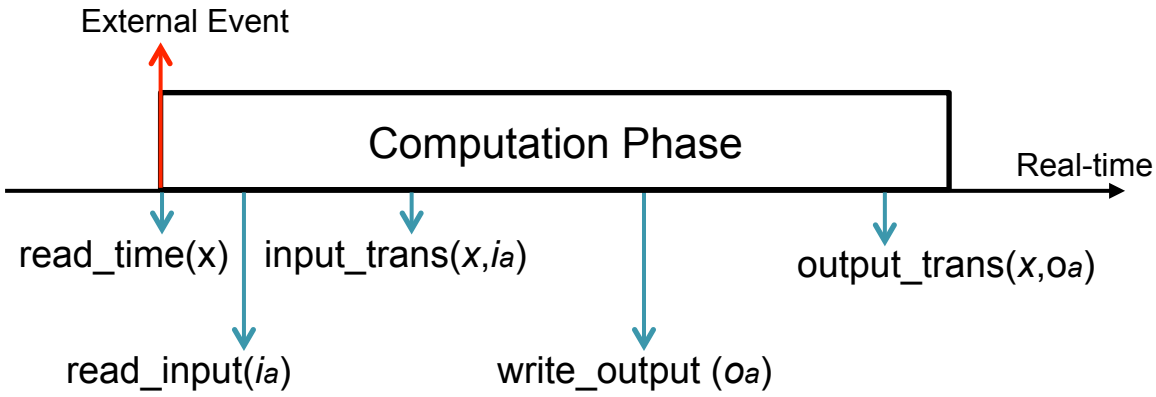
Penn Engineering

Gap: Synchrony Assumption in Modeling



- Synchrony Assumption
  - The program reacts to external events instantaneously.
  - Pros: greatly simplifies formal analysis of real-time systems.
  - Cons: real systems cannot guarantee the assumption due to computation delay.

1. Read Time
2. Read Input
3. Input-Transition
4. Write Output
5. Output-Transition



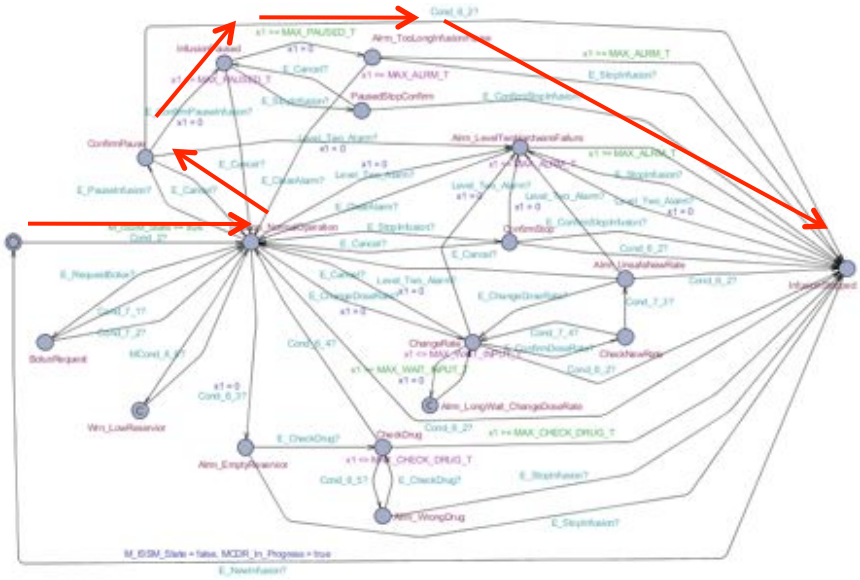
Types of the GPCA Pump Source Code

1. GPCA model code (Platform-independent)
   - GPCA model is synthesized into C-code using TIMES tool.
   - This code implements control-flow of the GPCA model depending on user-action and hardware conditions.
2. Glue code to interface to the target platform (Platform-dependent)
   - Clock implementation using the target platform APIs.
   - Environmental interface (for user and GPCA hardware).
3. Code for abstracted functionalities
   - Pump-motor driving code on transition to Infusion-Normal-Operation to inject drug to patient (e.g., providing electrical signal to the pump motor)
   - Code for updating dose rate on ChangeDoseRate state (e.g., maintaining variables for dose rate that is updated by user request)

Part 3: Validation

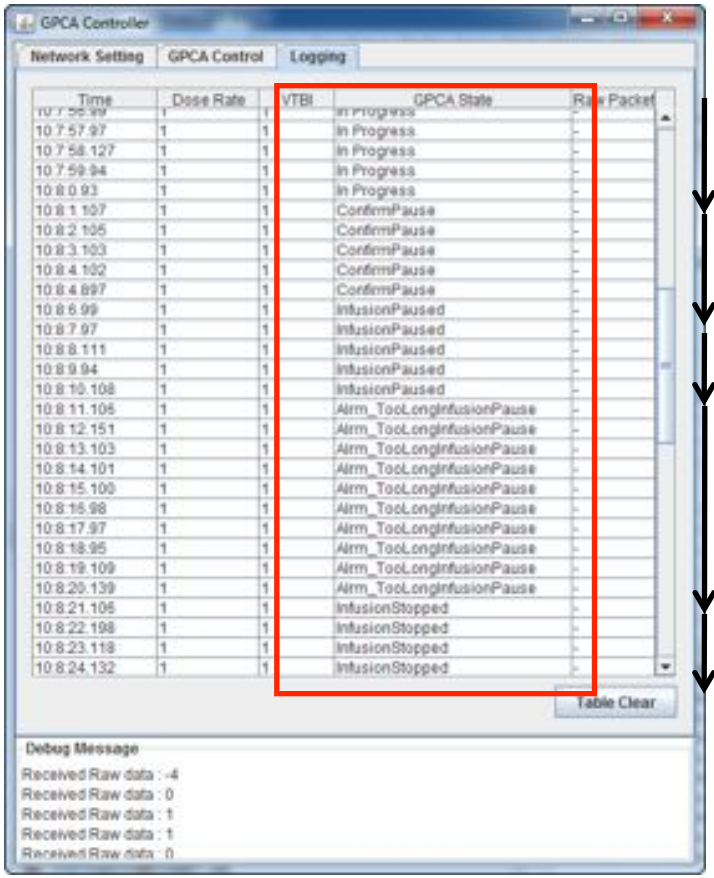
- Safety Requirement : The pump shall issue an alarm if paused for more then t minutes

<Model Trace>



The GPCA UPPAAL model transformed from FDA's GPCA model (Infusion Session Submachine)

<Implementation Trace>



The Tester screenshot

Penn Engineering

PRECISE

Abbott/Hospira Lifecare 4100 PCA PLUS II

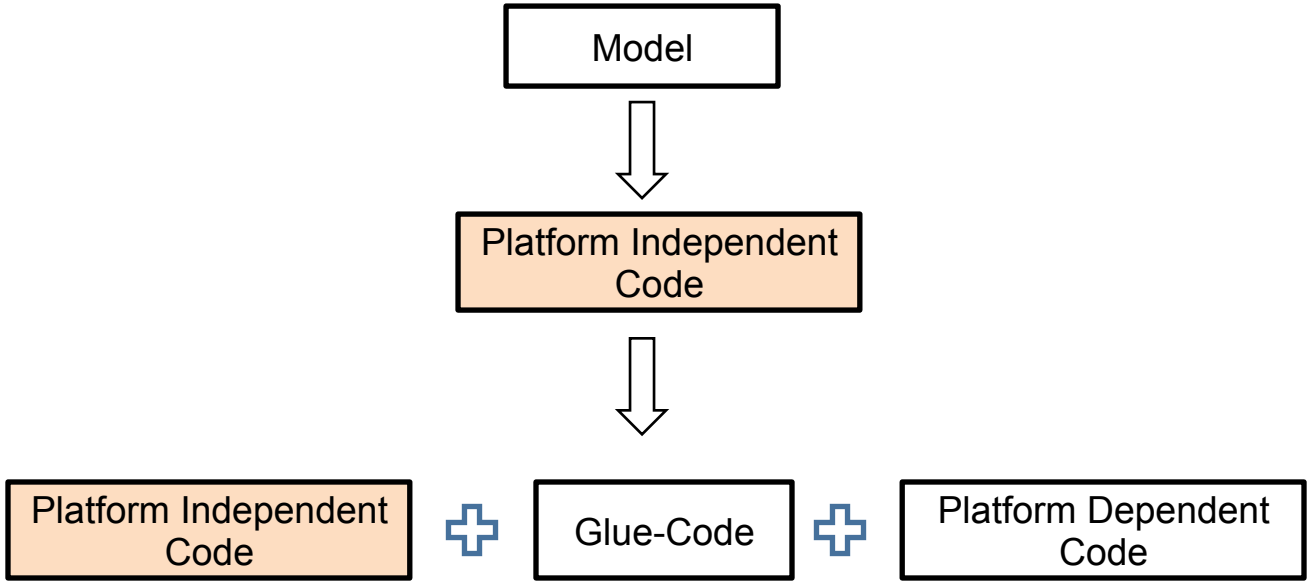
Baxter PCA II Syringe Pump

Penn Engineering

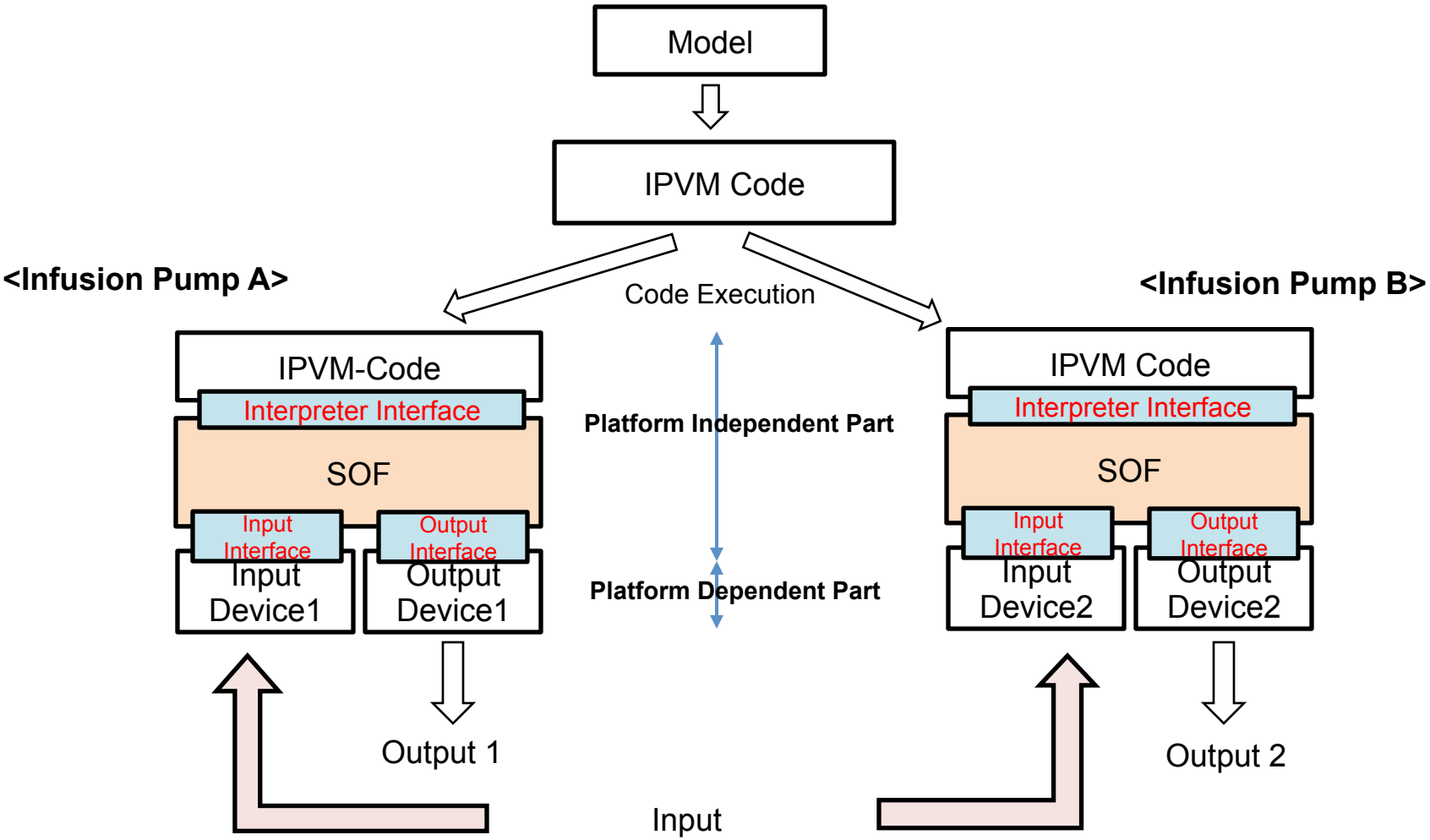
PRECISE
PENN RESEARCH IN EMBEDDED COMPUTING AND INTEGRATED SYSTEMS ENGINEERING

Challenge: time & i/o determinism

- How to ensure that a target platform *correctly* executes the generated code?
- What should be the notion of correctness?



