Model-based Safety Assessment using OCL and Petri Nets

Ricardo J. Rodríguez and Elena Gómez-Martínez
rj.rodriguez@unileon.es, egomez@babel.ls.fi.upm.es

universidad de león

Research Institute of Applied Sciences in Cybersecurity, University of León
León, Spain

Babel Group, ETSINF
Technical University of Madrid
Madrid, Spain

August 27, 2014

40th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)
Verona (Italy)

Agenda

1. Introduction
2. Previous Concepts
3. Safety Contracts Specification and Verification
   - Specification
   - Verification
4. Conclusions and Future Work

Agenda

1. Introduction
2. Previous Concepts
3. Safety Contracts Specification and Verification
   - Specification
   - Verification
4. Conclusions and Future Work

Introduction (I): Motivation

Safety assessment

- Needed by some systems (e.g. critical systems)
  - Industrial equipment, road vehicles, avionics...
  - Requirements specified by industrial standards (IEC-61508, ISO-26262, DO-178C)
- Later verification induces budget overruns
  - Example: Half of the overall costs in avionics software domain

RIASC

Introduction (II): Motivation

Contracts

- Commonly used to specify relationships between system components
- Pre- and post-conditions of a system component
- Refinement idea: safety contract
  - Assumptions; Guarantees
  - Aim: to assure a certain level of confidence of a component

Safety contracts can be used to specify safety standard requirements

Introduction (III): Our Approach

Rationale

- Safety contract specification in design phase: early validation → saves overruns
  - Using UML + UML profiles + OCL
    - UML State-Machine and UML Sequence diagrams: Dynamic part of the system
    - UML Class Diagram: Static one
    - MARTE profile: Performance system information
    - OCL: Specifying the safety contracts (assumptions, guarantees)

Introduction (III): Our Approach

Rationale

- Safety contract specification in design phase: early validation → saves overruns
  - Using UML + UML profiles + OCL
    - UML State-Machine and UML Sequence diagrams: Dynamic part of the system
    - UML Class Diagram: Static one
    - MARTE profile: Performance system information
    - OCL: Specifying the safety contracts (assumptions, guarantees)
- Verification through formal models
  - Petri nets (namely, Generalised Stochastic Petri nets)

Introduction (IV): Related Work

Related Work

- Specification
  - OCL already used: Either without verification, or without "formal" specification
  - UML profiles (SysML, OMEGA) : Express safety (or correctness) contracts
- Verification
  - Model-checking (ATL, Timed I/O Automata, AADL)

Agenda

1. Introduction
2. Previous Concepts
3. Safety Contracts Specification and Verification
   - Specification
   - Verification
4. Conclusions and Future Work

Previous Concepts (I)

UML and UML profiles

- Semi-formal modelling language

Previous Concepts (I)

UML and UML profiles

- Semi-formal modelling language
- Tailored for specific domains by profiling
  - Stereotypes: Concepts in the target domain
  - Tagged values: Stereotype attributes
- Enriches UML semantics, commonly used for NFPs specification

Previous Concepts (I)

