An Approach to Dynamic Context Discovery and Composition

Graham Thomson, Matthew Richmond, Sotirios Terzis and Paddy Nixon

Department of Computer and Information Sciences
University of Strathclyde
Livingstone Tower
Glasgow, UK, G1 1XH
E-mail: *firstname.lastname*@cis.strath.ac.uk

Abstract. As a variety of pervasive environments emerge, context-aware applications shall have to interact well with each of them. In this position paper, we propose extensions to the Strathclyde Context Infrastructure that gives context-aware applications the potential to adapt to such environments transparently. We present a vision of a context discovery technique based on automated semantic reasoning about context information and services. The technique will offer higher levels of scalability and of interoperability with different context environments that cannot be achieved with current methods.

1 Introduction

The ability for applications to be context-aware has been identified as a key characteristic for pervasive computing [1]. As a variety of pervasive environments emerge, context-aware applications shall have to interact well with each of them. Infrastructure that supports spontaneous interoperation between these environments is the next major research challenge [1].

In this position paper, we propose extensions to the Strathclyde Context Infrastructure (SCI) [2] that gives context-aware applications the potential to adapt to these different environments transparently. We present a vision of a context discovery technique based on automated semantic reasoning about context information and services.

By treating the problem of discovering and composing appropriate context entities as a special case of the more general problem of discovering and composing components in software engineering, we adapt previous research in software reuse [3] to dynamic context discovery. Our approach aims to exceed current context discovery techniques inspired by ideas from distributed systems research such as naming, directory services and simple trading services.

In the next section, three current approaches to discovery are briefly described. Following this, an outline of the current design of the Strathclyde Context Infrastructure is given, before going on to present our extensions to the infrastructure to achieve semantic context trading.

2 Current context discovery approaches

In this section, we look at several approaches to context discovery employed by current context systems.

Firstly, we look at A. Dey's context model [4]. Here, five context abstractions are proposed: Widgets - which are software components that provide applications with context information from their operating environment, Interpreters - which transform context information by raising its level of abstraction, Aggregators - which gather logically related context information and make it available within a single software component, Services - which execute actions on behalf of applications and Discoverers - which maintain a registry of the currently available widgets, interpreters, aggregators and services.

Applications can use discoverers to find a particular component with a specific name, or a class of components that match a specific set of attributes and/or services. This is equivalent to standard white pages and yellow pages lookup and is implemented as a simple centralised service. As noted by Dey [4], this implementation could be replaced by similar services such as Universal Plug and Play [5] or Jini [6].

In the approach presented by Chen et al. [7] advertisements of, and queries for resources are based on name specifications. A name specification is a set of tag/value pairs and any of the values may be context-sensitive.

For example, a camera on Alice's mobile phone may register itself with the following name specification:

```
   [sensor="camera", class="colour", room=$alice-locator:room,
building="Sudikoff"]
```

which would be matched by a query specified as:

```
   [sensor="camera", room="25", building="Sudikoff"]
```

when Alice is present in room 25 of the Sudikoff building.

Interesting work has been reported by Ponnekanti and Fox [8]. Although not exclusive to, but applicable to context discovery, they describe an approach to service interoperability that does not depend on standardised service interfaces. Here, a service is searched for, matching on its syntactic interface description. If a match is found it is used, if not, a 'glue directory' is searched for existing interface adapters (or a transitive chain of adapters) that enable the requested service to be satisfied from the set of currently available services. The required service interface may be satisfied by a lossy adapter, where not all, but the required portion of the interface is satisfied, or by a composition of several services.

Our proposal differs from, and potentially significantly improves upon, existing work in context discovery by focusing on the use of typing mechanisms and semantic annotations to enhance the retrieval of relevant context bindings in new environments.

3 The Strathclyde Context Infrastructure

For a full description of the SCI, please refer to [2]. Here, only a brief description is given that is sufficient for understanding the concepts discussed in this paper.