Approach: Infusion Pump Virtual Machine

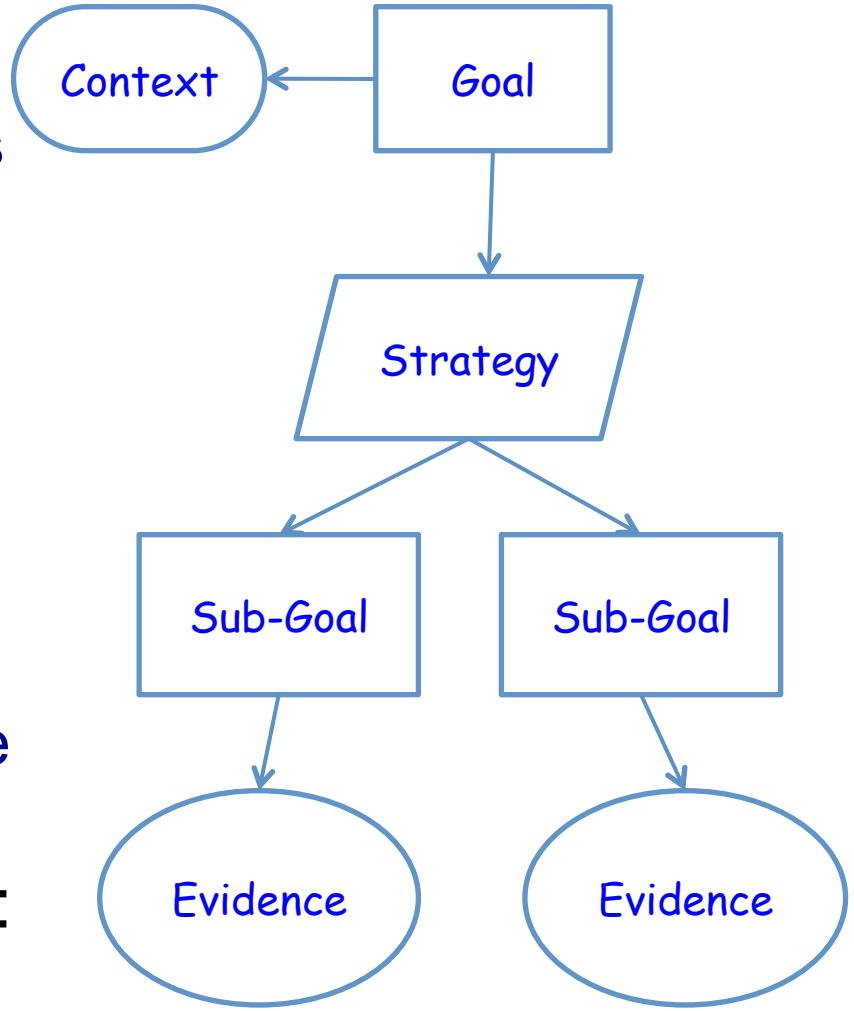


Current and Future Work

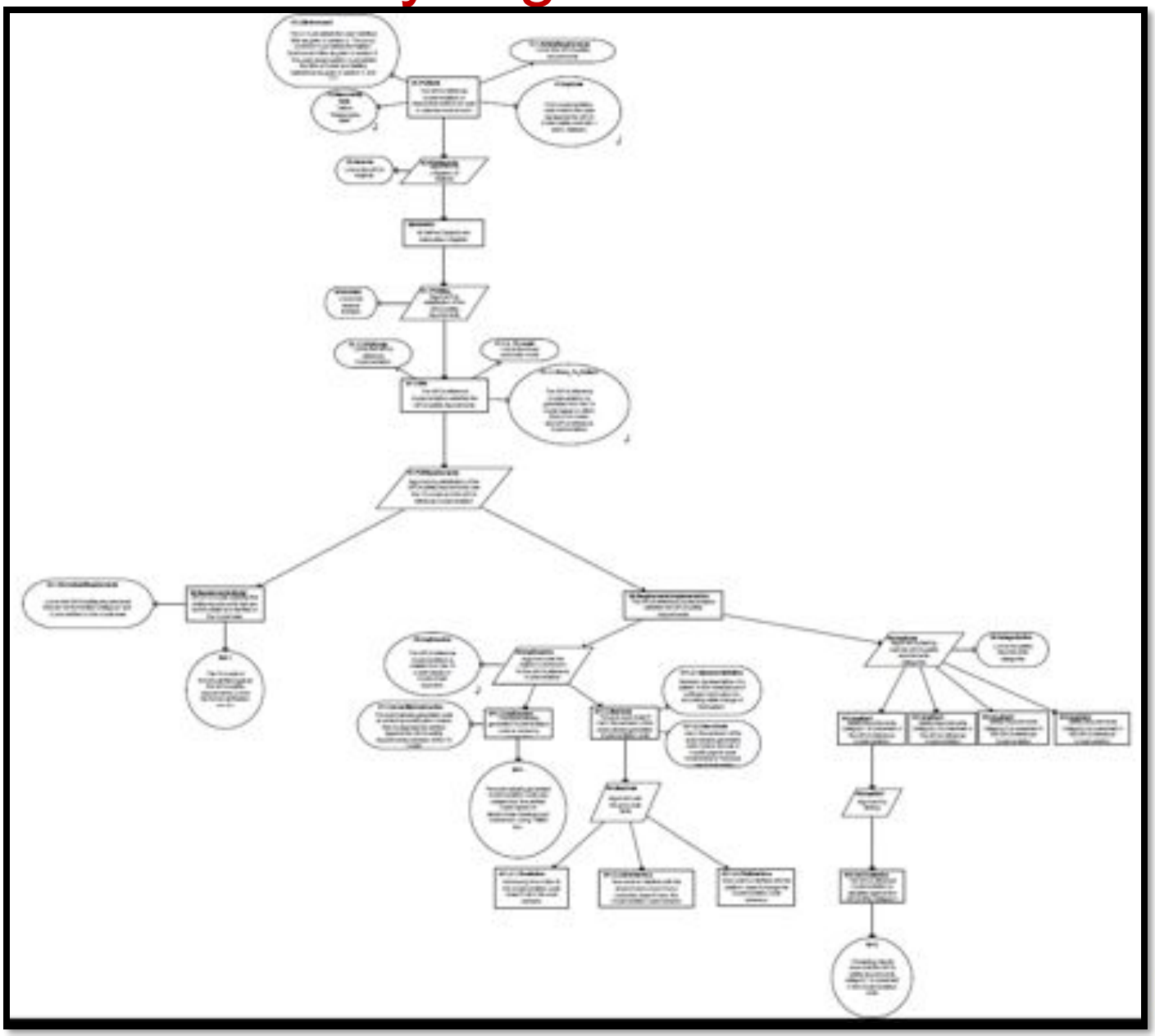
- Refine and complete the development…
  - Extend requirements to include security & privacy requirements
- Identify generic-platform dependent & specific-platform dependent glue code
  - How much need to be redone with a different pump hardware
- Assurance/safety cases for the GPCA reference implementation
  - Mock FDA submission

Assurance Cases

- To construct an assurance case we need to:
  - make an explicit set of claims about the system
  - produce the supporting evidence
  - provide a set of arguments that link the claims to the evidence
  - make clear the assumptions and judgments underlying the arguments
- Safety case is a special kind:
  - Claims are limited to safety



