Architectural Support for Context-Aware Applications: From Context-Models to Services Platforms

10/05/2012
Patrícia Dockhorn Costa
pdcosta@inf.ufes.br
www.inf.ufes.br/~pdcosta
Healthcare Scenario

- Mr. Janssen suffers from epilepsy
  - limitations in lifestyle
  - need of constant supervision
- Healthcare application
  - detects upcoming seizures
  - informs caregivers
  - increases quality of life
Context-Aware Application

- **Context**
  - the set of possibly interrelated conditions in which an entity exists
Context Concepts

- Context
- Context Model
- Context Modelling
- Context Information
- Context-Aware Application
- Situations
- Quality of Context
Context Concepts

- **Entity** 1..n
- **Context** 1..n
  - **Condition** 1..n
    - **Situation**
  - **Context information**
  - **Condition value**
    - **Situation information**
  - **Quality**

- **objects in the real world**
- **objects in the application**

Relations:
- Entity represents Context
- Context represents Condition
- Condition represents Situation
- Situation represents Context information
- Condition value represents Situation information
- Quality is related to objects in the application
Challenges

• Capturing context
  – aggregation
  – reasoning
  – inference

• Time sensitive

• Sensors are imperfect (Quality of Context)

• Sensors are distributed

• Applications are distributed (mobile)

• Application adaptation, reactivity

• Security, privacy
Objective

• Integrated solution for the development of context-aware applications:
  – Reference Architecture
  – Context Handling Platform
  – Context Modelling
Reference Architecture

- Context-Aware Patterns
  - Event-Control-Action pattern
  - Context Sources and Managers Hierarchy Pattern
  - Actions Pattern
Reference Architecture

• Components and Interfaces
  – Context sources
  – Context managers
  – Controllers
  – Action components
Context Handling Platform

end-user

user interaction

context information

context-aware application

generic!
Context Handling Platform

- Generalizes functionality that can be reused by several context-aware applications
- **Gathers context** information, performs context processing
- **Detects situations** in a distributed fashion
- Allows delegation of **application rules**
- Performs **adaptation** on behalf of applications
Context Handling Platform

(a)

user¹

user’s context¹

ip-a¹

ip-c¹

ip-d¹

application-specific components

ip-e¹

context handling platform

(b)

focusing on infrastructure interactions

user’s context

ip-d

application-specific components

ip-e

context handling platform
Context Handling Platform

application\(^1\)-specific components

context manager

context source\(^1\)

controller\(^1\)

action resolver\(^1\)

external action provider\(^1\)

user's context

external context processor

application\(^2\)-specific components

controller\(^2\)

action resolver\(^2\)

external action provider\(^n\)
Context Modelling

- location
- proximity
- temperature
- medical conditions

context-aware application
- locationValue (lat, long, alt)
- temperatureValue (Celsius)
- heartBeatValue (ECG)

models

end-user

user interaction

context information

context-aware application

• locationValue (lat, long, alt)
• temperatureValue (Celsius)
• heartBeatValue (ECG)
Context Modelling

- Abstractions to facilitate representation of context information
- **Situation** modelling (aggregation, reasoning)
- Automatic **code generation**
- Quality of Context (QoC)
Application’s universe of discourse and state-of-affairs

- Tourist Application

John has access to a Bluetooth network.
Context Modelling Requirements

• In order to support context-aware applications one needs amongst others (meta)models that define
  – Context and situation types and their relationships
  – The “imperfection” of context information (Quality of context)
  – Adaptation rules based on context and situations

• Context models should:
  – Support common understanding, problem-solving, and communication among the various stakeholders involved in application development
  – Represent context unambiguously
Goal of Context Modelling

- Provide basic **conceptual foundations** for context modeling, which allow context-aware application designers to represent (i) relevant elements of a context-aware application’s **universe of discourse**; and (ii) particular **state-of-affairs** of interest.

- We consider results from **foundational ontologies** to support our conceptual context modeling approach.
Foundational ontologies: related work
Foundational Context Concepts

- **Context**
  - the set of possibly interrelated *conditions* in which an *entity* exists
Foundational Context Concepts: situation

```
Substantial Universal

Entity

Context

isContextOf {subsets inheres in} hasContext {subsets bears}

1..* entities 1..* contexts

Situation Universal

Context Situation

+initialtime : Time
+finaltime : Time

Moment Universal
```
Foundational Context Concepts: situation

Diagram:

- **Context Situation**
  - **RelationalSituation**
    - **IntrinsicSituation**
    - **FormalRelationSituation**
  - **CombinedSituation**
    - **SituationOfSituations**
Foundational Context Concepts (UML profile)

- **Artifacts for specification**
  - **Context profile**: <<Entity>>, <<RelationalContext>>, <<IntrinsicContext>>, ...
  - **Situation profile**: <<IntrinsicSituation>>, <<FormalRelationSituation>, <<RelationalSituation>>, ...

