Multi-Agent Based Flexible Deployment of Context Management in Ambient Intelligence Applications

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

13th International Conference on Practical Applications of Agents and Multi-Agent Systems, Salamanca, 2015

Outline

Outline

Introduction: Problem

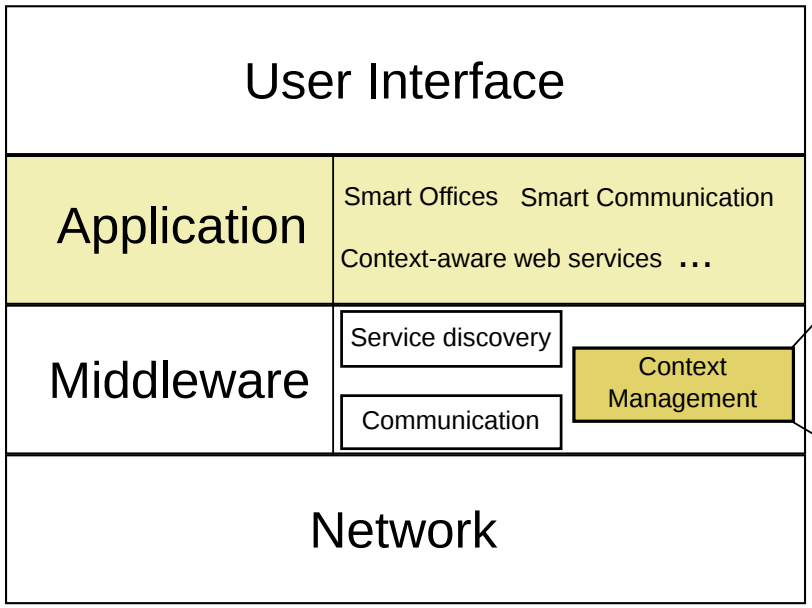
Introduction: Problem

- 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

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

Introduction: Context Management



Definition of Context (Dey, 2001)

Any information that can be used to characterize the situation of entities (i.e., whether a person, place or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves.

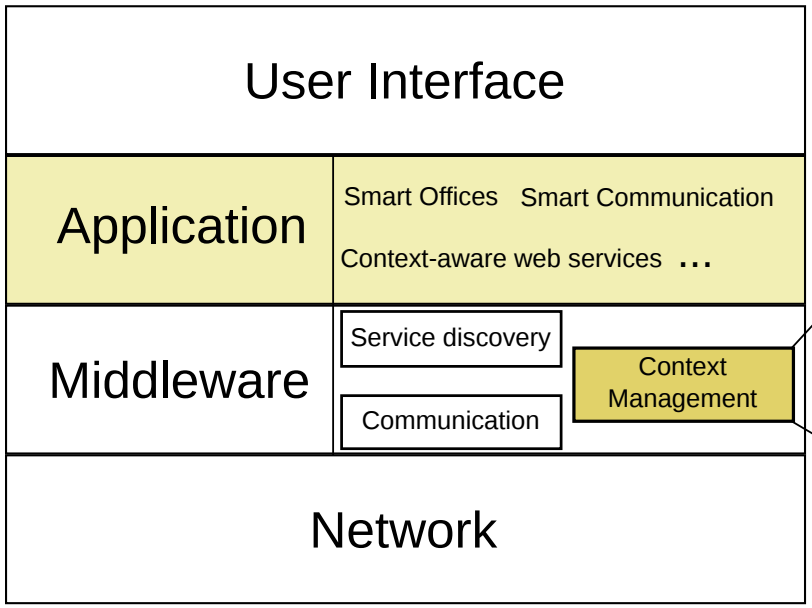
Context Management

Context representation and reasoning

Context provisioning
- **acquisition, coordination**
- **dissemination, usage**

Context Management Solution Deployment

Introduction: Context Management



Definition of Context (Dey, 2001)

Any information that can be used to characterize the situation of entities (i.e., whether a person, place or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves.

