Context-aware Semantic Middleware Solutions for Pervasive Applications

Alessandra Toninelli

alessandra.toninelli@unibo.it
Università degli Studi di Bologna
Department of Electronics, Information and Systems

PhD Course – Infrastructure and Supports for Wireless Systems
Università degli Studi di Bologna – Facoltà di Ingegneria
17 September 2009
Outline

1. Context-aware Systems
   - Background & Motivations
   - State of the Art

2. Context-aware Semantic Middleware
   - The Semantic Web
   - Semantic Metadata
   - Context-Aware Access Control
   - Socially Aware Policies

3. Conclusions
   - Summary
   - Emerging Directions
Context-Aware Computing

- **Scenario**
  - Heterogeneous devices & connection interfaces
  - Mobile users, devices & services
  - Variable interaction conditions (location, preferences, device, social setting)

- **Goal** → To support service/application provisioning by adapting to changing operating conditions

- **Solution** → Context-aware (middleware) systems
What is Context?

- Context is any information that can be used to characterize the state or the activity of an entity, and the environment where this entity operates.

  (based on [Dey, Salber & Abowd, 2001])

  - Any information in principle could be context, depending on the application.
  - Location, device status & capabilities, user preferences, environmental conditions, date/time, past context, ...

- Context Awareness is the ability of a system/application to use available context information to change its behavior accordingly.

  → context-aware adaptation
Supporting context-aware adaptation basically requires the definition of three building blocks:

1. Conceptual models to represent context information, and propagate it up to the application level.

2. Concepts and mechanisms to represent & enforce context-dependent behavior → adaptation.

3. Appropriate support services to manage context information & support context-aware adaptation.
Context Models

- Variable levels of abstraction & granularity
  - Context attributes vs. contexts as entities
  - Application-specific vs. general purpose (?)

- Several KR models
  - key-value pairs
  - object-oriented models (contexts as objects)
  - markup schemes (XML)
  - graphical models (UML)
  - logical models (LP, DL, fuzzy logic), including:
  - ontological models (→ Semantic Web)

- Reasoning
  - deriving inferred context info
  - dealing with imprecise/conflicting context
Context-aware Adaptation

- **Issues**
  - Specify context-dependent behavior strategies
  - Enforce context-dependent strategies when need (proactive vs. reactive)
    E.g., "show to the user only services currently available in his location".

- **System/Architecture level approaches**
  - Metadata

- **Software level approaches**
  - Reflection techniques (programming)
  - Aspect-oriented design (software engineering)

  *Note that architecture and software level approaches can be jointly applied*
Metadata to Support Context Awareness

- Metadata provide a declarative approach to describe:
  1. the structure/meaning of resources composing a system and their state → profiles
e.g., "this device belongs to Alessandra and its battery status is now 70%".
  2. the specification of management operations expressed at a high level of abstraction → policies
e.g., "if the device battery goes under 30% disable application X".

- Metadata allow to specify context and context-aware adaptation strategies without affecting the application logic → separation of concerns
Any context-aware system should support a standard set of functionalities:

- Pre-processing might include context aggregation and/or verification.
- Reasoning might take place at all horizontal levels.
- Context-aware adaptation typically takes place at application level.
Context Management Architectures

- **Context access & retrieval**
  - Direct sensor access (with a loose meaning of sensor)
  - Context server
  - Middleware

- **Context distribution & management**
  - Widgets (sensor-oriented)
  - Networked services (service-oriented)
  - Blackboard model (information-oriented)

- **Software application design**
  - Libraries
  - Toolkits
  - Frameworks/Infrastructures
Outline

1. Context-aware Systems
   - Background & Motivations
   - State of the Art

2. Context-aware Semantic Middleware
   - The Semantic Web
   - Semantic Metadata
   - Context-Aware Access Control
   - Socially Aware Policies

3. Conclusions
   - Summary
   - Emerging Directions
Some History

- First article by Berners-Lee, Hendler & Lassila (2001)

- Initially a lot of hype and several contributions...
  - AI (ontology languages, rules, reasoners)
  - Web-oriented applications (annotations, folksonomy, ...)
  - DB management (huge ontology KBs)

- But after some years things were getting too complicated to be usable :-(

- Towards a usable trade-off
  - Coexisting ontology languages, used for different purposes
  - Small interconnected ontologies vs. huge agreed ontologies
  - Inconsistent knowledge as a matter of fact

... "A little semantics goes a long way", Jim Hendler
The Core Model

- The Semantic Web is a web of data

- The Resource Description Framework (RDF) is a language for representing information about resources (identifiable) in the World Wide Web
  - by identifying things using Web identifiers (called URIs)
  - by describing resources in terms of simple properties and property values.