- **Products of specification**
  - **Context Models**: person, Temperature, GeoLocation, GeoLocationCoordinates, Device, etc
  - **Situation Models**: SituationFever, SituationConnected, SituationPresentation, etc
## Foundational Context Concepts: summarized

<table>
<thead>
<tr>
<th>Foundational concepts</th>
<th>context concepts</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Entity</strong></td>
<td></td>
<td>an object that bears context</td>
</tr>
<tr>
<td><strong>Context</strong></td>
<td></td>
<td>a particular condition that inheres in an entity</td>
</tr>
<tr>
<td><strong>Intrinsic Context</strong></td>
<td></td>
<td>a particular type of context that belongs to the essential nature of a single entity</td>
</tr>
<tr>
<td><strong>Relational Context</strong></td>
<td></td>
<td>a particular type of context that depends on the relation between distinct entities</td>
</tr>
<tr>
<td><strong>Contextual Formal Relation</strong></td>
<td></td>
<td>a relation that holds directly between two or more entities’ intrinsic values (qualities)</td>
</tr>
<tr>
<td><strong>Context Situation</strong></td>
<td></td>
<td>a composite concept that defines particular application’s state-of-affairs. It can be composed of entities, contexts, and other situations</td>
</tr>
<tr>
<td><strong>Intrinsic Situation</strong></td>
<td></td>
<td>a context situation composed of a single entity and one of its intrinsic contexts</td>
</tr>
<tr>
<td><strong>Relational Situation</strong></td>
<td></td>
<td>a context situation composed of at least two entities and their pertinent relational contexts</td>
</tr>
<tr>
<td><strong>Formal Relation Situation</strong></td>
<td></td>
<td>a context situation composed of a single entity type and two or more of its intrinsic contexts</td>
</tr>
<tr>
<td><strong>Situation of Situations</strong></td>
<td></td>
<td>a context situation composed of other context situations</td>
</tr>
</tbody>
</table>
Context Models: Entity Types

```
Context Models: Entity Types

<<Entity>>
IntangibleEntity

{ disjoint }

Application
Network

<<Entity>>
SpatialEntity

{ disjoint }

PointEntity
ContainerEntity

{ disjoint }

Device
Person
Building
Room
Vehicle

Home
Office

+ livesAt
+ worksAt
{ overlapping }

+ livesAt
+ worksAt

* *
* *
Intrinsic Context Types

- «Entity»
  - SpatialEntity

- Person
  - hasHeartRate
  - hasMood
  - hasActivity
  - hasMentalState

- «IntrinsicContext»
  - HeartRate
  - Mood
  - Activity
  - MentalState

- Desire
- Belief
- Intention
Intrinsic Context Types

- **SpatialEntity**
  - hasGeoLocation
  - Device
    - hasBatteryPower
      - IntangibleEntity
        - hasBandwidth
          - Network
            - hasBandwidth
              - Bandwidth
              - BatteryPower
              - GeoLocation
Relational Context Types

- **SpatialEntity**
  - Device
    - hasNetworkAvailability: NetworkAvailability
    - hasChannelAvailability: ChannelAvailability
    - hasChannelAvailability
      - «Relational Context»: ChannelAvailability
      - isChannelAvailableTo
      - *
  - hasNetworkAvailability
    - «Relational Context»: NetworkAvailability
    - isNetworkAvailableTo
    - *

- **IntangibleEntity**
  - Network
    - Channel
      - isNetworkAvailableTo
      - *
      - isChannelAvailableTo
      - *
Relational Context Types

- «Entity»
  - SpatialEntity
  - ContainerEntity
  - Person
  - Device

- «Relational Context»
  - Containment
  - Social Network
  - DeviceAvailability

- hasContainment
- hasSocialNetwork
- hasDeviceAvailability
- isDeviceAvailableTo
- isContainedIn

1..*
1..*
1..*
1..*
1..*
Contextual Formal Relations

