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)
- 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

TemperatureSensor

COSensor

FireDetector

Modeling

Safety standards

class FireDetector{
    // Generated Code
}

Source code

Functional safety of electrical/electronic/programmable electronic safety-related systems — Part 2: Software requirements

IEC 61508-3 Edition 2.0 2009-04

INTERNATIONAL STANDARD
NORME INTERNATIONALE
Overview - I

Functional application model (UML)

- 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
Overview - II

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

- **SensorCheck**
- **TemperatureSensor**
- **COSensor**

Perform automated M2M transformations

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

- **ProtectedAttribute**
  - - var : T

Software Engineering Group
Overview - III

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

TemperatureSensor

SensorCheck

COSensor

FireDetection

FireDetector

Perform automated M2T code generation

class FireDetector{
   //Generated Code
}

ProtectedAttribute

- var : T

Software Engineering Group
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

(a) Example class
- var : int
  + getVar() : int
  + setVar(val : int)

(b) CRC-based checksum applied to the var variable.
- <<CRCCheck>>
  var : int
  + getVar() : int
  + setVar(val : int)

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

- protectedVar : TVar
  + getProtected() : TVar
  + setProtected(val : TVar)

ProtectedAttribute

TVar, TFirstAC

var
<<bind>>
TVar -> int
Runtime behavior

```
ExampleClass  ContainingClass  ProtectedAttribute  AttributeCheck

        getExample() -->
        getProtected() -->

loop
[checkLeft == true]

check(toProtect, replicas) -->
check() -->

¬checkPassed : ACStatus ←

opt
[ACStatus != SUCCESS]

handleError() ←

toProtect ←
```

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

Software Engineering Group
Software Architecture

```
ErrorHandler
- singleton : ErrorHandler
+ handleErr(errId : char[*], errorCodes : ACStatus[*])
+ getInstance() : ErrorHandler

ACStatus
<<Enumeration>>
- CRC_FAILED
- MN_FAILED
- RANGE_UPPER_FAILED
- RANGE_LOWER_FAILED
- UPDATE_FAILED
- SUCCESS

TErrIdLen, TNumReplicas, TRestoreThreshold, TDefaultValue, TVar, TFirstAC

ContainingClass
- getExample() : int
+ setExample(val : int)

ProtectedAttribute
- toProtect : TVar
- replicas : TVar[*]
- errId : char[TErrIdLen]

+ init(val : TVar, errId : char[*])
+ getProtected() : TVar
+ setProtected(val : TVar)
- errorHandling(errorCodes : ACStatus[*])

<<Interface>>
AttributeCheck
- init(val : TVar, replicas : TVar[*])
- check(val : TVar, replicas : TVar[*]) : ACStatus
- update(val : TVar)

TUpdateInterval, TLowerBound, TUpperBound
UpdateCheck

RangeCheck

TNumAgreements

MNCheck

CRCCheck
- checksums : int
- [TNumChecksums]

TNumChecksums, TNumAgreements
```
Use Case: Development of a Fire Detection System
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


References