- RDF provides a common framework for expressing information so it can be exchanged between applications without loss of meaning

Contents of this and the following slide are partly borrowed from http://www.w3.org/2007/03/sw
RDFS introduces a formal semantic schema behind RDF vocabulary
  - OWL provides enhanced expressivity → ontologies

Semantic Tools
  - Ontology management systems (editing, versioning, mapping)
  - Knowledge management (storage & access to data)
  - Reasoner engines
Semantic Metadata

Profiles & policies can be used to:
- define and propagate context information
- define and enforce context-adaptation strategies

However the underlying context model is always *implicit* (i.e., it is only known to the developer/designer)

Semantic metadata allow the representation of context and context-aware adaptation strategies with unambiguous semantics

1. Context metadata can be exchanged between applications without loss of meaning
2. Automated reasoning can be performed to infer new context information
Original Research Directions

- PhD Thesis → Semantic technologies (metadata) to build context-aware middleware

1. **Context-aware discovery – mature**
   - Context is used to provide mobile users with a personalized view on services
   - Semantic-based profiles allow flexible matching between user requirements and service offer

2. **Security, mainly access control**
   - Context-aware policies

3. **Socially-aware applications**
   - Context is used to personalize user experience with social applications
   - Social characterization of context (the other way round)
Context-Aware Access Control Policies

Example

In case of emergency, any qualified physician located within the hospital is allowed to access Alice’s health protected information.

- Access control policies
  - High level directives defining who can access which resource under which circumstances

- Traditional policies based on identities/roles – static
- **Context-aware** access control policies
  - Definition of policies based on context
  - Use of semantic technologies to represent & reason about policies/contexts
Context-Aware Access Control Policies

Example
In case of emergency, any qualified physician located within the hospital is allowed to access Alice’s health protected information

- Access control policies
  - High level directives defining *who* can access *which resource* under *which circumstances*

- Traditional policies based on identities/roles – static

- **Context-aware** access control policies
  - Definition of policies based on context
  - Use of semantic technologies to represent & reason about policies/contexts
Proteus policies associate resources to *protection contexts*

- A protection context is a set of attributes & constrained values
- The current state is a set of attributes & values measured by "sensors"
  - A protection context is *active* if context assertions describing the current state match its context elements

Activation of protection contexts (and associated policies) allows access to a resource
Example

In case of emergency, any qualified physician located within the hospital is allowed to access Alice’s health protected information

PersonalEmergencyContext ≡ ProtectionContext ⊓ ∃ owner.Alice ⊓ ∃ requestor.InHospitalQualifiedPhysician ⊓ ∃ resource.AliceHPI ⊓ ∃ environment.PersonalEmergency

< Dr. Green, located, EmergencyRoom >
< CurrentState, environment, PersonalEmergency >

- Context & policy representation based on Description Logic
Proteus Middleware Architecture

- OWL-DL to represent contexts and policies
- Java prototype v1.0
  - Pellet 1.5 DL reasoner via OWL API & SPARQL
  - PEM implementation via Java Security Manager extension (JAAS)
  - CM implemented on top of Contory context provisioning platform
Usable Security on Mobile Phones

- Relevant use cases in mobile phone usage modeled as *access control* issues:
  - access to user status information (status sharing)
  - access to the user’s attention (call filtering)

- Access control policies
  - High level directives defining *who* can access *which resource* under *which circumstances*

- Why is it difficult for mobile phone users to express their policies?
  - Users are not able to handle technical details – *only details*?
  - Users don’t understand policy models they are supposed to use
  - User strategies are *strongly* dependent on their social context – software systems are not
Design Requirements for Usable Policies

User understandable policies → Usable policy-based framework

1. *Intensional* definition of policies
2. *Socially aware* modeling of access conditions
   - Need for a social vocabulary describing the user’s environment
   - Created and maintained by the phone
3. Support for specifying policies only when needed
   Deferred policy specification
4. Support for viewing/grouping policies based on different criteria