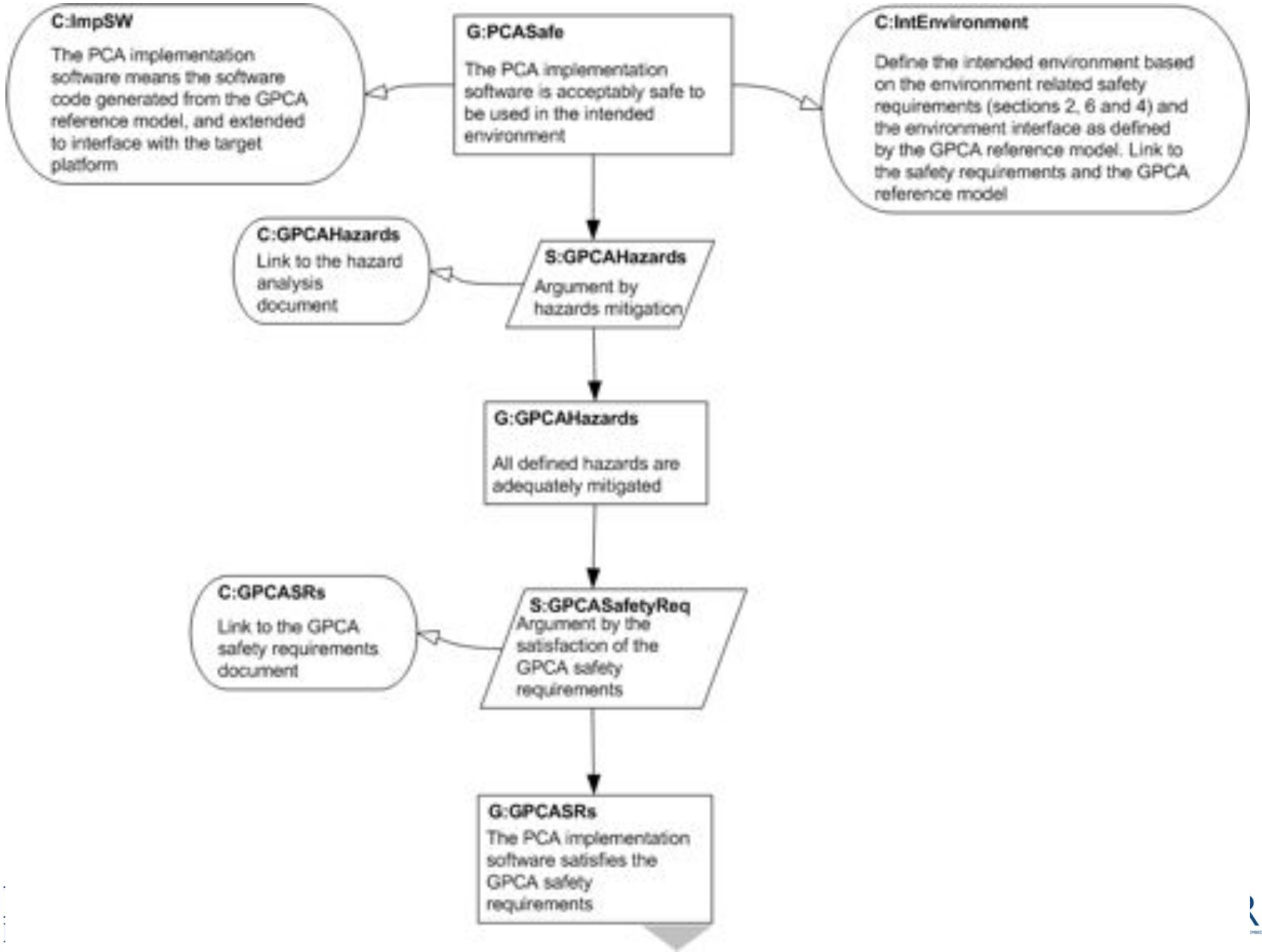
The GPCA Safety Argument



Penn Engineering

PRECISE
PENN RESEARCH IN EMBEDDED COMPUTING AND INTEGRATED SYSTEMS ENGINEERING

The GPCA Safety Argument

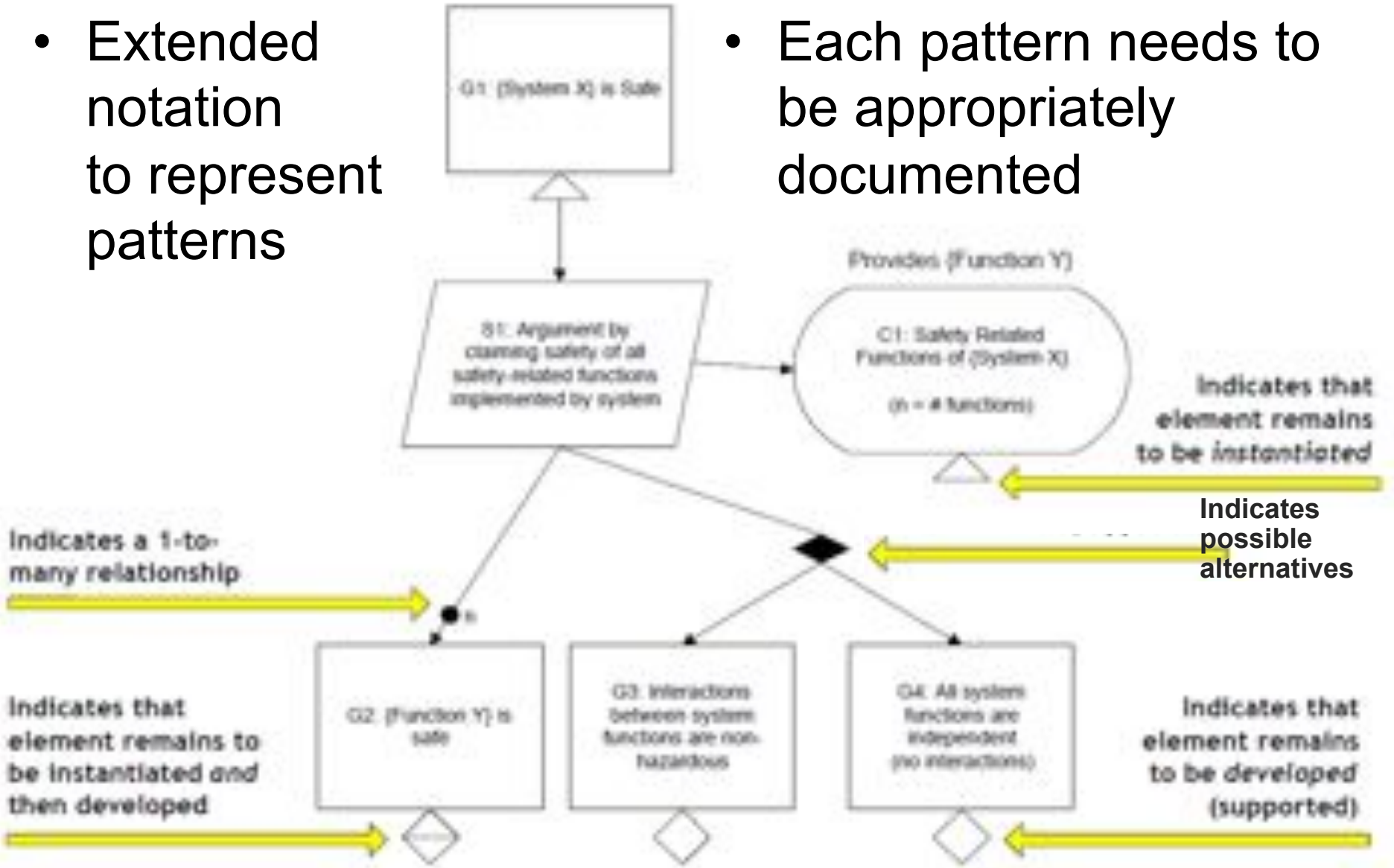


Current Work

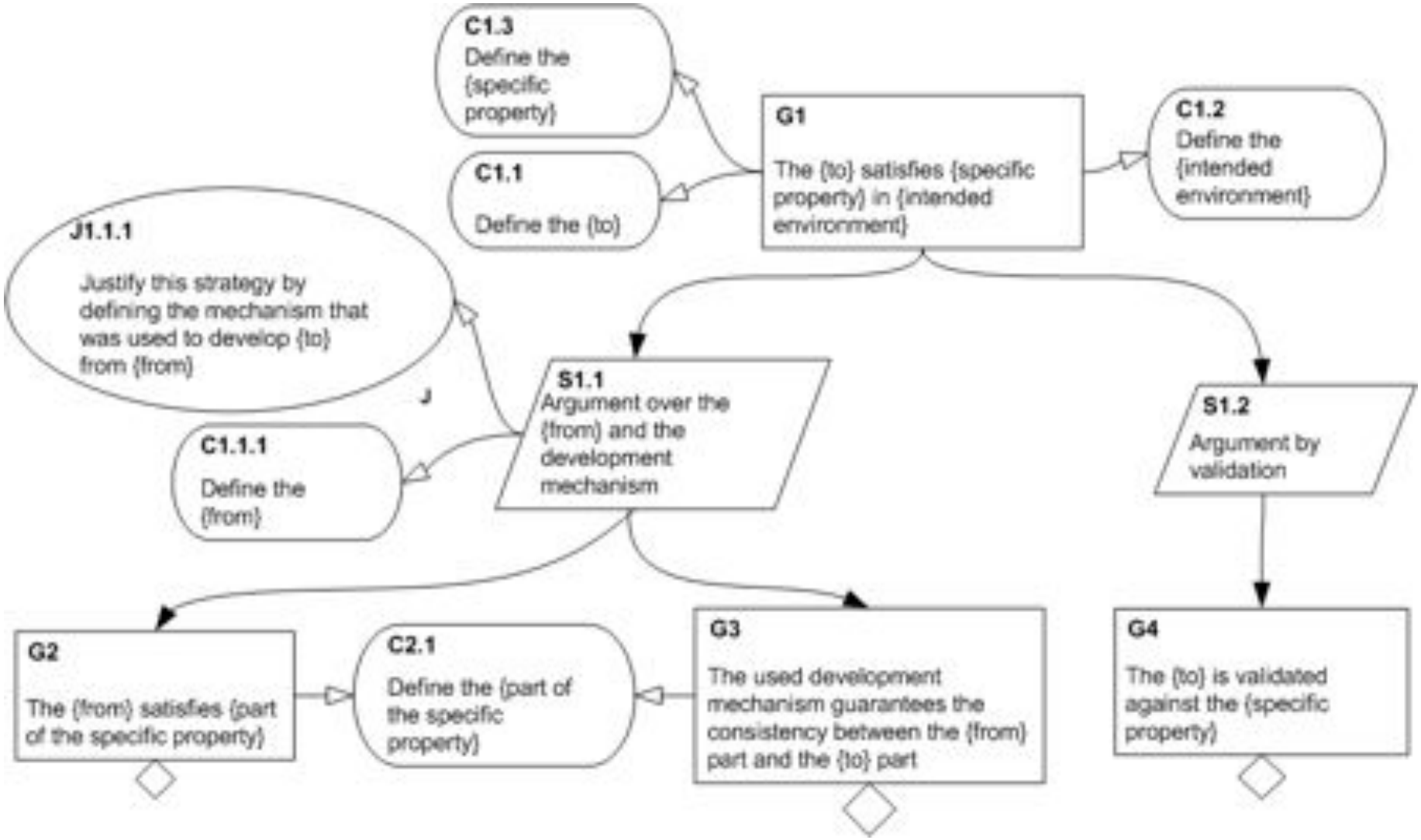
- Define a pattern for model-driven development approaches (MDD pattern)
- How to identify gaps in (GPCA) assurance cases
- How to evaluate (GPCA) assurance cases

Assurance Case Patterns

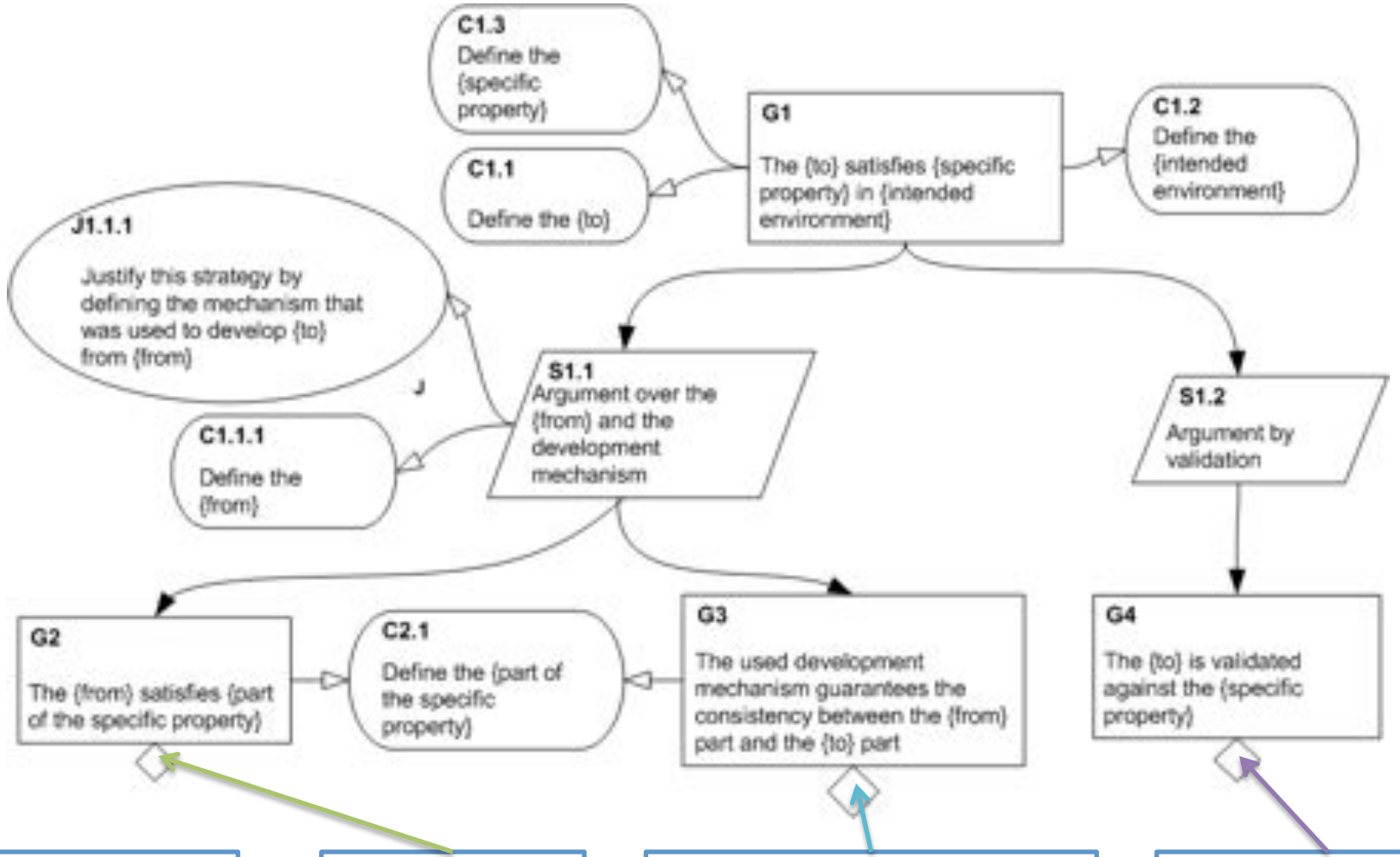
- Extended notation to represent patterns
- Each pattern needs to be appropriately documented



MDD pattern (from-to pattern)

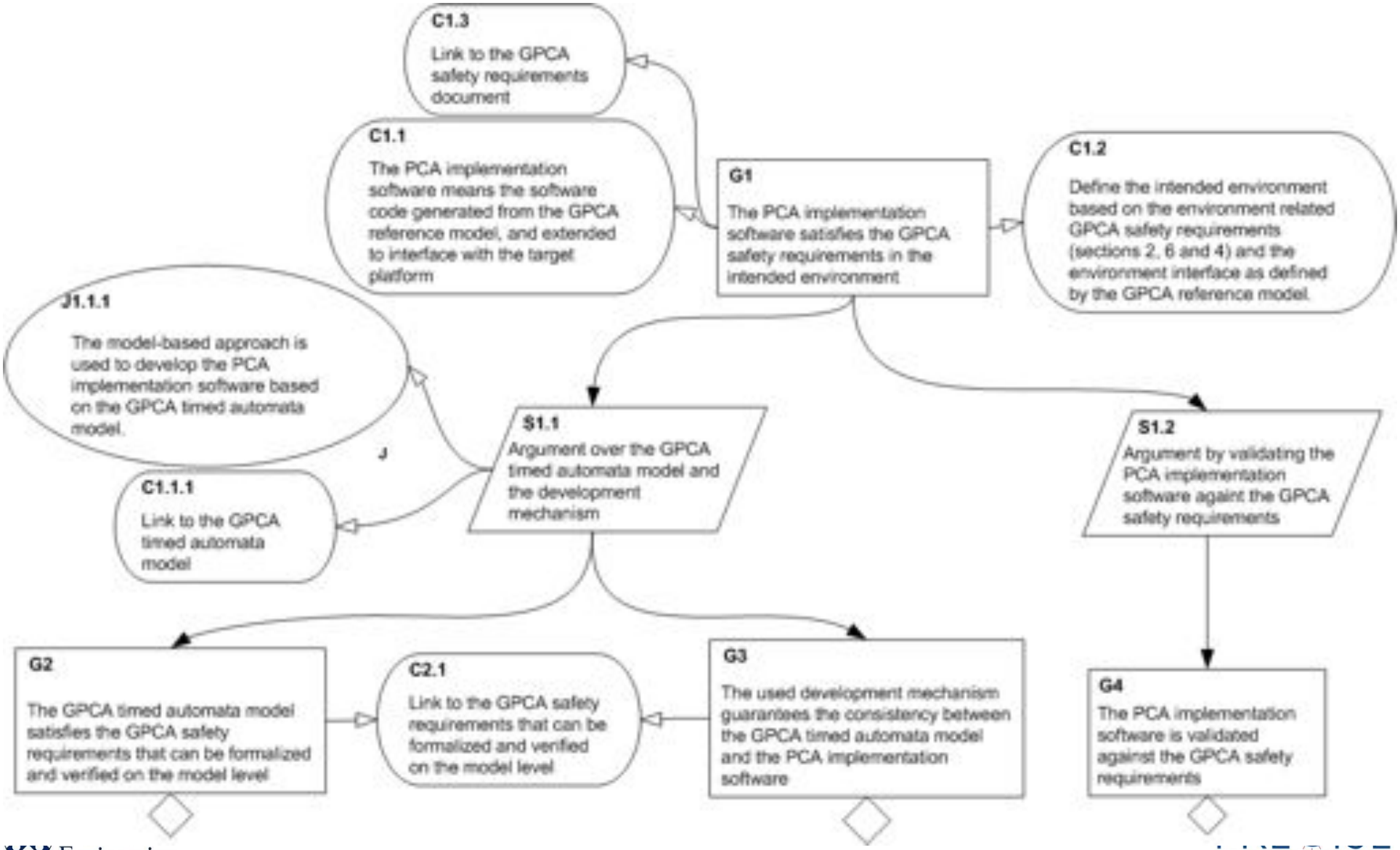


Mapping the Model-Based Approach to the MDD pattern



(1) modeling the system → (2) verifying this model → (3) transformation the model into an implementation → (4) validating the implementation