The Strathclyde Context Infrastructure (SCI) is organised into two distinct layers. The upper layer of the infrastructure is a network overlay of partially connected nodes and is referred to as the SCINET. The lower layer of the infrastructure concerns the contents of each node, which consists of entities (People, Software, Places, Devices and Artefacts) responsible for producing, managing and using contextual information, and is referred to as a range.

The SCINET

The SCINET is concerned with managing interactions that take place between two or more ranges in order to provide appropriate contextual information. It is likely that entities that exist in one range may be interested in consuming contextual information from entities in other ranges.

The Structure of a Range

The main element of the SCI is the Range. A Range is a bounded physical or logical area.

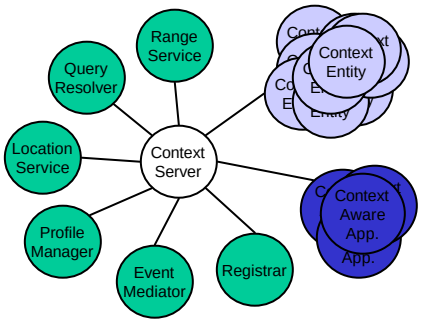

Fig. 1. The structure of a Range.

Each Range contains a single Context Server, which manages three types of components; Context Entities, Context Utilities and Context Aware Applications (see Fig. 1).

The Context Server (CS) is the most important component of a Range. It manages the other components and provides the means of communicating with other Ranges in a network.

A Context Entity (CE) is a lightweight software component for representing an entity within the infrastructure. A CE maintains a Profile for its entity that contains meta-data describing the entity. For entities that provide a service, the CE may also maintain an Advertisement describing the services that this entity can provide to other entities.

A Context Aware Application (CAA) is an application that has the ability to pull or be pushed contextual information to or from the infrastructure. A CAA communicates with the CS by way of a Query, which is represented as a short XML document.

The Context Utilities (CU) is a set of specialist services that help the CS in the management of a Range. These include a Range Service which is responsible for detecting the arrival into and departure of entities from a Range, and a Query Resolver which provides the means to take a high level query and decompose it into a useful configuration of Context Entities.

4 Semantically Enhanced Context Trading

For the SCI, we wish to have models for context that abstract away from the details of individual sensors and sensor types. This would allow contextualised services to utilise context information from a large variety of sensors of differing types [9]. To achieve this however, we must address the issues of how an application will locate the appropriate set of sensors in its operating environment, and how the application can compose different sensors to produce the required context information.

Current research into these issues is inspired by ideas from distributed systems research (see [10] for example) and focus on common basic mechanisms like naming, directory services or simple trading services. We wish to harness the power of semantic component trading [3] in tackling these issues in context-aware computing systems.

It is easy to imagine a context entity as a software component. In fact, we could consider the problem of discovering and composing the appropriate context entities to be a special case of the more general problem of discovering and composing components in software engineering [11]. This allows us to apply the semantic component trader concepts directly.

Currently in the SCI, context information is represented as a configuration of context entities (software representations of sensors). What we hope to achieve is to be able to express requests for context information at an abstract level, that is, in terms of basic context elements and context operators instead of concrete context entities and configurations. Basic context elements would be an abstraction of context entities, and context operators structures for composition of basic context elements to build higher-level contexts.

To facilitate queries of this form, we replace the context server of the range with a semantically enhanced context trader. The context trader performs similarly to current trading services in that it takes a request for context information and returns a list of possible configurations sorted by some user-specified properties. However, the context trader differs from current trading services in that context entities include a behavioural specification as part of their type description, and that matching of suitable entities is based on specification matching techniques, such as plug-compatibility [12] and behavioural subtyping [13].

As an example, a context-aware application may request nearby devices that can display PNG images with a configuration containing the basic context elements 'Location Service', 'Shortest Path Algorithm' and 'PNG Display'. The trader would determine that this configuration is equivalent to one containing 'Route Planner' and 'PNG Display', and search for context entities that can instantiate either configuration. One such instantiation could be a route planner entity that can be used directly along with a composition of a PNG to BMP converter and a digital picture frame that accepts BMP images.

