Grounded Contextual Reasoning enabling Innovative Mobile Services

Herma van Kranenburg [1], Alfons Salden[1], Tom Broens[2] and Johan Koolwaaij[1]

[1]Telematica Instituut, Enschede, Netherlands; [2]University of Twente, the Netherlands

Abstract - This paper reflects our findings on the technological feasibility of a mobile service scenario. We will show that despite the scenario being quite ordinary for end-users, the required service support functionality is rather complex. The realisation hinges on intricate grounded contextual reasoning about location information, user profiles and other situational context information. In particular, existing engineering methods and state-of-the-art technologies very well enable, for example, timely warning of a sales manager, who is late for and on his way to a business meeting. Furthermore, such techniques readily enable recommending the 'best' alternative travel schemes such that he arrives in time for his meeting. We report on our findings with implementing a sophisticated location reasoner of such a commuter-assistant that ultimately aligns commuting and meeting preference schemes of the sales manager.

1. Introduction

Applications can be enriched by being aware of the end-user context. It enables applications to be tailored to the needs and situation of users. Especially in the case of mobile applications and moving end-users this adds to an increased user appreciation and perception of added value services. For application providers, context aware services offered by an innovative service infrastructure are beneficial. This enables them to develop context aware applications with a short time to market. Such context aware services typically provide context gathering and interpretation functionality. Examples of context gathering services concern information on location, transportation means, traffic jams, priority and business-meeting schedules, while context interpretation typically concerns inferring transportation speed, projected arrival time and alternative travelling schemes. With services pro-actively responding to upcoming relevant changes in the environment, users are even better served. Such attentiveness and anticipatory aspects are part of a ubiquitous attentive system [1] that we aim at.

Contextual reasoning is a key element in context aware services; it involves processing network context. Here a network does not only concern a heterogeneous storage, computation and communication network (including end-devices) with middleware and application services and platforms living on top. Social and business networks may as well be parts of such a network. Of course, those networks are generally intertwined, attached or coupled to make a dynamic complex network. Thus, innovative services should be capable of reorganising and coordinating current and future networks at and surrounding the mobile node of an individual or group of end-users. The latter means that on the basis of analyses of various types of contextual information objects a selection and decision has to be made - at run-time - about those services to be offered or delivered to a user. For truly optimal contextual service offerings grounded contextual reasoning is a necessity. In the case of our scenario it means that on the basis of business meeting schedules and personal profiles and preference schemes a robust hierarchy of alternative ways of commuting to work have to be presented to the manager keeping in mind the manager's next business meeting and his current situation.

The basic principles of contextual reasoning are explained in [2]. In simple words, context reasoning is about deducing entailed contextual information from the various sources of context information. In the case of our context aware mobile services the problem of contextual reasoning concerns understanding which, why, when and how specific inference mechanisms compliant with multiple network contexts are used by and are valuable to people. In this respect, reasoning about heterogeneous network contexts for pervasive environments can certainly profit from the expertise residing in AI with regard to integration of heterogeneous knowledge and databases [3]. However, despite the large amount of work on context reasoning, we are very far from a general accepted network theory for contextual reasoning. So far, special forms of contextual reasoning have been proposed for specific purposes, but no one has succeeded (yet) in unifying all this work in a sensible single theory. The result has been a fragmentation of interests, methodologies and technical tools. In section 3, we give an overview of these and discuss this further.

Our paper is organised as follows. In section 2, we present and analyse our commuter-assistant scenario: our manager has just missed his train, but his mobile agent is perfectly aware of his whereabouts and the traffic and business situation he is in, and suggests viable alternatives by using contextual reasoning methods. In section 3 and 4, we review the principles, engineering methods and state-of-the-art technologies that can help in realising contextual reasoning in our scenario. We report on our first experiences and experimental results in building such a commuter-assistant system in section 5.

Manuscript received March 23, 2005. This work is part of the Freeband Awareness project (http://awareness.freeband.nl/). Freeband is co-sponsored by the Dutch government under contract BSIK 0325.

2. COMMUTER-ASSISTANT SCENARIO

This section describes a scenario in which contextual reasoning plays an important role to assist sales manager Ferdinand in choosing the best means of transport based on his business schedule and real-time public transport information. It demonstrates the complexity of the network context providers and reasoning required.

It is a cloudy morning and Ferdinand is in a hurry. Ferdinand is a sales manager of a major financial institute. Because of all the traffic jams, he prefers to travel by public transport. This morning he has an appointment with two customers for a major business deal and he does not want to come late.
Ferdinand steps out of the bus and notices that he is rather late for his train connection. He runs to the train station, but before even entering the station, he already gets a notice from his mobile device that his train has left. At least that saves him a run to the platform, and even better, the system automatically gives him alternative travel schemes:
"Ferdinand, you can travel by **taxi** or by **bus**. Taxi is more expensive (€*15*,-) and normally it would get you *on time* to your appointment. However, at this moment there is a large queue of people waiting for a taxi. So at this moment, the bus is the best alternative. It is less expensive (€ *4*,-) but will bring you *10 minutes* late to your appointment. The bus is standing on *platform 5* and leaves in 3 *minutes*."
Ferdinand decides to go for the bus, and hence he is scheduled to arrive 10 minutes late at his appointment. The system informs the customers that Ferdinand will be 10 minutes late and that he is currently in the bus on highway N35. Although Ferdinand will be late on his appointment the waiting time for his customers is minimized and they have timely feedback on his status.

To realise such a commuter-assistant scenario one has to keep track of the actual and past user travel preferences and service issues, such as arrival and departure times, location, platforms, meeting times, costs, etcetera. This requires the following context providers and services:

- Travel Preference Scheme Provider;
- Business Meeting Scheme Provider;
- Train Service Provider;
- Taxi Service Provider;
- Bus Service Provider;
- Location Provider.

