UML-based Model-Driven Code Generation of Error Detection Mechanisms

ICSEA 2020
Porto, Portugal

Lars Huning, Padma Iyenghar, Elke Pulvermüller
correspondence: lhuning@uos.de

Presenter resume

- B.Sc. Computer Science (2016)
- M.Sc. Computer Science (2018)
- Since 2018: Research assistant and PhD student in the Software Engineering group of Osnabrück University
- PhD topic: Automatically generating source code for safety mechanisms from UML model representations
- Currently on the lookout for a co-advisor of my PhD thesis

Outline

- Introduction
- Overview
- Related Work
- MDD Code Generation for Error Detection
- Use Case
- Conclusion and Future Work

Introduction

Safety-critical applications

IEC

IEC 61508-3

Edition 2.0 2010-04

INTERNATIONAL STANDARD

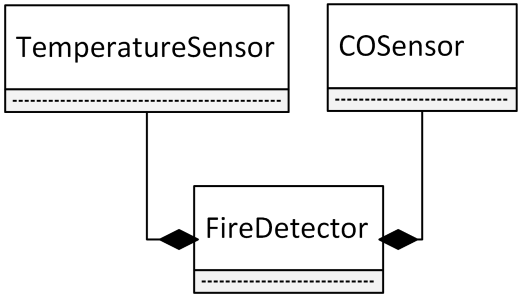
NORME INTERNATIONALE

BASIC SAFETY PUBLICATION
PUBLICATION FONDAMENTALE DE SÉCURITÉ

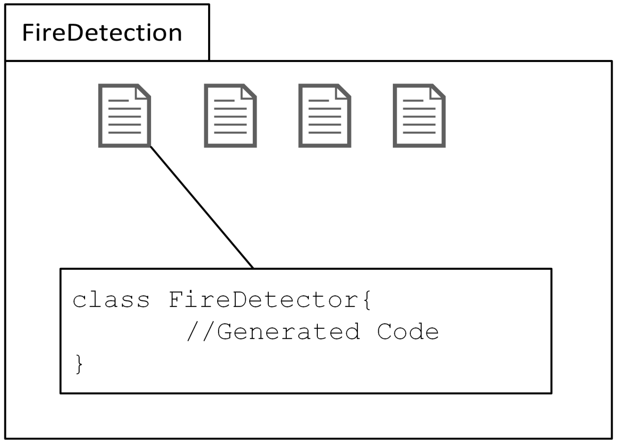
Functional safety of electrical/electronic/programmable electronic safety-related systems –
Part 3: Software requirements

Sécurité fonctionnelle des systèmes électriques/électroniques/électroniques programmables relatifs à la sécurité –
Partie 3: Exigences concernant les logiciels

Safety standards



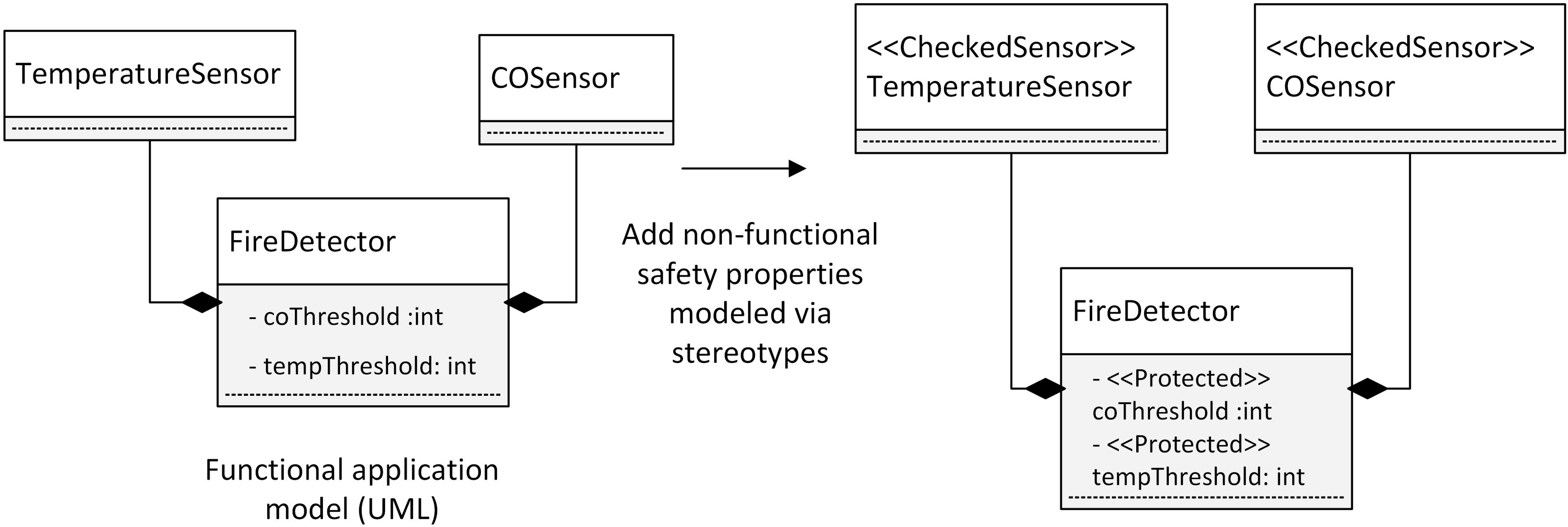
Modeling



```
class FireDetector{
        //Generated Code
}
```