UML and UML profiles

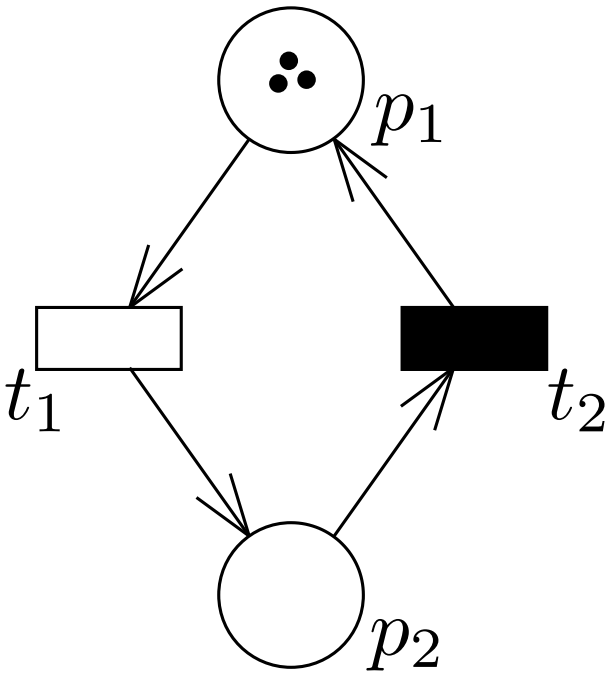
- Semi-formal modelling language
- Tailored for specific domains by profiling
  - Stereotypes: Concepts in the target domain
  - Tagged values: Stereotype attributes
- Enriches UML semantics, commonly used for NFPs specification
- Profile examples:
  - Modelling and Analysis of RT and Embedded systems (MARTE)
    - Generic Quantitative Analysis Model framework, `gaStep` stereotype (activity durations)
  - Dependability Analysis and Modelling (DAM)
  - Security Analysis and Modelling (SecAM)

Previous Concepts (II)

- UML + MARTE not suitable for performance evaluation or model-checking
- Formal models may help for this goal
  - UML + MARTE $\rightarrow$ Petri nets (namely, Generalised Stochastic PN)

Previous Concepts (II)

- UML + MARTE not suitable for performance evaluation or model-checking
- Formal models may help for this goal
  - UML + MARTE → Petri nets (namely, Generalised Stochastic PN)



GSPN

- Bipartite graph
- Places (circles, p_X)
- Transitions (bars, t_X)
  - Immediate ($t = 0$)
  - Timed (exponential, deterministic firing distributions)
- Arcs (with directions, and weight)
- Tokens

Agenda

1. Introduction
2. Previous Concepts
3. Safety Contracts Specification and Verification
   - Specification
   - Verification
4. Conclusions and Future Work

Safety Contracts Specification and Verification (I)

- Safety Contract Fragment (SCF) $\mathcal{S} = \langle \mathcal{A}, \mathcal{G} \rangle$
  - Assumptions ($\mathcal{A}$): Expected to be met by the component's environment
    - $\mathcal{A} = \mathcal{A}^{+} \bigcup \mathcal{A}^{*}$, OR and AND safety constraints, respectively
  - Guarantees ($\mathcal{G}$): Component's behaviour under such an environment

Safety Contracts Specification and Verification (I)

- Safety Contract Fragment (SCF) $\mathcal{S} = \langle \mathcal{A}, \mathcal{G} \rangle$
  - Assumptions ($\mathcal{A}$): Expected to be met by the component's environment
    - $\mathcal{A} = \mathcal{A}^{+} \bigcup \mathcal{A}^{*}$, OR and AND safety constraints, respectively
  - Guarantees ($\mathcal{G}$): Component's behaviour under such an environment

How can we express a SCF, usually expressed in text form, in OCL within a UML model?

Safety Contracts Specification and Verification (I)

- Safety Contract Fragment (SCF) $\mathcal{S} = \langle \mathcal{A}, \mathcal{G} \rangle$
  - Assumptions ($\mathcal{A}$): Expected to be met by the component's environment
    - $\mathcal{A} = \mathcal{A}^{+} \bigcup \mathcal{A}^{*}$, OR and AND safety constraints, respectively
  - Guarantees ($\mathcal{G}$): Component's behaviour under such an environment

How can we express a SCF, usually expressed in text form, in OCL within a UML model?

