

Integrating Safety and Security Requirements into Design of an Embedded System

Saad Zafar¹ and R. G. Dromey²

¹Institute for Integrated and Intelligent Systems

²Software Quality Institute

Griffith University

Australia

Topics

- Overview
- Background
- GSE design strategy
- GSE design process
- AIP case study
- Conclusion
- References
- Acknowledgements
- Questions and comments

Overview

- Modern embedded systems^[1,2,3]
 - Typically have to satisfy critical properties like safety and security
 - Must employ modeling techniques that facilitate easy and early validation and verification of critical properties
- Genetic Software Engineering (GSE)
 - Provides a platform for modeling, analyzing and integrating safety and security requirements

Background

- Isolation of safety and security requirements engineering^[4,5,6,7]
 - Safety and security analysis assumed to be performed in isolation from each other
 - Problems
 - Inadequate understanding of safety and security semantics
 - Lack of common terminology
 - Disparate processes
 - Incompleteness in requirements
 - Incompleteness in requirements characteristics like verifiability
 - Ambiguity in requirements

Background

- Safety and security

- Common characteristics^[8]

- Protect valuable assets from harm
 - Specified as constraints on system
 - System level properties
 - Typically require very high level of assurance

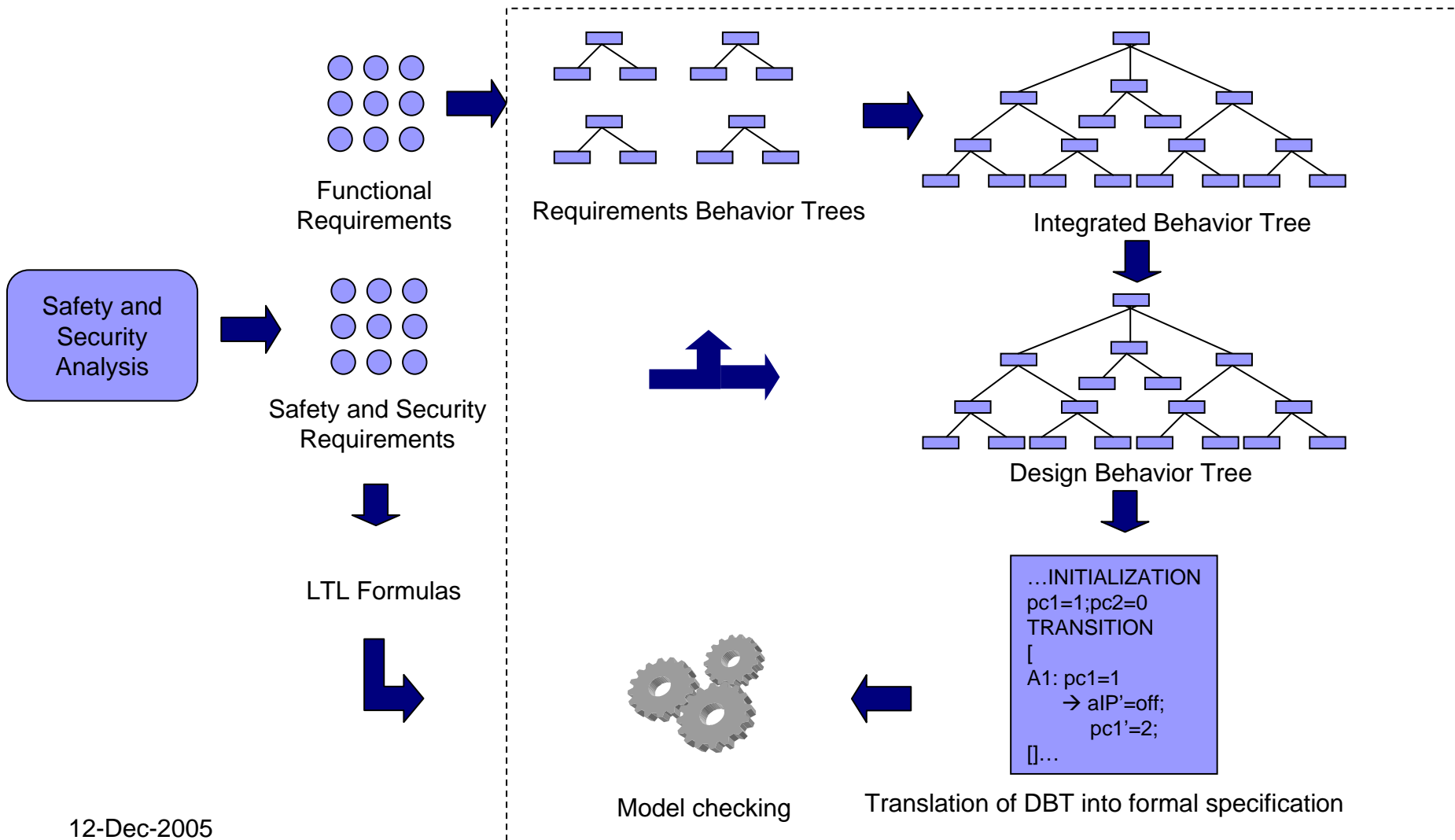
- Development strategy

- Uniform approach to systems development

GSE Design Strategy^[9]

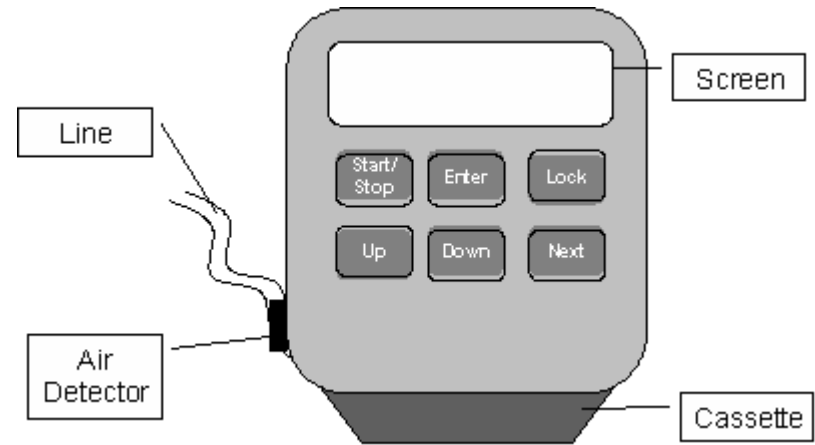