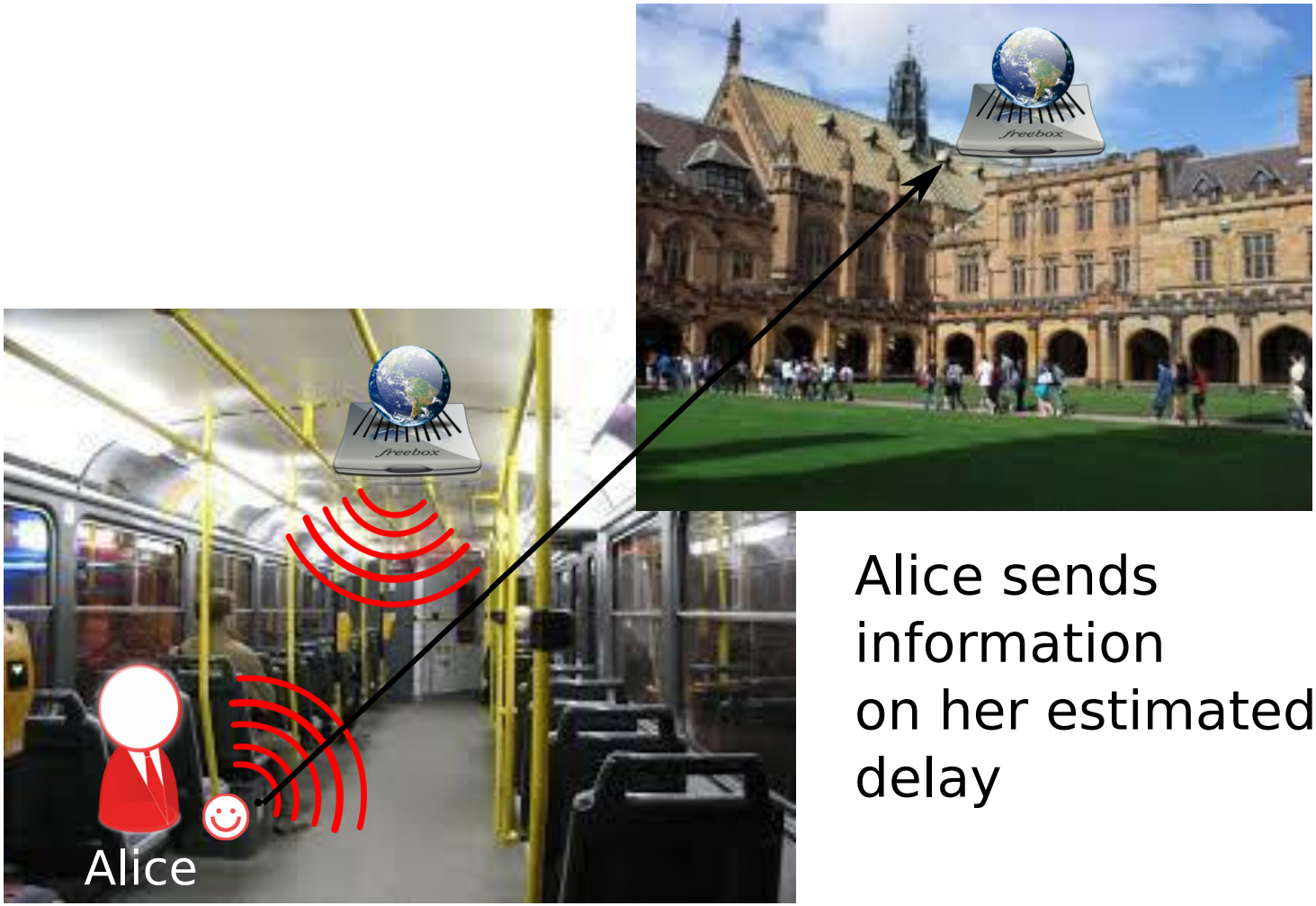
Context Management

Context representation and reasoning

Context provisioning
- **acquisition, coordination**
- **dissemination, usage**

Context Management Solution Deployment

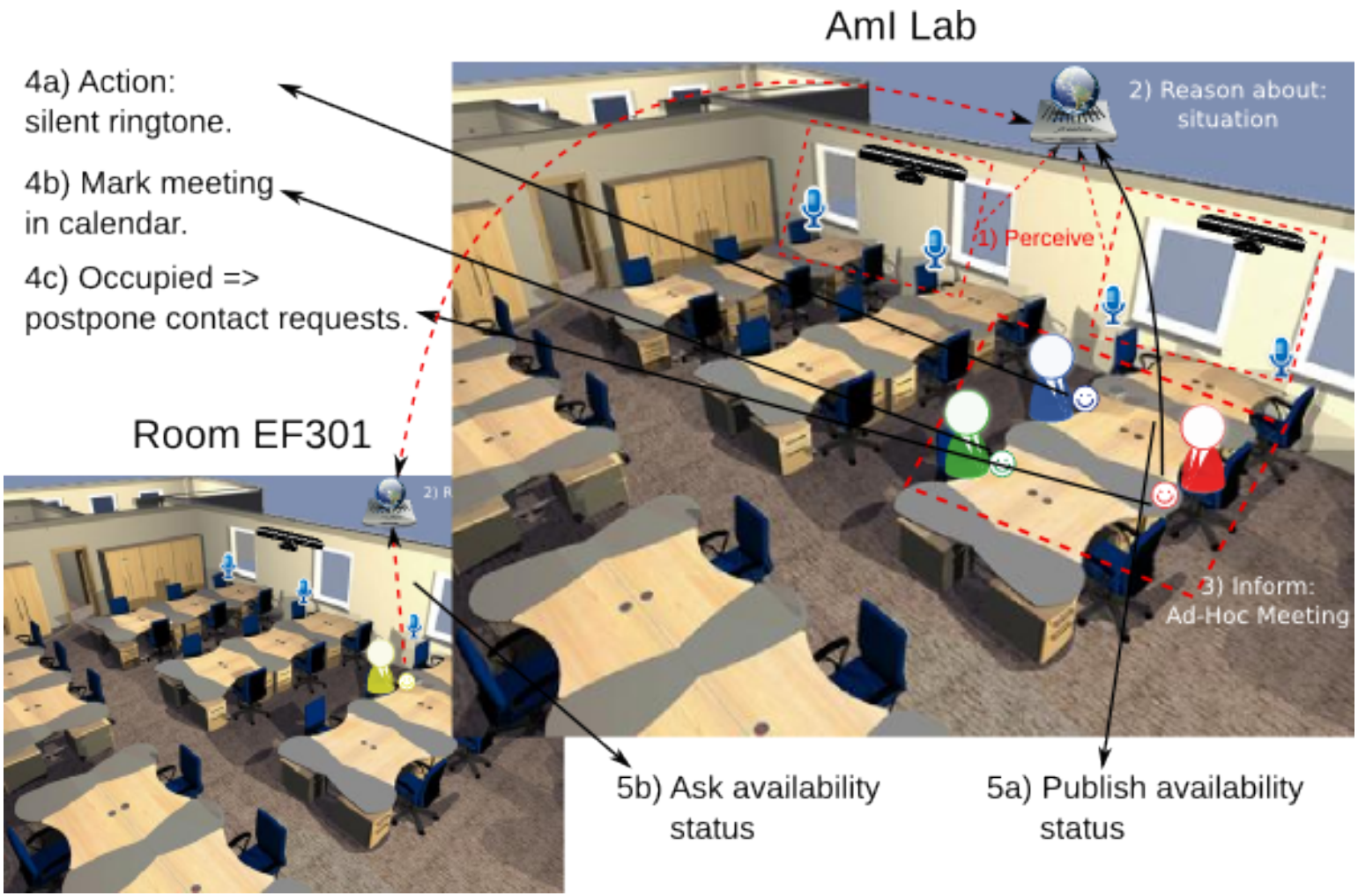
Introduction: Motivational Scenario



Alice sends information on her estimated delay

Introduction: Motivational Scenario

Introduction: Motivational Scenario



Introduction: Issues

- Alice's smartphone interacts with many *different* context-aware services: Tram, University Course Activity Service, AmI-Lab Management Server, her own local 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) an other service

Introduction: Issues

- Alice's smartphone interacts with many *different* context-aware services: Tram, University Course Activity Service, AmI-Lab Management Server, her own local 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) an other service

- What is a good selection of fundamental units for context management?
- What is a good set of **abstractions** for **structuring the use** of various **context management services** in an AmI application, depending on **changing relevancy** of context?

Introduction: Main Goals

- Develop a Context Management Middleware (CMM) based on design principles from the **Multi-Agent Systems**, **Semantic Web** and **Software Service Component** fields.
- Focus on flexible provisioning, ease of deployment/configuration

- **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** fields.
- Focus on flexible provisioning, ease of deployment/configuration