SCF expressed with information from a UML Class Diagram

Safety Contracts Specification and Verification (I)

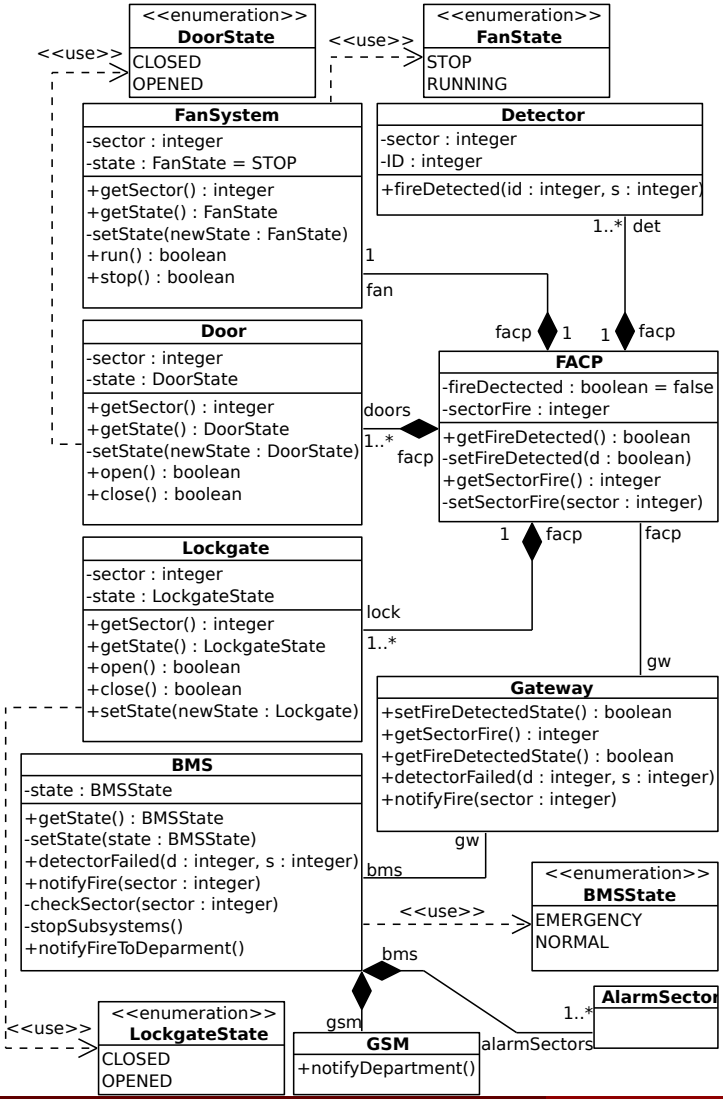
- Safety Contract Fragment (SCF) $\mathcal{S} = \langle \mathcal{A}, \mathcal{G} \rangle$
  - Assumptions ($\mathcal{A}$): Expected to be met by the component's environment
    - $\mathcal{A} = \mathcal{A}^{+} \bigcup \mathcal{A}^{*}$, OR and AND safety constraints, respectively
  - Guarantees ($\mathcal{G}$): Component's behaviour under such an environment

How can we express a SCF, usually expressed in text form, in OCL within a UML model?

SCF expressed with information from a UML Class Diagram
(we have currently moved to UML Composite diagram, relating a SCF to component input/output ports...)

Safety Contracts Specification and Verification (II)

Running example: Fire prevention system in a hospital building (1)



- *Building Management System* (BMS)
  - Controls air conditioner, lights and elevators, etc.
- *Fire Alarm Control Panel* (FACP): communicates with BMS via a Gateway
  - An area is divided in sectors
  - A sector is composed of:
    - Environmental detectors, fire doors, lockgates and ventilation system fans

Safety Contracts Specification and Verification (III)

Running example: Fire prevention system in a hospital building (2)

1. *When a fire is positively detected in some sector, eventually the system reaches emergency state*; and
2. *When a fire is detected in a sector s, the lock gates of s are eventually closed*

Safety Contracts Specification and Verification (III)

Running example: Fire prevention system in a hospital building (2)

1. *When a fire is positively detected in some sector, eventually the system reaches emergency state*; and
2. *When a fire is detected in a sector s, the lock gates of s are eventually closed*

$$\mathcal{S}_1 = \langle \{FACP.getFireDetected()\}, \{BMS.getState() = EMERGENCY\} \rangle$$

$$\mathcal{S}_2 = \langle \{FACP.getFireDetected() \wedge Lockgate.getSector() = Lockgate.sector.fcp.getSectorFire()\}, \{Lockgate.getState() = CLOSED\} \rangle$$