- Bridges gaps between informal and formal requirements specification
- Support early validation and verification
 - Uses simple graphical notation with formal semantics
 - Support for formal verification of requirements
- Bridges the gap between problem space and solution space
 - Systematic refinement of integrated requirements into system design
- Strategy for managing complexity
 - Reducing load on our short-term memory

GSE Design Process



AIP Case Study

- An overview^[10]
 - A safety-critical device
 - Used for drug therapy for patients who are away from direct care of medical practitioners
 - Delivers drug to patient at programmed infusion rate
 - Security concern – must avoid unauthorized programming of the pump
- System hazards
 - Drug under-/non-delivery
 - Air embolism
 - Drug overdose



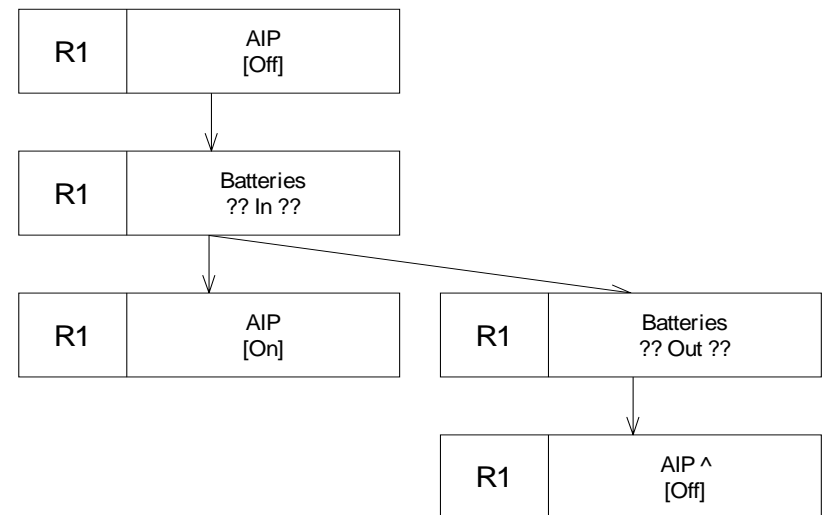
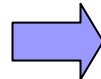
Hazard	Cause
Under-delivery or non-delivery of drug	<ul style="list-style-type: none"> - Obstruction or kinks in the line - Incorrect setting of the pump's infusion rate - Incorrect calculation drug infused
Air embolism	<ul style="list-style-type: none"> - Presence of air in the line
Over delivery of drug	<ul style="list-style-type: none"> - Incorrect setting of the pump's infusion rate - Incorrect calculation of drug infused

AIP Case Study

■ Requirements specification

- Building system out-of-its requirements
- Requirements traceability
- Requirements validation

No.	Requirement
R1	AIP system is turned on when batteries are put in and is turned off when batteries are out.

