Towards Context-Aware Propagators

Language Constructs for Context-Aware Adaptation Dependencies

Engineer Bainomugisha, Wolfgang De Meuter, Theo D'Hondt

Programming Technology Lab, Vrije Universiteit Brussel, Belgium
Goal

Ensuring consistency during system adaptation process

- Language support for selection of applicable adaptations
- (Re)definition of adaptation dependencies
A Context-Aware System

Context-aware Fire Sprinkler System (CaFSS)

D
Combustible metals

C
Electrical fires

B
Flammable liquids

A
Paper, wood

Class of fire

Outdoor
Location

Temperature

3
A Context-Aware System

Context-aware Fire Sprinkler System (CaFSS)

Class of fire:
- **A**: CO2
- **B**: Flammable liquids
- **C**: Electrical fires
- **D**: Combustible metals

Location:
- Outdoor

Temperature:
- 405°
A Context-Aware System

Context-aware Fire Sprinkler System (CaFSS)

Class of fire:
- A: Electrical fires
- B: Flammable liquids
- C: Combustible solids
- D: Electrical fires

Location:
- Outdoor

Temperature:
- 405°C
A Context-Aware System

Context-aware Fire Sprinkler System (CaFSS)

Class of fire:
- A: Flammable liquids
- B: Paper, wood
- C: Electrical fires
- D: Combustible metals

Temperature:
- Sprinkle at maximum
- Send SMS
- Trigger alarm

Location:
- Outdoor

- Powder
- Water
- CO2
- Foam
Existing Work

• Context-Oriented Programming (COP) (Hirschfeld et.al.)
  – Programming language support

• Context-Oriented Domain Analysis (CODA) (Desmet et.al.)
  – High-level modelling approach for context-aware systems
CaFSS using CODA

- CO2
- Foam
- Water
- Powder

Adaptation
CaFSS using CODA

CaFSS

- fire class A
- fire class B
- fire class C
- fire class D

CO2
Foam
Water
Powder

Context
Adaptation
CaFSS using CODA

Adaptation dependencies

- CO2
- Foam
- Water
- Powder

Excludes

Context

Adaptation
CaFSS using CODA

Issues

• Fixed dependencies e.g. CO2 always excludes Foam
• Expressing all possibilities becomes cumbersome
Context-Aware Dependencies

CO2

Foam

Excludes

D1

Includes

D2

Requires

D3

Foam

Outdoor

405°
Context-Aware Dependencies

• Multiple dependencies can coexist

\[ D := (D_1 \; D_2 \; D_3) \]
Context-Aware Dependencies

• Multiple dependencies can coexist

\[ D := (D_1 D_2 D_3) \]

Contradictions
System Adaptation Issues

Need for Explicit Support for:
System Adaptation Issues

Need for Explicit Support for:

• Expression of context and adaptations
System Adaptation Issues

Need for Explicit Support for:

• Expression of context and adaptations

• (Re)Definition of adaptation dependencies
System Adaptation Issues

Need for Explicit Support for:

• Expression of context and adaptations
• (Re)Definition of adaptation dependencies
• Management of dependencies’ contradictions
Context-Aware Propagators

Propagators model (Radul, & Sussman, 2009)

- Conventional programming

- One source for a variable’s value
Context-Aware Propagators

• Conventional programming

- One source for a variable’s value

• Propagators model
  (Radul, & Sussman, 2009)

+ Multiple sources for the value
+ Coexisting values for a variable
Context-Aware Propagators

Ci → P₁ → Ai → P₂ → Di

Context information → Mappings → Adaptation → Dependency definition → Dependencies
Context-Aware Propagators

Adaptations Reasoner

Ci → P₁ → Ai → P₂ → Di

- Ci: Context information
- P₁: Mappings
- Ai: Adaptation
- P₂: Dependency definition
- Di: Dependencies
CaFSS using Propagators

(define kb (make-cell))
(define class-of-fire (make-cell))
(define temp (make-cell))
(define location (make-cell))
(define dependencies (make-interface-cell))
(CaFSS-reasoner
  kb class-of-fire temp location dependencies)

(content dependencies)
;;;Cell Value >>No Adaptation available

(add-content kb *context-adaptation-kb*)
(add-content class-of-fire 'Class-B-Fire)
(content dependencies)
;;;Cell Value: (CO2)
CaFSS using Propagators

$P_m$ - Mappings propagator

$P_d$ - Dependencies propagator

Interface cell
CaFSS using Propagators

\( P_m \) - Mappings propagator
\( P_d \) - Dependencies propagator
CaFSS using Propagators

$P_m$ - Mappings propagator
$P_d$ - Dependencies propagator
CaFSS using Propagators

$P_m$ - Mappings propagator

$P_d$ - Dependencies propagator

Diagram showing the relationship between Fire A, $P_m$, $P_d$, and Interface cell with $CO_2$. The $P_m$ and $P_d$ nodes are connected to each other and to the $CO_2$ nodes, indicating the flow of information or data between these components.
CaFSS using Propagators

$P_m$ - Mappings propagator
$P_d$ - Dependencies propagator
CaFSS using Propagators

P\textsubscript{m} - Mappings propagator
P\textsubscript{d} - Dependencies propagator
CaFSS using Propagators

\[ P_m - \text{Mappings propagator} \]
\[ P_d - \text{Dependencies propagator} \]
CaFSS using Propagators

Fire A

Fire B

\( P_m \) - Mappings propagator

\( P_d \) - Dependencies propagator

CO2

Foam

405°

Interface cell

D1: CO2- XFoam

Pm - Mappings propagator

Pd - Dependencies propagator
CaFSS using Propagators

- Dependencies change depending on the context
- Multiple dependencies may contradict each other
Managing Dependencies’ Contradictions

- Dependency “Decorations”
- Multidirectional Computation
- Merging
Summary

- Ensuring consistency through adaptation dependencies

- Context-aware dependencies: multiple dependencies may coexist

- Managing contradictions using propagators
Ongoing

- Further investigation on constraint propagation
- Implementation
- Propagators in a distributed setting
- Application to other scenarios
Thank you

Questions?

ebainomu@vub.ac.be

http://prog2.vub.ac.be/~ebainomu/
References