Safety Contracts Specification and Verification (IV)

Running example: Fire prevention system in a hospital building (3)

SCF $\mathcal{S} = \langle \mathcal{A}, \mathcal{G} \rangle$ to OCL

- $\mathcal{A}, \mathcal{G}$: Relate a private class attributes, thru. setter/getter methods
- But OCL is defined in a concrete class
- Assume $\mathcal{G}$ relates a private class attribute, thru. a setter method

$$\mathcal{S}_1 = \langle \{FACP.getFireDetected()\}, \{BMS.getState() = EMERGENCY\} \rangle$$

$$\mathcal{S}_2 = \langle \{FACP.getFireDetected() \wedge Lockgate.getSector() = Lockgate.sector.fcp.getSectorFire()\}, \{Lockgate.getState() = CLOSED\} \rangle$$

Safety Contracts Specification and Verification (IV)

Running example: Fire prevention system in a hospital building (3)

SCF $\mathcal{S} = \langle \mathcal{A}, \mathcal{G} \rangle$ to OCL

- $\mathcal{A}, \mathcal{G}$: Relate a private class attributes, thru. setter/getter methods
- But OCL is defined in a concrete class
- Assume $\mathcal{G}$ relates a private class attribute, thru. a setter method

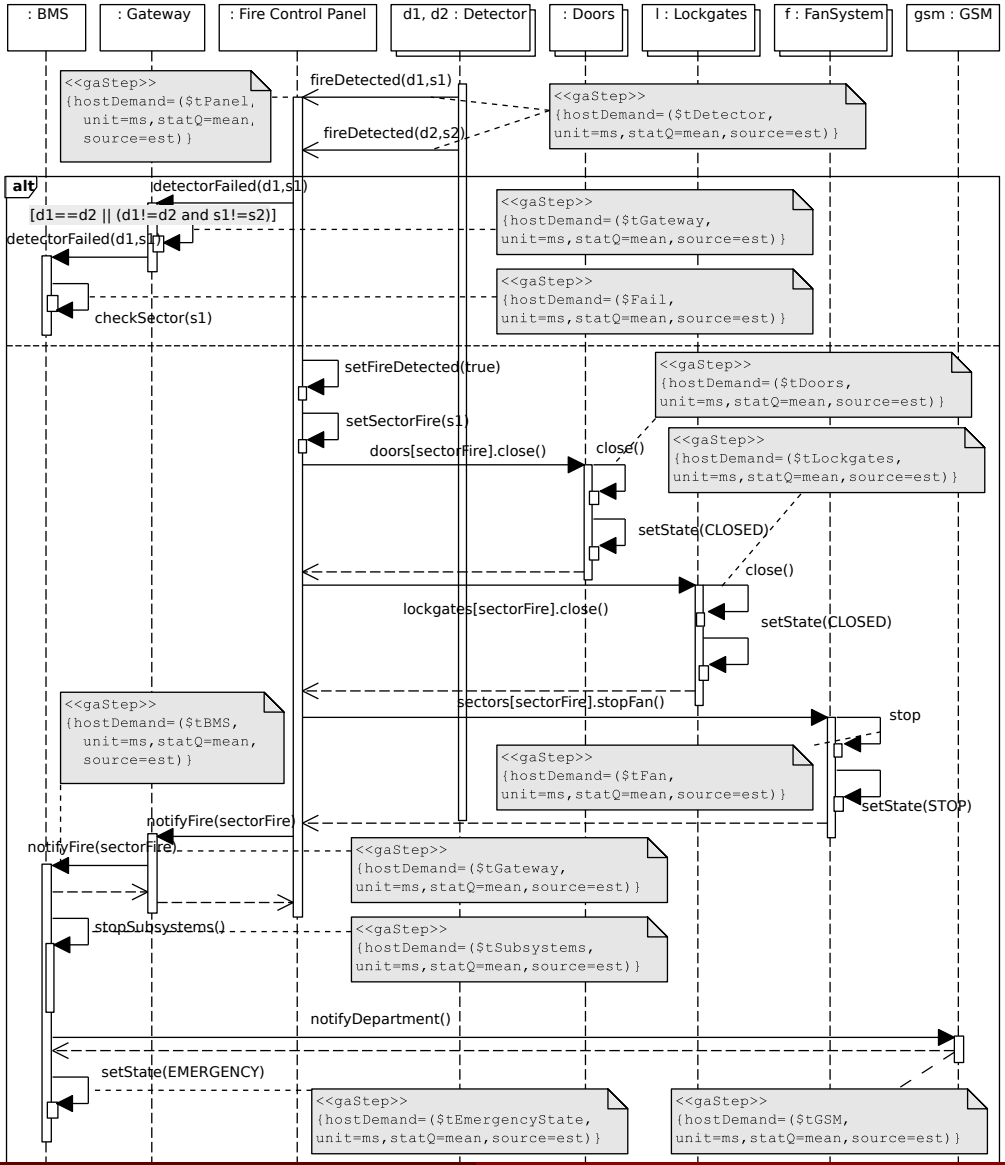
$$\mathcal{S}_1 = \langle \{FACP.getFireDetected()\}, \{BMS.getState() = EMERGENCY\} \rangle$$