- **Why these objectives?**

Outline

CONSERT Middleware Architecture

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

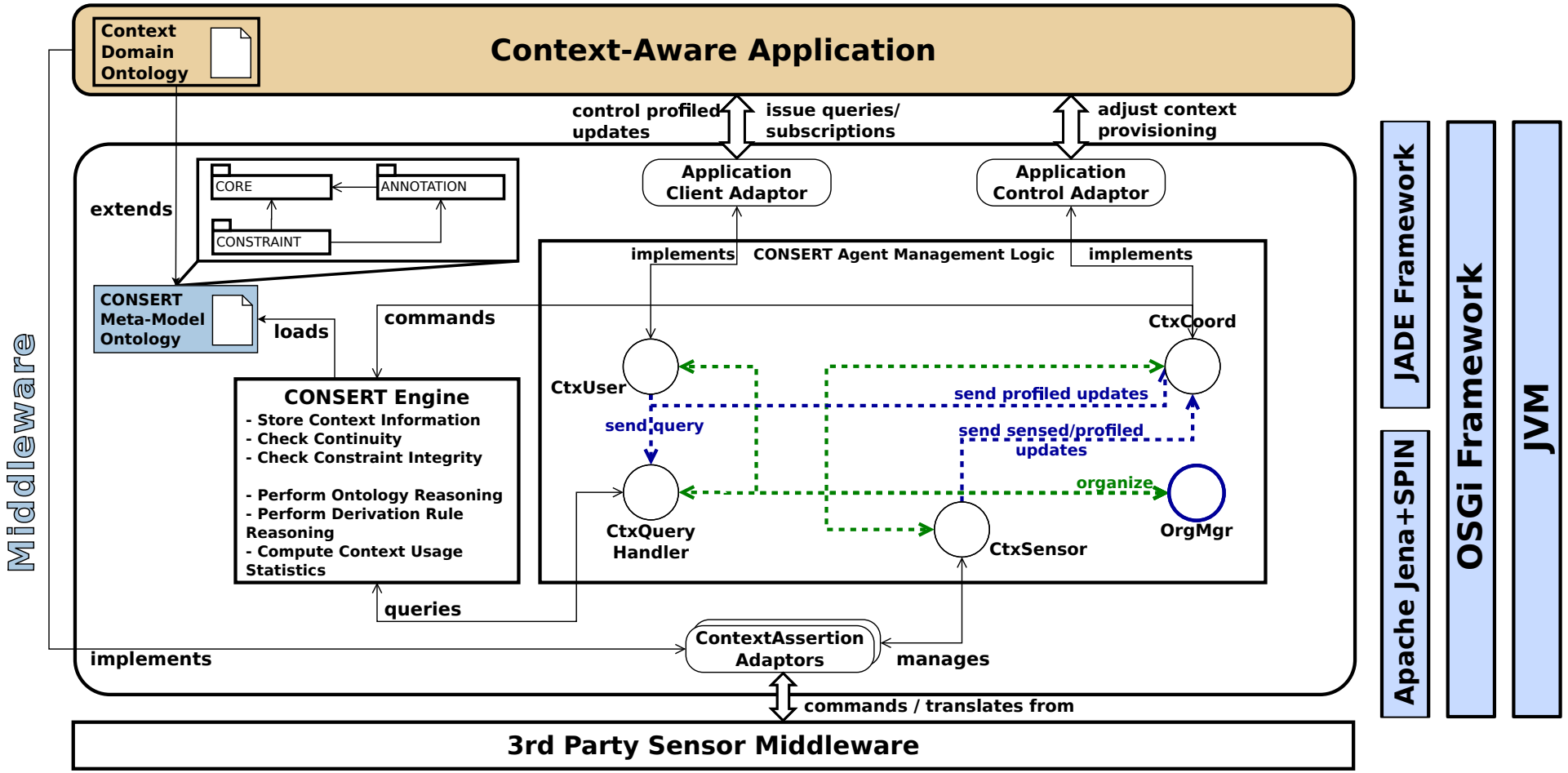
CONSERT Middleware Architecture

- CONSERT = CONtext asSERTion [Sorici et al., 2015]
- **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



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

CONSERT Middleware Agents

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)

Context Management Unit (CMU)

- Element of control encapsulation

Context Management Unit (CMU)

- Element of control encapsulation
- **Instances** of context provisioning agents are grouped into **mangement units** that are assigned to handle a specific *provisioning aspect* (acquisition, coordination, context consumption)

Context Management Unit (CMU)

- Element of control encapsulation
- **Instances** of context provisioning agents are grouped into **mangement 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 AmI-Lab management server)

Deployment Structuring Elements

- **Idea**: create **link** between **multi-dimensionality of context information** and **CMU** assigned to service it

Deployment Structuring Elements

- **Idea**: create **link** between **multi-dimensionality of context information** and **CMU** assigned to service it
- **ContextDimension**: a *privileged direction* (e.g spatial, activity-related, relational) along which an application structures its context provisioning process.

Deployment Structuring Elements

- **Idea**: create **link** between **multi-dimensionality of context information** and **CMU** assigned to service it

- **ContextDimension**: a *privileged direction* (e.g spatial, activity-related, relational) along which an application structures its context provisioning process.
  - Alternatively: the context information type whose instances are most likely to cause a **shift in relevant context**
  - E.g.: *locatedIn(Person, UniversitySpace)*, *engagedIn(Person, CourseActivity)*

Deployment Structuring Elements

- **Idea**: create **link** between **multi-dimensionality of context information** and **CMU** assigned to service it

- **ContextDimension**: a *privileged direction* (e.g spatial, activity-related, relational) along which an application structures its context provisioning process.
  - Alternatively: the context information type whose instances are most likely to cause a **shift in relevant context**
  - E.g.: *locatedIn(Person, UniversitySpace)*, *engagedIn(Person, CourseActivity)*
- **ContextDomain**: a *logical partition* of the global application context model along a chosen ContextDimension
  - E.g.: *Tram*, *AmI-Lab*, *CS_Lecture*