It is important to note that these are broad classes of context providers, differing - even within a class - in quality of context, usage characteristics, costs, and availability. To make this more concrete, consider the diversity of location providers for the commuter in our scenario:

- GPS location provider, residing on his mobile device, communicating with his GPS receiver, good precision, no costs, available outdoors and limited by battery power;
- Network location provider, service residing in the telecom operator domain, accessible via GPRS/UMTS, available with broad coverage, limited precision, costs are involved, requires subscription and authorisation;
- Local location provider, uses locally available communication channels (Bluetooth, Wi-Fi, RF, GPRS) to spot the identity of other nearby devices/cells/tags and does a lookup in a central registry to obtain the location of the static ones (if available), precision and costs depends on the communication channel, availability limited by nearby devices and coverage of the central registry;
- P2P location provider scans via a small area network (e.g. Bluetooth) for nearby devices that offer a location service, works everywhere, usually free of charge, availability depending on peers.

To present Ferdinand timely with alternative travel schemes the commuter-assistant system needs to collect data from several context providers and reason with the contextual objects to do a rightful interpretation and come up with optimal transportation recommendations. Necessary steps include:

- Collecting time-schedules and meeting-priority;
- Collecting available travel services and their transport timetables;
- Collecting locations (of Ferdinand, train, available taxis);
- Collecting traffic situation (like traffic jams);
- Inferring price and expected arrival time at the meeting's location for the available travel service providers (e.g. 2 taxi providers and 1 bus);
- Inferring the impact of selecting either of the available travel provider service offerings (e.g. projected time of too-late arrival with the current situation, risk of another delay yet to occur);
- Inferring hierarchical ordered list with best two alternative travel schemes.

The service of the commuter-assistant system to Ferdinand might seem simple to an end-user. Nevertheless, the analysis in this section makes clear that quite innovative and complicated service support functionality on context gathering and reasoning is required to realise it.

3. REASONING FRAMEWORK

In the following paragraphs we first present the main *principles* of contextual reasoning followed by the most relevant engineering methods for grounding and *categorising contexts* robustly. Using these principles and categorisation approaches suitable *context representation* and *reasoning methods* are presented.

3.1 Principles

Contextual reasoning exploits relationships among local structures engendered by different contexts. For example, the business scheme and the travel preferences of the sales manager determine how to weigh the different travel service alternatives and how to present them. Herein two general principles underlie contextual reasoning [3]:

- Principle of locality - reasoning about local structures takes place in partial, approximate, and perspective representations of the world;

- Principle of compatibility - constraints on the reasoning process are in line with one or more context-models.

Contextual reasoning for advanced mobile services boils down to making (fuzzy) truth assignments for such a service on the basis of contextual parameters provided and assessed by various network entities. These network entities can for example yield:

- Heterogeneous network contexts (e.g. "He missed the train and thus is confronted with a specific travel service schedule and has the following upcoming business meetings");
- Hierarchies of nested network contexts (e.g. "He prefers an ordering of alternative travel services according to his preferences.").

Raw network data can form low-order context for a mobile service, while interpretations and combinations of lower-order contexts can yield higher-order contexts. Relevant low or high-order network contexts for a mobile service can be constructed on the basis of network models. Such models can generate hierarchies of nested network representations of various granularities that can serve network context representations. In other words, such models can yield various categorisations of network contexts that in turn can be used for contextual reasoning purposes.

3.2 Context categorisation

In contextual reasoning about mobile service provisioning the reasoning engine can take various network contexts as constraints for making appropriate inferences. Following our scenario network contexts can involve various actual travel service schedules, but also business meeting and preference schemes of an end-user. An inference of what mobile service to deliver given those contexts can be expressed in terms of a concatenation of logical operations on the actual schedules and schemes of the end-user.

The commercial success of proactive and attentive mobile services will be mainly determined by user acceptance. Latter acceptance depends in turn on whether such services are grounded by the actual behaviours of end-user with current and envisioned mobile services. This implies that the application behaviours of the advanced mobile services should reflect the expectations and the needs of the end-user as much as possible. This requires a grounding of network contexts as well as inference rules used by the reasoning engine. The question arises how to ground those network contexts and inference rules.

Grounding of context and inference rules can come about by rigorous complex network modelling [4]. Modelling of the grounding process certainly should precede and should implement any conceptual model for context aware applications [5]. For our purposes we can restrict ourselves to classical engineering methods well known in natural language processing and computer vision, where contextual grammars together with lexical contexts help reveal and ground semantic meaning of a word in a sentence [6]. In the case of our scenario these engineering methods can ground network contexts robustly by aggregating, processing and predicting user preferences given business meeting schemes and actual travel service schedules. By applying network-specific clustering techniques robust categorisation and dimensionality reduction of the categorisation of network situations and contexts can be achieved. Such categorisations also concern multimodal dialogues and possible schemes among systems and humans [7], like the presentation schemes for the alternative travel service schedules.

Approaches for obtaining grounded and robust categorisation of network contexts can be based on template matching, Principal Component Analysis (PCA), structural matching, neural networks, Bayesian Network Classifiers [8], decision trees, K-Nearest Neighbours, Hidden Markov Models (HMM [9]), dynamic scale-space paradigms, reinforcement learning, support vector machines (SVM) [4]. In the case of our commuting sales manager scenario a geometry underlying the social network is still lacking. Most of the more sophisticated engineering methods such as PCA and syntactic or structural matching need a well-defined geometry to be applicable and to be superior to methods based on HMM and Bayesian Network Classifiers. Diverse types of system information (like location obtained via GPS) can be collected over long time periods to trace the whereabouts of e.g. the sales manager. Furthermore, these locations can be gathered in conjunction with other human-system interactions of the sales manager with his commuter-assistant system agents. Locations and human-system interaction and decision behaviours can subsequently readily be incorporated into a HMM. A Bayesian Network Classifiers allows in addition to predict a customer's interaction and decision behaviours. Thus also his preference schemes concerning his desired commuter-assistant system service, e.g. the preferred hierarchy of presented commuting alternatives given a traffic jam and a business schedule of the sales manager, can be distilled.