Source code

Overview - I



TemperatureSensor

COSensor

FireDetector

- coThreshold :int
- tempThreshold: int

Add non-functional safety properties modeled via stereotypes

<<CheckedSensor>> TemperatureSensor

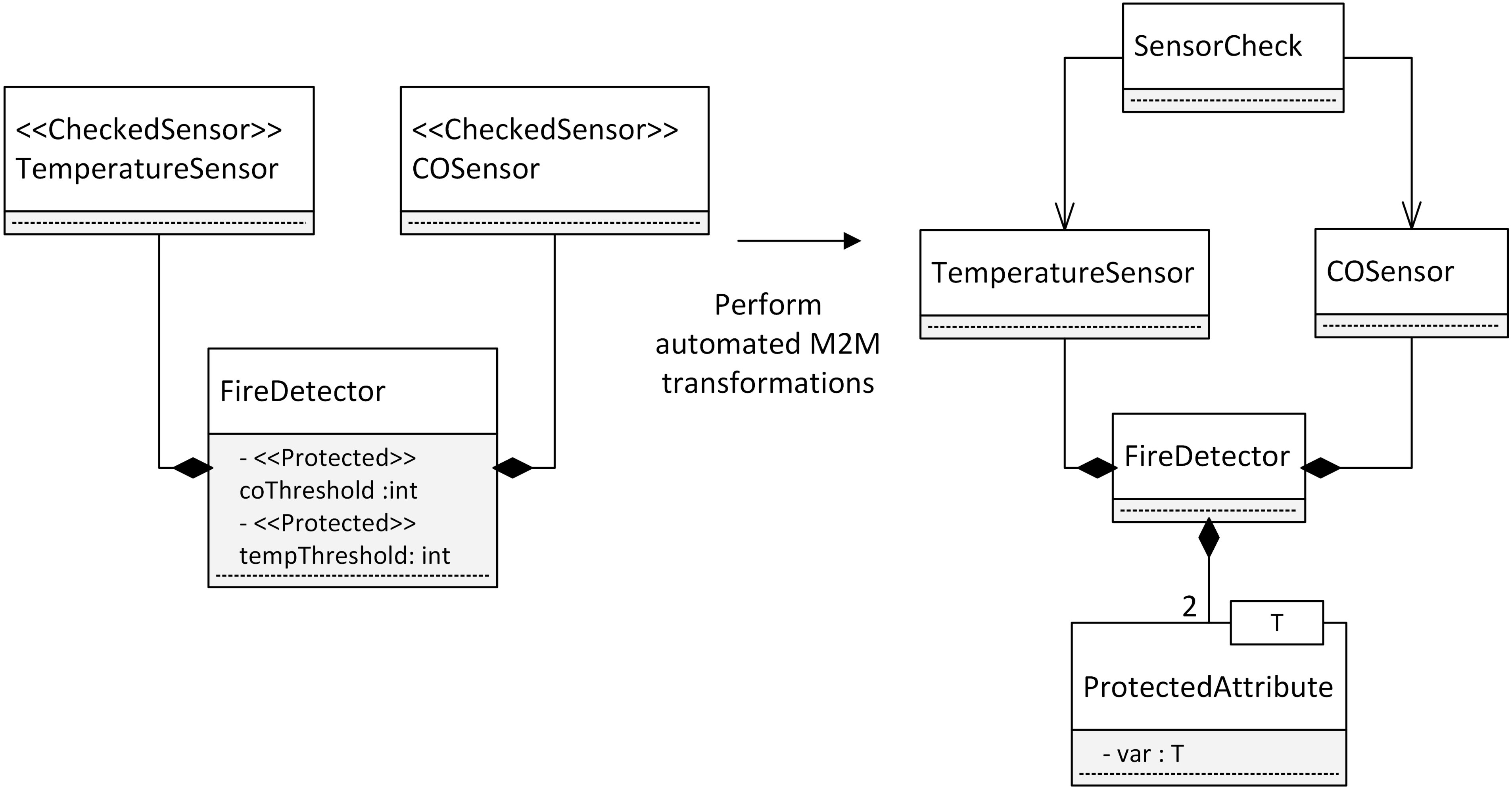
<<CheckedSensor>> COSensor

FireDetector

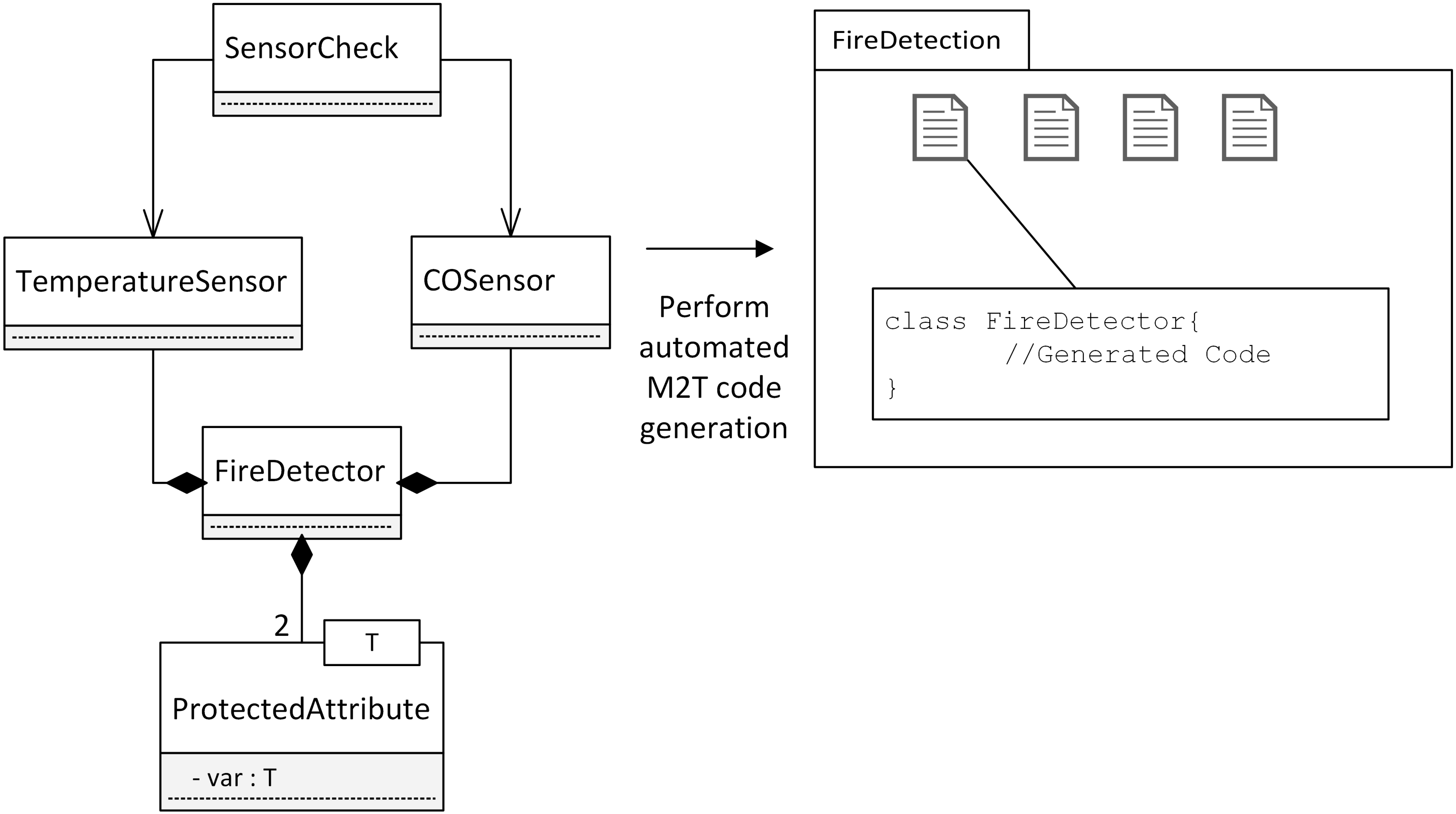
- <<Protected>> coThreshold :int
- <<Protected>> tempThreshold: int

Functional application model (UML)

Overview - II


<<CheckedSensor>>
TemperatureSensor
<<CheckedSensor>>
COSensor
FireDetector
- <<Protected>>
coThreshold :int
- <<Protected>>
tempThreshold: int
Perform
automated M2M
transformations
SensorCheck
TemperatureSensor
COSensor
FireDetector
2
T
ProtectedAttribute
- var : T