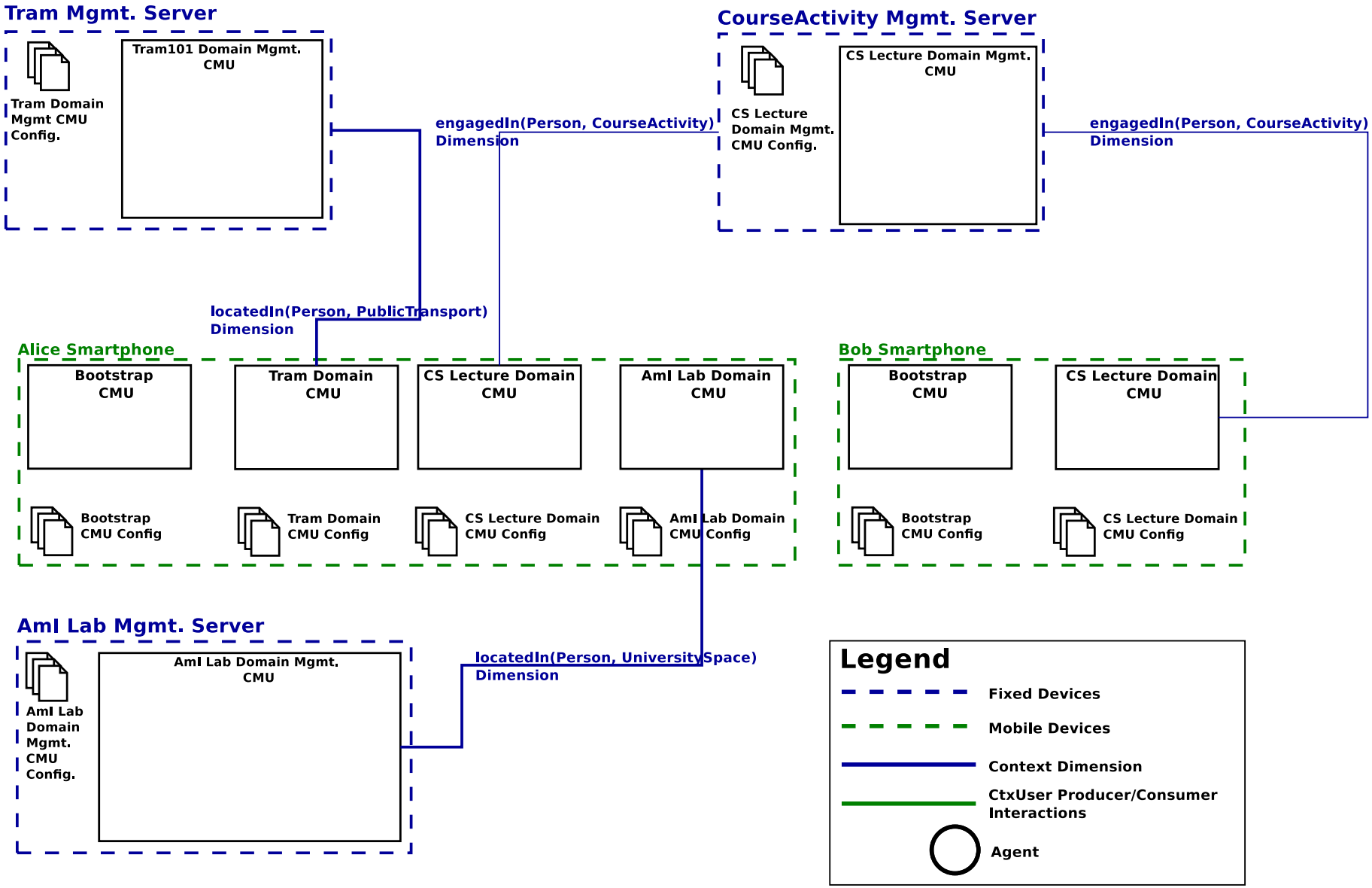
Deployment Structuring Elements

- **Idea**: create **link** between **multi-dimensionality of context information** and **CMU** assigned to service it

