Policy-based Adaptation of Context Provisioning in Ambient Intelligence

Alexandru Sorici$^{1,2}$  Gauthier Picard$^1$  Olivier Boissier$^1$  Adina Magda Florea$^2$

$^1$Laboratoire Hubert Curien UMR CNRS 5516, Institut Henri Fayol, MINES Saint-Etienne, France  
  sorici@emse.fr

$^2$University Politehnica of Bucharest, Department of Computer Science, 313 Splaiul Independentei, 060042 Bucharest, Romania

6th International Symposium on Ambient Intelligence, Salamanca, 2015
1 Introduction
   • Context Provisioning
   • Research Objectives

2 CONSERT Middleware
   • Architecture
   • Context Provisioning Policy Definition
   • Provisioning Protocols and Policy Execution

3 Conclusions and Future Work
Outline

1 Introduction
   - Context Provisioning
   - Research Objectives

2 CONSERT Middleware
   - Architecture
   - Context Provisioning Policy Definition
   - Provisioning Protocols and Policy Execution

3 Conclusions and Future Work
Introduction: Context Provisioning

Definition
Context Provisioning is the process of managing the units involved in enabling the flow of context information from producers to consumers thereof.

Main Life Cycle
- Context Acquisition
- Context Modeling / Reasoning / Coordination
- Context Dissemination

Complementary Functionality
- Context Producer Discovery
- Mobility Management (e.g., interaction session, handovers)
- Context Access Management
- Provisioning Adaptability (structural, functional)

Sorici et al. (UPB, EMSE) Policy-based Adaptation in AmI ISAml 2015
Definition

**Context Provisioning** is the process of managing the units involved in enabling the flow of context information from producers to consumers thereof.
Introduction: Context Provisioning

Definition

Context Provisioning is the process of managing the units involved in enabling the flow of context information from producers to consumers thereof.

Main Life Cycle
- Context Acquisition
- Context Modeling / Reasoning / Coordination
- Context Dissemination

Complementary Functionality
- Context Producer Discovery
- Mobility Management (e.g. interaction session, handovers)
- Context Access Management
- Provisioning Adaptability (structural, functional)
- ...
Introduction: Example Scenario
Introduction: Example Scenario
Introduction: Example Scenario
Introduction: Example Scenario

1) Perceive

2) Reason about: situation

3) Inform: Ad-Hoc Meeting
Introduction: Example Scenario

1) Perceive

No requests for noise, posture, temperature or luminosity for past 5 min.
What is a good abstraction for the fundamental units of context management?
Introduction: Context Provisioning Issues

- What is a good abstraction for the fundamental units of context management?

- What is a good mechanism to engineer the adaptability of the context provisioning process?
Introduction: Context Provisioning Issues

- What is a **good abstraction** for the fundamental units of context management?

- What is a **good mechanism** to engineer the adaptability of the context provisioning process?

- How to support application development by ensuring **flexibility** and **ease of use** of the adaptation mechanism?
Introduction: Main Goals

- Develop a Context Management Middleware (CMM) based on design principles from the Multi-Agent Systems, Semantic Web and Software Service Component domains.
  - **Agents**: units of control encapsulation for each provisioning aspect with potential for increased autonomy.
  - Guide and adapt agent provisioning behavior through declarative policies.

- Why these objectives?
Introduction: Main Goals

- Develop a **Context Management Middleware (CMM)** based on design principles from the **Multi-Agent Systems**, **Semantic Web** and **Software Service Component** domains.
  - **Agents**: units of **control encapsulation** for each provisioning aspect with **potential for increased autonomy**.
  - Guide and adapt agent provisioning behavior through **declarative policies**.

- Why these objectives?
Outline

1 Introduction
   - Context Provisioning
   - Research Objectives

2 CONSERT Middleware
   - Architecture
   - Context Provisioning Policy Definition
   - Provisioning Protocols and Policy Execution

3 Conclusions and Future Work
CONSERT Middleware Architecture

- **CONSERT** = **CONtext as SERTion** [Sorici et al., 2015b]
- **Multi-Agent Based Architecture**: use **MAS design principles** as a **good fit** for this **engineering problem**

Why Agents?
Conceive the provisioning units as: autonomous, reactive, proactive and socially interacting entities.