The PCA Safety Case – Instance of the MDD pattern



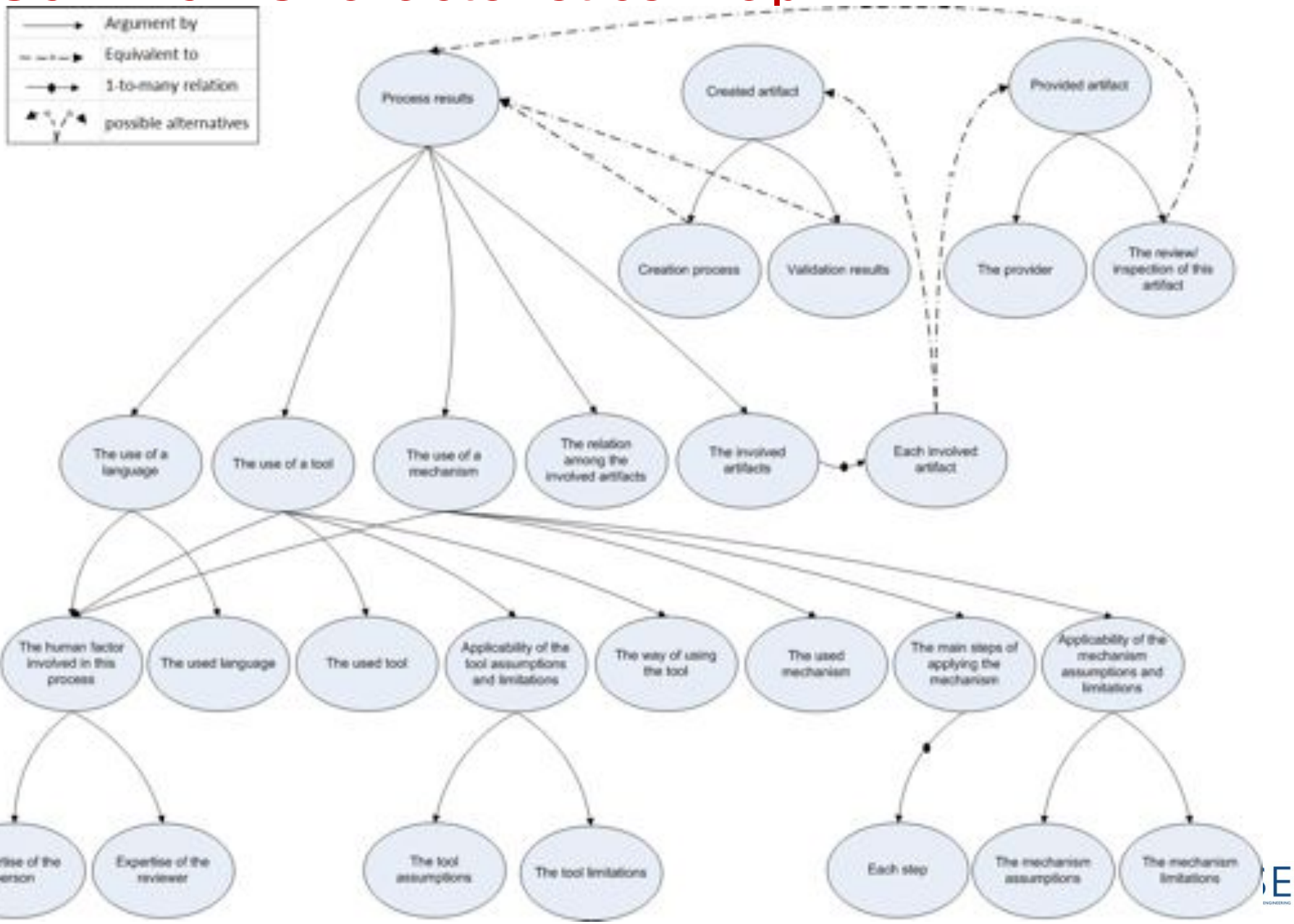
Confidence Arguments

- Separate safety argument from confidence argument
- Safety argument
  - Reasoning about safety of the system
    - E.g.: why this hazard sufficiently unlikely to occur? Does the testing results show that?
- Confidence argument
  - Reasoning about confidence in safety argument, assumptions, evidence
    - E.g.: is that testing exhaustive? Is there sufficient confidence in the testing? Is the model checking tool trustworthy?

Confidence Arguments Construction

- We need a mechanism to
  - Systematically construct confidence arguments
  - Identify safety gaps (assurance deficits)
- Generalize experience from GPCA case study
  - Identify common characteristics of concepts needed in confidence argument
  - Summarize relationship between the concepts in a map
    - We target trustworthiness
    - Another aspect is appropriateness, which can be handled similarly

Common Characteristics Map



CCMap Instantiation

C:TAmodel

Link to the GPCA timed automata model

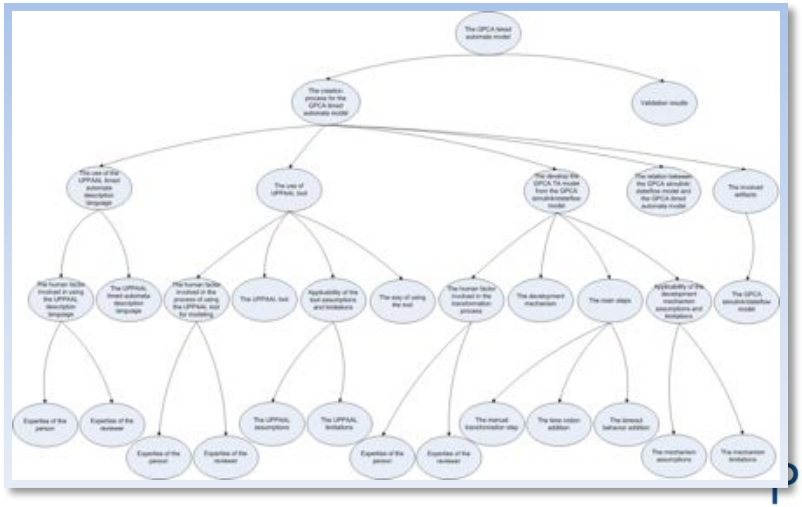
- Given a safety argument element
  - pick the corresponding map node
  - Unroll the map
  - Find affected map nodes, repeat