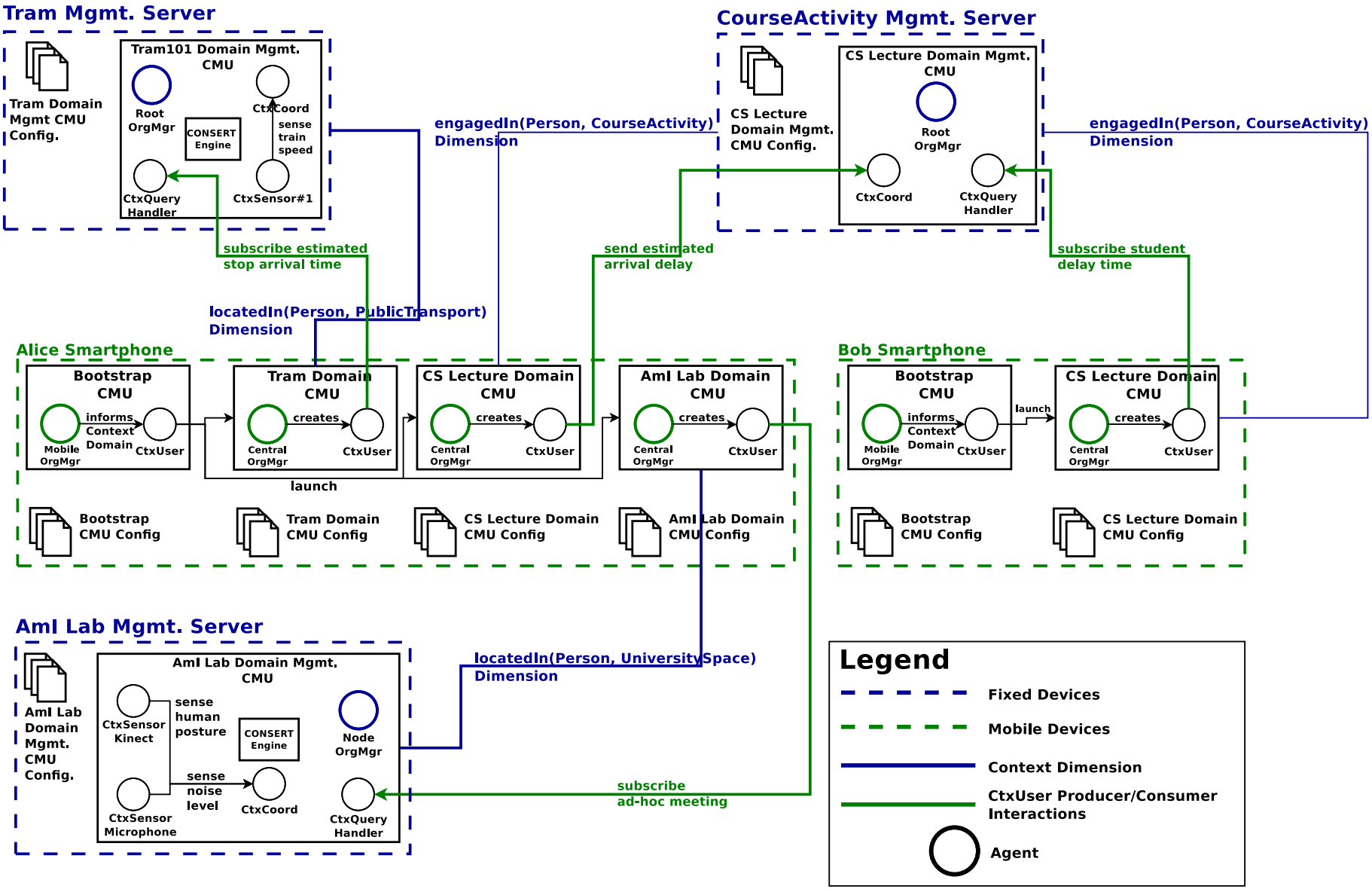
- **ContextDimension**: a *privileged direction* (e.g spatial, activity-related, relational) along which an application structures its context provisioning process.
  - Alternatively: the context information type whose instances are most likely to cause a **shift in relevant context**
  - E.g.: *locatedIn(Person, UniversitySpace)*, *engagedIn(Person, CourseActivity)*
- **ContextDomain**: a *logical partition* of the global application context model along a chosen ContextDimension
  - E.g.: *Tram*, *AmI-Lab*, *CS_Lecture*

- **one-to-one mapping** from a ContextDimension + ContextDomain pair to a CMU assigned to its management