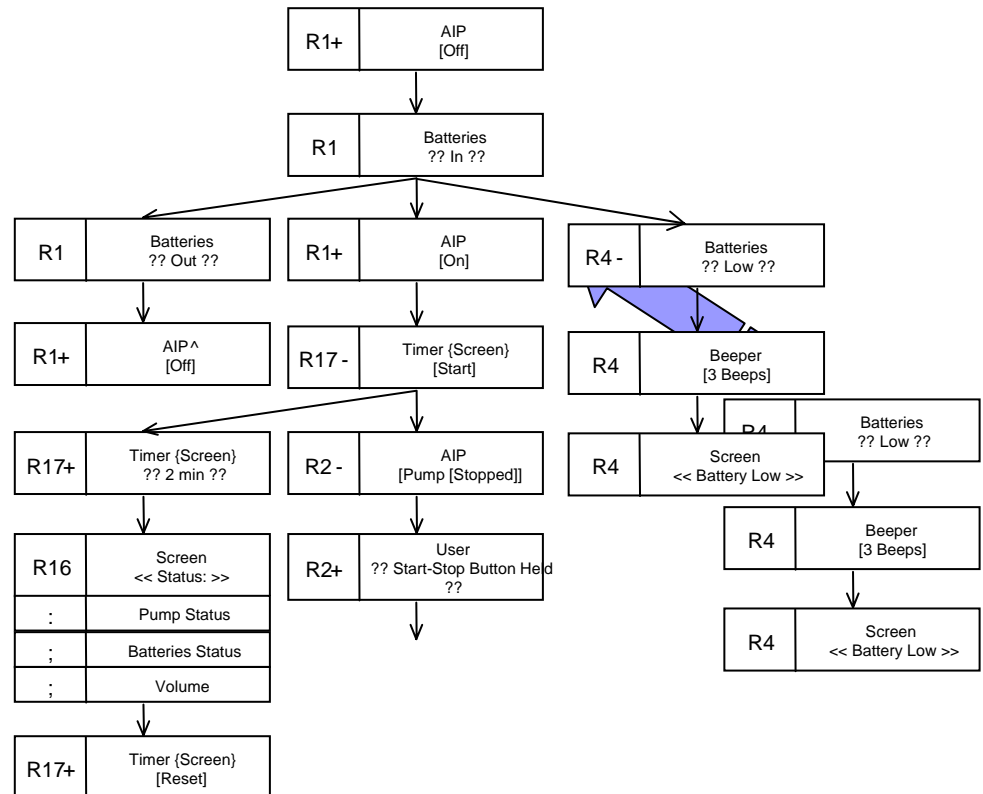


AIP Case Study

■ Requirements integration

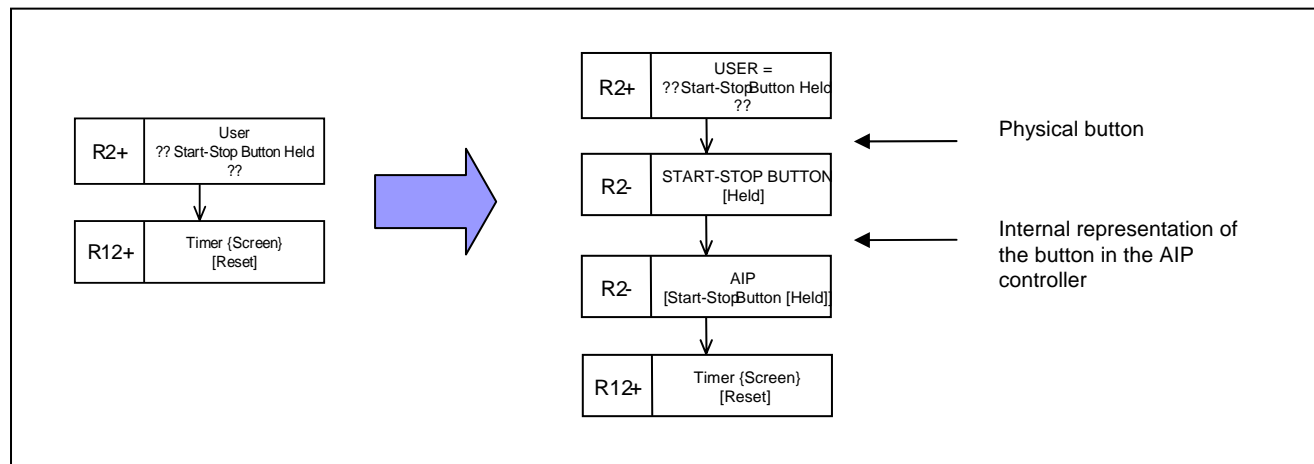
- Early defect detection
 - Resolve integration problems, e.g. missing pre-conditions