The common characteristics map

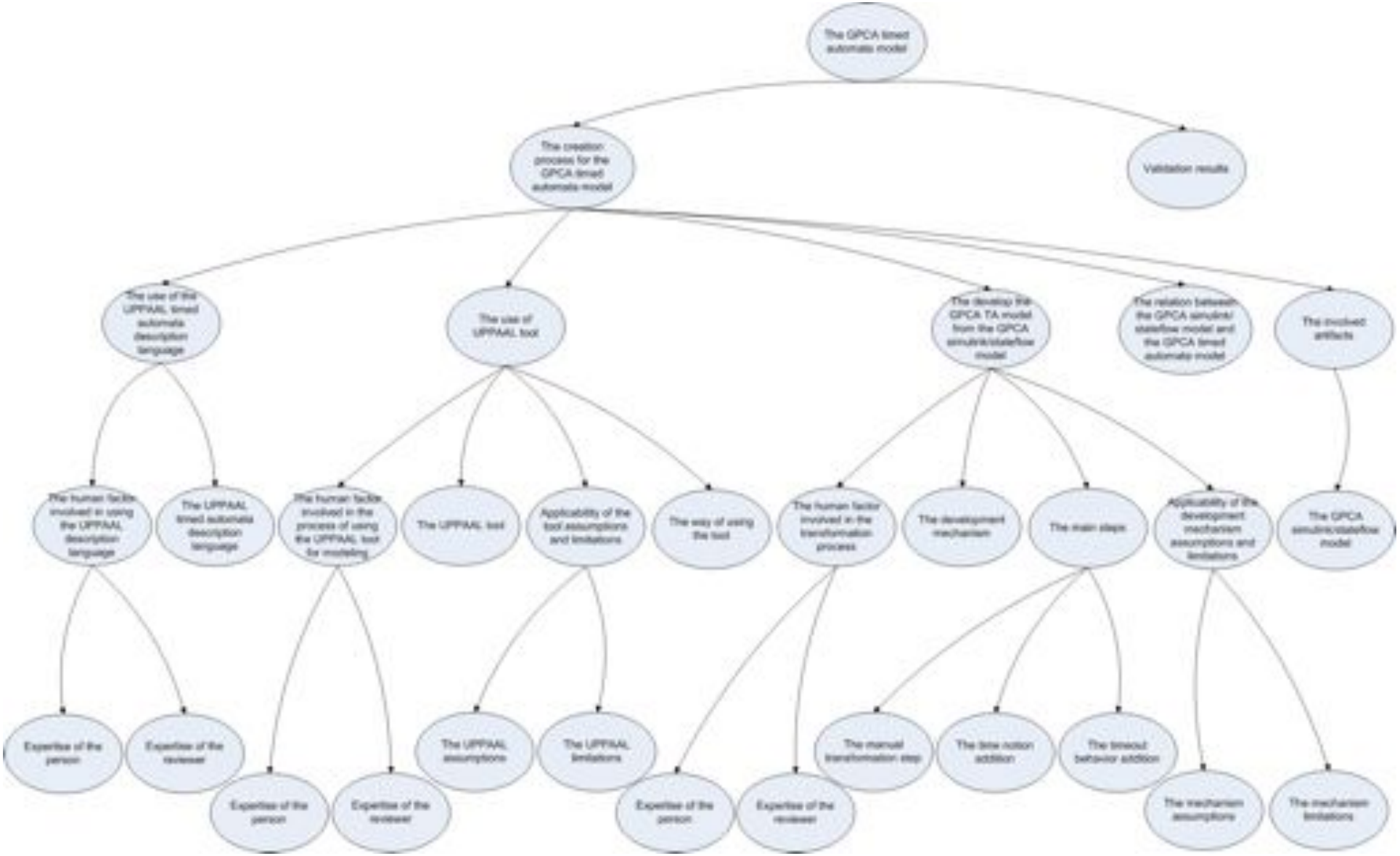
The Safety Argument Context or Evidence

AND

Instantiate the common characteristics map



Instantiated CCMap

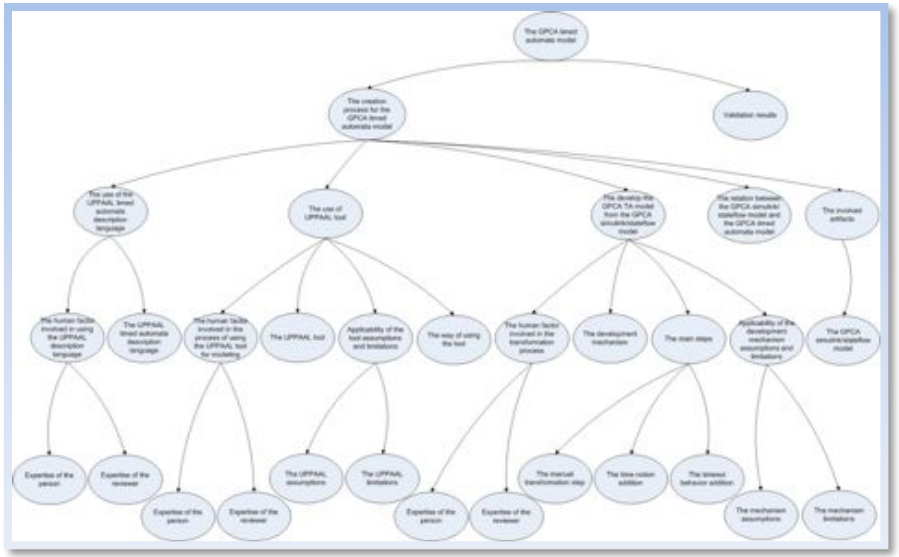


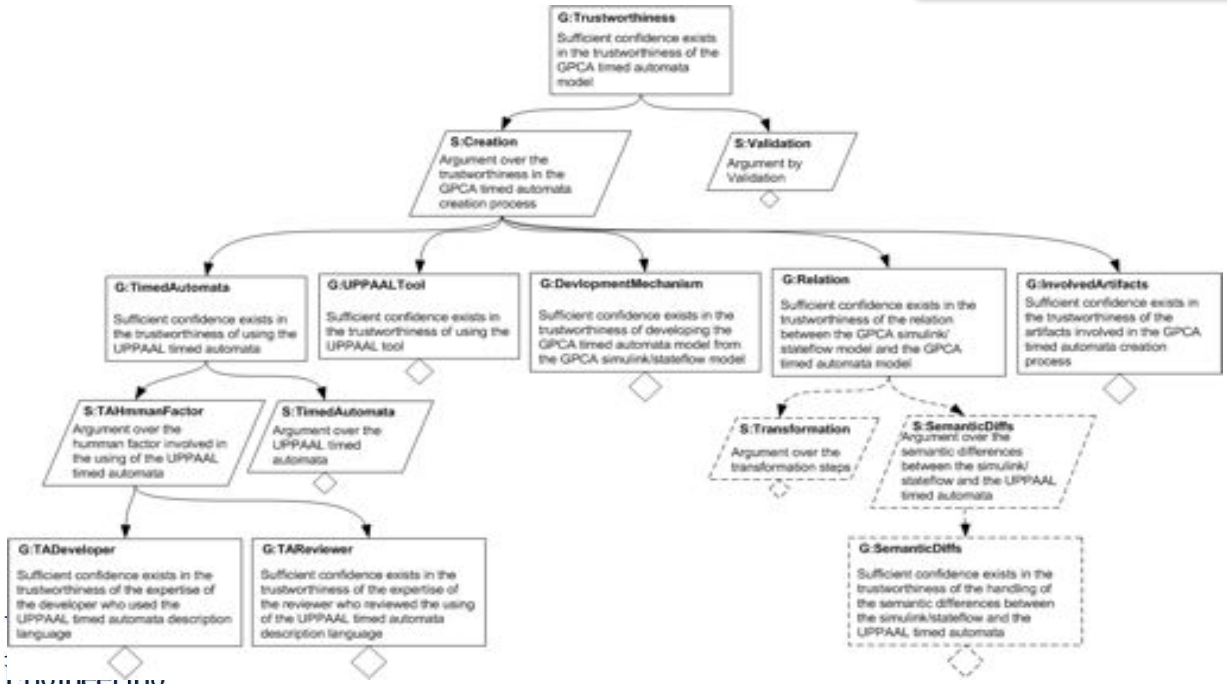
Penn Engineering

PRECISE
PENN RESEARCH IN EMBEDDED COMPUTING AND INTEGRATED SYSTEMS ENGINEERING

Generate Confidence Argument

- Near-isomorphic structure



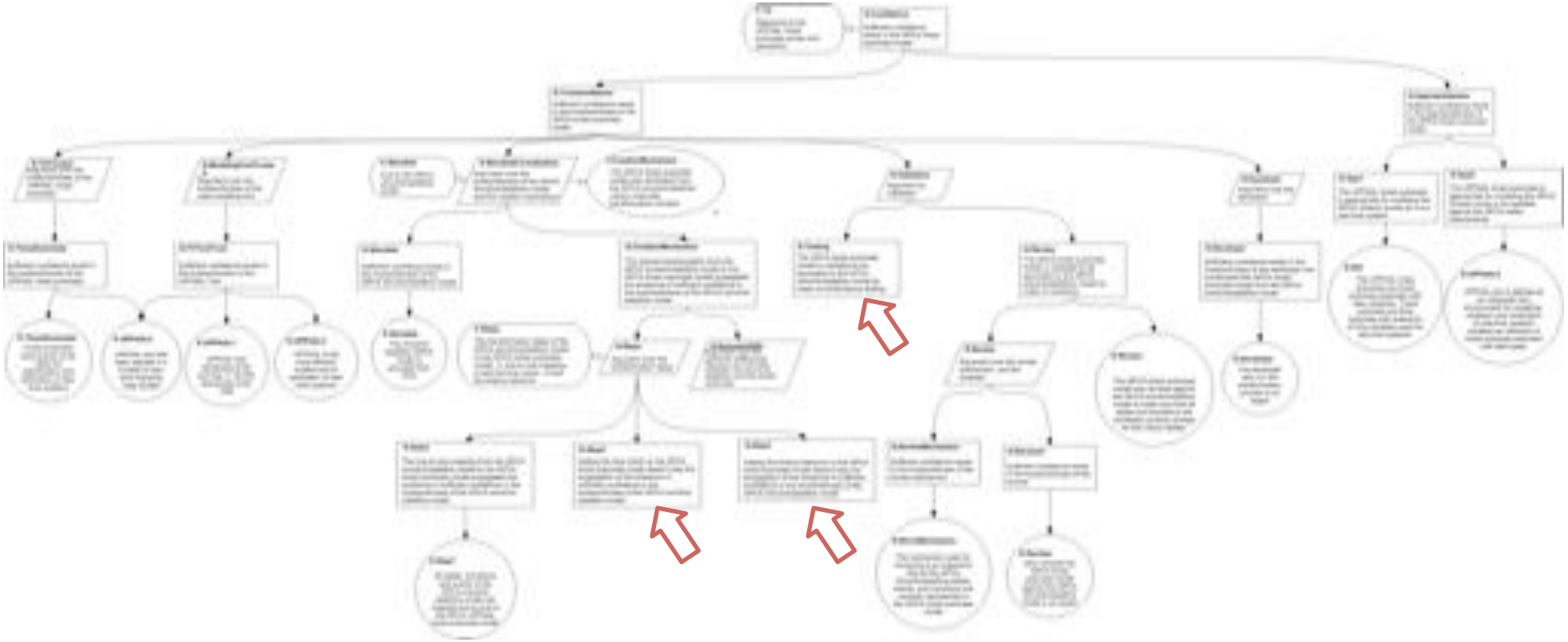


Engineering

PRECISE
PENN RESEARCH IN EMBEDDED COMPUTING AND INTEGRATED SYSTEMS ENGINEERING

Identify safety gaps

- Look for branches that do not end with evidence nodes



Evaluate the Safety Argument

- Assurance cases are, by their nature, often subjective.
- One of the purposes of assurance case development, therefore, is to facilitate mutual acceptance of this subjective position.
- The goal of assurance case evaluation, therefore, is to assess if there is a mutual acceptance of the subjective position.

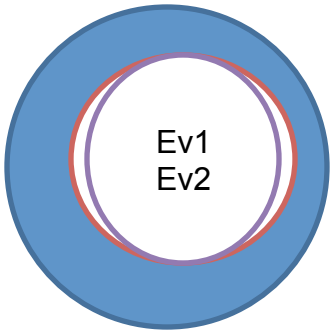
- Need an approach/method
  - Experts should only be required to express their opinions about the basic elements in argument structures (e.g., assumptions, evidences)
  - A systematic mechanism should provide a way to aggregate the opinions to communicate a message about the argument overall sufficiency.

Evaluate the Safety Argument

- The proposed method consists of two steps
  - Step 1: Assign degree of belief in the sufficiency and insufficiency of the basic elements of the argument
  - Step 2: Aggregation
    - Starting from the leaves,
      - aggregate the degree of beliefs in the sufficiency/insufficiency in the premises (e.g., the evidence)
      - to obtain the degree of belief in the sufficiency/insufficiency in the conclusion (i.e., the goal).
    - Repeat the process until the top-level goal has been reached

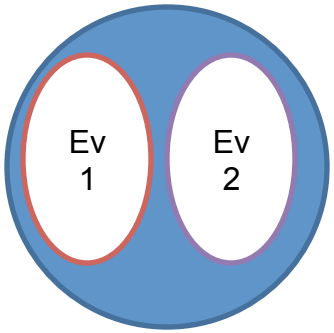
How to evaluate Safety Argument

- The argumentation type



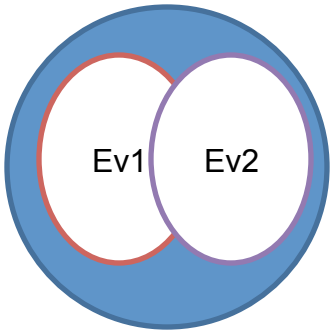
Case #1

Alternative



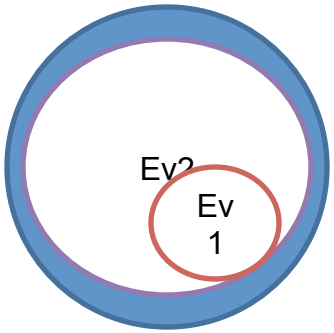
Case #2

Disjoint



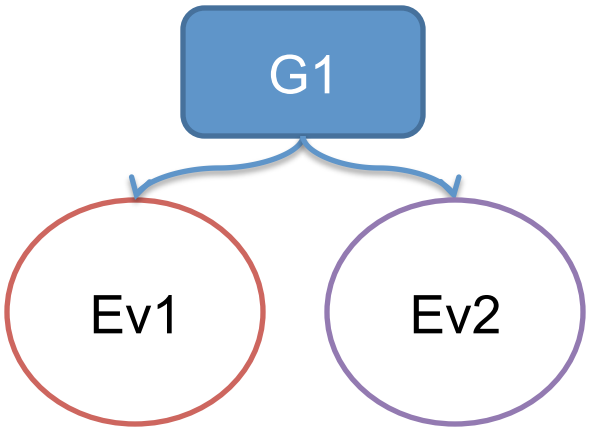
Case #3

Overlap



Case #4

Containment



Penn Engineering

PRECISE

Medical-Device Plug-and-Play

Characteristics

- Medical devices gaining communication capabilities
- Devices still operate independently
- Standardized interaction between devices non existent
- Full benefit of communication capabilities not being realized

Current

Future

MD PnP: Interoperable medical devices based on plug-n-play!

Vendor neutrality based on open medical device interfaces

www.mdpnp.org

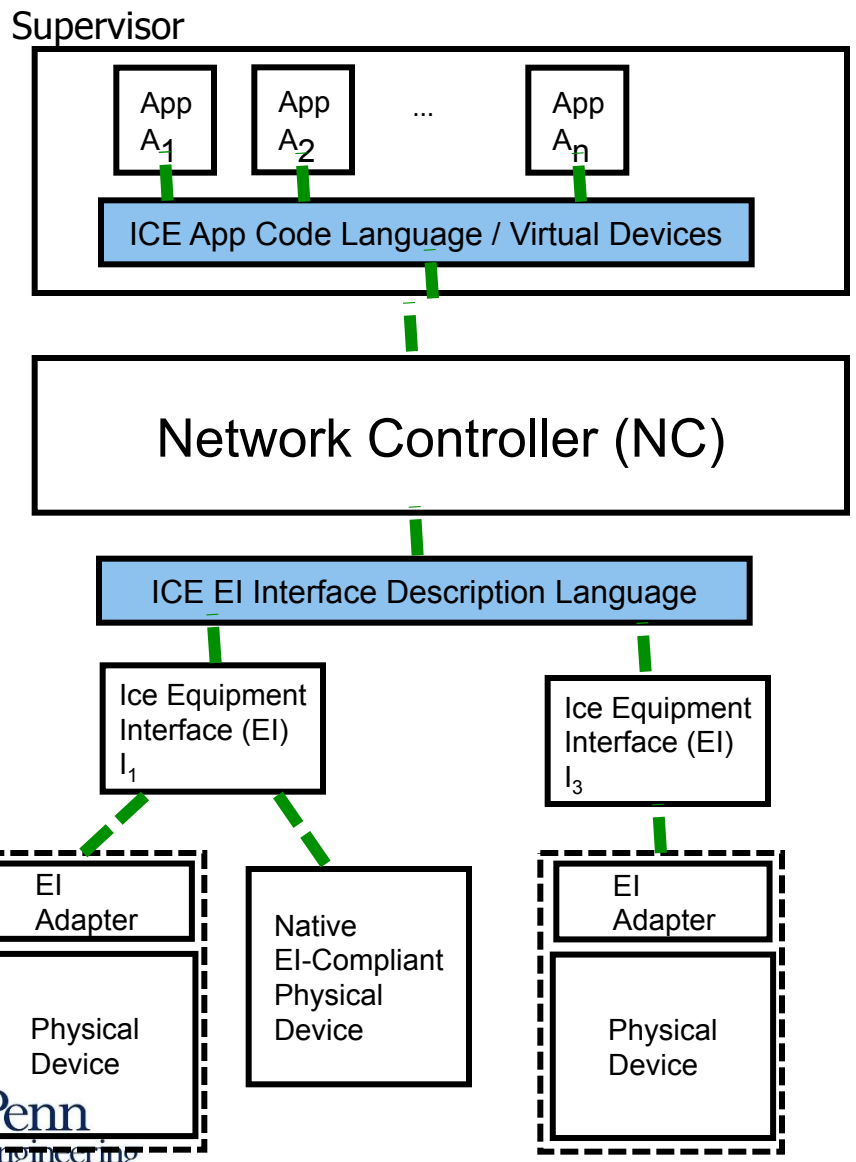
Advantages

- Improve Patient safety
- Safety interlocks
- Complete, accurate medical records
- Reduce errors
- Context awareness
- Rapid deployment

Penn Engineering

PRECISE

Integrated Clinical Environment (ICE)