The reader may be sceptical of this approach, thinking that the level of formal specification required may be very complex, and therefore too difficult. In our approach however, we aim to strike a balance between rigorous behavioural specifications and ease of use. This balance is achieved by utilising a rigorously specified domain vocabulary in the specification process. In every application domain there is a set of core terms that is widely used and their meaning is unambiguously clear within the domain. These terms form the domain vocabulary and its use in the context specification process means that we don't have to formally specify the whole domain and we can focus our formal specification only to the variations around the domain terminology. So, in order to exploit semantically enhanced context trading, we need to specify the domain vocabulary for sensor technology and define meaningful semantic relationships between its terms.

A natural choice for defining these vocabularies and relationships is to use ontologies. There are many ontological description languages that may be possible candidates for our system (see [14][15][16] for example), but we shall have to investigate the needs of our system first. If our ontological language is too powerful, there may be cases where equivalent relationships between basic context elements are not provable. Ontologies also allow us to easily link domain specific vocabularies together, such that a term could be known to be a more specific version of a term contained in another vocabulary.

We illustrate the envisioned high-level design of the context trader in Fig. 2. With context information requests formed in terms of basic context elements and context operators, the semantic context trader can search the relationship graph for compatible basic elements and operators, and form all equivalent context expressions. Then after examining the set of available context entities in the current range (using the context entity repository), the trader can determine which equivalent context expressions can be instantiated. These context entities

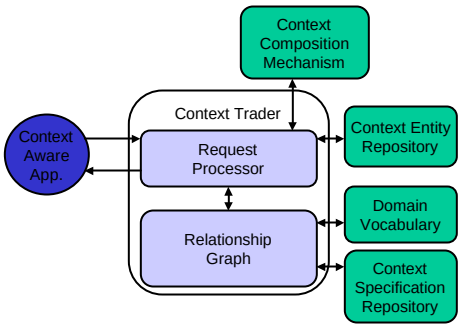

Fig. 2. The structure of a context trader.

and an accompanying composition strategy would then be passed to the context composition mechanism to construct the resulting set of configurations.

By introducing the semantically enhanced context trader, we aim to improve both precision and recall of the context selection process. Precision is improved as specification matching offers stronger assurances that the context information provided by the configurations that the trader returns is what we asked for. Recall is improved by a combination of context composition, and by being able to identify a greater set of compatible context entities through the rich set of semantic relations recorded by the trader.

We plan to experiment with a prototype implementation of the modified infrastructure. The result of which we hope will be the identification of an appropriate set of basic context elements that deliver the required context information, and a satisfactory set of context operators. Essentially, this would form a context *'lingua franca'* that does not constrain the development of context-aware applications while offering improved levels of interoperability and scalability that will be required for large-scale pervasive environments.

Application developers may wish to express context information in an application domain specific way that is most appropriate for them (and their customers) to use. In this scenario, we imagine that an administrator defines local composition and transformation strategies to translate the domain specific context requests down to an expression of basic elements and operators. When this domain specific request form is presented to the current context trader, it may know how to do the required transformation and satisfy the request locally, or it may act as a broker and forward it to the 'home' trader for processing.

Other 'local' compositions and transformations may include those specified by users themselves. For example, popular configurations for the current range may be suggested ahead of others, or the last used configuration for a particular user and application may be reselected upon re-entering the current range. As

the nomadic context-aware applications pass from range to range, this local information maybe carried with them, or ranges may transfer this information directly from range to range, giving way to evolutionary learning at each range.

5 Summary

We have presented a technique for dynamic context discovery based on a semantically enhanced trading mechanism.

This work is based on previous research in component trading in software reuse. We consider the problem of discovering and composing appropriate context entities to be a special case of the more general problem of discovering and composing components in software engineering.

Requests for context information are specified at an abstract level–in terms of basic context elements and context operators–to allow contextualised services to utilise context information from a large variety of sensors of differing types.

We include a behavioural specification as part of a basic context element's type description that is exploited to increase both the precision and recall of selected context configurations. This is achieved by examining a rich set of semantic relationships between the basic context elements to determine which of the available context entities can be composed to provide the required context information.