$$\mathcal{S}_2 = \langle \{FACP.getFireDetected() \wedge Lockgate.getSector() = Lockgate.sector.fcp.getSectorFire()\}, \{Lockgate.getState() = CLOSED\} \rangle$$

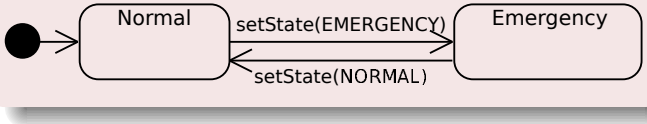
```
context BMS::setState(state: BMSState): void
  inv: gw.facp.getFireDetected() implies state = EMERGENCY

context Lockgate::setState(newState: LockgateState): void
  inv: gw.facp.getFireDetected() and
         self.getSector() = self.sect.fcp.getSectorFire()
         implies newState = CLOSED
```

Safety Contracts Specification and Verification (IV)



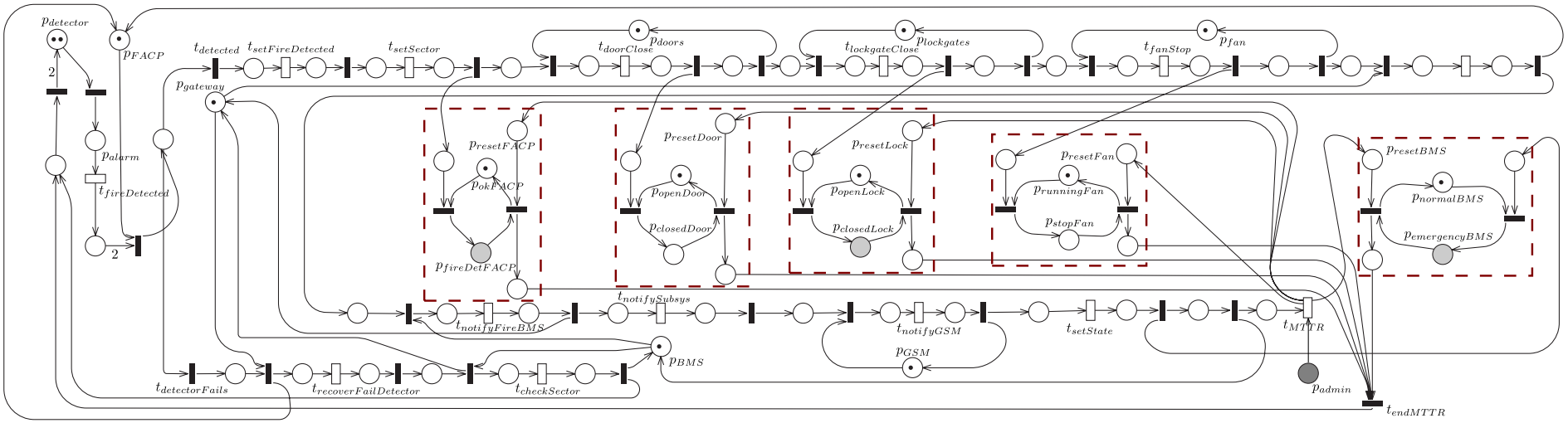
UML-SM of the BMS



- UML-SM for each state change: Firewall doors, air lockgates, air fan system

Safety Contracts Specification and Verification (V)

Running example: Fire prevention system in a hospital building (4)



- UML-SM: red-dashed boxes
- Validation of $\mathcal{S}_1, \mathcal{S}_2$ by checking place marking probabilities (light-grey highlighted)
  - $\mathcal{S}_1$: places $p_{fireDetFACP}$ and $p_{emergencyBMS}$
  - $\mathcal{S}_2$: places $p_{fireDetFACP}$ and $p_{closedLock}$ (second constraint of the assumption assumed to be always fulfilled)

Agenda

1. Introduction
2. Previous Concepts
3. Safety Contracts Specification and Verification
   - Specification
   - Verification
4. Conclusions and Future Work

Conclusions and Future Work (I)

- Early safety verification helps to:
  - Detect potential problems contradicting safety requirements
  - Save budget overruns
- Safety requirements expressed as safety contracts

Contributions

- UML diagrams to describe the system
  - UML-CD: Static part
  - UML-SD, UML-SM: dynamic part
  - UML profiles: Performance specification
  - OCL: Express safety contracts
    - Safety Contract Fragments: Assumptions, guarantees
- Formal model to verify safety contracts
  - Petri nets (namely, GSPN)
  - Verification by checking marking probabilities

