Multi-Agent Based Context Management in Ambient Intelligence Applications

Alexandru Sorici\textsuperscript{1,2} Gauthier Picard\textsuperscript{1} Adina Magda Florea\textsuperscript{2}

\textsuperscript{1}Laboratoire Hubert Curien UMR CNRS 5516, Institut Henri Fayol, MINES Saint-Etienne, France sorici@emse.fr

\textsuperscript{2}University Politehnica of Bucharest, Department of Computer Science, 313 Splaiul Independentei, 060042 Bucharest, Romania

May 28, 2015
1 Introduction
   - Problem
   - Motivational Scenario
   - Objectives

2 CONSERT Middleware
   - Context Representation Approach
   - Context Provisioning Approach
   - Middleware Deployment Approach

3 Conclusions and Future Work
Outline

1 Introduction
   - Problem
   - Motivational Scenario
   - Objectives

2 CONSERT Middleware
   - Context Representation Approach
   - Context Provisioning Approach
   - Middleware Deployment Approach

3 Conclusions and Future Work
Introduction: Problem

The Ambient Intelligence vision considers very broad scenarios (e.g. ISTAG Scenarios [Ducatel et al., 2001])

User can have short-lived, context-aware interactions with many unrelated and heterogeneous applications and services

Insufficient work on means to effectively structure and dynamically deploy the multitude of context management services required by an application

Sorici et al. (UPB, EMSE)
Introduction: Problem

- The Ambient Intelligence vision considers very broad scenarios (e.g. ISTAG Scenarios [Ducatel et al., 2001])
- User can have short-lived, context-aware interactions with many unrelated and heterogeneous applications and services
The Ambient Intelligence vision considers very broad scenarios (e.g. ISTAG Scenarios [Ducatel et al., 2001])

- User can have short-lived, context-aware interactions with many unrelated and heterogeneous applications and services

- Insufficient work on means to effectively structure and dynamically deploy the multitude of context management services required by an application
Alice sends information on her estimated delay.
Introduction: Motivational Scenario
Introduction: Motivational Scenario

4a) Action: silent ringtone.
4b) Mark meeting in calendar.
4c) Occupied => postpone contact requests.

Room EF301

Aml Lab

2) Reason about: situation
1) Perceive
3) Inform: Ad-Hoc Meeting

5a) Publish availability status
5b) Ask availability status
Alice’s smartphone interacts with many *different* context-aware services: Tram, University Course Activity Service, AmI-Lab Management Server.

The smartphone uses local context information itself (e.g. Alice’s profile).

Information obtained from one service can both *influence* (e.g. tram speed with class delay time) and *be independent of* (e.g. AmI-Lab interactions) another service.
Alice’s smartphone interacts with many different context-aware services: Tram, University Course Activity Service, AmI-Lab Management Server.

The smartphone uses local context information itself (e.g. Alice’s profile).

Information obtained from one service can both influence (e.g. tram speed with class delay time) and be independent of (e.g. AmI-Lab interactions) another service.

How does the application structure the use of different context information services (tram, university course activity management, AmI laboratory)?

How can the provisioning process be configured and adapted to the dynamic usage of context information?
Introduction: Main Goals

- Develop a **Context Management Middleware (CMM)**
- Focus on **expressive modeling, flexible provisioning, ease of deployment/configuration**

- **Why these objectives?**
Introduction: Main Goals

- Develop a Context Management Middleware (CMM)
- Focus on expressive modeling, flexible provisioning, ease of deployment/configuration

Why these objectives?
1 Introduction
   • Problem
   • Motivational Scenario
   • Objectives

2 CONSERT Middleware
   • Context Representation Approach
   • Context Provisioning Approach
   • Middleware Deployment Approach

3 Conclusions and Future Work
Our approach to the presented challenges and objectives

- Use MAS, Semantic Web and service component based design principals and technologies as a good engineering fit for CMM development
CONSORT = CONtext asSERTion

CONSORT Meta-Model characteristics [Sorici et al., 2015b]:
- 3 ontology-based modules providing uniform representation support for context modeling concerns (content, meta-properties - including QoC, constraint representation)
- Flexible modeling of content and annotations (statements of variable arity, semantics for annotation combination during inference)
CONSERT Reasoning Engine characteristics [Sorici et al., 2015b]:

- Reasoning cycle which favors temporal inference and a complete view of semantically distinguishable situations
- Reasoning implementation supporting customizable constraint resolution and derivation scheduling
- Capability of introspection → statistics of dynamic context information usage
Multi-Agent Based Architecture

Key aspect: use MAS design principles as a good fit for an engineering problem

Why Agents?
- Conceive the provisioning units as: autonomous, reactive, proactive and socially interacting entities
- Use existing, mature development frameworks (JADE) to tackle communication infrastructure

⇒

- 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 CMM - Provisioning

Context-Aware Application

Middleware

CONSORT Meta-Model Ontology

CONSORT Engine
- Store Context Information
- Check Continuity
- Check Constraint Integrity
- Perform Ontology Reasoning
- Perform Derivation Rule Reasoning
- Compute Context Usage Statistics