No.	Requirement
R4	When battery is low, system sends three beeps and displays battery low message on screen.



AIP Case Study

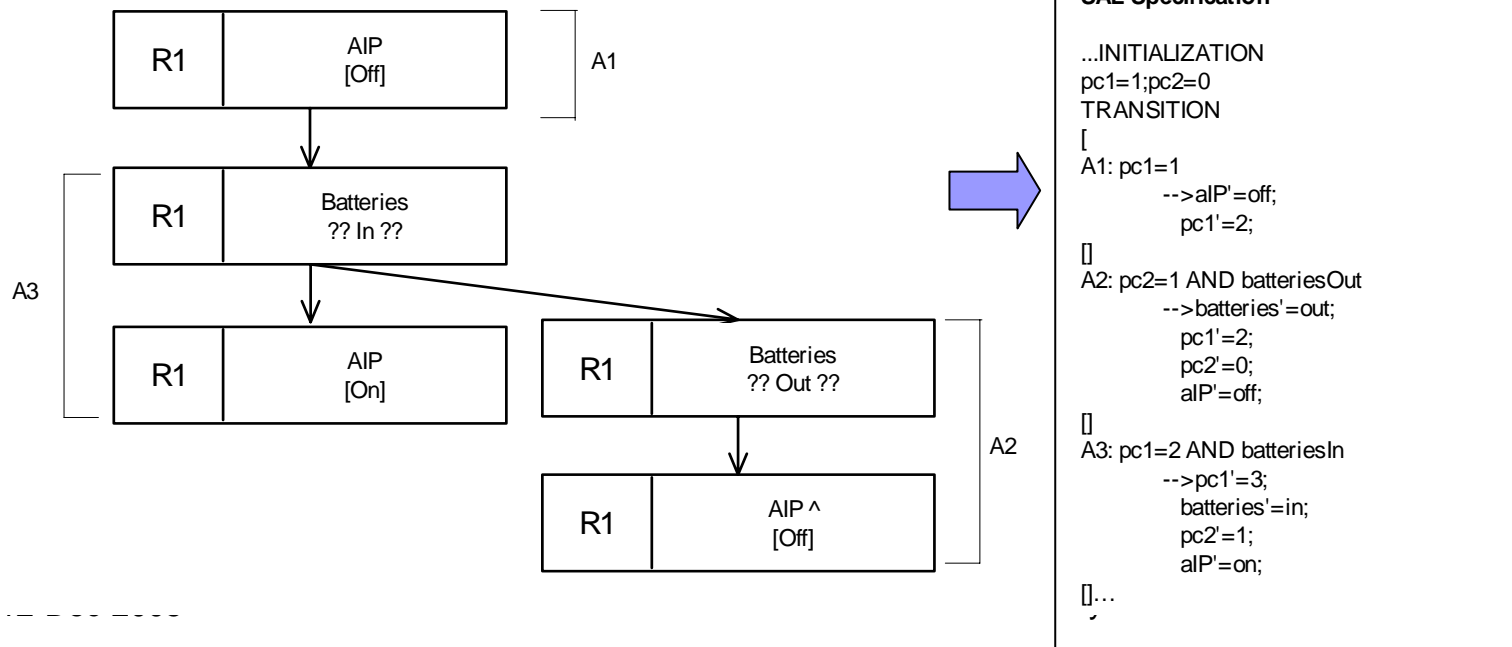
- Deriving system design from requirements
 - Systematically refining requirements
 - Decoupling of operator, sensor and actuator behavior
 - Separation of synchronous and asynchronous message passing
 - Identification of atomic actions
 - Identification of critical regions and interrupt handling