The network contexts grounded by above-mentioned engineering techniques can form inputs for a model of Propositional Logic of Context (PLC) [3] that maps each context sequence into a set of partial truth assignments. Such a PLC can in turn be embedded in Local Models Semantics / MultiContext Systems (LMS / MCS).

3.3 Context representation

Knowledge representation is at the heart of the modern research in artificial intelligence (AI [e.g. 19]). Its principles and techniques are applicable to contextual reasoning problems related to mobile services. Knowledge representation schemes capture namely essential features of a problem domain and make that information accessible to a problem-solving procedure. In addition knowledge representation languages help humans to understand the problem-solving procedure.

A major problem in network modelling is handling complexity. A proven method coping with this is data abstraction consistent with the network model [4]: the representation of only that information needed for a given contextualisation purpose should be retained at a certain level of aggregation. An AI representation language must support

such more human-like qualitative problem solving (rather than quantitative), reasoning (rather than calculation) and organising large and varied amounts of knowledge (rather than implementing a single well-defined algorithm).

In order to cope with qualitative reasoning the bindings of variable names, objects and values should be handled in a highly dynamic fashion. E.g. adaptive (learning) systems taking advantage of above network models have a good ability for generalisation allowing them to correctly apply learned knowledge to novel situations.

Typically, representing a problem domain requires a large amount of highly structured interrelated knowledge. E.g. not only components of an object (like human and car), but also their interrelation and combined parts should be described. Taxonomic information and semantic relationships are required. Ontologies are ways to capture both meaning and relationships: an ontology is a formal specification of a conceptualization [10] where concepts are distinguished by axioms and definitions stated in a logic. Therefore an ontology is somewhat similar to a thesaurus, dictionary or glossary, yet with much greater detail and structure that enable computers to process its content. By establishing a common vocabulary among applications, ontologies support the sharing and reuse of formally represented knowledge. Ontologies can be categorized by scope as exemplified by the layered architecture of ontologies developed within the IST-Project WonderWeb, see Figure 1. Ontologies at lower layers provide representation requirements for the higher layers, whereas ontologies at the upper layers provide design guidelines to the lower layers. They are typically distinguished as so-called foundational, core ontologies and application specific ontologies.:

- *Foundational Ontologies*: contain high-level domain-independent concepts (broad coverage),
- *Core Ontologies:* provide domain-specific infrastructure (medium coverage),
- *Application Ontologies:* relate concepts and properties in domain of interest (small coverage).

Foundational ontologies together with the more abstract parts of core ontologies are sometimes referred to as upper ontologies.

By providing repositories of standardised knowledge representation primitives, upper ontologies foster the semantic interoperability in distributed information systems. Additionally, the aligning to upper ontologies can provide a solid underpinning to application ontologies and may help to exclude terminological and conceptual ambiguities resulting from unintended interpretations. Last but not least the principles and engineering methods of sections 3.1 and 3.2 clearly make operational and ground a robust representation of such ontologies [4].

A higher-level notion of structure (beyond just using collections of predicates or similar formalisms) helps to deal with complex concepts in a coherent fashion. Semantic networks are suitable for such. Here, an algorithm for reasoning about the domain in question can make relevant associations simply by following the links in the network. In addition links can indicate class memberships that allow properties attached to a class description to be inherited by all members of the class. Again the engineering methods of section 3.2 make operational and ground robust semantic networks [4].

The notion of context is widely studied in different areas of AI. More on the work of formalisation of context in AI can be found in [11,12] and on contextual reasoning in multi-context systems in [13]. Examples of contextual reasoning with regard to the integration of heterogeneous knowledge and databases can be found in [14,15,16].

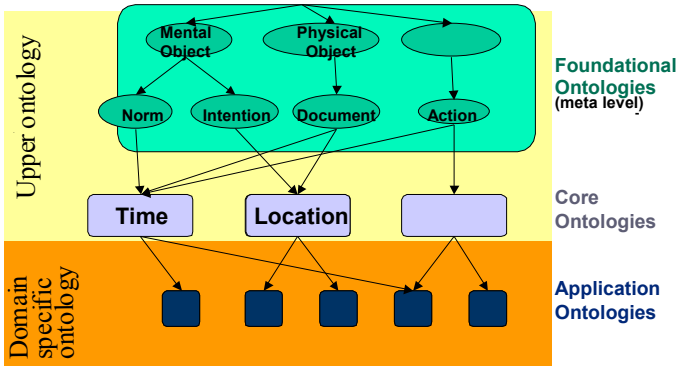

Figure 1. Ontology structuring.

3.4 Reasoning methods

Contextual reasoning serves a selection and decision making process concerning which applications and services to offer or to provide a network entity given a particular situational context. The basis of any contextual reasoning method in pervasive computing is a network model (see also section 3.2) that describes and predicts the states and dynamics of all past, existing and future network entities. Such a model actually defines the reasoning methods that are applicable. Furthermore, it defines also possible contexts along which or to which those reasoning methods may be subjected.

Logically well-founded ontologies do not only offer ways for describing a domain of interest (see section 3.3), but also allow to reason about the represented information. Ontology reasoning in terms of domain specific conceptualization requires logic-based inference systems that have been well studied within the field of knowledge representation in the AI community in the past. Description Logics (DLs) as a decidable fragment of first-order predicate logic turned out to be an adequate formalism for representing and reasoning about expressive ontologies [17]. DLs form the formal foundation of W3C's Web Ontology Language (OWL) [18].