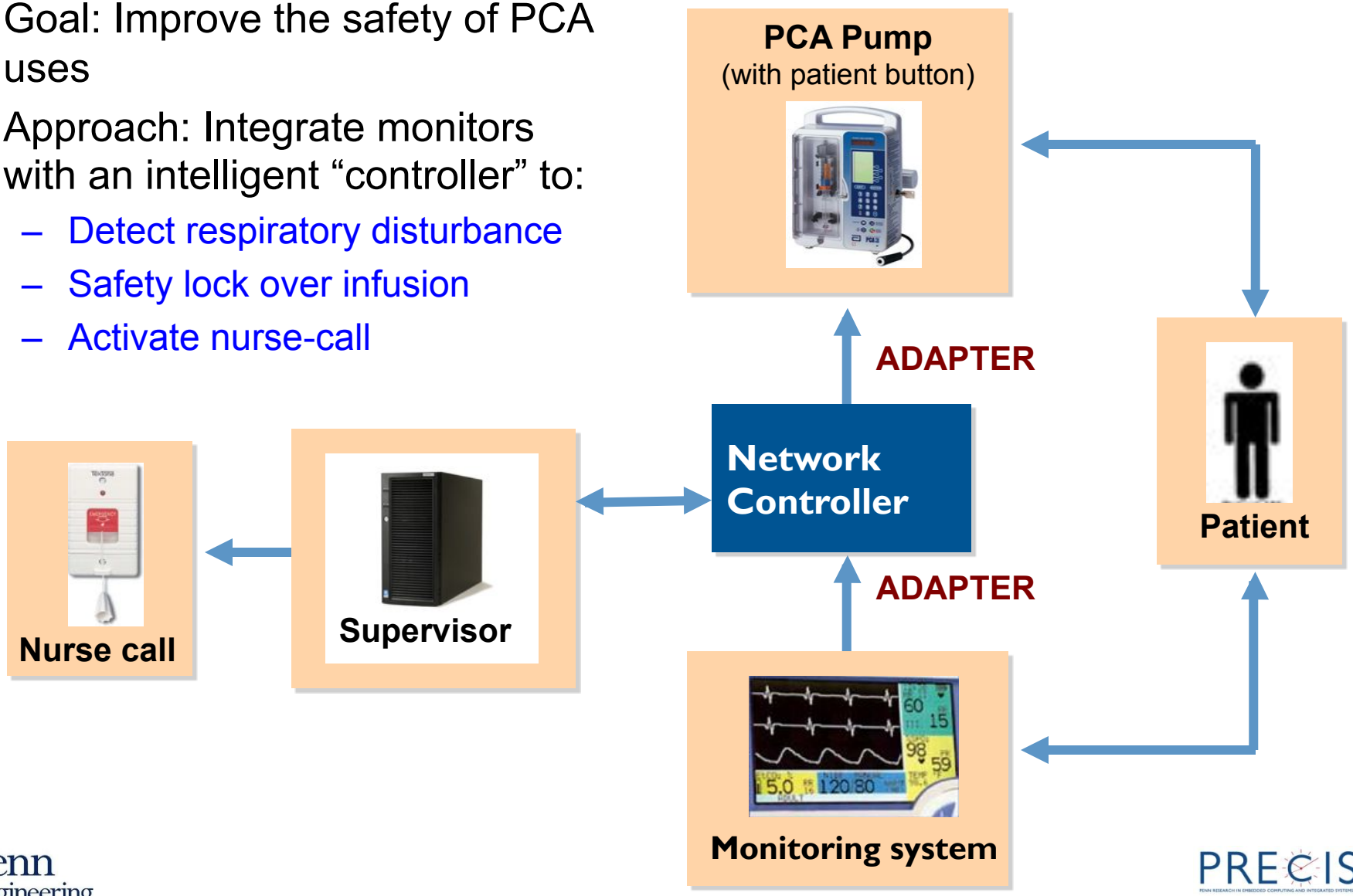
- ASTM Standard F2761-2009 for ICE defines a high-level architecture and functional concept
- Subsequent standards are intended to provide specific functional and interfacing requirements for components
- The ICE architecture standard is the focal point for FDA's evaluation of MAP concepts in future medical systems
  - A key element of this evaluation is moving from regulation of "systems as a whole" to component-wise regulation

Penn Engineering

PRECISE

PCA Closed-loop System

- Goal: Improve the safety of PCA uses
- Approach: Integrate monitors with an intelligent "controller" to:
  - Detect respiratory disturbance
  - Safety lock over infusion
  - Activate nurse-call



Penn Engineering

PRECISE

Virtual Medical Devices (VMD)

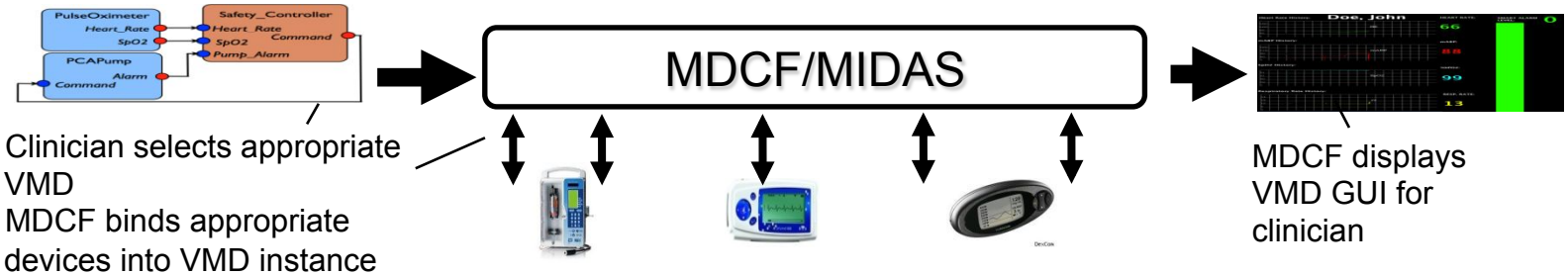
- **MD PnP** enables the concept of **Virtual Medical Devices:**
  - A set of medical devices coordinating over a network for clinical scenario.

Device Coordination Algorithm

Medical Device Types

Virtual Medical Device (VMD)

- VMD does not physically exist until instantiated at a hospital.
- The Medical Device Coordination Framework (MDCF) is prototype middleware for managing the correct composition of medical devices into VMD.



- Clinician selects appropriate VMD
- MDCF binds appropriate devices into VMD instance

MDCF displays VMD GUI for clinician

Penn Engineering

PRECISE

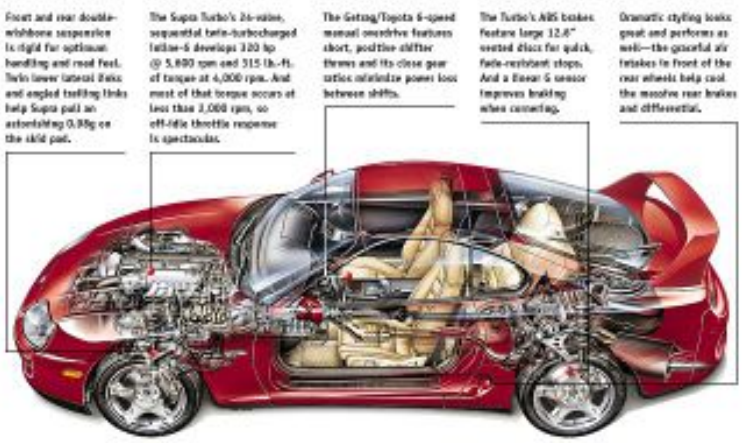
Certification of VMD App

- Safety analysis of the VMD model relies on assumptions about
  - Devices that comprise the VMD
  - Interoperability infrastructure
- Current regulatory approach:
  - Certify each instantiation of VMD app
    - fixed medical devices, network, middleware, etc.
- Alternative approach:
  - Certify VMD app based on abstract interfaces
  - Certify devices on interface satisfaction
  - System can use any certified component

Traditional safety critical systems...

Aerospace

Nuclear



Automotive

Penn Engineering

PRECISE

System Integration

In other safety critical domains, there is a typically a prime contractor that is responsible for integration and system-level verification and validation.

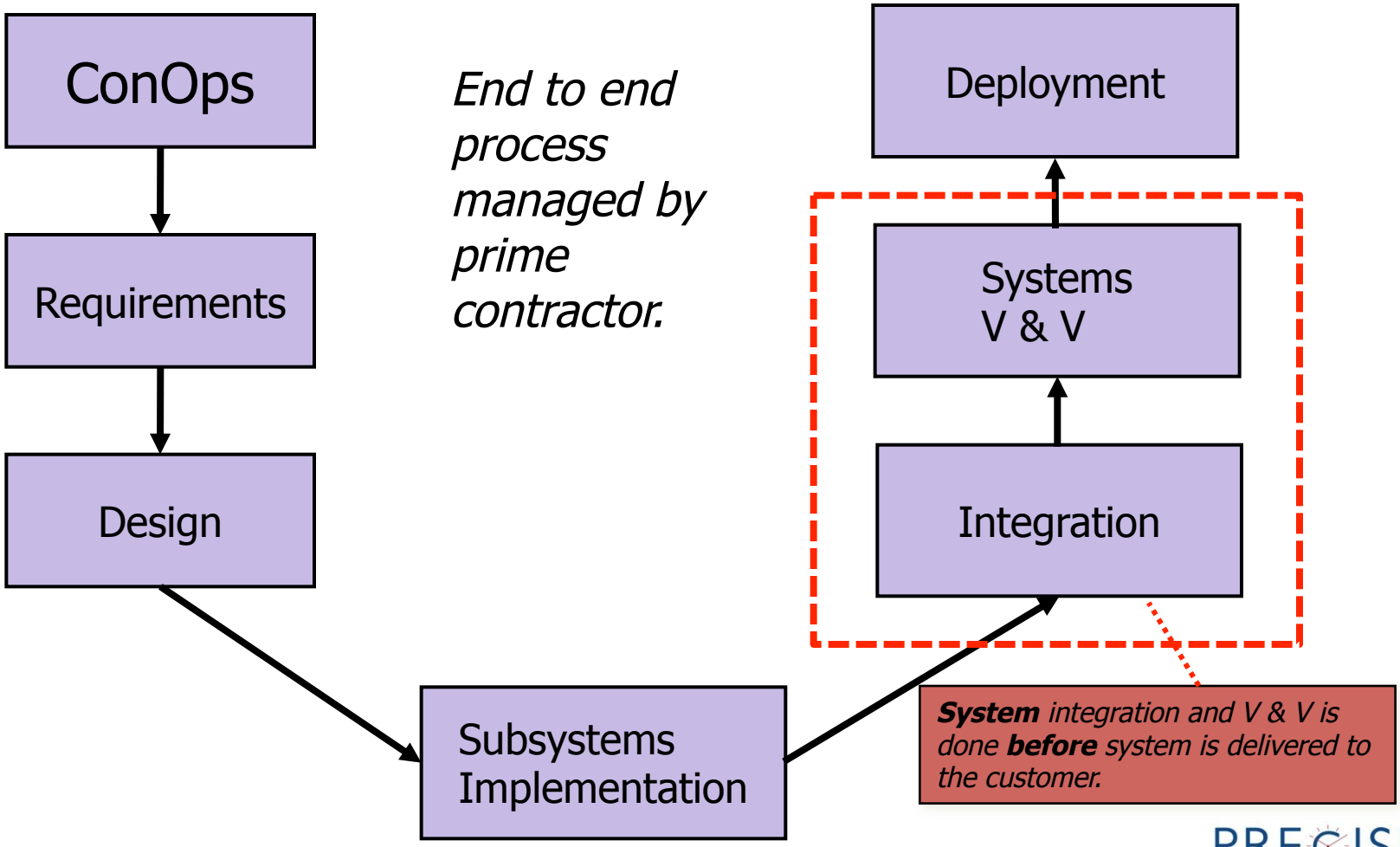
- Integration is performed *before* deployment with full knowledge and behavior of components being integrated
- Integrator has expert-level technical knowledge of components & system behavior
- Responsible for overall system
  - Verification & Validation
  - Safety arguments
  - Certification

787 Final Assembly Integrator - The Boeing Company

As Prime Contractor/Integrator for the final assembly of the composite 787 Dreamliner in Everett, WA,

System Integration

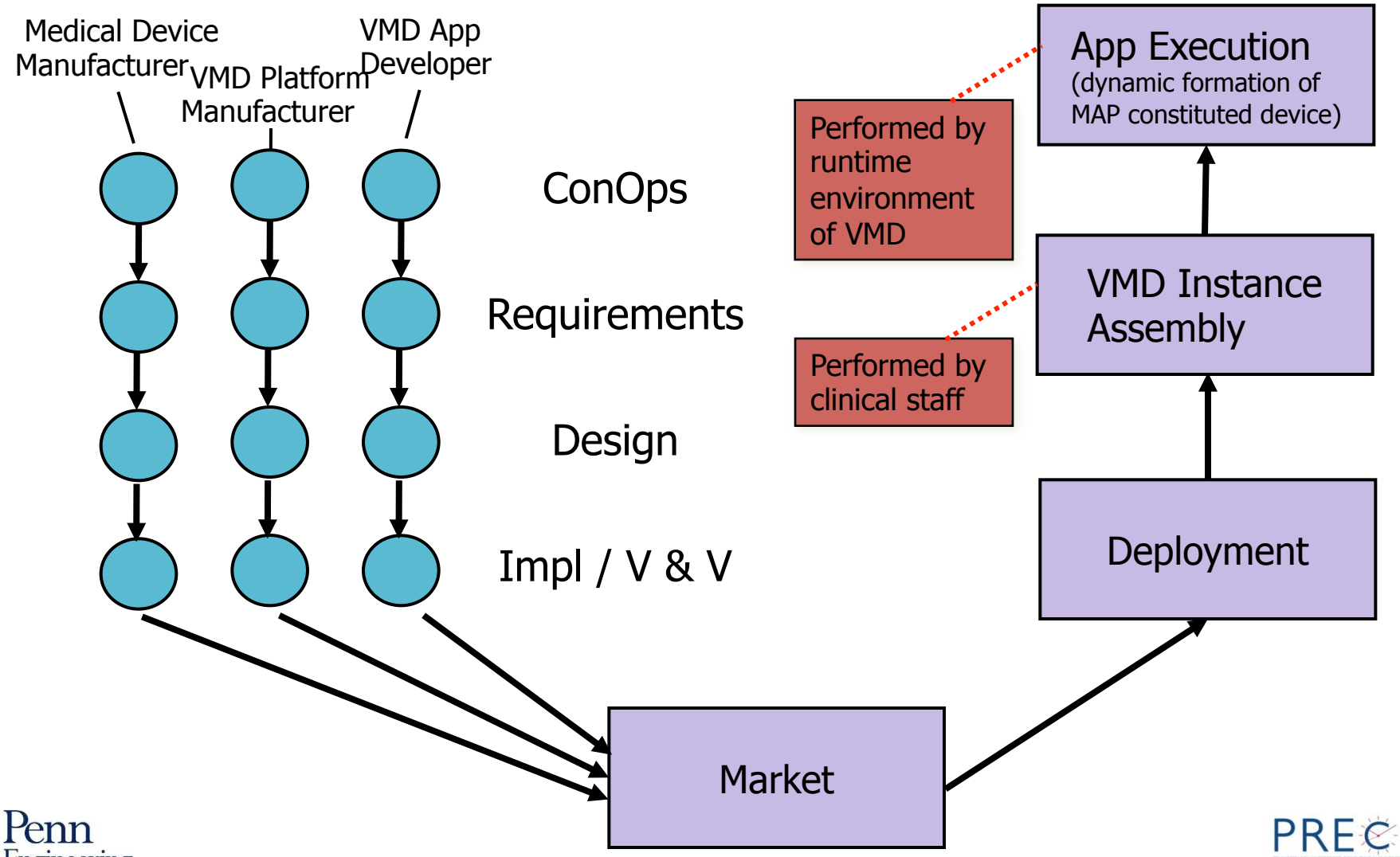
In other safety critical domains, there is a typically a prime contractor that is responsible for integration and system-level verification and validation.