AIP Case Study

■ Model Checking

- Automated translation of requirements into SAL specification language



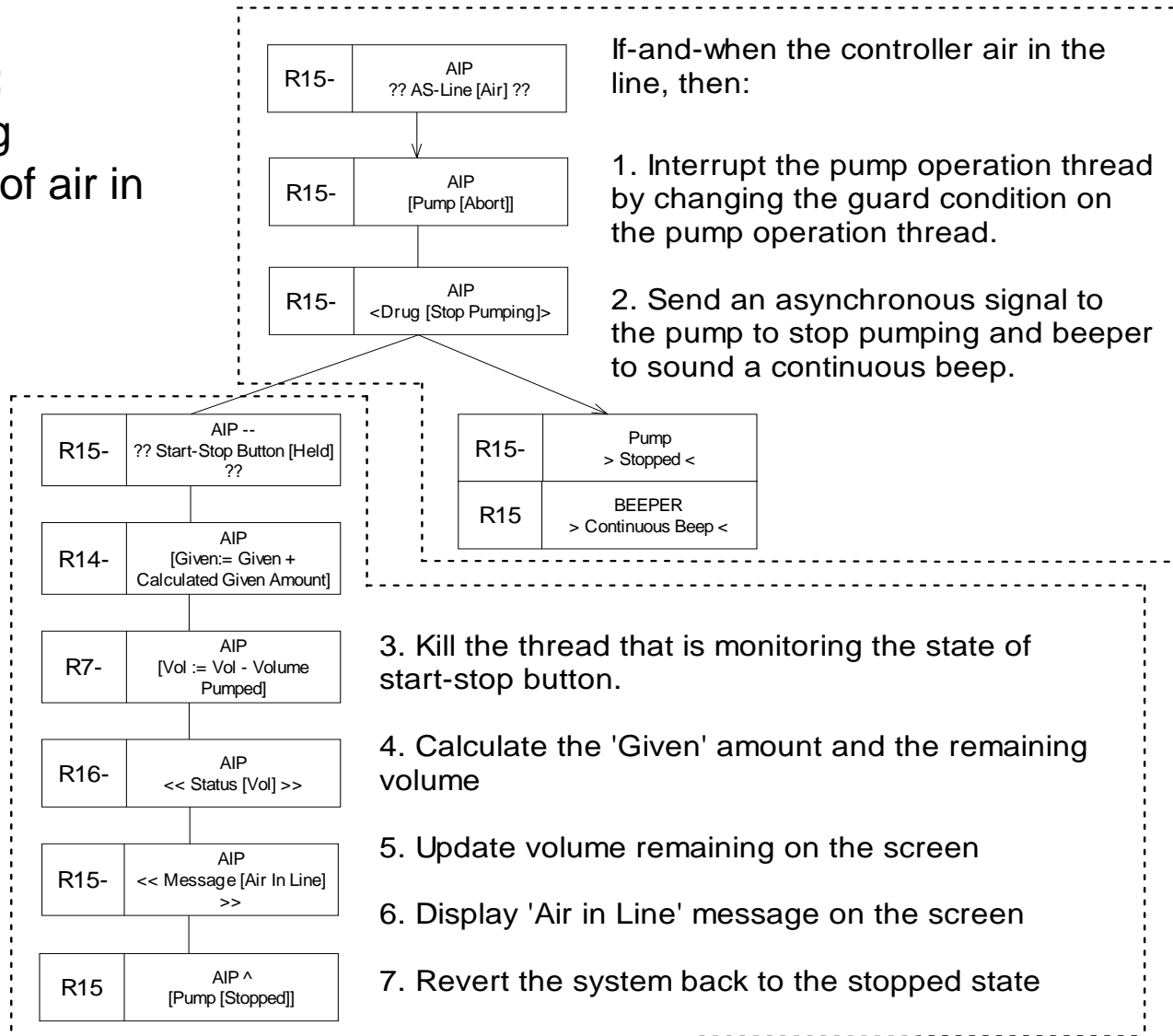
AIP Case Study

- Modeling safety properties
 - Pump must be stopped if air is detected.
 - Pump must be stopped if there is blockage in the line.
 - There should not be any discrepancy between the amount of drug calculated as given and the actual amount of drug infused.