In simple words, contextual reasoning is about interpreting a specific issue in a complex network given particular contextual constraints, e.g. which transportation means to choose and suggest given the schedule and preferences of sales manager Ferdinand. Such a reasoning process may yield an instance of a key element needed in a further selection or decision process, e.g. when the taxi is suddenly delayed. Such a key element may in turn be expressed in terms of a grounded upper-contextual reasoning serving selection and decision making about related issues.

In the interpretation process, use can be made of various reasoning techniques (e.g. rule-based systems, model-based

reasoning, case-based reasoning, see e.g. [19]) and knowledge bases (containing both general knowledge as well as case-specific information) that are consistent with a particular network model. For example in a rule-based system, this knowledge is represented in the form of *if… then…* rules. The inference engine is essentially an interpreter for the knowledge base; it applies the knowledge to actual problems. Reasoning can deal with partial context information such that implicit information can be inferred. Typically the upper context ontology uses standard vocabulary, like OWL, and is supported by of the shelf reasoning components like RACER [20] (see section 4).

Reasoning allows entailed (or "new") knowledge to be inferred from a set of facts and rules. Essential to any intelligent entity is the ability to derive additional knowledge from a world description. It is impossible to store an inflexible description of every possible situation, hence abstract descriptions of classes of objects and situations (see section 3.3) are used to formulate and reason about. For example, the rule "for all X, X is clear if there does not exist a Y such that Y is on X" allows the system to infer - from given facts - whether or not there is something on top of another object. The variables make the intelligent system as general as possible.

The ability to learn from examples, experiences, or high-level instructions (instead of being hard-coded) depends on the application of meta-knowledge. Meta-knowledge is about "knowing about what you know". E.g. an intelligent system should not only know things but also should know what it knows. It should be able not only to solve problems but also to explain how it solved the problems and why it made certain decisions. It should recognize the limitations of its knowledge and learn from its interactions with the world.

The commonsense human approach to problem solving supports the hypothesis that exploring alternative choices solves NP (Nondeterministically Polynomial) problems. This problem-solving approach underlies the state space search technique. This technique provides a means for automating intelligent or heuristic strategies for selectively exploring the problem-solution space. Here intelligence and heuristics guide the search along lines that have a high probability of success while avoiding wasted or apparently stupid efforts.

A state space may be searched in two ways: either from the given data of a problem instance toward a goal ('data-driven') or from a goal back to the data (goal-driven'). Data-driven search is also referred to as forward chaining; goal-driven as backward chaining. In forward chaining you start with the given facts of the problem and a set of rules for changing state. Search proceeds by applying rules to facts to produce new facts, until a satisfactory goal condition ("*relevant* context information") is reached. In backward chaining one starts with the goal and considers the rules that could be used to generate that goal and tries to determine what conditions must be true to use them. These conditions become then the new goal for search, until the working backwards ends at the fact of the problem. Again the engineering methods of section 3.2 can make operational and ground data-driven and goal-driven searches [4].

Applied to our commuter-assistant scenario: The inference and hierarchical ordering mechanisms are the most crucial from a contextual reasoning perspective. The former and the collection mechanisms are initiated by the secretary-commuter system whenever inconsistencies occur between the actual travel scheme of the sales manager and those of the travel service providers. For the Travel Preference Scheme Provider a dynamic user profiling management system is indispensable. It may do history management and apply several engineering methods about past travel service interactions and decisions of the sales manager. This allows to distil grounded and ordered travel schemes related to time, locations, inconsistencies between actual travel schedule of the sales manager and those of the travel service assistants and business meeting schedules [6]. Furthermore, it allows the commuter-assistant on the basis of the above contexts to infer and to display at run-time this preferred order of travel alternatives by a concatenation of event condition action rules. In order to apply those context-dependent rules several technologies exist, which are surveyed in the next section.

4. ENABLING TECHNOLOGIES

Machine languages, reasoning engines and tools can help to support contextual reasoning on categorisations of various networks. They can provide mechanisms to represent data, applications and services, and enable to reason on data by applications or services. In our commuter-assistant scenario such enabling technologies can help contextualize the travel services on the basis of the business schedules and the preferences of the sales manager.

4.1 Machine languages

To reason on context by means of machines, it should be defined in some formal language that can be parsed by an automaton. Many of such machine languages have been proposed for intuitively representing concepts like business schedules, preferences and travel services in expert systems, planning systems, knowledge management systems, etc. Formal logics and languages provide a more or less natural translation of those concepts into objects or agents of a related computational model.

Nowadays, there exist several languages that can handle both exchanging objects across the World Wide Web, as well as dealing with cooperation among heterogeneous agents. Examples of these are Knowledge Interchange Format (KIF) [21], Open Knowledge Base Connectivity (OKBC) [22], XML-based Ontology exchange Language (XOL) [23], Resource Description Framework (RDF) with RDF data query language (RDQL [24]), Darpa Agent Markup Language (DAML) plus Ontology Inference Layer (OIL) (DAML + OIL) and Web Ontology Language (OWL), see also Figure 2.

KIF was one of the first knowledge representation languages. It enables the interchange of knowledge among disparate programs.

OKBC is a frame-based language. It provides a uniform model of Knowledge Representation Systems (KRSs) supporting networked as well as direct access to knowledge bases. The Foundation of Intelligent Physical Agents (FIPA) uses the OKBC representation.

XOL enables the exchange of ontology definitions among different systems like database systems, ontology development tools or application programs.

RDF provides a lightweight ontology system to support the exchange of knowledge on the Web. It can be used to for instance to describe statements like "Ferdinand … late for his train connection". Key element of this approach is the relation (called 'predicate') between concepts that can be traversed to reason on concepts (e.g. context **"**This morning he has an appointment with some customers for a major business deal and he does not want to come late."). RDQL enables specification of queries to retrieve certain concepts, e.g. preferences and travel services, from an ontology given a particular context.