Penn Engineering

PRECISE

VMD Development & Assembly



VMD Characteristics

In other safety critical domains, there is a typically a prime contractor that is responsible for integration and system-level verification and validation.

- Integration is performed *before* deployment with full knowledge and behavior of components being integrated
- Integrator has expert-level technical knowledge of components & system behavior
- Responsible for overall system
  - Verification & Validation
  - Safety arguments
  - Certification

With VMDs, there is **no** prime contractor that is responsible for integration and system-level verification and validation.

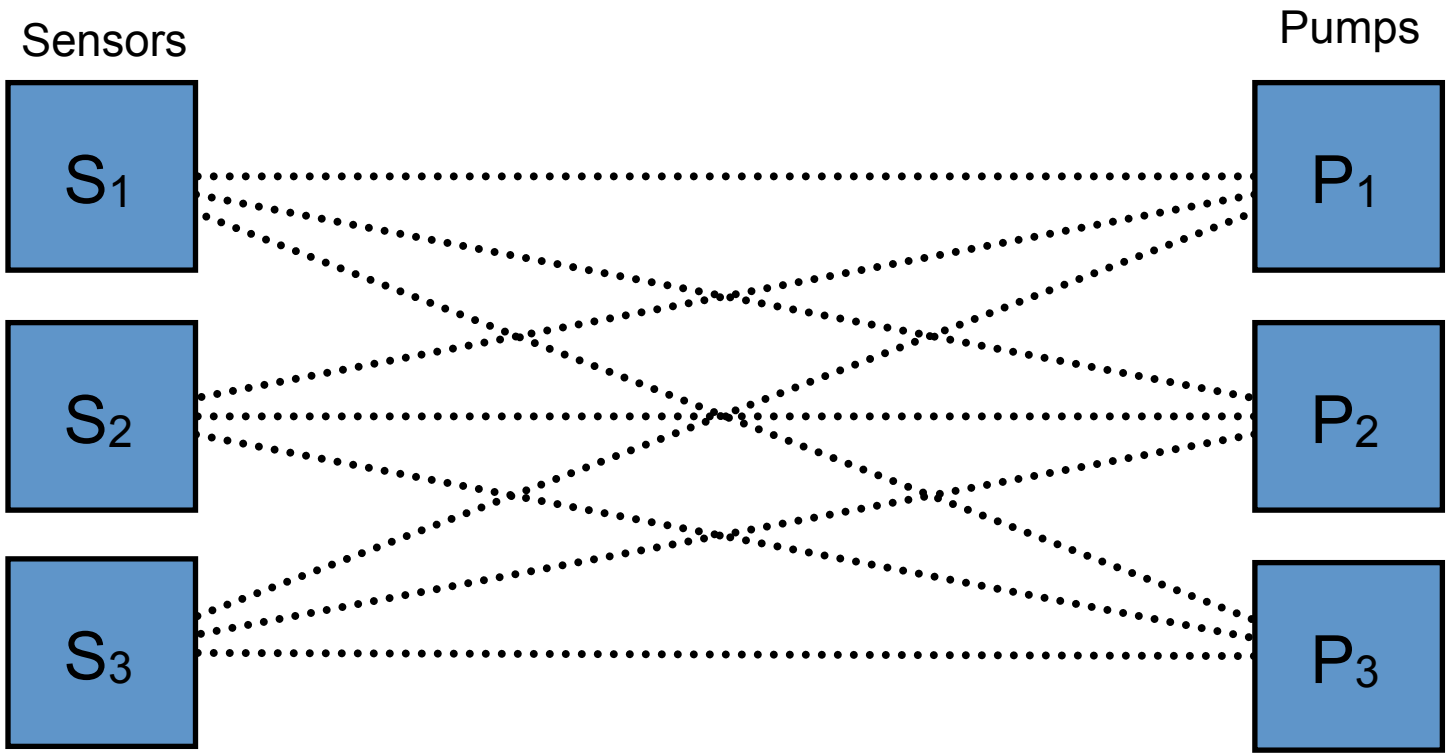
- Assembly is performed *after* deployment
- Assembler (hospital staff) **does not have** expert-level technical knowledge of components & system behavior
- **App developer** is responsible for overall system safety arguments
- Platform services (compatibility checks) assist in determining **at app launch time** if platform and attached devices satisfy requirements of app
- The app's directions for assembly of the platform constituted device are stated **only in terms of properties/ capabilities that are exposed on the interfaces** of the platform and devices.

Regulatory Process

- "pair-wise" approval
  - Approve every possible permutation of devices forming a composite medical system
  - It is simply not viable
- "component-wise" approval
  - Approve each system component
    - "component x is safe for its intended use in its intended use environment"
    - Part of component x's intended use is to interacts with other components according to *their* intended use

Pairwise Approval / Certification

Example "interoperable" device ecosystem 3 different (model/manufacturer) SpO2 monitors, 3 different (model/manufacturer) PCA infusion pumps:

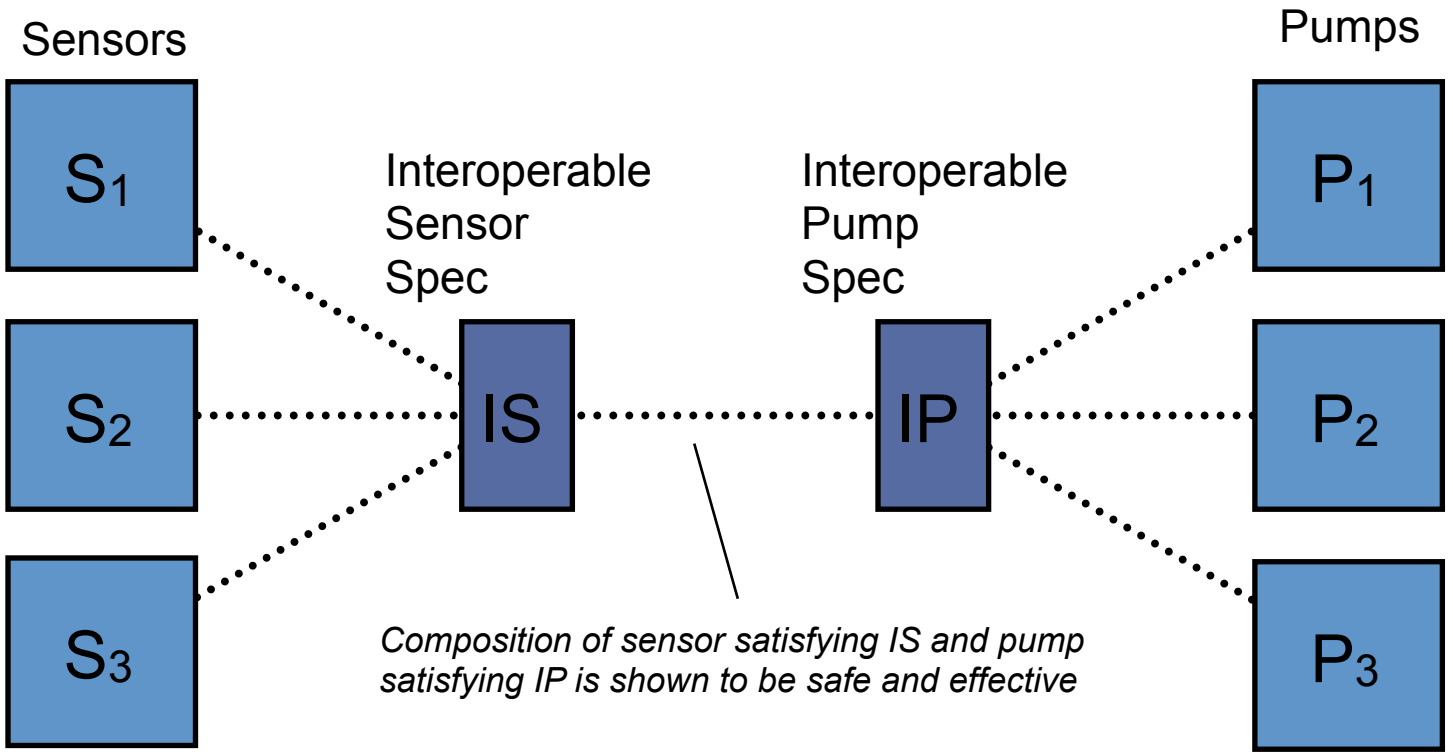


Each sensor must be approved or certified for use with each pump and vice versa. This is burdensome for manufacturers and regulators

Certification or approval relationship

Interface-based approval / certification

Example "interoperable" device ecosystem 3 different (model/manufacturer) SpO2 monitors, 3 different (model/manufacturer) PCA infusion pumps:

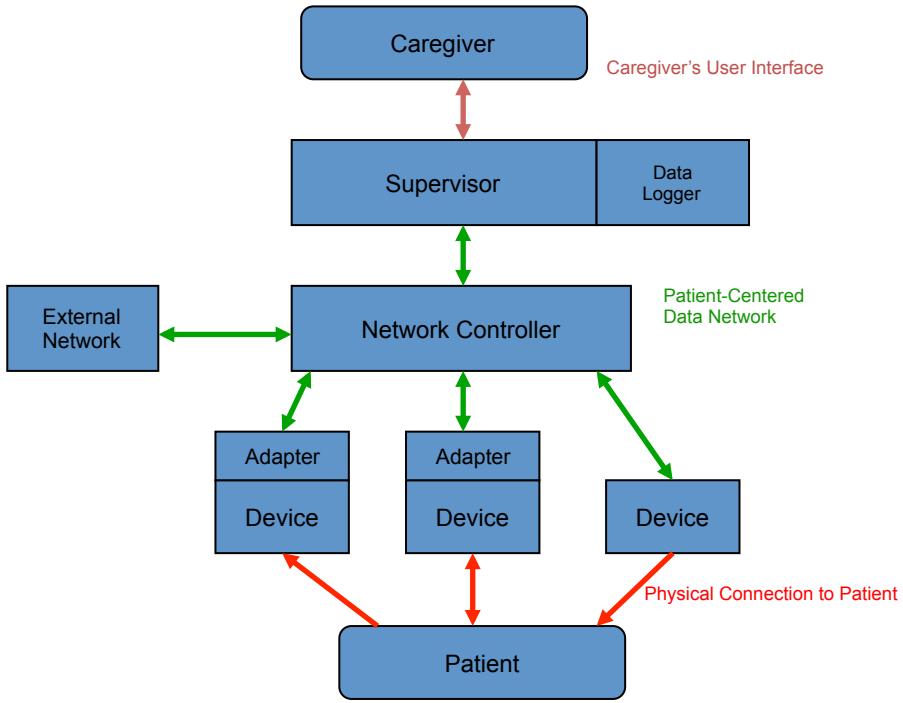


Each sensor (or pump) only needs certification or approval w.r.t. the interface spec. Additionally, the ecosystem can grow without forcing recertification (or re-approval) of previously analyzed devices