AIP Case Study

- Modeling safety properties
 - DBT Design
 - Thread 1: Pumping operation
 - When pump is in *running* state:
 - Pumping operation occurs in a loop (controlled by a timer component)
 - The timer activates the pump every time it counts up to 'pump time' (based on infusion rate)
 - Thread 2: Detect air in the line
 - Interrupt pumping operation when air in the line is detected
 - Thread 3: Detect line blockage
 - Interrupt pumping operation when the line is blocked
 - Thread 4: Monitor start-stop button
 - Interrupt pumping operation when start-stop button is held long enough

Thread 2: Monitoring presence of air in the line



AIP case study – Modeling safety properties

AIP Case Study

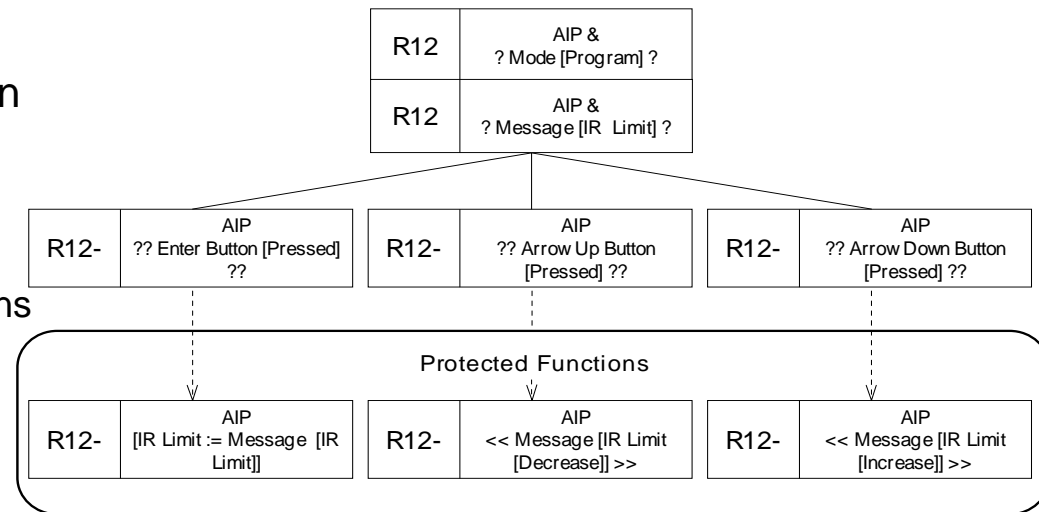
■ Modeling access control

Screen	Patient Mode			Clinic Mode			Program Mode		
	Buttons			Buttons			Buttons		
	↵	↑	↓	↵	↑	↓	↵	↑	↓
IR Upper Limit	x	x	x	x	x	x	Set	Inc.	Dec.
IR	x	x	x	Set	Inc.	Dec.	x	x	x
Given	x	x	x	Clear	x	x	x	x	x

Legend: ↵= Enter, ↑ = Arrow-up, ↓ = Arrow-down, IR = Infusion rate

AIP Case Study

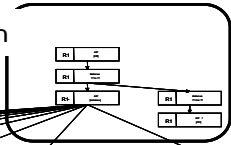
- Modeling access control
 - Support for requirements validation
 - Graphical representation
 - Integrated view of the system
 - Improved readability
 - Constraints on functions
 - Context driven functionality
 - Support for formal verification



Functions accessible in program mode

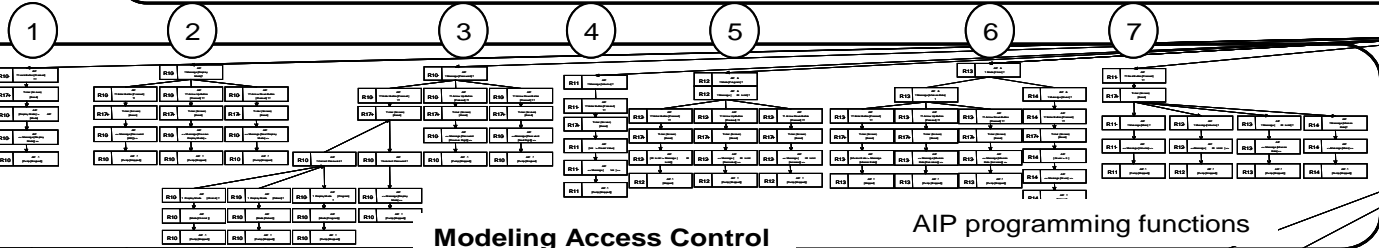
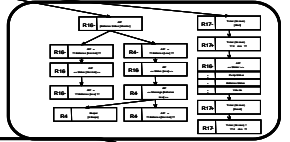
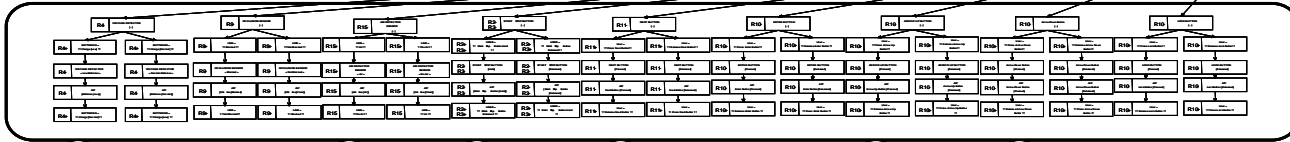