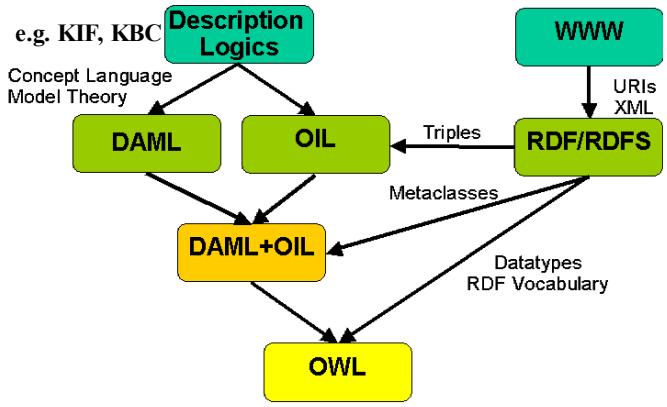

Figure 2. Language extensions and dependency between data models, logical paradigms and standards.

DAML + OIL is a joint standard for specifying and exchanging ontologies. Its definition is based on existing frame-based language such as OKBC, XOL RDF and RDF Schema with richer modelling primitives.

OWL is a W3C standard for ontology and metadata representation that facilitates machine interpretability of Web content and enables applications processing this content. It is based on DAML+OIL, XML and RDF/RDF-S. OWL is defined as three sublanguages: OWL Lite, OWL DL and OWL Full with a growing level of expressiveness and complexity. Choosing between them is based on one's requirements in engineering ontologies. Generally speaking, the design decision treats the trade-off between the expressiveness of OWL versus an efficient OWL reasoning support. OWL Lite and OWL DL represent the decidable fragment of OWL with good tool and complete reasoning support.

Currently OWL is becoming the standard for specifying metadata. Advantages of OWL-DL are its expressivity, monotonicity, and decidability. Disadvantages are its language complexity and lack of support for property chaining and it does not allow for procedural attachments. It has however, good tool support and enables us to reason on concepts by traversing relationships. Therefore, for our scenario, OWL-DL is the most feasible specification language to use and in our limited scenario we might even suffice for OWL-lite.

4.2 Reasoning engines

An individual expert system applies its inference engine to a database of knowledge in search for solutions to a given problem. Distributed knowledge management systems rely on the co-operative capabilities of various individual expert systems to solve more complex problems. In our scenario, one can readily imagine that contextual reasoning and cooperation is needed whenever the sales manager is abroad and still using his commuter-assistant service e.g. to go to a conference. Having an appropriate meta-ontology available that relates the different ontologies across the globe, reasoning systems can make appropriate decisions to adapt or activate an application or service on the basis of an observed context all represented in the ontology at hand. These reasoning systems can be rule-based, constraint-based or probabilistic.

Rule-based reasoning engines can use Prolog [19], a subset of first order logic, as a logic language to define rules. For example, open source and Java-based JTP [25] provides besides a hybrid reasoning system architecture also a reasoning system component library that enable rapid building, specializing, and extending of (backward-chaining and forward chaining) reasoners based on such rules. Each reasoner in a JTP hybrid reasoning system can embody special-purpose algorithms and maintain JTP system's knowledge. Another example is Jess (Java Expert System Shell) [26] that supports the development of rule-based systems, which can be tightly coupled to code written in the Java language.

Constraint-based reasoning engines use the inherent constraints in a problem to rule out impossible alternatives before and during the search for a solution [27]. They are in particular valuable in our scenario in which the system is simultaneously confronted with several scheduling, planning, resource allocation and routing problems. There exists a complete environment for the design and development of such decision support systems called ChiP [28].

The choice for every component (e.g. reasoning engine) in the reasoning process depends heavily on the choice of machine languages. As we chose for OWL as machine language the other components should comply with this language. E.g. in [29] an implementation of a context aware application is done by using OWL, in combination with RACER and JESS.

4.3 Tools

Several OWL tools exist such as API's/parsers like Jena [30], OWL API [31] and OWLP[32], and graphical OWL editors of ontologies, like Protégé [33] and SWOOP [34]. Another tool is KAON [35] that enables ontology creation and management. It provides a framework for building

applications that require scalable and efficient ontology-based reasoning. RACER [20] provides a description logic based reasoner for managing and reasoning about ontologies. It is OWL-compatible and has semantic middleware for proof checking, ontology validation and classification.

For rapid development of context aware mobile services there exist the Context Broker Architecture (CoBrA) [36] or SOUPA [37], and GAIA [38] infrastructure. Making use of our context categorizations, contextual reasoning methods and appropriate ontology they could help realizing our scenario very quickly.

CoBrA is an agent-based architecture for supporting context aware systems. Central is a context broker who maintains a shared network model, ontology and a common policy language on behalf of a network of agents (being SW agents, context providers, context brokers, context aware service providers, users and customers), applications, services, communication and computational networks, and devices. The CoBrA and SOUPA ontology are both designed to model and to support pervasive computing context aware applications and services and are OWL compliant. The extensible CoBrA model can represent, manipulate and access context information for the purpose of context aware service provisioning like in our scenario. First-order probabilistic logic, high-order logic and Bayesian network methods can be applied for further reasoning about various context information important for both controlling and regulating attentively and proactively existing and novel context aware services.

GAIA is a CORBA-based development infrastructure for building interacting autonomous agents. It provides event, presence, discovery and naming services to agents. With increasing network complexity there is a high need for ways of sustaining robust service discovery, matchmaking, interoperability and context awareness of agents. A GAIA compatible ontology has been developed for just that purpose [39]. It can sustain - despite the rise in complexity of even the overall network dynamics - descriptions of agents and their relations, rules or policies for agents and their relations, and the different types of contextual information. This way GAIA can augment configuration management, discovery and matchmaking, HCI, interoperability and context aware services. Thereto, GAIA ensures that the Ontology Server maintains the ontology. For checking logically consistency of the ontological concepts, logical retrieval and (context) reasoning about the autonomous agents the CORBA FaCT Reasoning Engine is used. For dynamic discovery purposes the Ontology Server registers regularly with the CORBA Naming Service.