Overview - III



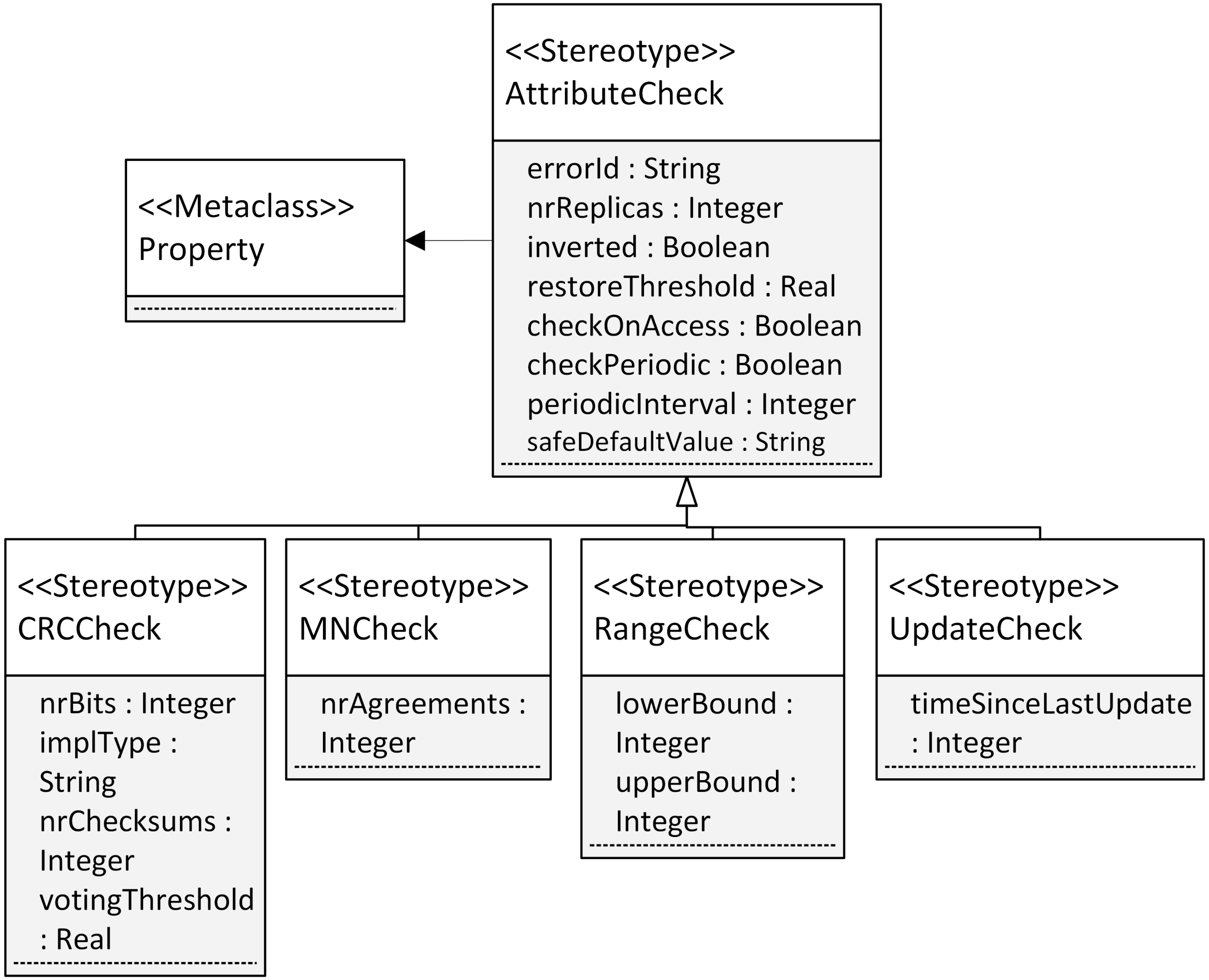
Perform automated M2T code generation

```
class FireDetector{
        //Generated Code
}
```

Related Work

- Basic UML → C++ MDD Tools (Rhapsody, 2020; Enterprise Architect, 2020)
- Advanced code generation for UML (Sunitha and Samuel, 2019)
- Modeling of embedded aspects (MARTE, 2008; Bernadi et al., 2011; SAFURE, 2017)
- MDD Generation of other safety mechanisms (Huning et al., 2019; Huning et al., 2020)
- Code generation for error detection (Trindade et al., 2014; Pezze and Wuttke, 2016)

UML Profile for error detection mechanisms



<<Metaclass>>
Property

<<Stereotype>>
AttributeCheck

errorId : String
nrReplicas : Integer
inverted : Boolean
restoreThreshold : Real
checkOnAccess : Boolean
checkPeriodic : Boolean
periodicInterval : Integer
safeDefaultValue : String

