A component-based approach for Context-Aware Systems specification

Brahim Djoudi, Chafia Bouanaka, Nadia Zeghib
Plan

• Introduction

• Context-Aware Systems Specification Framework
  • Context-aware Systems Specification
  • A MAUDE-based Language For Context-aware Adaptive Systems Specification
  • Context-Aware Management Environment

• Conclusion
Introduction

Context-aware Adaptive

The basic idea of Context-aware Adaptive systems is that they anticipate – based on context – what system functionalities the user will need and execute.
Introduction

A number of researches have been invested in validating and verifying system adaptation behavior. These studies are based on:

- Tree-based location models
- Ambient Calculus
- Bigraphs theory
- Colored Petri Nets
- Graph-based approach
A **generic** model for Context-aware systems should satisfy the following requirements:

• The model should provide a reusable context infrastructure.
• It might facilitate representing context information and its relationship with system entities.
• It might conform to user requirements.
• The model should be doted with a formal basis to perform the required reasoning for correctness and model consistency.
Context-Aware Systems Specification Framework
Context-Aware Systems Specification Framework

Our main technical contribution consists of defining a systematic process for context-aware systems modelling. In particular, the proposed model provides:

⇒ A meta-layer on top of Maude for specifying relevant contextual information by defining a Maude-based domain specific language called **CTXs-Maude**,

⇒ A dynamic runtime management for context/system interaction.
Context-aware Systems Model

✓ A layered model is defined

✓ To simplify context handling and processing
An Adequate Formal Basis for Context-aware systems specification

A reasonable and desirable formal method to use for this scope should have the following important features:

• It should be powerful enough to capture the principal models of computation and specification,
• It should be executable in order to validate model semantics,
• It should be able to work at different levels of abstraction,
• It should provide a refinement mechanism so that one can focus on a few concepts at a time and (possibly) deliver the language semantics specification at different refinement phases.

Maude Language

Meta-Programming

Reflectivity
Since Maude (Clavel et al., 2008) allows specifying modules with user-definable syntax by exploiting its reflection and meta-programming properties, we define a domain specific language on top of core Maude that introduces new constructors to allow specifying context-aware adaptive systems.

We Propose A New Language, Called Ctxs-maude (Context-aware Systems Using Maude)
Running Example: Pervasive Computing Smart Home

• All concepts introduced in our model are illustrated through an example of a Smart home context-aware system. Smart home is a kind of future home where pervasive computing emerges anywhere to facilitate our daily life.
Running Example: Pervasive Computing Smart Home

• The considered Smart home is composed of two sub-systems:
  • The Fixed system: is an intelligent system able to perform a universal remote control on all devices within the house. It can be used to lock, unlock all the doors and windows; it can be used to set alarm automatically based on the schedule fixed on the smart PDA etc.
  • Smart PDA (Personal Digital Assistant) or Smart phone. Besides, the fixed system is permanently connected to the hand held smart PDA, allowing it to access smart house devices.
Functional Layer

- A set of **Components** that provide system core functionalities.
- A component comprises a set of **Provided/Required Ports**, each one containing a set of interfaces for **Services** required or provided by the component.
- **Components Connections** are dynamically established whenever one component is providing the service and the other is requesting it.
- A **Configuration** is an instance of the defined **Architecture**, containing instances of already declared components types.
Functional Layer

Components:

```vhdl
component CptID1 is PortID1 endcp
component CptID2 is PortID2 endcp
    ....
architecture Archid is CptID1 CptID2 endarch
```

Example:

```vhdl
component PDA is PS endcp
component FS is SE ... endcp .
    ....
architecture SH is PDA FS Wr_Watch... endarch
```
Functional Layer

Provided Ports

```
inport PortID is
  provides Service: parameter -> result.
  ...
Implementation ImpModule endp
```

Example

```
inport SE is
  provide Lightning :
    SHStat -> String.
  implementation SHImpl endp
```
Functional Layer

Provided Ports

outport PortID is
requests Service: parameter ->
result.
    ... endp.

Example

outport PS is
request ChangeWakeUp:
    SHStat -> String.
endp
CTXs-MAUDE Context element

Context IdContext is
 CTXDescrip: /*Sensor URL .*/
 CTXState: /*Context States*/
    State:
    CTXValue -> /*Context Value.*/.
    Actions:
    /*Action declaration.*/.
    endact
    endState

