Modal Abstraction View of Requirements for Medical Devices Used In Healthcare Processes

Heather M. Conboy    George S. Avrunin    Lori A. Clarke

School of Computer Science
University of Massachusetts Amherst, USA

SEHC 2013
Problem: Medical device requirements often depend on the healthcare processes in which the device is used. Since such processes are often complex, critical requirements may be specified inaccurately, or even missed.

Goal: Automatically derive device requirements from
- the overall process requirements
- a formal model of the process in which the device is used
Medical device requirements often depend on the healthcare processes in which the device is used. Since such processes are often complex, critical requirements may be specified inaccurately, or even missed. An overall process model can be formalized to provide derived device requirements that can be used to help:

- Developers understand how the expected process could impact device design.
- Certify that a device will be safe when used in the expected process.
- Select a device to be used in a process.
Example: “Smart” Infusion Pump
Used During In-Patient Surgery Process

**Overall process requirement:**
“Patient must never be administered a medication overdose”

**Pump:**

**Surgery process:**
Example: “Smart” Infusion Pump
Used During In-Patient Surgery Process

Overall process requirement:
“Patient must never be administered a medication overdose”

Pump:

Surgery process:
Example: “Smart” Infusion Pump Used During In-Patient Surgery Process

Overall process requirement:
“Patient must never be administered a medication overdose”

Pump:

Surgery process:
Example: “Smart” Infusion Pump
Used During In-Patient Surgery Process

Overall process requirement:
“Patient must never be administered a medication overdose”

Pump:

Surgery process:
Example: “Smart” Infusion Pump Used During In-Patient Surgery Process

Overall process requirement:
“Patient must never be administered a medication overdose”

Pump:

Drug Library:
- ICU

Drug:
- Drug 1

Dose:
- 100 units

Surgery process:

PRE-OP
OR (Operating room)
ICU (Intensive care unit)
Example: “Smart” Infusion Pump Used During In-Patient Surgery Process

Overall process requirement:
“Patient must never be administered a medication overdose”

Pump:

Surgery process:

PRE-OP
OR (Operating room)
ICU (Intensive care unit)
Example: “Smart” Infusion Pump
Used During In-Patient Surgery Process

Overall process requirement:
“Patient must never be administered a medication overdose”

Pump:

Drug Library:
- OR

Drug:
- Drug 1

Dose:
- 100 units

Surgery process:
Example: “Smart” Infusion Pump Used During In-Patient Surgery Process

Overall process requirement:
“Patient must never be administered a medication overdose”

Pump:

Drug Library:
- OR

Drug:
- Drug 1

Dose:
- 100 units

Since not ICU, no alert!

Surgery process:

PRE-OP

OR
(Operating room)

ICU
(Intensive care unit)
Previous Work:
Our Process-based Requirement Derivation Approach

Overall process model
Devicemodel
Process model in which
device is used
Little-JIL [Cass2000]

Interface synthesis method employs model checking and learning algorithms
[Beyer2005, Gianakopoulou & Păsăreanu2009]
FLAVERS model checker [Dwyer2007]
L* learner [Agnluin1987, Rivest & Schapire1989]

Overall process requirement
"never overdose"

Finite state automaton (FSA)

Process-based requirement deriver

Derived device requirement for pump
Previous Work:
Our Process-based Requirement Derivation Approach

Overall process model
Device model
FSA
Process model in which device is used
Little-JIL [Cass2000]

Interface synthesis method employs model checking and learning algorithms
[Beyer2005, Gianakopoulou&Păsăreanu2009]
FLAVERS model checker [Dwyer2007]
L* learner [Angluin1987, Rivest&Schapire1989]

Overall process requirement
“never overdose”

Finite state automaton (FSA)

Derived device requirement
for pump
Overall process requirement
"never overdose"

Finite state automaton (FSA)
Previous Work: 
Our Process-based Requirement Derivation Approach

Overall process requirement
"never overdose"

Process-based requirement deriver

Derived device requirement
for pump

Finite state automaton (FSA)