```
«Entity»
  SpatialEntity

hasGeoLocation

«IntrinsicContext»
  GeoLocation

+ coordinates

+ latitude : Real
+ longitude : Real
+ altitude : Real

+ nearness(in entity1 : GeoLocationCoordinates, in entity2 : GeoLocationCoordinates) : Boolean
+ distance(in entity1 : GeoLocationCoordinates, in entity2 : GeoLocationCoordinates) : Real
```
Contextual Formal Relations

**Datatype**: GeoLocation

- latitude : Real
- longitude : Real
- altitude : Real
- nearness(in entity1 : GeoLocation, in entity2 : GeoLocation) : Boolean
- distance(in entity1 : GeoLocation, in entity2 : GeoLocation) : Real

**Datatype**: GeoLocationCoordinates

- + latitude : Real
- + longitude : Real
- + altitude : Real
- + nearness(in entity1 : GeoLocationCoordinates, in entity2 : GeoLocationCoordinates) : Boolean
- + distance(in entity1 : GeoLocationCoordinates, in entity2 : GeoLocationCoordinates) : Real

**Class**: SpatialEntity

- hasGeoLocation

**Class**: ContainerEntity

- hasSpatialCoordinates

**Class**: SpatialCoordinates

- + dimension

**Datatype**: SpatialDimension

- + radius : Real
- + containment(in point : GeoLocationCoordinates, in container : GeoLocationCoordinates, in dimension : SpatialDimension) : Boolean
- + containment(in entity1 : GeoLocationCoordinates, in dimension1 : SpatialDimension, in entity2 : GeoLocationCoordinates, in dimension2 : SpatialDimension) : Boolean
Context Situation Types: Situation Fever

\{ Context SituationFever inv:
  temp = person.hasTemperature AND
  person.hasTemperature.value > 38 \}
Context Situation Types: Context Types

Person

- MsnUser
  - hasMsnStatus
    - «IntrinsicContext» MsnStatus
      - status: MsnStatusEnum
        + online
        + busy
        + beRightBack
        + away
        + inCall
        + outToLunch

- SkypeUser
  - hasSkypeStatus
    - «IntrinsicContext» SkypeStatus
      - status: SkypeStatusEnum
        + online
        + offline
        + skypeMe
        + away
        + notAvailable
        + doNotDisturb
Intrinsic Situation Types: SituationAvailable

```
{ Context SituationAvailable inv:
  (skypeUser = msnUser) AND
  ((not skypeUser.oclIsUndefined()) AND (skypeUser.skypeStatus = skypeStatus) AND
   (skypeStatus.value = "Online") OR (skypeStatus.value = "SkypeMe")))
OR
  ((not msnUser.oclIsUndefined()) AND
   (msnUser.msnStatus = msnStatus) AND
   (msnStatus.value = "Online")
   OR (msnStatus.value = "BeRightBack")))
```
Formal Relation Situation Types: SituationWithinRange

{ Context SituationWithinRange inv: 
  entityA.hasGeoLocation = locationA AND 
  entityB.hasGeoLocation = locationB AND 
  locationA.value \rightarrow distance(locationB.value) < range }
Situation of Situations Types: SituationSwitch

{ Context SituationSwitch inv:
  (wlan.device = bluetooth.device) AND
  (wlan.device.hasConnection.network.oclIsTypeOf(WLAN)) AND
  (bluetooth.device.hasConnection.network.oclIsTypeOf(Bluetooth)) AND
  (bluetooth.initialtime - wlan.finaltime < 1) }

```
    «Relational Situation»  wlan  «Situation Of Situations»
     SituationConnected     1      0..1
                  bluetooth  1      0..1
```
Situation of Situations Types: SituationSwitch

{ Context SituationDuration inv:
  ((not SituationWithinRange.finaltime.isOclUndefined()) AND
  (SituationWithinRange.finaltime - SituationWithinRange.initialtime > 60))
  OR
  ((SituationWithinRange.finaltime.isOclUndefined()) AND)
  (Time.now() - SituationWithinRange.initialtime > 60))}
Context Information Models

```
<datatype>
GeoLocationCoordinates
+ latitude : Real
+ longitude : Real
+ altitude : Real
+ nearness(in entity1 : GeoLocationCoordinates, in entity2 : GeoLocationCoordinates) : Boolean
+ distance(in entity1 : GeoLocationCoordinates, in entity2 : GeoLocationCoordinates) : Real

<datatype>
GeoLocationMeasurement
+ entityID : String
+ geoLocation : GeoLocationCoordinates
+ precision : RangePrecision
+ freshness : Freshness
+ origin : Origin
+ probabilityOfCorrectness : Real

<datatype>
RangePrecision
+ minValue : Real
+ maxValue : Real

<datatype>
Freshness
+ creationtime : Time
+ temporalResolution : RangePrecision

<enumeration>
Origin
+ sensed
+ derived
+ profiled

conceptual model

context information model
```
Context Information Models

conceptual model

context information model

«datatype»

SituationAvailableMeasurement
+ personID : String
+ msnStatus : MsnStatusEnum
+ skypeStatus : SkypeStatusEnum
+ probabilityOfCorrectness : Real
+ freshness : Freshness
Context Information Models