5. Realisation and Experience

To demonstrate the concepts and techniques for context reasoning that are outlined in this paper, we have built a demonstrator that implements the first part of the scenario mentioned in section 2, namely that part of the scenario where the commuter is notified of the fact that he has just missed his train. Even such a simple notification requires substantial intelligence in gathering relevant context information and reasoning about that information. Due to the fact that information, such as the train is leaving now, is probably not available directly, it has to be obtained from other indirect sources.

The commuter in our scenario has a number of location providers at his disposal that are all wrapped as a semantic web service that relates local concepts to a central and widely accepted ontology. All these location providers can be discovered, and advertise their interface description as well as other relevant parameters like costs, accuracy et cetera. We have implemented a number of example context wrappers (compare the functional view [40] in Figure 3), including:

- GSM GPS location wrapper, which is a wrapper around a Bluetooth-enabled GPS receiver connected to a Symbian 7 phone. It delivers location information.
- GSM Cell ID wrapper, which offers information on the GSM cell in which a Symbian 7 phone is currently located. Together with a GSM cell to location converter this could be used as a source of location information as well.
- Messenger wrapper, which connects to Windows Messenger and delivers information on the status of a user on a PC or laptop.
- Outlook wrapper, which connects to Microsoft Outlook. It can give information on the scheduled activities of a user and the location of those activities.

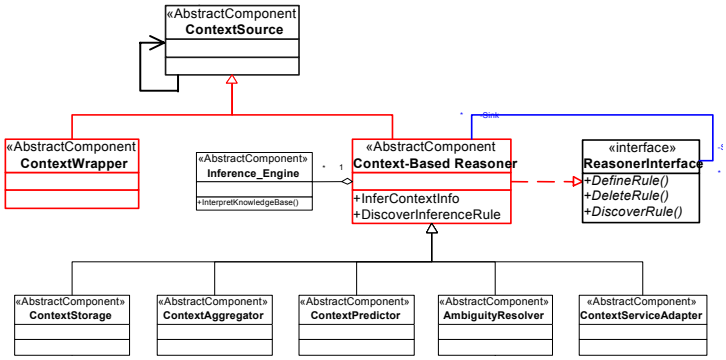

Figure 3. Functional view of context sources: various context-based reasoners and wrappers all deliver context information and inference rules can be defined at the reasonerInterface.

These wrappers hide platform- and device specific details from the generic infrastructure. Amongst the wrappers above are three that can act as a source of location information, in various forms, that we use for our reasoning validation case on the commuter-assistant scenario.

One of the platform specific components allows Symbian 7 phones to advertise an 'Awareness ContextProvider' service to the outside world, see the screenshot in Figure 4. This means that software running on another Bluetooth enabled device can scan for devices that offer this service. They can then connect and (if allowed by the owner of the device) retrieve context information from that device in a peer-to-peer fashion.

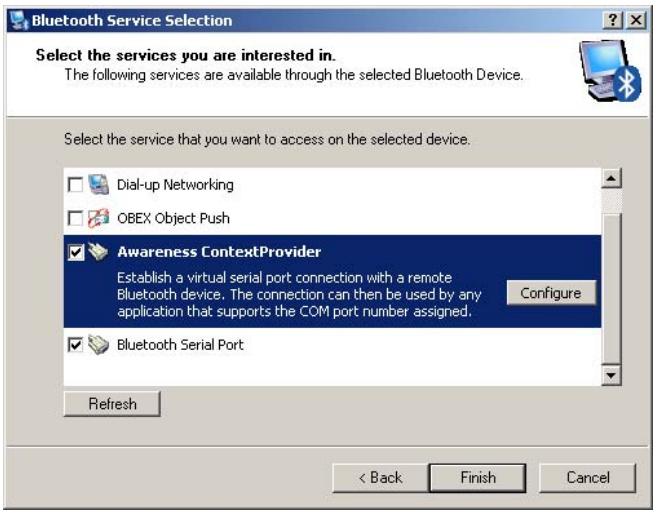

Figure 4. Bluetooth services available on a device.

Depending on the user profile and the current task, a relevant location provider is selected from a hierarchy of such providers to obtain location information. For example, the user profile indicates that costs should be minimal. The result is that the context manager uses the GPS Location Provider when the commuter is walking towards the station, but switches to the P2P Location Provider when the GPS looses its fix. When that does not result in a viable location, it switches to the Local Location Provider (which might cost some money, see the discussion in section 2) and it uses the Network Location Provider as a backup if everything else fails.

In Figure 5, we display a specific constellation of a set of context providers and reasoners to deliver this event information (train has left) irrespective of the communication channel (in the example: GPRS or Wi-Fi). With the help of a context reasoner it becomes possible to switch from a GPRS location provider (cell-ID based) to a local location provider (using the Wi-Fi hotspot's MAC address as input), when the user is roaming seamlessly from GPRS to Wi-Fi, and vice versa. This is done by using a context reasoner component that monitors a third context provider for changes in the type of network used by the end-user. Upon such a change, it switches seamlessly to another location provider.