<<Stereotype>>
CRCCheck

nrBits : Integer
implType : String
nrChecksums : Integer
votingThreshold : Real

<<Stereotype>>
MNCheck

nrAgreements : Integer

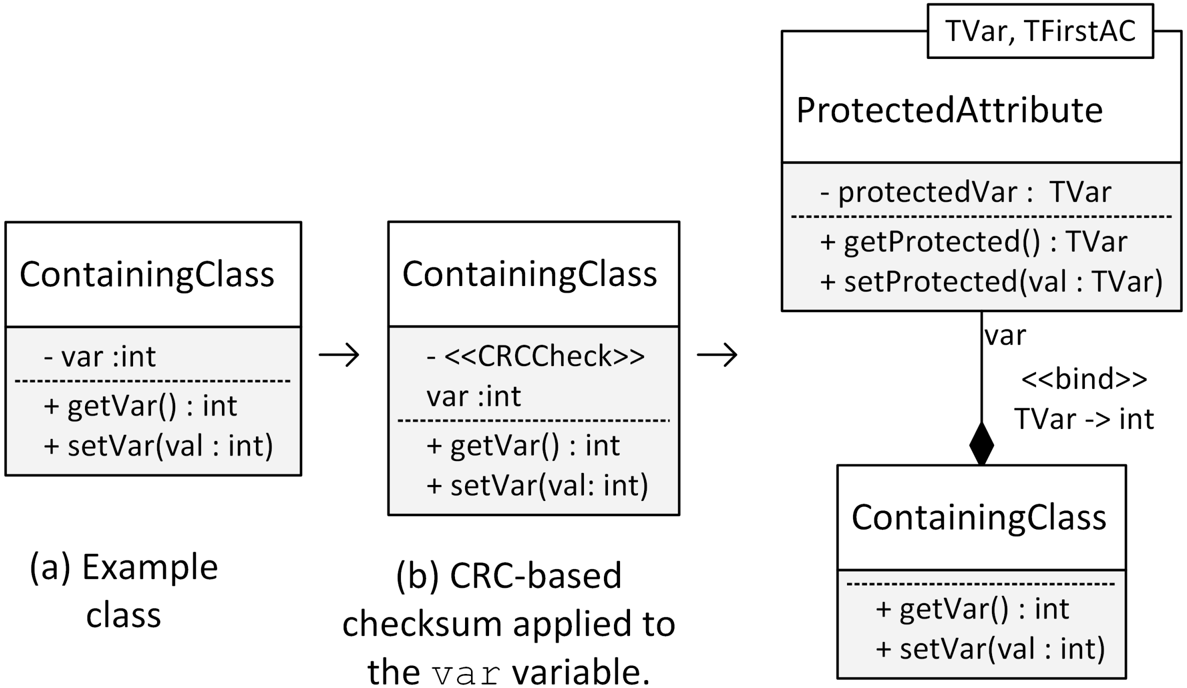
<<Stereotype>>
RangeCheck

lowerBound : Integer
upperBound : Integer

<<Stereotype>>
UpdateCheck

timeSinceLastUpdate : Integer

Basic concept for transparent generation



ContainingClass
- var :int
+ getVar() : int
+ setVar(val : int)

→

ContainingClass
- <<CRCCheck>> var :int
+ getVar() : int
+ setVar(val: int)

→

TVar, TFirstAC
ProtectedAttribute
- protectedVar : TVar
+ getProtected() : TVar
+ setProtected(val : TVar)

var
<<bind>>
TVar -> int

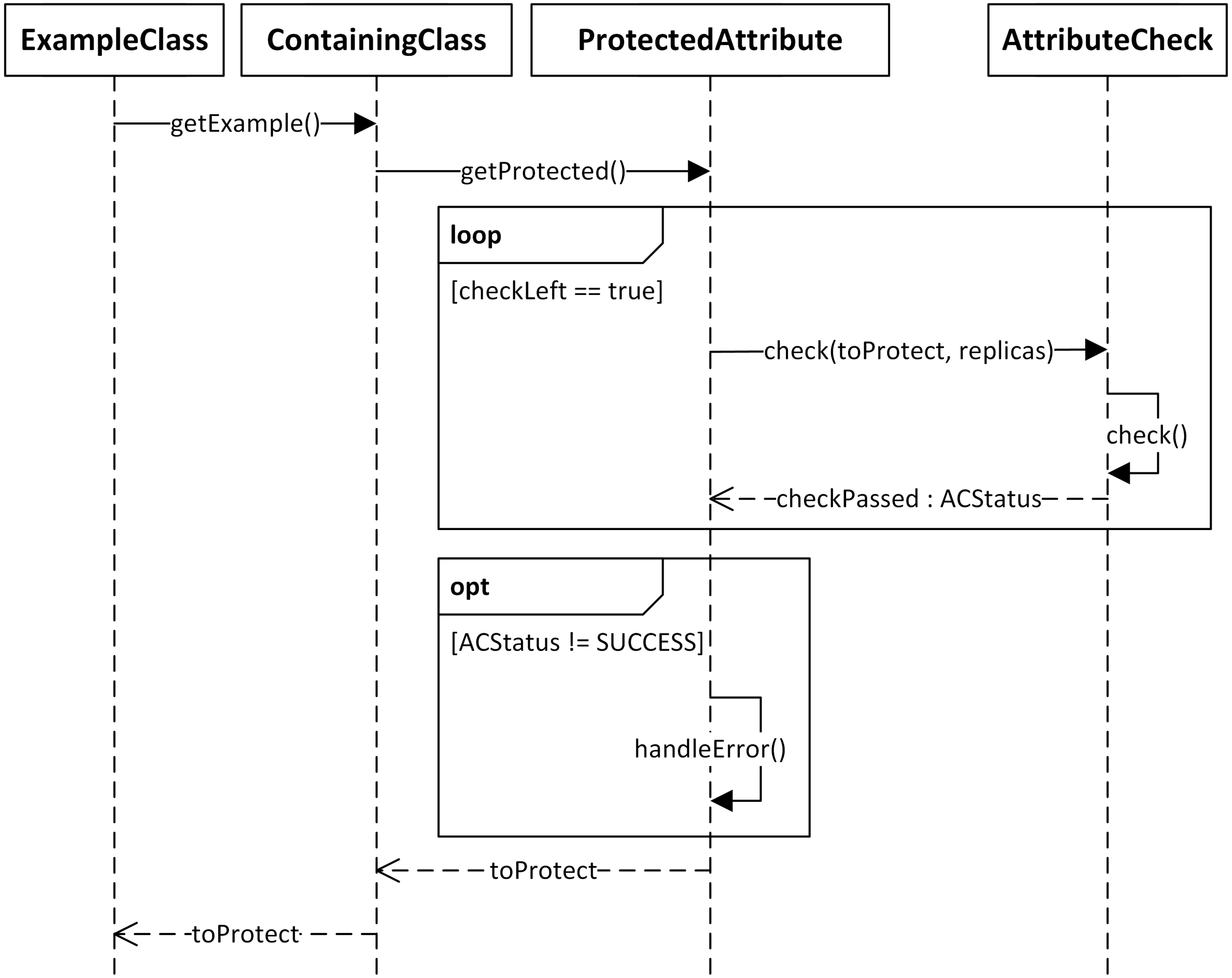
ContainingClass
+ getVar() : int
+ setVar(val : int)