Conclusions and Future Work (I)

A last remark

- Final effort must be done in implementation
  - Assure it matches the system model, or otherwise it may lead the system to an unsafe system

Conclusions and Future Work (I)

A last remark

- Final effort must be done in implementation
  - Assure it matches the system model, or otherwise it may lead the system to an unsafe system

Future Work

- Formalise the model transformation
- Explore other models (such as Othello to enable LTL model-checking)
- Use UML-profiled annotations to account for time specification during verification

Conclusions and Future Work (I)

A last remark

- Final effort must be done in implementation
  - Assure it matches the system model, or otherwise it may lead the system to an unsafe system

Future Work

- Formalise the model transformation
- Explore other models (such as Othello to enable LTL model-checking)
- Use UML-profiled annotations to account for time specification during verification

Acknowledgements

- ARTEMIS JU nSafeCer, nº 295373

Work improved

Last improvements done...

- UML Composite diagram to specify the system
- $\mathcal{S} = \langle \mathcal{A}, \mathcal{G} \rangle$
  - $\mathcal{A}$: Relates input ports of a component
  - $\mathcal{G}$: Relates output ports of a component
  - Still AND/OR formulae...
- Transformation to OCL invariant (with $\mathcal{A}$ implies $\mathcal{G}$)
- $p \Rightarrow q \Leftrightarrow \neg p \vee q$
  - The latter conditions are verified in PN (marking probability)

Model-based Safety Assessment using OCL and Petri Nets

Ricardo J. Rodríguez and Elena Gómez-Martínez
rj.rodriguez@unileon.es, egomez@babel.ls.fi.upm.es

universidad de león

Research Institute of Applied Sciences in Cybersecurity, University of León
León, Spain

Babel Group, ETSINF
Technical University of Madrid
Madrid, Spain

August 27, 2014

40th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)
Verona (Italy)