Deployment Structuring Example



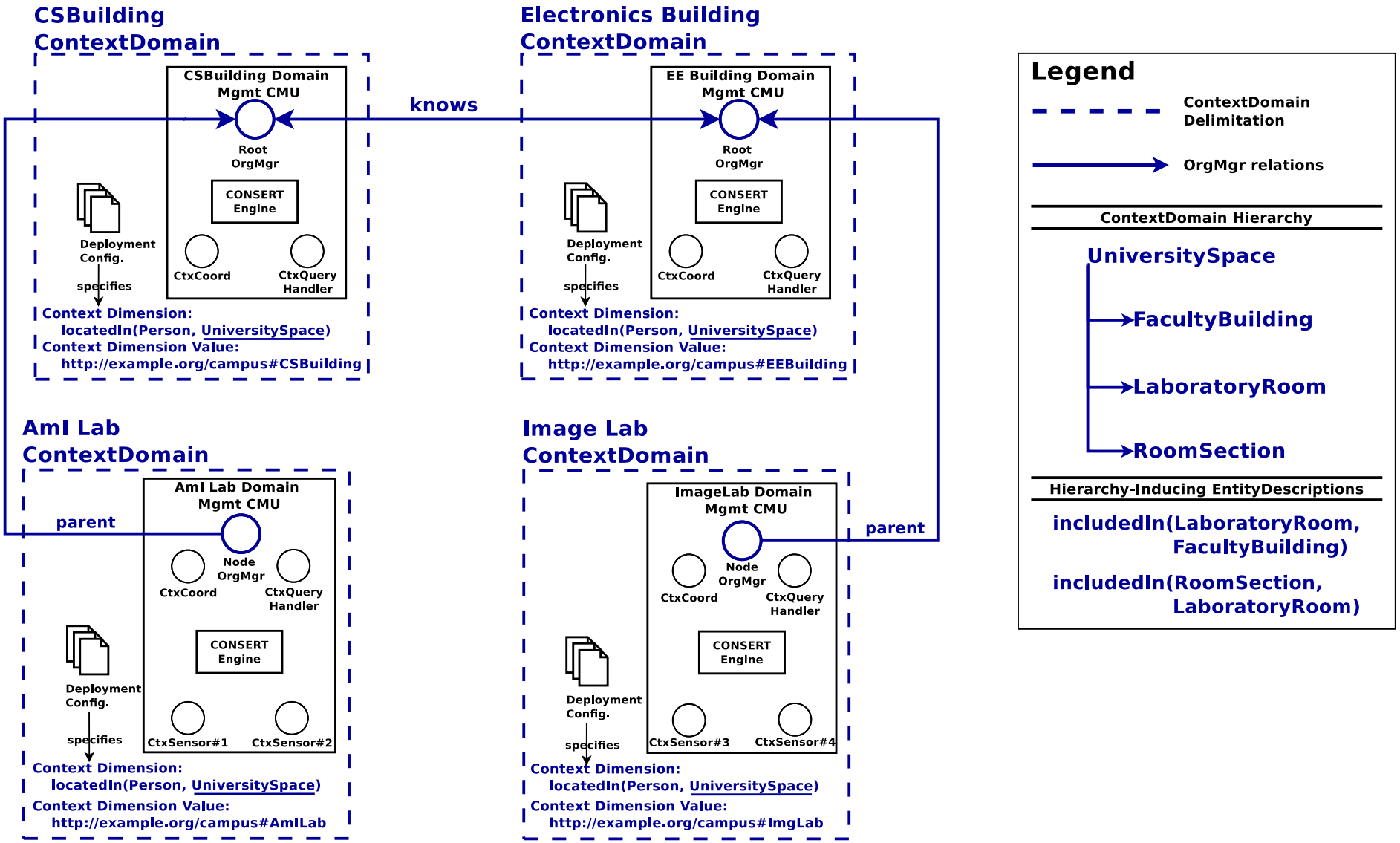
Deployment Structuring Example



Deployment Schemes

- 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 $\Rightarrow$ multiple CMUs

Hierarchical Deployment Example



Deployment Policies

- Set of parameters that specify deployment configurations
  - **Platform Config**: JADE Container setup which hosts one or more CMUs
  - **Context Domain Config**: information about the ContextDomain structure and the context model partition it contains
  - **Agent Config**: information about the provisioning agents that compose the CMU assigned to service the ContextDomain
- Parameter vocabulary implemented as an ontology

Deployment Policy Examples

Platform Config

```
:Container_AmILab
  a orgconf:AgentContainer ;
  orgconf:containerHost "localhost";
  orgconf:containerPort 1099;
  orgconf:hasMTPHost "localhost";
  orgconf:hasMTPPort 7778 ;
  orgconf:platformName "EF210".
```