(a) Example class

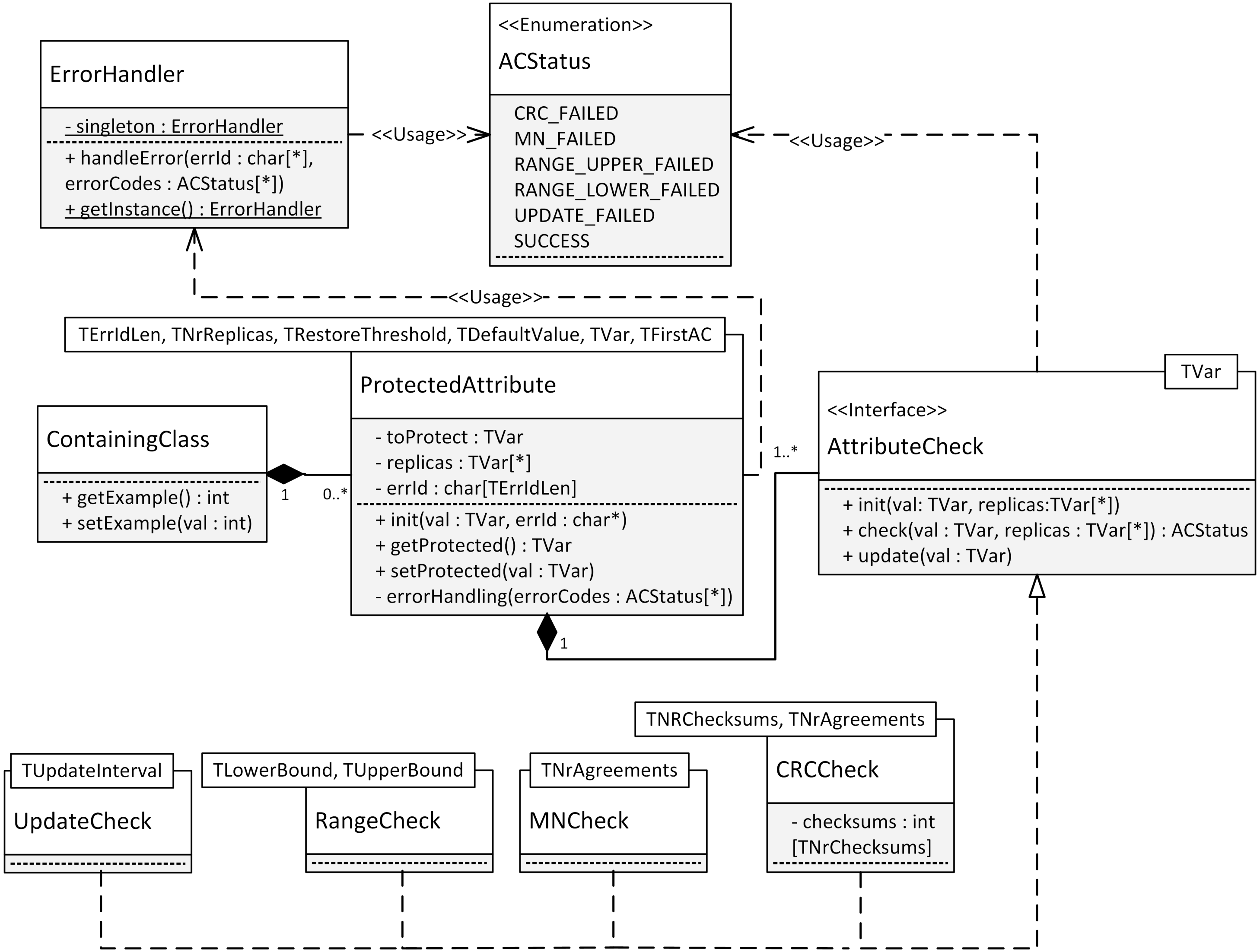
(b) CRC-based checksum applied to the `var` variable.