Overall process model
Device model
FSA

Process model in which device is used
Little-JIL [Cass2000]

Interface synthesis method employs model checking and learning algorithms
[Beyer2005, Gianakopoulou&Pășăreanu2009]
FLAVERS model checker [Dwyer2007]
L* learner [Angluin1987, Rivest&Schapire1989]
Previous Work: Our Process-based Requirement Derivation Approach

Overall process model
Device model
Process model in which device is used
Little-JIL [Cass2000]

Finite state automaton (FSA)

Interface synthesis method employs model checking and learning algorithms
[Beyer2005, Gianakopoulou&Păsăreanu2009]
FLAVERS model checker [Dwyer2007]
L* learner [Angluin1987, Rivest&Schapire1989]
Previous Work:
Our Process-based Requirement Derivation Approach

Overall process model

Device model
FSA

Process model in which device is used
Little-JIL [Cass2000]
Previous Work:
Our Process-based Requirement Derivation Approach

- Overall process model
  - Interaction diagram
  - Process model in which device is used
  - Little-JIL [Cass2000]

- Device model
  - FSA

- Derived device requirement
  - for pump

- Finite state automaton (FSA)

- Interface synthesis method
  - employs model checking and learning algorithms
  - [Beyer2005, Gianakopoulou&Păsăreanu2009]
  - FLAVERS model checker [Dwyer2007]
  - L* learner [Angluin1987, Rivest&Schapire1989]
**Previous Work:**

Our Process-based Requirement Derivation Approach

**Overall process model**

- Device model
- FSA
- Process model in which device is used
- Little-JIL [Cass2000]

**Interface synthesis method** employs model checking and learning algorithms

- [Beyer2005, Gianakopoulou&Păsăreanu2009]
- FLAVERS model checker [Dwyer2007]
- L* learner [Angluin1987, Rivest&Schapire1989]

**Derived device requirement** for pump

- Finite state automaton (FSA)

**Overall process requirement** "never overdose"

1. `setDose_HIGH, start_alert, start_ok`
2. `setDose_HIGH`
3. `setDose_HIGH`
4. `setDose_LOW`
5. `setDose_LOW`
6. `setDose_LOW`
Derived device requirement for pump
Previous Work:
Our Process-based Requirement Derivation Approach

**Overall process requirement**

"never overdose"

**Finite state automaton (FSA)**

**Process-based requirement deriver**

**Derived device requirement**

for pump

**Process model in which device is used**

Little-JIL [Cass2000]

**Device model**

FSA

**Interface synthesis method**

employs model checking and learning algorithms

[Beyer2005, Gianakopoulou&Păsăreanu2009]

FLAVERS model checker [Dwyer2007]

L* learner [Angluin1987, Rivest&Schapire1989]
Preliminary Investigation of Our Approach

- Implemented a toolset that supports this approach and evaluated the approach by applying the toolset to two simplified case studies [Conboy2010]

- Further evaluated the approach by applying the toolset to two complex case studies in the elections domain

Derived device requirements were useful, but . . .

* Image Copyright 2007 Steven Rodriguez
Introduction and Background
Modal Abstraction View
Summary and Future Work

Scalability is a Challenge

Complexity can increase along many dimensions
- e.g., overall process requirements, process models, device models

Two notable limitations are:
- Derivation performance often does not scale well
- Derived device requirements often reflect the complexity and are difficult to understand
Scalability is a Challenge

Complexity can increase along many dimensions

- e.g., overall process requirements, process models, device models

Two notable limitations are:

- Derivation performance often does not scale well
- **Derived device requirements often reflect the complexity and are difficult to understand**
Device considered to be modal if different settings enable different behaviors

- Medical devices are often modal
  - e.g., Pump issues alerts based on the setting of the drug library

**Proposed technique:** Create modal abstraction view that decomposes a device requirement into pieces based on its modes

- Each piece captures a given mode’s behaviors
- Transitions among the pieces capture the mode changes

**Potential benefit:**

Should reduce the cognitive load for viewers since they can consider each piece separately
How to Create the Modal Abstraction View