AIP Initialization



Threads monitoring batteries status and screen update timer

Threads monitoring user and sensor inputs

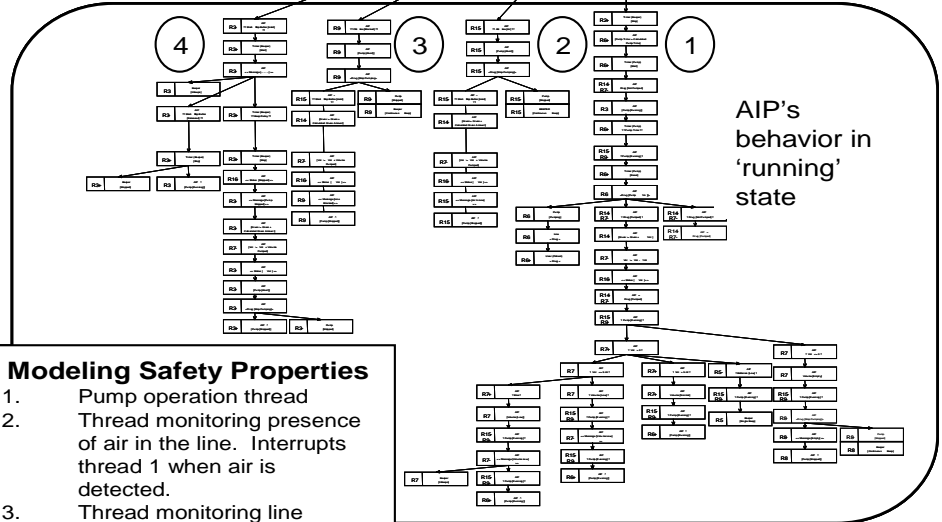


AIP's behavior in 'stopped' state

AIP programming functions

Modeling Access Control

	Screen	Mode	Button	Effect
1	Any	Any	Lock	Display current mode
2	Mode	Any	←	Display password screen
			↑	Display previous mode
			↓	Display next mode
3	Password	Any	←	1. Set the displayed mode if password is correct, or 2. Display the selected mode
			↑	Display previous digit
			↓	Display next digit
4	Volume	Any	←	Reset volume
			←	Set IR limit
5	IR Limit	Prog.	↑	Increment IR limit
			↓	Decrement IR limit
			←	Set IR
6	IR	Clinic	↑	Increment IR
			↓	Decrement IR
			←	Clear given amount
7	Any	Any	Next	Scroll through available screens



AIP's behavior in 'running' state

Modeling Safety Properties

1. Pump operation thread
2. Thread monitoring presence of air in the line. Interrupts thread 1 when air is detected.
3. Thread monitoring line blockage. Interrupts thread 1 when line is blocked.
4. Thread monitoring start-stop button. Interrupts thread 1 when button is held long enough.

AIP Case Study

- Formal verification of safety and security properties
 - Hazard of drug under/non-delivery
 - Safety properties
 1. If the line is blocked the pump should not pump.
 2. If the pump is stopped then the drug volume must be re-calculated.
 - Security properties
 3. The infusion rate must not be set in patient mode.
 4. The infusion rate must not be set in program mode.