Example

Context HealthState is
  CTXDescrip: Monitoring Heartbeat, blood pressure levels of the body.
  CTXState:
    State: CTXValue -> AbnormalState.
    Actions:
    ExeAction invoke /IdInstance= wr_watch /Port= HS /Request: SendSignal(AbNormal)
    endact
    endState
  endctxState
endctx
Some adaptation operations require the presence of two or more different contexts states. This is declared by HighCTX.
**CTXs-MAUDE Composite Context**

```
HighCTX IdContext is /* HighContext */

HCTXState StateID: /* HighStates */
  BCTXStates :/* BasicContext States*/
    basic ContextID1 '/ Value ' and
    basic ContextID2 '/ Value '.
  ...
  Actions: /*Action declaration.*/.
  endact
endHCTXSt
...
endHctx

HighCTX Auto-lighting is
  HCTXState Lighting:
    BCTXStates :
      Location / Room_Entrance and
      Environment / Dark.
    Actions:
      ExeAction invoke /IdInstance= PDA
      /Port= PH /Request: Lightening (Room).
  endact
endHCTXSt
endHctx
```

Example
Actions clause is a parameter used to generate an on the fly strategy to be performed on system structure and behaviour. Two categories of actions are defined:

- The **first category** acts on system state by enforcing it to **execute specific operations**; and is declared using the keyword `ExeAction`.

- The **second category** acts on **system structure** to modify its actual configuration; it uses keywords `CptAction`, `PrtAction`...; for adding new instances, ports, connections, removing an existing one and so on.
Behavourial Action Syntax

Actions :

ExeAction_/*Action name*/. 
/IdInstance=/*Instance Name*/. 
/Port=/*Port identifier*/. 
/Request: /*Service call*/.
Endact

Example

Actions :

ExeAction invoke 
/IdInstance = PDA 
/Port = PS 
/Request: ChangeWackUp (Time). 
Endact
CTXs-MAUDE Adaptation Action Specification

**Structural Action Syntax**

Actions:

CptAction Actionid /*Action Name*/.

/Component (_)/ /*Component Type*/.

Set IdInstance = _. /*Instance id*/.

Endact

**Example**

Actions:

CptAction NewInstance

/Component (air-conditioner)

Set IdInstance = AC.

Endact
**Port Action Syntaxe**: 

Actions:

```
PrtAction Actionid /*Action Name*/
/Port (_) /*Port identifier*/.
inComponent(_). /*Component Type*/.
```

Endact

**Example**: 

Actions:

```
PrtAction AddPrt
/Port = SE3
   inComponent( FS).
Endact
```
Context-Aware Management Environment

To allow context management and establishing dynamic interaction canals between instances of context and functional components, the management layer is composed of two independent interfaces:

- Context interface,
- Functional interface.
Context-Aware Management Environment

Context Loop is composed of:

- **Input**
  - System/Context commands
  - System data specification

- **Context Module, Functional Loop**
  - CTXs-Maude
  - Context Specification

- **Output**
  - System State
  - Resulting data
Context-Aware Management Environment

Functional Loop is composed of:

Input → System Module, Configuration → Output

where a configuration is described by:

ConfigID – Instances- Connections
Context/Functional-levels interactions

The Interactions between Context and Functional layers is realized via on-the-fly strategies that are dynamically generated from Context and system states information.

![Diagram showing the process of Context/Functional-levels interactions](image)
Context/Functional-levels interactions

A set of operations is defined to generate On the fly strategies:

⇒ checkCtx and checkValue to check if the newly introduced context identifier and the corresponding context value exist in the context module declaration.
ActionProcessing: It recursively applies adaptation actions on the current system state using metaXapply operation,

\[
\text{eq ActionProcessing}(
'\text{ExeAction actionId '/IdInstance= idins '/Port= idprt '/Requet: req '},
\text{RestActionCode , SystemStat ) = ActionProcessing} \left( \text{RestActionCode , downTerm( getTerm(}
\text{metaXapply( upModule( 'FMod ,false),}
\text{SystemStat, actionId ,none,0, unbounded,0})
\right), SystemStat \right).
\]
We consider the situation when the wrist watch records any reading that is out of the threshold limits.

**Context**: HealthState/Value: Abnormal_State.

This situation triggers a SendSignal service that sends a signal to the fixed system, and automatically alerts user and/or searches for a nearby hospital using GPS to report that there is an emergency.
Exemple 2

• User work office schedules an emergency meeting at night.

• The user has an open or free slot.

• By scheduling the early slot in user day, his Smart PDA asks the fixed system of the smart home to subsequently change user wakeup time but being still within the permitted boundaries.

• The fixed system sets the alarm on based on the location where user sleeps.
Our Approach Advantages

Compared to existing approaches, our model has the following advantages:

• The separation of concerns between context model elements and system model reduces the complexity and increases model reusability and maintainability.

• The introduced specification language CTX-Maude supports modelling context-aware systems hierarchical structures.
Our Approach Advantages

• The description and the representation of context and adaptation tasks become easier using CTXs-Maude.

• The Management runtime layer allows users to select adaptation action to be applied without caring about how these actions are implemented and executed. This feature gives users great facility and flexibility when using CTXs-Maude.
THANKS' FOR YOUR ATTENTION