```
context information model

<datatype>
SituationAvailableMeasurement
+ personID : String
+ msnStatus : MsnStatusEnum
+ skypeStatus : SkypeStatusEnum
+ probabilityOfCorrectness : Real
+ freshness : Freshness
```

```
conceptual model

<role>
MsnUser
  hasMsnStatus

<role>
SkypeUser
  hasSkypeStatus

<IntrinsicContext>
MsnStatus
+ status : MsnStatusEnum

<IntrinsicContext>
SkypeStatus
+ status : SkypeStatusEnum

<IntrinsicSituation>
SituationAvailable
0..1

msnUser 0..1

skypeUser 0..1

msnStatus 0..1

skypeStatus 0..1
```
Situation Realization

- **Rule-based approach**
  - Fits nicely the nature of situation detection
  - Rules (OCL invariants) are repeatedly applied to a collection of facts (context information)

- **Jess**
  - Shadow facts
  - Main components: working memory and rule-base
Situation Realization (overview)

- **Template level (design-time)**
  - Application designer
  - Situation models (UML + OCL)
  - Situation models (Java classes)
  - Situation models (OCL invariants)

- **Instance level (run-time)**
  - Universe of discourse
  - State of affairs
  - Service users
  - Context sources

- **Specification**
  - Context models (UML class diagram)
  - Situation models (UML class diagram)

- **Realization**
  - Context models (Java classes)
  - Situation models (Java classes)
  - Situation fact templates (Jess)
  - Situation fact templates (Jess)
  - Situation detection rules (Jess)
  - Working memory
  - Rule set
  - Rule engine
  - App. entity

- **Condition**
  - Condition 1
  - Condition 2
  - Condition 3
Situation Lifecycle

Detects continuously situations of a situation type using pattern matching implemented by the rule engine

DetectSituation

[situation fact does not exist and invariant holds]

CreateSituationFact

DetectEndSituation

Invariant holds
Situation fact exists

DeactivateSituationFact

[invariant no longer holds]
## Situation Detection

<table>
<thead>
<tr>
<th>Creation Rule</th>
<th>Deactivation Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>(situation type invariant)</code></td>
<td><code>(not (situation type invariant))</code></td>
</tr>
<tr>
<td><code>(not (situation exists))</code></td>
<td><code>(situation exists)</code></td>
</tr>
<tr>
<td><code>=&gt;</code></td>
<td><code>=&gt;</code></td>
</tr>
<tr>
<td><code>create (situation)</code></td>
<td><code>deactivate (situation)</code></td>
</tr>
<tr>
<td><code>[RaiseEvent()]</code></td>
<td><code>[RaiseEvent()]</code></td>
</tr>
</tbody>
</table>