Exploit research into message-based, communicative-act centric interaction protocols to address communication infrastructure concerns ⇒ Good encapsulation of the logic for each provisioning aspects with potential for increased provisioning autonomy

Message based communication with complete handling of success and failure cases
CONSERT Middleware Architecture

- CONSERT = CONtext asSERTion [Sorici et al., 2015b]
- Multi-Agent Based Architecture: use MAS design principles as a good fit for this engineering problem
- Why Agents?
  - Conceive the provisioning units as: autonomous, reactive, proactive and socially interacting entities
  - Exploit research into message-based, communicative-act centric interaction protocols to address communication infrastructure concerns
    \[\Rightarrow\]
  - Good encapsulation of the logic for each provisioning aspects with potential for increased provisioning autonomy
  - Message based communication with complete handling of success and failure cases
CONSERT Middleware Architecture

Context-Aware Application

- Control profiled updates
- Issue queries/subscriptions
- Adjust context provisioning

Middleware

- Context Domain Ontology
- CONSERT Meta-Model Ontology
- CONSERT Engine
  - Store Context Information
  - Check Continuity
  - Check Constraint Integrity
  - Perform Ontology Reasoning
  - Perform Derivation Rule Reasoning
  - Compute Context Usage Statistics
- Extends
- Loads
- Commands
- Queries

CONSERT Agent Management Logic

- Application Client Adaptor
- Application Control Adaptor
- CtxCoord
- CtxQuery Handler
- CtxSensor
- OrgMgr
- ContextAssertion Adaptors
- Manages
- Commands / Translates from

Legend
- Provisioning agent interactions
- OrgMgr interactions

Agent

3rd Party Sensor Middleware

Sorici et al. (UPB, EMSE) Policy-based Adaptation in AmI ISAml 2015
CONSERT Middleware Agents

Multi-Agent Based Architecture: 4 provisioning agents + 1 management agent

Provisioning Agents

- **CtxSensor Agent**: manage interactions with sensors (based on sensing policies), communicate with CtxCoord to send updates and receive provisioning tasking commands
- **CtxCoord Agent**: coordinate processing of context information
  - Create and control CONSERT Engine
  - Use coordination policies to determine what sensor updates and inferences are active and how (e.g. with which frequency) updates must be sent
Provisioning Agents

- **CtxQueryHandler Agent**: disseminate context information, answer to queries and subscriptions. Can work in local or federated mode.
- **CtxUser Agent**: connection with application logic
  - Send queries and subscriptions
  - Act as prosumer: provide static or profiled ContextAssertions

Management Agent

- **OrgMgr Agent**:
  - Control deployment and life cycle of provisioning agents (i.e. create, start, stop, destroy provisioning agents)
  - Maintain overview of distributed deployment (if the case) + manage query/updates routing
Context Provisioning Policies

- Guide the behavior of provisioning agents (especially CtxCoord and CtxSensor)

- Consist of a set of **parameters** (key-value attributes) and a set of **control rules** (developer defined)

- **Implemented using Semantic Web Technologies**
  - Ontology-based parameter vocabulary
  - SPARQL-based rule definition
Context Sensing Policies

- Specify initial settings for **how** sensed context information is updated
- 2 parameters: **update-rate**, **update-mode** (change-based, time-based)
Specify initial settings for how sensed context information is updated

2 parameters: **update-rate**, **update-mode** (change-based, time-based)

```owl
:presenceSensingPolicy
  a sensorconf:SensingPolicy ;
  coordconf:forContextAssertion
    ex:sensesBluetoothAddress ;
  sensorconf:hasUpdateMode
    coordconf:time-based ;
  sensorconf:hasUpdateRate 2 .

:luminositySensingPolicy
  a sensorconf:SensingPolicy ;
  coordconf:forContextAssertion
    ex:sensesLuminosity ;
  sensorconf:hasUpdateMode
    coordconf:change-based ;
  sensorconf:hasUpdateRate 0 .
```
Context Coordination Policies

- Define the adjustable aspects of the context provisioning process
- Allow for a rule-based mechanism for controlling the adjustment (adaptation)

- **Control Parameters**: Setup the CONSERT Engine, specify enabled updates and update modes
- **Control Rules**: alter control parameters according to dynamic use of context information
Provisioning Control Parameters

- Parameters may be general or context information type specific.
Parameters may be general or context information type specific

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Values</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>assertion enabling</td>
<td>true/false</td>
<td>Specify if assertion updates are enabled by default.</td>
</tr>
<tr>
<td>ont. reasoning interval</td>
<td>number in seconds</td>
<td>Time span between calls to ontology reasoner.</td>
</tr>
<tr>
<td>TTL</td>
<td>number in seconds</td>
<td>Time to live for any ContextAssertion in the runtime storage</td>
</tr>
<tr>
<td>integrity constraint resolution</td>
<td>String in enumeration</td>
<td>Identifier of the service handling integrity constraint resolutions</td>
</tr>
<tr>
<td>uniqueness constraint resolution</td>
<td>String in enumeration</td>
<td>Identifier of the service handling uniqueness constraint resolutions</td>
</tr>
<tr>
<td>observation_window</td>
<td>number in seconds</td>
<td>Length of time window over which context usage statistics are computed</td>
</tr>
<tr>
<td>inference scheduling service</td>
<td>String in enumeration</td>
<td>Identifier of service providing priority scheduling for ContextDerivationRules</td>
</tr>
</tbody>
</table>
Provisioning Control Rules