For a given derived device requirement represented as an FSA:

1. Manually identify mode change events:
   - setLib_ICU,
   - setLib_OR
For a given derived device requirement represented as an FSA:

1. Manually identify mode change events:
   - setLib_ICU,
   - setLib_OR
How to Create the Modal Abstraction View

For a given derived device requirement represented as an FSA:

1. Manually identify mode change events:
   - setLib_ICU,
   - setLib_OR

2. Decompose the FSA based on the mode change events
How to Create the Modal Abstraction View

For a given derived device requirement represented as an FSA:

1. Manually identify mode change events:
   - setLib_ICU,
   - setLib_OR

2. Decompose the FSA based on the mode change events
How to Create the Modal Abstraction View

For a given derived device requirement represented as an FSA:

1. Manually identify mode change events:
   - `setLib_ICU`,
   - `setLib_OR`

2. Decompose the FSA based on the mode change events
How to Create the Modal Abstraction View

For a given derived device requirement represented as an FSA:

1. Manually identify mode change events:
   - setLib_ICU,
   - setLib_OR

2. Decompose the FSA based on the mode change events
How to Create the Modal Abstraction View

For a given derived device requirement represented as an FSA:

1. Manually identify mode change events:
   - `setLib_ICU`
   - `setLib_OR`

2. Decompose the FSA based on the mode change events

3. Draw the view highlighting mode change transitions and pieces for each mode
How to Create the Modal Abstraction View

For a given derived device requirement represented as an FSA:

1. Manually identify mode change events:
   - `setLib_ICU`
   - `setLib_OR`

2. Decompose the FSA based on the mode change events

3. Draw the view highlighting mode change transitions and pieces for each mode
How to Create the Modal Abstraction View

For a given derived device requirement represented as an FSA:

1. Manually identify mode change events:
   - `setLib_ICU`, `setLib_OR`

2. Decompose the FSA based on the mode change events

3. Draw the view highlighting mode change transitions and pieces for each mode
Related Work: Modes

- **Model checking used to discover mode errors** [e.g., Crow2000]: Generates a counterexample, not a derived requirement

- **Other requirement derivation approaches track modes** [e.g., Combefis2011]: No view that takes into account the modes

- **Visualization techniques to improve understandability**
  - **Modecharts** [e.g., JaharianMok1994] basically map from modal abstractions to low-level source code:
    Our approach maps from low-level FSAs to modal abstractions
  - **Alternative views** [e.g., WhittleJayaraman2010] that decompose the requirements based on different high-level features (e.g., interaction overview diagrams)
Summary

- Developed a tool to automatically create the **modal abstraction view** of a derived requirement and applied this tool to case studies
  - Seems to improve understandability

Need to do a more careful evaluation of the derivation and view

- Other domains
- Wider variety of overall process requirements
- More complex processes
- More complex “devices”
  - e.g., *computerized physician order entry system*: Use modes to reflect the access control policies for the various types of hospital staff
Future Work:

To Improve the Understandability of the Derived Requirements

- Decompose modes based on **sub-modes**
- Create **alternative views** that highlight different high-level features, e.g.,
  - sub-processes (e.g., operating room)
  - human participants (e.g., nurse)
- Consider **other requirement representations**, e.g.,
  - statecharts [Harel87]
  - modecharts [JaharianMok94]
**Future Work:**

**To Improve the Scalability of the Deriver**

- **Optimize** derivation
- **Incrementally refine** overall process model based on previous derivations
Future Work:
To Improve the Scalability of the Deriver

- **Optimize** derivation
- **Incrementally refine** overall process model based on previous derivations

```
Overall process model
Process model in which pump is used
Process-based requirement deriver + optimizations

"never overdose"
overall process requirement
FSA

Pump model
FSA

Interactions

Process model in which pump is used
Little-JIL

"never underdose"
overall process requirement
FSA

"never overdose"
derived pump requirement
FSA

"never overdose or underdose"
derived pump requirement
FSA
```