*Warning: necessary but (probably) not sufficient*
Access Control Policies - State of the Art

- Powerful representation model for access control decisions
- Research on policy specification, management & enforcement
  - Different logical approaches (LP, ILP, DL, abduction, ...)
  - Semantic policies (e.g. KAoS, Rei)
  - Conflict analysis & consistency check
  - Policy refinement

- Limitations of existing policy languages/models
  - Little support for *social vocabulary* in policy definition
  - Social data remain *fragmented* (no semantic interconnection)
  - Inflexible policy definition and retrieval (*fixed* place, time & application)
A Socially Aware Policy Model

- A policy is modeled as a set of attributes and constrained values defining:
  - *what* (the resource)
  - *who* (the requestor - a socially aware description)
  - the *context* (when, where, why)
  - *how* (the access modality)

- The current state is modeled as a set of attribute-value pairs
- A policy is *in effect* if the current state values match the definition of the policy constraints.
Graph-based Representation of Policies

Example

My phone will ring if a friend of mine calls me

- A policy is represented as a set of RDF statements and/or
- SPARQL triple patterns in case of variables (to be matched at policy enforcement time)
Graph-based Policy Evaluation

- The current state is represented as a set of RDF statements (triples)
- RDF/SPARQL policy graphs are matched against current state triples
Why RDF as a Policy Language

- Semantic languages are well suited to represent and reason over social information (e.g., FoaF, CoaC)
- Social data constantly evolve - semantic languages are extensible
- RDF allows **semi-structured** data models
  - Humans tend to organize their knowledge in concepts and relations – typically flexible and adaptive wrt. to the situation

- *This is our modeling choice*
  Other KR models/languages might serve the purpose
Why a Graph-based Policy Model

- RDF graphs allow to create and **semantically interconnect** social data from different sources
  - Different applications running on the user’s behalf (e.g., calendar, address book, facebook)
  - Phones belonging to different users (e.g., colleagues, family)

- The policy graph can be browsed in multiple directions and starting from different applications
  - From the address book - *which status info can this person access?*
  - From the calendar - *who can call (access) me during this event?*
  - From the document manager - *who can see this document in which situation?*
  - And from the policy manager, too
Prototype Implementation

- Policy evaluation
  - Piglet + OINK (Nokia) to navigate and query RDF graphs – Python

- Prototype mobile phone application (address book) – ongoing work
  - Android platform (open source) – Java
  - Cross-application/deferred specification of policies
Outline

1. Context-aware Systems
   - Background & Motivations
   - State of the Art

2. Context-aware Semantic Middleware
   - The Semantic Web
   - Semantic Metadata
   - Context-Aware Access Control
   - Socially Aware Policies

3. Conclusions
   - Summary
   - Emerging Directions
Summary

- Context-aware systems at the state of the art
  - Context representation models
  - Representation & enforcement of context-aware adaptation strategies
  - Context management support services

- Semantic metadata to build context-aware middleware
  - Meaningful exchange of context information → interoperability
  - Reasoning to infer new context knowledge → flexibility and adaptation

- Semantic approaches require a tradeoff between complexity and expressivity
  - Performance (reasoning takes time and resources)
  - Scalability (when dealing with thousands of triples)
Emerging Research Directions

- Focus on \textit{information} in distributed systems design
  - Wide-scale integration of devices/services based on interoperable information exchange (including context)
  - Interoperability is moving at the information level – Semantic Web as key technology?

- Mobile phones as truly ubiquitous devices

- Emerging trends in ubiquitous applications
  - Social characterization of context – Social Semantic Web
  - Network-based approaches rising up to the application level – Web 2.0 and others
  - Cloud computing – infrastructure/middleware as a service
Open Issues

- Managing the "web of data"
  - Provenance tracking – where does this data graph come from?
  - Graph partitioning – how can we split this graph without loss of information?
  - Link metadata to data (e.g., link access control policies to controlled resources)
  - Information overload – usability and performance issues

- Novel approaches to (semantic) context-aware middleware should consider the network and socially aware nature of pervasive applications
Thank you

Thanks for listening...

... Questions?

- http://lia.deis.unibo.it/Staff/AlessandraToninelli/research.htm
- http://lia.deis.unibo.it/Staff/AlessandraToninelli/publications.htm