- Make use of context knowledge base snapshots and CONSERT Engine usage statistics to express conditions for altering of provisioning control parameters
- Implemented as SPARQL Query Templates (using SPIN\(^1\))

\(^1\)http://spinrdf.org
Provisioning Control Rules

- Make use of context knowledge base snapshots and CONSERT Engine usage statistics to express conditions for altering of provisioning control parameters
- Implemented as SPARQL Query Templates (using SPIN\textsuperscript{1})

CONSTRUCT {
  _:b0 a :StopAssertionCommand.
  _:b0 :forContextAssertion ?assertion.
}
WHERE {
  ?stat a :AssertionSpecificStatistic.
  ?stat :isDerivedAssertion true.
  ?stat :nrSubscriptions 0.
  FILTER (?time > ?elapsedThreshold).
}

coord:ControlPolicy
coord:hasStopAssertionCommand [
  a :QueryAbsenceAssertionCancellation;
  arg:contextAssertion ami:sensesLuminosity;
  arg:elapsedTimeThreshold 300;
];

Figure: SPARQL expression of derivation cancellation rule template (left) and control rule assignment (right)

\textsuperscript{1}http://spinrdf.org
Context Provisioning Protocols - Sensing Chain

![Diagram of Context Provisioning Protocols]

- **CtxSensor**
  - Retrieve Sensing Policy
  - Publish Assertions
    - list of enabled Assertions
  - Sensor update
    - Assertion Update
      - Tasking Command
        - Run Provisioning Control Rules
  - Analyse which Context Assertions are currently enabled
Context Provisioning Protocols - Request Chain

- **CtxSensor**
- **CtxCoord**
- **CtxQueryHandler**
- **CtxUser**

**Protocol Steps:**
1. **RegisterQueryHandler**
   - ACK Registration
2. **Register Query User**
   - ACK Registration
3. **Make Query**
   - Assertions
   - Enabled
   - Run Query
   - Inform Query Result
4. **Inform Static/Profiled Update**
   - ACK Update
Context Coordination Policy Execution

- CtxCoord agent uses control parameters to set up the CONSERT Engine and the default active CtxSensor agents
- CtxCoord agent runs control rules every observation window seconds
**Context Provisioning Policy Execution**

- CtxCoord agent uses control parameters to set up the CONSERT Engine and the default active CtxSensor agents.
- CtxCoord agent runs control rules every *observation_window* seconds.
- CtxCoord agent requests *snapshot of context knowledge* base and *context usage statistics* from CONSERT Engine.
- Control rules are partitioned into *execution groups*:
  - *Execution groups* are run in a *developer-specified order*.
  - Rule outcomes from later groups *overwrite* contradictory outcomes from rules in previous groups ⇒ ensure *control rule output consistency*. 

---

*Sorici et al. (UPB, EMSE) Policy-based Adaptation in AmI*
Outline

1. Introduction
   - Context Provisioning
   - Research Objectives

2. CONSERT Middleware
   - Architecture
   - Context Provisioning Policy Definition
   - Provisioning Protocols and Policy Execution

3. Conclusions and Future Work
Conclusions and Future Work

- CONSERT Middleware focuses on **flexibility** through **agent-based encapsulation** of provisioning aspects (more in [Sorici et al., 2015a])
- Address **ease of development** for context provisioning adaptation through declarative policies
Conclusions and Future Work

- CONSERT Middleware focuses on flexibility through agent-based encapsulation of provisioning aspects (more in [Sorici et al., 2015a])
- Address ease of development for context provisioning adaptation through declarative policies

- Exploit multi-agent potential for autonomy by introducing Context Level Agreements (CLAs)
  - CtxCoord, CtxSensor agents have individual goals (e.g. reduce workload, save energy) which are valued against request characteristics (e.g. required accuracy, needed freshness) from a CtxUser
  - Control of CLA establishment needs to be integrated in provisioning policies
    - Increase specificity level of control rules to individual context providers
    - Use observed Quality-of-Context to enhance expressiveness of control rule conditions

THANK YOU!

Questions?