Figure 6 and Figure 7 show screenshots taken from the mobile terminal showing the situations of detection of a access network change by the communication monitor. It alerts the commuter reasoner, which in turn changes the location providers with the result being that Ferdinand also has location information while being inside the station building. Moreover, this location information is provided on the lowest costs possible due to his user preference setting.
The commuter reasoner also has access to other context providers, including one that wraps the commuter's schedule. With that information it uses an external service provider, in this case a public transport planner, to plan Ferdinand's optimal travel schedule. As a next step, the context reasoning comes into play. We use a rule-based system for reasoning, with an OWL ontology as knowledge base. From the travel schedule (that is accepted by the commuter), his location and the current time, it derives the fact that the commuter is bound for a certain train. This can be expressed as a rule at the commuter reasoner interface (compare Figure 3 [40]). The rule being "if user is moving in the direction of the scheduled location of a transport vehicle mentioned in his travel schedule in the assigned time slot, he is bound for that vehicle". The context of bound-to is described in the OWL ontology. When that rule is evaluated positively, an event triggers the process that starts the monitoring of the particular transport vehicle.

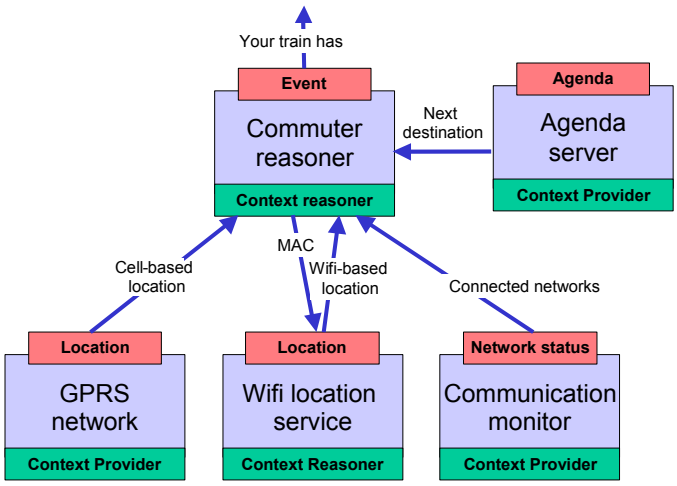

Figure 5. A specific example of a provider-reasoner constellation to deliver a notification that his train has left, based on location information of a group of users with the same destination.

The monitoring of this vehicle (in this case the train) could be as easy as setting a trigger in a central service offered by the railroad company upon the departure of a particular train. In most countries, such a service does not exist, but fortunately there are alternatives in case based reasoning. This method uses the group of users of the same system that are also bound for the same train. Their collective location information can be used – in an anonimized way - to infer that the train has left [41]. This can also be formulated in a rule based reasoning system: if the locations of a substantial subset of the group of users bound for a train match, that matching location is the location of the train. And if the location of the train moves away from the departure station, the train has left. And if the train has left without the commuter, he gets a notification of that very fact. Inclusion of such system dynamics as the behaviour and situation of others bound for the same train is our next goal in implementation. This extends on the state-space search, which we know is not sufficient for our goal, since we need rather a characterization of system dynamics [4]. Further we will extend the validation of our designed reasoner components, including the ambiguity resolver with multiple location providers

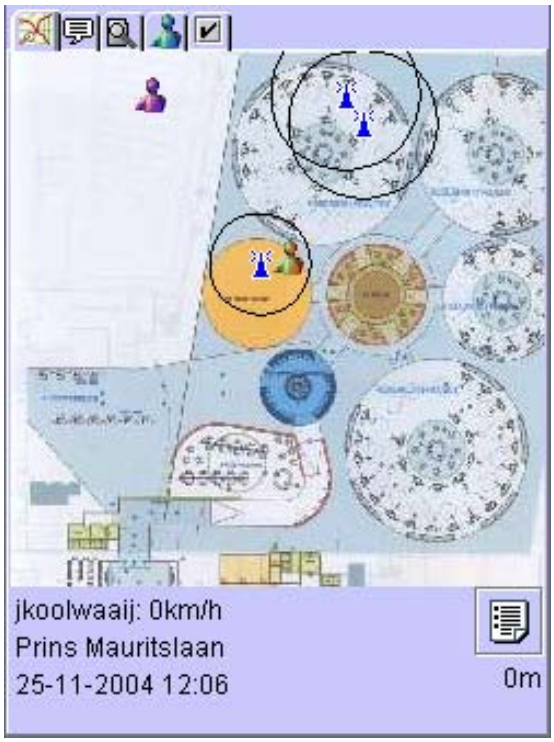

Figure 6. The initial situation: Ferdinand is outside the station IP-connected via GPRS; the location provider is the GPRS network-operator (based on cell-ID).

Figure 7. The situation after Ferdinand entered the train-station, coverage area of a public Wi-Fi hotspot. A seamless handover has taken place from GPRS to Wi-Fi, and based on that (network) context information the context reasoner has decided to switch from location provider as well.

6. CONCLUSIONS

In this paper, we described context-based reasoning – from principles and methods, needed grounding and representation means to existing tool, ontologies, and languages support. We analysed a commuter-assistant scenario with respect to the context provisioning and reasoning required and reported on our first validation findings. Our context-reasoning components are designed in a generic architecture for context processing entities, managed by a context provider. The different sources of context information, being various reasoners and wrappers, support intelligent processing of context information. Several location sources are wrapped as semantic web services, provider advertisements are realised and intelligent provider-reasoner constellation is implemented enabling optimal location provider selection allowing for further optimal location interpretation. This enables the first part of our scenario: notifying the commuter that he has missed his train. Our future work is dedicated to a further implementation of reasoning components that enable aligning commuting and meeting preference schemes. It includes an ambiguity resolving component, capable of solving multiple, possibly contradicting location objects.

ACKNOWLEDGEMENTS

The authors thank Martin Wibbels and Hans Zandbelt for their contributions.

REFERENCES

1 H.van Kranenburg, A.Salden, H.Eertink, R. Van Eijk, J. de Heer, Ubiquitous attentiveness – enabling context aware mobile applications and services, Proc. EUSAI 2003, LNCS 2875, Ambient Intelligence, pp. 76-87, 2003.