(c) Primitive `var` variable replaced by wrapper class (`ProtectedAttribute`) that contains a CRC-based error detection mechanism.

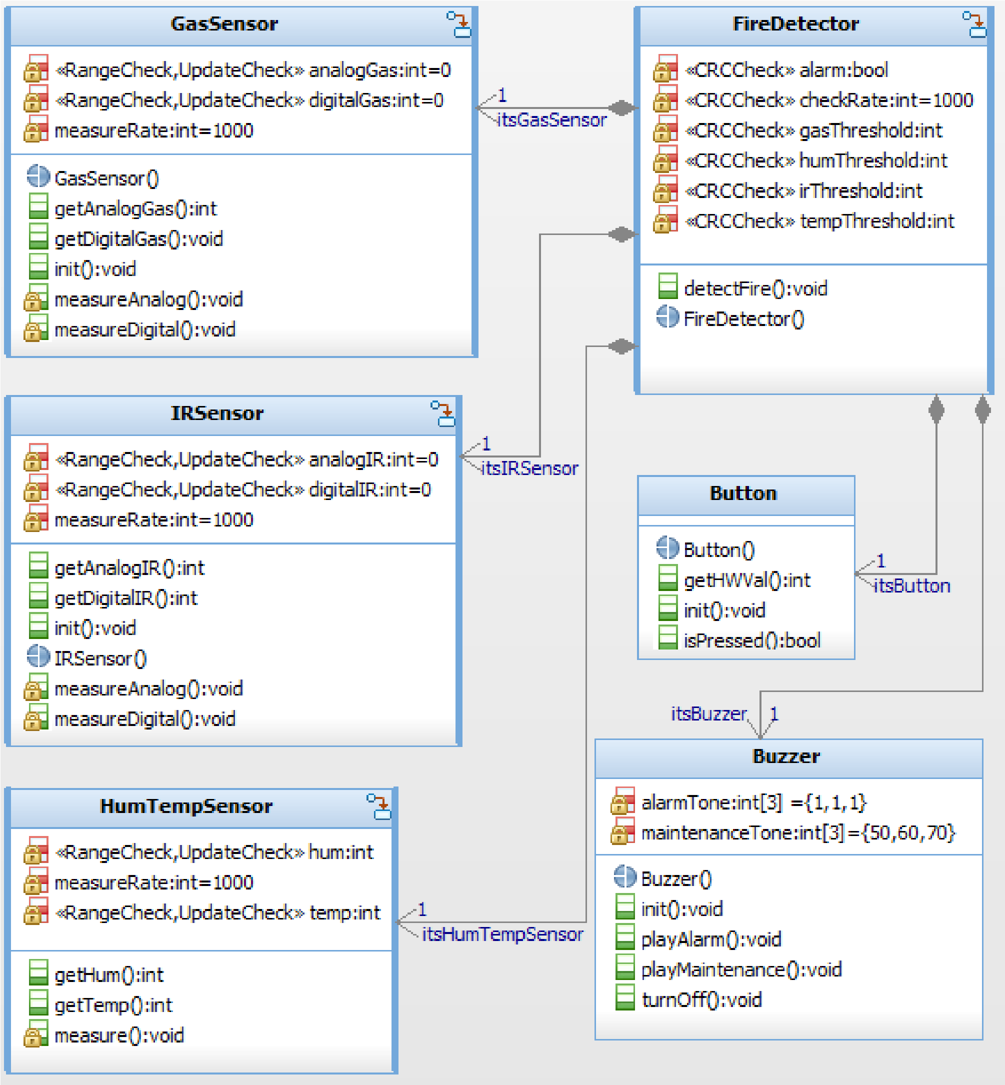
Runtime behavior



Software Architecture


ErrorHandler
- singleton : ErrorHandler
+ handleError(errId : char[*], errorCodes : ACStatus[*])
+ getInstance() : ErrorHandler
<<Usage>>
<<Enumeration>>
ACStatus
CRC_FAILED
MN_FAILED
RANGE_UPPER_FAILED
RANGE_LOWER_FAILED
UPDATE_FAILED
SUCCESS
<<Usage>>
<<Usage>>
TErrIdLen, TNrReplicas, TRestoreThreshold, TDefaultValue, TVar, TFirstAC
ProtectedAttribute
- toProtect : TVar
- replicas : TVar[*]
- errId : char[TErrIdLen]
+ init(val : TVar, errId : char*)
+ getProtected() : TVar
+ setProtected(val : TVar)
- errorHandling(errorCodes : ACStatus[*])
ContainingClass
+ getExample() : int
+ setExample(val : int)
1
0..*
1
1..*
TVar
<<Interface>>
AttributeCheck
+ init(val: TVar, replicas:TVar[*])
+ check(val : TVar, replicas : TVar[*]) : ACStatus
+ update(val : TVar)
TUpdateInterval
UpdateCheck
TLowerBound, TUpperBound
RangeCheck
TNrAgreements
MNCheck
TNRChecksums, TNrAgreements
CRCCheck
- checksums : int [TNrChecksums]

Use Case: Development of a Fire Detection System



GasSensor

«RangeCheck,UpdateCheck» analogGas:int=0
«RangeCheck,UpdateCheck» digitalGas:int=0
measureRate:int=1000

GasSensor()
getAnalogGas():int
getDigitalGas():void
init():void
measureAnalog():void
measureDigital():void

FireDetector

«CRCCheck» alarm:bool
«CRCCheck» checkRate:int=1000
«CRCCheck» gasThreshold:int
«CRCCheck» humThreshold:int
«CRCCheck» irThreshold:int
«CRCCheck» tempThreshold:int

detectFire():void
FireDetector()

IRSensor

«RangeCheck,UpdateCheck» analogIR:int=0
«RangeCheck,UpdateCheck» digitalIR:int=0
measureRate:int=1000

getAnalogIR():int
getDigitalIR():int
init():void
IRSensor()
measureAnalog():void
measureDigital():void

Button

Button()
getHWVal():int
init():void
isPressed():bool

HumTempSensor

«RangeCheck,UpdateCheck» hum:int
measureRate:int=1000
«RangeCheck,UpdateCheck» temp:int

getHum():int
getTemp():int
measure():void

Buzzer

alarmTone:int[3] ={1,1,1}
maintenanceTone:int[3]={50,60,70}

Buzzer()
init():void
playAlarm():void
playMaintenance():void
turnOff():void

Conclusion and Future Work

- Conclusion:
  - Specify error detection mechanism with UML stereotype