Agent Config

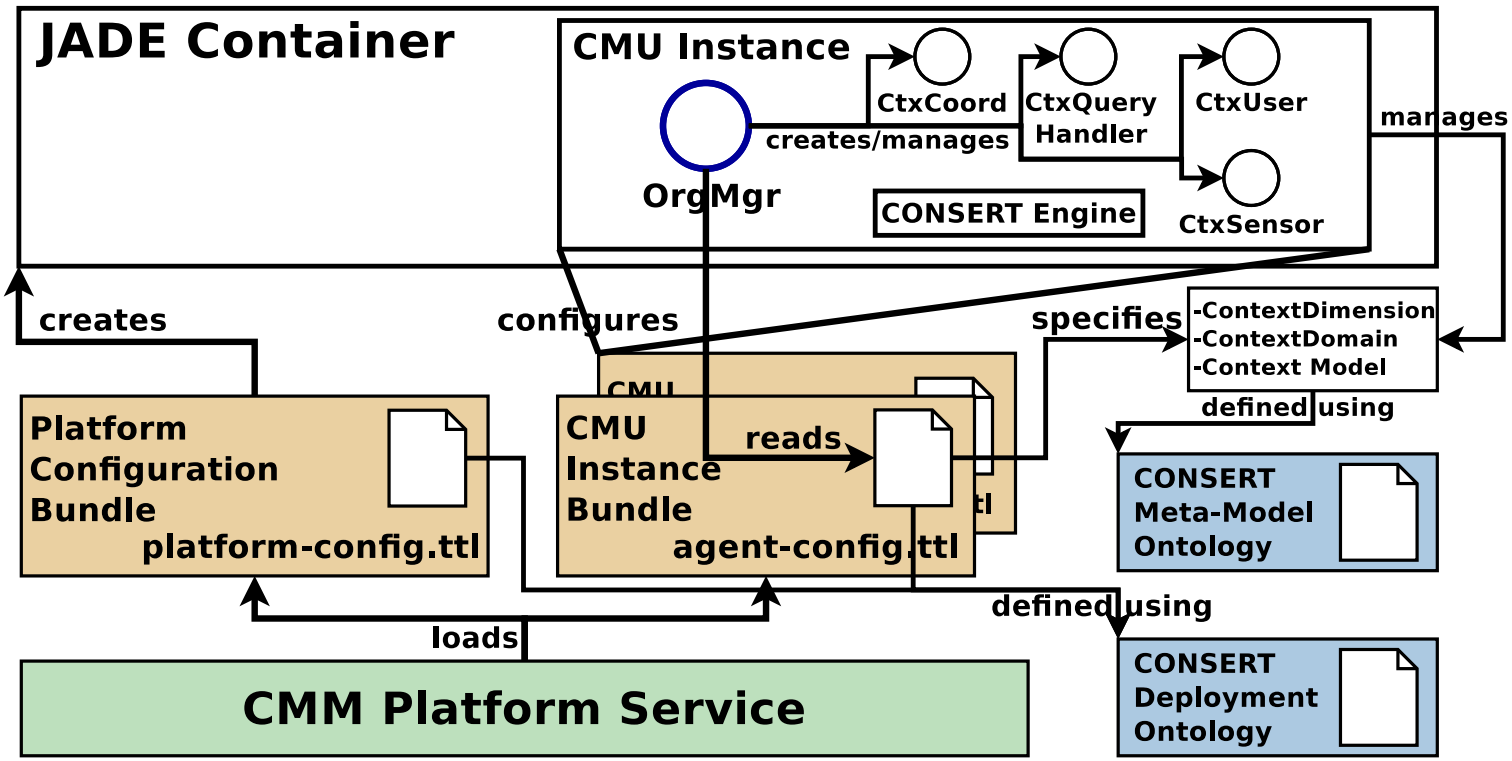
```
:CtxCoord_AmILab
  a orgconf:CtxCoordSpec ;
  orgconf:hasAgentAddress :CtxCoord_AmILab_Address ;
  orgconf:hasControlPolicy  [
    a  orgconf:AgentPolicy ;
    orgconf:hasPolicyDocument [
      a  orgconf:ContentDocument ;
      orgconf:documentPath "etc/cmm/coordconfig.ttl"
    ]
  ] .
```

ContextDomain Config

```
:AmILab_Domain
  a  orgconf:ContextDomain ;
  orgconf:hasDomainDimension person:locatedIn ;
  orgconf:hasDomainRangeEntity amilab:LaboratoryRoom ;
  orgconf:hasDomainRangeValue amilab:AmI-Lab ;
  orgconf:hasDomainHierarchyProperty
    space:spatiallySubsumedBy ;
  orgconf:hasDomainHierarchyDocument
    [ a  orgconf:ContentDocument ;
      orgconf:documentPath
        "etc/cmm/domain-hierarchy-config.ttl"
    ] ;
  orgconf:hasContextModel :AmILab_ContextModel .
```

```
:AmILab_ContextModel
  a  orgconf:ContextModelDefinition ;
  orgconf:hasModelCoreDocument  [
    a orgconf:ContentDocument ;
    orgconf:documentURI
      "http://purl.org/net/amilab/core"
  ] ;
  ...
```

Deployment Configuration



- CMU Instance can be: installed, started, stopped, uninstalled
- Life cycle state depends on the assertion of corresponding ContextDimension + ContextDomain
- Interaction between Platform Service and OrgMgr agent

Outline

Conclusions

- Introduce **explicit**, **model dependent** elements to support **structuring** of context management according to **dynamic use** of handled context information
- Focus on:
  - **Flexibility** of deployment scheme
  - **Ease of development / configuration** through use of declarative policies and service component based design

Conclusions

- Introduce **explicit**, **model dependent** elements to support **structuring** of context management according to **dynamic use** of handled context information
- Focus on:
  - **Flexibility** of deployment scheme
  - **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

Future Work

- From experience implementing the scenario $\Rightarrow$ need for **tooling** (e.g. an IDE) to ease effort of defining deployment configuration.

Future Work

- From experience implementing the scenario ⇒ need for **tooling** (e.g. an IDE) to ease effort of defining deployment configuration.

- 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

References I

Ducatel, K., Bogdanowicz, M., Scapolo, F., Leijten, J., and Burgelman, J.-C. (2001).
Scenarios for ambient intelligence in 2010.
Office for official publications of the European Communities.

Sorici, A., Picard, G., Boissier, O., Zimmermann, A., and Florea, A. (2015).
Consert: Applying semantic web technologies to context modeling in ambient intelligence.
Computers & Electrical Engineering.

THANK YOU!

Questions?