No.	LTL Formula	Result
1.	$G((\text{line}=\text{blocked}) \Rightarrow (X(\text{pump}=\text{pStopped})))$	Proved
2.	$G((\text{aIP_Drug}=\text{stopPumping}) \Rightarrow (\text{aIP_Vol}=\text{vCalculated}))$	Proved
3.	$G((\text{aIP_Mode}=\text{patient}) \Rightarrow \text{NOT} (\text{aIP_InfusionRate}=\text{setInfusionRate}) \text{ AND } (\text{aIP_MSG}=\text{mInfusionRate}))$	Proved
4.	$G((\text{aIP_Mode}=\text{program}) \Rightarrow \text{NOT} (\text{aIP_InfusionRate}=\text{setInfusionRate}) \text{ AND } (\text{aIP_MSG}=\text{mInfusionRate}))$	Proved

AIP Case Study

- Formal verification of safety and security properties
 - Hazard of drug overdose
 - Safety properties
 1. If the pump is stopped then the drug volume must be re-calculated.
 - Security properties
 2. The upper limit for infusion rate must not be set in patient mode.
 3. The upper limit for infusion rate must not be set in clinic mode.

No.	LTL Formula	Result
1.	$G((aIP_Drug=stopPumping) \Rightarrow (aIP_Vol=vCalculated))$	Proved
2.	$G((aIP_Mode=patient) \Rightarrow NOT (aIP_IRLimit=setIRLimit) AND (aIP_MSG=mIRLimit))$	Proved
3.	$G((aIP_Mode=clinic) \Rightarrow NOT (aIP_IRLimit=setIRLimit) AND (aIP_MSG=mIRLimit))$	Proved

AIP Case Study

- Formal verification of safety and security properties
 - Hazard of air embolism
 - Safety properties
 1. If there is air in the line the pump should not pump.
 2. If the drug volume is zero pump should be set to running mode.

No.	LTL Formula	Result
1.	$G((\text{line}=\text{air}) \Rightarrow (X(\text{pump}=\text{pStopped})))$	Proved
2.	$G((\text{pump}=\text{running}) \Rightarrow \text{NOT} (\text{aIP_Volume} = \text{empty}))$	Not Proved

Future Directions

- Safety and security goal specification using BTs
- Support for hazards and threat analysis in GSE
- Support for timing and performance analysis for DBT
- Automated failure mode and effect analysis (FMEA) based on BT specification
- Automated code generation

Conclusion

- Integrated view of safety and security requirements
- Role of formal verification
- Systematic and uniform design process
- Early defect detection
- Requirements traceability
- Early verification of safety and security requirements
- Easy validation of requirements
- Systematic translation of informal requirements
- Systematic refinement of requirements specification
- Managing complexity of the design process

References

1. J. C. Knight, "Safety Critical Systems: Challenges and Directions," presented at 24th International Conference on Software Engineering, ICSE 2002, Orlando, Florida, 2002.
2. P. T. Devanbu and S. G. Stubblebine, "Software Engineering for Security: A Roadmap," presented at International Conference on Software Engineering (ICSE 2000) - Future of SE Track, 2000.
3. R. Lutz, "Software engineering for safety: A roadmap," presented at The Future of Software Engineering, 2000.
4. I. Sommerville, "An Integrated Approach to Dependability Requirements Engineering," presented at 11th Safety-Critical Systems Symposium, Bristol, 2003.
5. A. Burns, J. McDermid, and J. Dobson, "On the Meaning of Safety and Security," *Computer Journal*, vol. 35, pp. 3-15, 1992.
6. J. Rushby, "Critical System Properties: Survey and Taxonomy," Computer Science Laboratory, SRI International SRI-CSL-93-1, 1994.
7. D. G. Firesmith, "Common Concepts Underlying Safety, Security, and Survivability Engineering," Software Engineering Institute, Carnegie Mellon University CMU/SEI-2003-TN-033, December 2003.
8. N. G. Leveson, *Safeware: System Safety and Computers*: Addison-Wesley Publishing Company, 1995.
9. R. G. Dromey, "Genetic Design: Amplifying Our Ability to Deal With Requirements Complexity," *Lecture Notes in Computer Science*, vol. 3466, pp. 95-108, 2005.
10. Deltec, "Operator's Manual, CADD-Legacy ® 1 Ambulatory Infusion Pump, Model 6400," vol. 2005: Smiths Medical MD, Inc., 2005.

Acknowledgements

■ Funding

- Partially funded by Australian Research Council (ARC) under the ARC Centres of Excellence program

■ Contribution

- Assistance by Dr. Lars Grunske and Ninsansala Yatapanage in model checking the AIP design

■ Suggestions

- Constructive suggestions by our colleagues in the Dependable Complex Computer-based Systems (DCCS)



Questions and comments