CORE
ANNOTATION
CONSTRATN

Middleware

extends

loads

commands

implements

queries

3rd Party Sensor Middleware

JADE Framework

OSGi Framework

Apache Jena+SPIN

JVM

Core Domain Ontology

Loaders: CONSERT Engine

Application Client Adaptor

Application Control Adaptor

issues queries/subscriptions

adjust context provisioning

control profiled updates

organize

process sensed/profiled updates

considers context profiled updates

organizes

issues profiled updates

manages

commands / translates from

ContextAssertion Adaptors

3rd Party Sensor Middleware

Sorici et al. (UPB, EMSE) MAS Context Management in AmI May 28, 2015 13 / 29
CONSERT CMM - Provisioning

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 Management Unit (CMU): unit of control encapsulation

- **Instances** of context provisioning agents can be grouped into management units that are assigned to handle a specific *provisioning aspect* (acquisition, coordination, context consumption)

E.g.:
- CtxSensor + CtxUser agents can be grouped and deployed on a *prosumer* machine (e.g. Alice’s smartphone)
- CtxCoord + CtxQueryHandler - grouped and deployed on a coordination machine (e.g. the Aml-Lab management server)
Context Provisioning Policies [Sorici et al., 2015a]

- 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
Sensing Policies (CtxSensor agents)
- Specify initial settings for how sensed ContextAssertions are updated
- 2 parameters: update-rate, update-mode (change-based, time-based)

Coordination Policies (CtxCoord agents)
- Control Parameters:
  - Setup CONSERT Engine (e.g. active inferences, constraint resolution service configuration, TTL, OBSERVATION_WINDOW)
  - Set enabled ContextAssertion updates and update-modes
- Control Rules: alter control parameters according to dynamic use of context information
Context Provisioning Protocols - Sensing Chain

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

Diagram showing interactions between different components:
- CtxSensor
- CtxCoord
- CtxQueryHandler
- CtxUser

Key interactions:
- RegisterQueryHandler
- ACK Registration
- Register Query User
- ACK Registration
- Make Query
- Assertions Enabled
- Run Query
- Inform Query Result
- Inform Activation
- Run Query
- Inform Query Result
- Tasking Command
- StartAssertion Updates
- ACK Tasking Command
- Activate Assertion
- Inform Activation
- Send Static/Profiled Update
- ACK Update
Idea: create link between multi-dimensionality of context information and CMU assigned to service it
Idea: create link between multi-dimensionality of context information and CMU assigned to service it

ContextDimension: a privileged direction (e.g. spatial, activity-related) along which an application structures its context provisioning process.

ContextDomain: a logical partition of the global application context model along a chosen ContextDimension
ContextDimensions + ContextDomains + CMUs allow us to consider two deployment schemes:

- **Centralized:**
  - a single (default) ContextDomain
  - a single CMU handling context provisioning

- **Decentralized:**
  - one or more ContextDimensions and ContextDomains
  - ContextDomains can be organized in a *flat* or *hierarchical* manner
  - Comprises both fixed and mobile nodes ⇒ multiple CMUs
Outline

1 Introduction
   • Problem
   • Motivational Scenario
   • Objectives

2 CONSERT Middleware
   • Context Representation Approach
   • Context Provisioning Approach
   • Middleware Deployment Approach

3 Conclusions and Future Work
Focus on:

- **Flexibility** of the context model and the deployment scheme
- **Provisioning adaptability** through policy based specifications
- **Ease of development / configuration** through use of declarative policies and service component based design

Implementation: use of Semantic Web, MAS and OSGi technologies as a **good engineering fit** for middleware goals

**Why:** good response to the need of supporting context-aware application development
Current validation through implementation of the AmI-Lab scenario using a simulated environment (using the iCasa platform\(^1\))

\[ \Rightarrow \text{need to perform real time experiment in AmI-Lab} \]

\(^1\)http://adeleresearchgroup.github.io/iCasa/
- Current validation through implementation of the AmI-Lab scenario using a simulated environment (using the iCasa platform\(^1\))
  - \(\Rightarrow\) need to perform real time experiment in AmI-Lab

- From experience developing the scenario
  - Need for tooling (e.g. context model IDE, syntactic sugar for derivation rule writing, automated code generation for simple query/subscription usage)

\(^1\)http://adeleresearchgroup.github.io/iCasa/
Current validation through implementation of the AmI-Lab scenario using a simulated environment (using the iCasa platform\(^1\))

⇒ need to perform real time experiment in AmI-Lab

From experience developing the scenario

Need for tooling (e.g. context model IDE, syntactic sugar for derivation rule writing, automated code generation for simple query/subscription usage)

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

Established CLAs influence where and when CMUs are deployed

\(^1\)http://adeleresearchgroup.github.io/iCasa/


THANK YOU!

Questions?