  - Source code for mechanism is generated automatically
  - Generation is transparent due to wrapper class
- Future Work:
  - Performance Evaluation
  - More safety mechanisms
  - Combination with safety assurance cases for certification
  - Apply concept to other non-functional properties

References

Rhapsody (2020). IBM. Rational Rhapsody Developer. https://www.ibm.com/us-en/marketplace/uml-tools (accessed 20th August 2020).

Enterprise Architect (2020). Enterprise Architect. https://sparxsystems.com/products/ea/index.html (accessed 20th August 2020).

Sunitha, E. and Samuel, P. (2019). Automatic Code Generation From UML State Chart Diagrams. IEEE Access, 7:8591–8608.

OMG MARTE (2008). A UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded Systems. Technical report, Object Management Group.

Bernardi, S., Merseguer, J., and Petriu, D. (2011). A dependability profile within MARTE. Software and System Modeling, 10:313–336.

SAFURE (2017). Architecture models and patterns for safety and security. Deliverable D2.2 from EU-research project SAFURE. https://safure.eu/publicationsdeliverables (accessed 20th August 2020).

References

Huning, L., Iyenghar, P., and Pulvermueller, E. (2019). UML specification and transformation of safety features for memory protection. In Proceedings of the 14th International Conference on Evaluation of Novel Approaches to Software Engineering, Heraklion, Crete, Greece. INSTICC, SciTePress.

Huning, L., Iyenghar, P., and Pulvermueller, E. (2020). A UML profile for automatic code generation of optimistic graceful degradation features at the application level. In Proceedings of the 8th International Conference on Model-Driven Engineering and Software Development, MODELSWARD, Valetta, Malta. INSTICC, SciTePress.

R. Trindade, L. Bulwahn, and C. Ainhauser, “Automatically generated safety mechanisms from semi-formal software safety requirements,” in Computer Safety, Reliability, and Security, A. Bondavalli and F. Di Giandomenico, Eds. Cham: Springer International Publishing, 2014, pp. 278–293.

M. Pezze and J. Wuttke, “Model-driven generation of runtime checks for system properties,” International Journal on Software Tools for Technology Transfer, vol. 18, no. 1, Feb 2016, pp. 1–19.