```python
(situation type invariant)
(not (situation exists))
=>
create (situation)
[RaiseEvent()]
```
Mappings
Context Models => Java

- Octopus (www.klasse.nl/octopus/index.html)
  - Generates java code from UML classes, and statically checks OCL constraints
  - UML classes to Java classes
  - Associations to class attributes
    - One-to-one (one attribute in each class)
    - One-to-many (one of the attributes is a collection)
    - Many-to-many (both attributes are collections)
  - Subsets association
## Situation Models => Java + Jess

<table>
<thead>
<tr>
<th>OCL language</th>
<th>Jess language</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>(ObjectType (OBJECT ?object))</td>
</tr>
<tr>
<td>object.pdatatype</td>
<td>(ObjectType (OBJECT ?object) (pdatatype ?pdatatype))</td>
</tr>
<tr>
<td>object object</td>
<td>(ObjectType(^1) (OBJECT ?object(^1)) (object(^2) ?object(^2)))</td>
</tr>
<tr>
<td>object object object</td>
<td>(ObjectType(^2) (OBJECT ?object(^2)) (object(^3) ?object(^3)))</td>
</tr>
<tr>
<td>object datatype</td>
<td>(ObjectType(^1) (OBJECT ?object(^1)) (datatype ?pdatatype))</td>
</tr>
<tr>
<td>object.datatype.pdatatype</td>
<td>(ObjectType(^1) (OBJECT ?object(^1)) (datatype ?datatype)) (DataType (OBJECT ? datatype) (pdatatype ?pdatatype))</td>
</tr>
<tr>
<td>object.object.object.datatype.pdatatype</td>
<td>(ObjectType(^1) (OBJECT ?object(^1)) (object(^2) ?object(^2))) (ObjectType(^2) (OBJECT ?object(^2)) (datatype ?datatype)) (DataType (OBJECT ?datatype) (pdatatype ?pdatatype))</td>
</tr>
<tr>
<td>Object-&gt;collection</td>
<td>(ObjectType (OBJECT ?object) (collection ?collection))</td>
</tr>
</tbody>
</table>
Context Models => Java

public class Entity...
public class Context...
public class IntrinsicContext extends Context...
public class SpatialEntity extends Entity...
public class PointEntity extends SpatialEntity...
public class ContainerEntity extends SpatialEntity...
person class Person extends PointEntity...
person class Building extends ContainerEntity...
person class GeoLocation extends IntrinsicContext...
person class SpatialCoordinates extends IntrinsicContext...

(defrule entertrue_situation_contained )
(defrule enterfalse_situation_contained )

public class SituationType...
person class SituationContained extends SituationType...
(defrule entertrue_situation_contained
  (Person (OBJECT ?person) (hasGeoLocation ?person_hasGeoLocation))
  (GeoLocation (OBJECT ?locationPerson &:(eq ?locationPerson ?person_hasGeoLocation)))
  (Building (OBJECT ?building) (geoLocation ?building_hasGeoLocation))
  (GeoLocation (OBJECT ?locationBuilding &:(eq ?locationBuilding ?building_hasGeoLocation)))
  (Building (OBJECT ?building) (spatialCoordinates ?building_hasSpatialCoordinates))
  (SpatialCoordinates (OBJECT ?spatialCoord &:(eq ?spatialCoord ?building_hasSpatialCoordinates)))
  (GeoLocation (OBJECT ?locationPerson) (location ?locationPerson_coordinates))
  (GeoLocation (OBJECT ?locationBuilding) (location ?locationBuilding_coordinates))
  (SpatialCoordinates (OBJECT ?spatialCoord) (dimension ?spatialCoord_dimension))
  (test (call context_control: SpatialDimension Containment ?locationPerson_coordinates
                                      ?locationBuilding_coordinates ?spatialCoord_dimension))
  (not (SituationContained (OBJECT ?st) (person ?person) (building ?building) (finalTime nil))))
  =>
  (bind ?SituationContained (new situation_control:SituationContained ?person ?building))
  (definstance SituationContained ?SituationContained)
(defrule enterfalse_situation_contained
  (not (and (Person (OBJECT ?person) (hasGeoLocation ?person_hasGeoLocation))
           (GeoLocation (OBJECT ?locationPerson & (eq ?locationPerson ?person_hasGeoLocation)))
           (Building (OBJECT ?building) (geoLocation ?building_hasGeoLocation))
           (GeoLocation (OBJECT ?locationBuilding & (eq ?locationBuilding ?building_hasGeoLocation)))
           (Building (OBJECT ?building) (spatialCoordinates ?building_hasSpatialCoordinates))
           (SpatialCoordinates (OBJECT ?spatialCoord & (eq ?spatialCoord ?building_hasSpatialCoordinates)))
           (GeoLocation (OBJECT ?locationPerson) (location ?locationPerson_coordinates))
           (GeoLocation (OBJECT ?locationBuilding) (location ?locationBuilding_coordinates))
           (SpatialCoordinates (OBJECT ?spatialCoord) (dimension ?spatialCoord_dimension))
           (test (call context_control.SpatialDimension Containment ?locationPerson_coordinates
                  ?locationBuilding_coordinates ?spatialCoord_dimension)))
  => (SituationContained (OBJECT ?st) (person ?person) (building ?building) (finaltime nil))
  
  (call ?SituationContained deactivate)
Distribution

• Service-oriented approach: components encapsulate Jess engines, and situation information is exchange by means of the component services

• DJess: separate engines virtually share working memory. Rule engines running on different nodes can apply rules on shared facts
Conclusions

- **Context models** help understanding context concepts and how they relate to each other.
- Context models are static.
- **Situations** allow one to define *state-of-affairs of concern* for context-aware applications.
- **Behaviours** can be defined in terms of how the system evolves from situation to situation!
- Situations can be used to define *conditions that trigger a rule system*, as, e.g., in ECA rules.
- Situations can be composed of situations themselves:
  - modularization of the situation models, improving organization and reuse of situation specifications.
Conclusions (2)

• Situation realization is rule-based
  – Allows attentive situation detection as opposed to query-based solutions

• Model-driven approach
  – Specification elements are systematically mapped to realization elements
  – UML as a mature technology in model-driven developments