2 M. Benerecetti, P. Bouquet, and C. Ghidini, Contextual Reasoning Distilled, Journal of Theoretical and Experimental Artificial Intelligence, 12(3): 279--305, July 2000.

3 L. Serafini, P. Bouquet, Comparing formal theories of context in AI, v. 155, i. 1-2, p. 41 – 67, 2004.

4 A. H. Salden and J. de Heer, Natural Anticipation and Selection of Attention within Sustainable Intelligent Multimodal Systems by Collective Intelligent Agents, Proceedings SCI 2004, Orlando, Florida, USA.

5 H. Ailisto, P. Alahuhta, V. Haataja, V. Kyllönen, M. Lindholm, Structuring Context Aware Applications, Ubicomp workshop, Gothenburg, Sweden, 2002.

6 S. Harnad, The Symbol Grounding Problem, Physica D, 42, pp. 335-346.

7 J. Chai, S. Pan, M.X. Zhou, K. Houck, Context-based Multimodal Input Understanding in Conversational Systems, *Proc. ICMI'02* (Pittsburgh, Pennsylvania, 2002), IEEE Computer Society, 87-92.

8 D. Astuti, Context Inference, Seminar on Research Themes in Context aware Computing, Department of Computer Science, Un. of Helsinki, Finland, 2004.
9 D. Ashbrook, T. Starner, Learning Significant Locations and Predicting User Movement with GPS, 6th Int. Symposium on Wearable Computers, 2002.
10 T. Gruber. A translation approach to portable ontology specifications. Knowledge Acquisition, 5(2):199–220, April 1993.
11 V. Akman and M. Surav, Steps toward formalising context, AI Magazine, 1996, pp 55-72.
12 J. McCarthy, Notes on formalizing context, Proc. IJCAI'93, 1993.
13 F. Giunchiglia, contextual reasoning, Epistemologia, special issue on I Linguaggi e le Macchine, 1993, XVI:345-364.
14 Farquhar, A. Dappert, R. Fikes, W. Pratt, Integrating Information Sources Using Context Logic, Proc. of AAAI Spring Symp. on Information Gathering from Distributed Heterogeneous Environments, 1995.
15 J. Mylopoulos, R. Motschnig-Pitrik, Partitioning Information Bases with Contexts, Proc.3rd Int. Conf. on Cooperative Information Systems, 1995.
16 C. Ghidini and L. Serafini, Model Theoretic Semantics for Information Integration, Proc. AIMSA'98, volume 1480 of *LNAI*, 1998.
17 S. Staab et al. Handbook on Ontologies. Springer, 2004.
18 W3C. OWL Web Ontology Language Reference. http://www.w3.org/2004/OWL/
19 G.F. Luger and W.A. Stubblefield, Artificial Intelligence. Structures and strategies for complex problem solving. Addison Wesley Longman, 1998
20 http://www.sts.tu-harburg.de/~r.f.moeller/racer/
21 American National Standard for KIF, 1995, http://logic.stanford.edu/kif/specification.html
22 V.K. Chaudri, Open Knowledge Base Connectivity, version 2.0.3, Apr'98, http://www.ai.sri.com/~okbc/
23 Pangea systems, Artificial Intelligence center, XML-based Ontology exchange Language specs, vs0.4, Aug'99, http://www.ai.sri.com/pkarp/xol/xol.html
24 http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/
25 R Fikes, G. Frank, J. Jenkins, JTP – A System Architecture and Component Library for Hybrid Reasoning. Un. of Stanford, 2003. http://www.ksl.stanford.edu/software/JTP
26 Jess – the Rule Engine for the Java Platform. http://herzberg.ca.sandia.gov/jess/
27 R. Bartak, An online guide to Constraint Programming. http://kti.ms.mff.cuni.cz/ ~bartak/constraints/index.html
28 CHiP: Constraint Handling in Prolog. http://www.cosytec.com/production_scheduling/chip/optimization_product_chip.htm
29 M. Wagner, W. Kellerer, An Ontology Reasoning Use Case for Situation-aware Services, WWRF #11, Oslo, June 2004.
30 Jena, http://www.hpl.hp.com/semweb/jena2.htm, http://jena.sourceforge.net/ and http://www.hpl.hp.com/semweb/jena.htm
31 OWL API, http://owl.man.ac.uk/api.shtml
32 OWLP, http://www-db.research.bell-labs.com/user/pfps/owlp/
33 http://protege.stanford.edu/
34 MindSwap, SWOOP, http://www.mindswap.org/2004/SWOOP/
35 http://kaon.semanticweb.org/
36 H. Chen et al., An Intelligent Broker for Context aware Systems, Adjunct Proc. of Ubicomp 2003, http://cobra.umbc.edu/ontologies.html
37 H. Chen et al., SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications, Proc. Int. Conf. on Mobile and Ubiquitous Systems: Networking and Services, August 2004. http://pervasive.semanticweb.org/
38 M. Roman, C.K. Hess, R. Cerqueira, A. Ranganathan, R.H. Campbell, K Nahrstedt, GAIA: A Middleware Infrastructure to Enable Active Spaces, IEEE Pervasive Computing, vol. 1, no. 4, pp. 74-83, 2002.
39 A. Ranganathan, R.E. McGrath, R.H. Campbell, M.D. Mickunas, Ontologies in a Pervasive Computing Environment, Proc.18th Int. Joint Conf. on Artificial Intelligence, Workshop on Ontologies and Distributed Systems, 2003.
40 H. van Kranenburg, H. Eertink, Processing heterogeneous context sources, Proc. SAINT 2005, Next Generation IP-based Service Platforms for Future Mobile Systems workshop (Trento, Italy), ISBN 0-7695-2263-7 (IEEE), pp. 140-143, 2005.
41 B. Hulsebosch, A. Salden, M. Bargh. Context-based Service Access for Train Travelers, Proc. EUSAI 2004, Eindhoven, Netherlands.