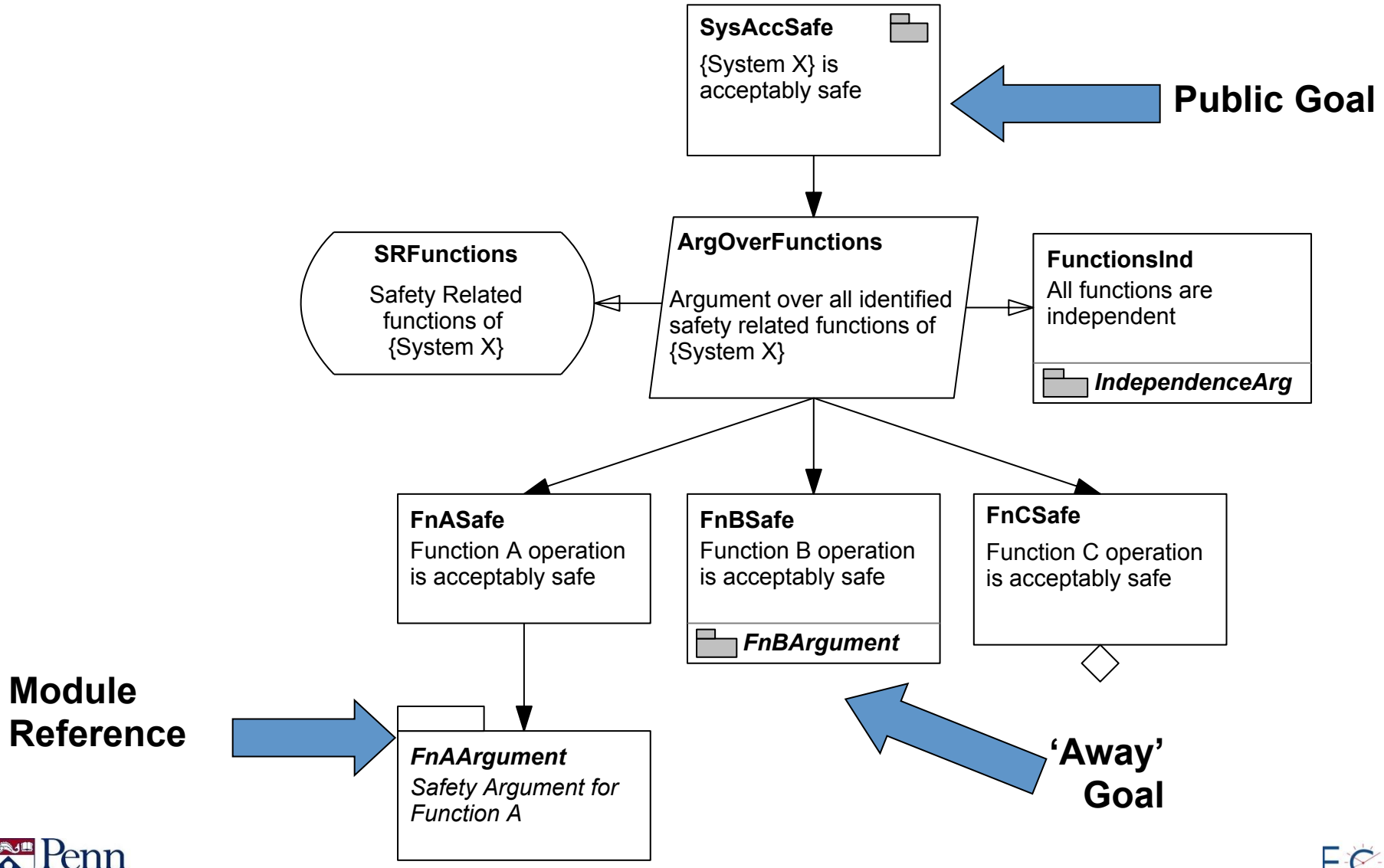
..... Certification or approval relationship

Modular Assurance Case

- The assurance case for a system of systems would be an assurance case of assurance cases (i.e., tree of trees)



Modular Assurance Case --Example

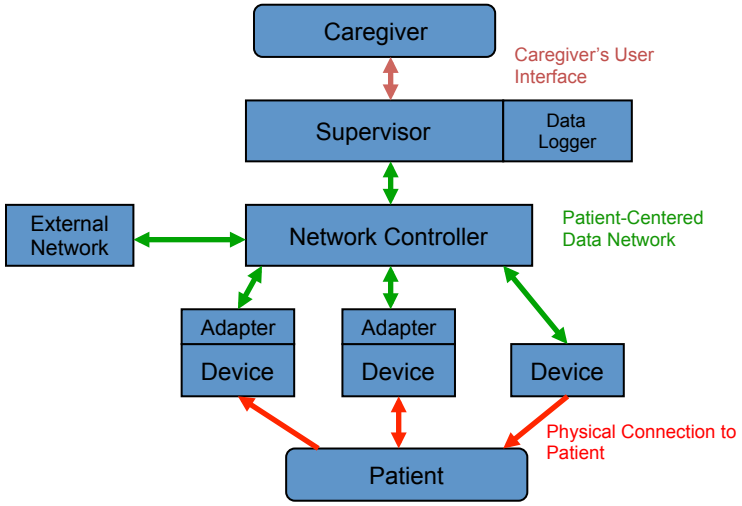


Penn Engineering

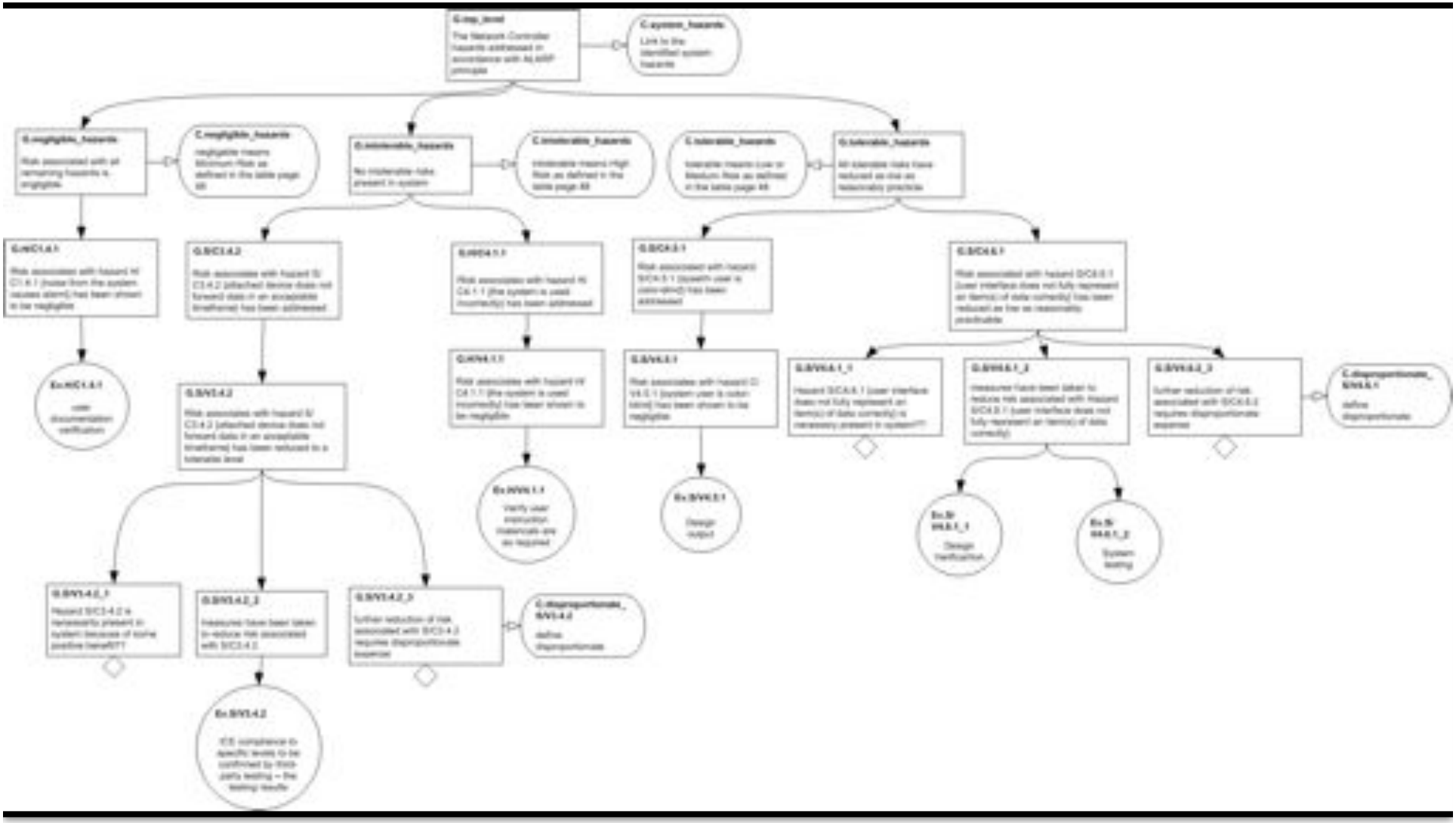
PRECISE

Modular Assurance Case

- An assurance case for the supervisor
- An assurance case for the Network Controller
- An assurance case for each device
- An assurance case for each virtual medical device (VDM) app
  - It is safe
  - It is compliant with VDM interfaces



The network controller safety case

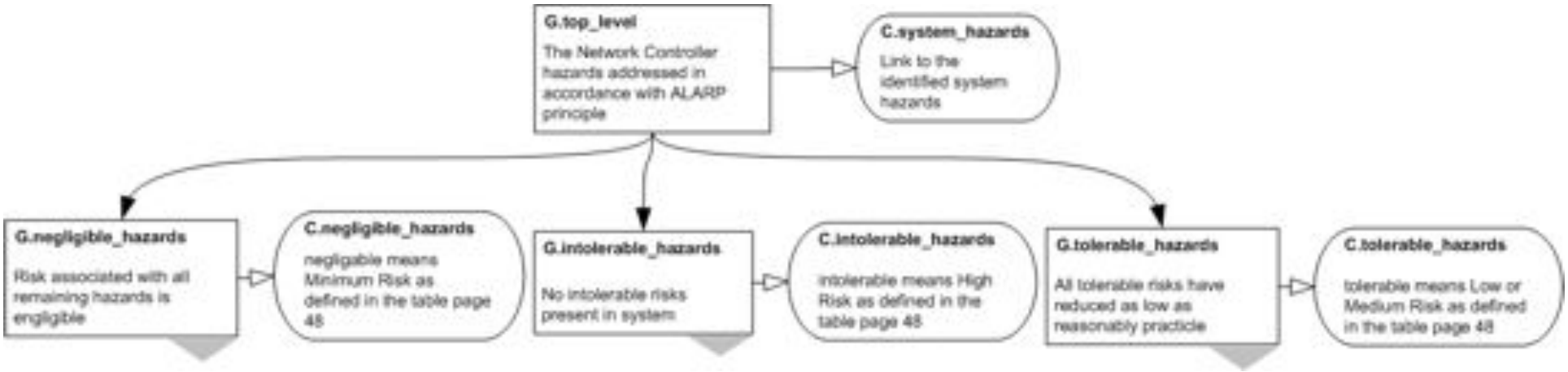


Penn Engineering

PRECISE
PENN RESEARCH IN EMBEDDED COMPUTING AND INTEGRATED SYSTEMS ENGINEERING

The network controller safety case

- The top part



Summary

- In order for assurance cases to work in practice, we need to
  - Develop effective ways to construct them
  - Systematically assess the arguments
- Based on our experience with the GPCA case study
  - MDD pattern
  - The safety gaps identification
  - Evaluation mechanism
- Assurance cases for MDPnP
  - Construct a modular assurance case (assurance case of assurance cases)

MCPS Research at PRECISE Center

- High-confidence medical software systems
  - Model-based development
  - Open source reference implementation of GPCA (Generic Patient-Controlled Analgesia) infusion pump
  - Pacemaker and heart modeling and analysis
  - Mental models
- Medical device interoperability
  - Security and Privacy
- Smart alarms & clinical decision support
- Physiological closed-loop systems
  - Safe controllers
- Assurance and Certification
  - Evidence-based certification
  - Blackbox recorder for medical device

MCPS Team Members

- Penn, SEAS
  - Insup Lee (PI)
  - Rajeev Alur
  - Rahul Mangharam
  - George Pappas
  - Rita Powell
  - Oleg Sokolsky
- Penn, UPHS/SoM
  - William Hanson, III, MD
  - Margaret Mullen-Fortino, RN
  - Soojin Park, MD
  - Victoria Rich, RN, PhD
- Penn, Sociology, SAS
  - Ross Koppel
- MGH/CIMIT
  - Julian Goldman, MD
- Minnesota
  - Mats Heimdahl
  - Nicholas Hopper
  - Yongdae Kim
  - Michael Whalen
- Waterloo
  - Sebastian Fischmeister
- Collaborators
  - John Hatcliff, KSU
  - Paul Jones, FDA
  - Sandy Weininger, FDA
  - Zhang Yi, FDA

- CPS: Large: Assuring the Safety, Security and Reliability of Medical Device Cyber Physical Systems (NSF CNS-1035715)
- NSF FDA Scholar In Residence (NSF CNS-1042829)

- Affiliated Project: Medical Device NIH/NIBIB Quantum Grant: Development of a Prototype Healthcare Intranet for Improved Health Outcomes (PI: Julian Goldman)

Acknowledgements

- Domain knowledge comes from our collaborators at the UPenn Hospital and MD PnP program at CIMIT
  - Julian Goldman, MD
  - Margaret Mullen-Fortino, RN
  - Soojin Park, MD
- VMD concepts are developed in collaboration with KSU (John Hatcliff) and MD PnP Quantum (Julian Goldman) project

- Some of the slides are courtesy of Julian Goldman, John Hatcliff, Oleg Sokolsky, and Andrew King
- Work support by NSF CPS and NIH grants

THANK YOU!

http://precise.seas.upenn.edu