We utilise a rigorously specified domain vocabulary so that we focus our formal specification effort on the variations around the domain terminology.

The challenge will be to identify an appropriate set of basic context elements that deliver the required context information, a satisfactory set of context operators as well as a suitable sensor domain vocabulary that makes our system attractive to use and permit simple translations from application domain specific context requests to our basic context elements and operators form.

The relationships between our approach and current research into ontological description languages remains to be explored, as does providing suitable methods to facilitate unobtrusive inter-range knowledge sharing.

Through all the mechanisms mentioned above, we hope that the context trader shall allow fluid interoperation of context-aware applications between the variety of different context aware environments and separate application domains that shall be encountered in a large-scale pervasive environment, while placing the minimum of constraints on context-aware application developers and users.

6 Acknowledgements

This work is supported by the EU FP5 GLOSS Project (IST-2000-26070), in collaboration with Trinity College Dublin, Université Joseph Fourier, and the University of St Andrews, and through research student scholarships from the University of Strathclyde.

References

1. T. Kindberg and A. Fox. System software for ubiquitous computing. *IEEE Pervasive Computing*, 1(1), 2002.
2. Richard Glassey, Graeme Stevenson, Matthew Richmond, Feng Wang, Paddy Nixon, Sotirios Terzis, and Ian Ferguson. Towards a middleware for generalised context management. In *1st International Workshop on Middleware for Pervasive and Ad-Hoc Computing (MPAC03)*, June 2003.
3. S. Terzis and P. Nixon. Component location and the role of trading in large scale distributed systems. *Proceedings of the 5th International Symposium on Software Engineering for Parallel and Distributed Systems (PDSE 2000)*, 2000.
4. Daniel Salber Anind K. Dey and Gregory D. Abowd. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Anchor article of a special issue on context-aware computing in the Human-Computer Interaction (HCI) Journal*, 16(2-4), 2001.
5. U. Glässer and Y. Gurevich and M. Veanes. Universal Plug and Play Models. Technical Report MSR-TR-2001-59, Microsoft Research, June 2001.
6. Ken Arnold. The jini architecture: dynamic services in a flexible network. In *Proceedings of the 36th ACM/IEEE conference on Design automation conference*, pages 157–162. ACM Press, 1999.
7. Guanling Chen and David Kotz. Context-sensitive resource discovery. In *First IEEE International Conference on Pervasive Computing and Communications (PerCom'03)*, March 23 - 26, 2003.
8. Shankar R. Ponnekanti and Armando Fox. Application-service interoperation without standardized service interfaces. In *First IEEE International Conference on Pervasive Computing and Communications (PerCom'03)*, March 23 - 26, 2003.
9. Hans-W. Gellersen, Albrecht Schmidt, and Michael Beigl. Multi-sensor context-awareness in mobile devices and smart artefacts. Available at: citeseer.nj.nec.com/gellersen02multisensor.html.
10. A. Dey. *Providing Architectural Support for Building Context-Aware Application*. PhD thesis, Georgia Institute of Technology, Atlanta, 2000.
11. F. Mili H. Mili and A. Mili. Reusing software: Issues and research directions. *IEEE Transactions On Software Engineering*, 21(6), 1995.
12. Amy Moormann Zaremski and Jeannette M. Wing. Specification matching of software components. *ACM Transactions on Software Engineering and Methodology*, 6(4):333–369, October 1997.
13. B.H. Liskov and J.M. Wing. A behavioural notion of subtyping. *ACM Transactions on Programming and Systems*, 16(6), November 1991.
14. D. Fensel, F. van Harmelen, I. Horrocks, D. McGuinness, and P. Patel-Schneider. Oil: An ontology infrastructure for the semantic web. *IEEE Intelligent Systems*, 16(2):38–44, 2001.
15. Michael Smith, Chris Welty, and Deborah McGuiness. Owl web ontology language guide. http://www.w3.org/TR/2003/CR-owl-guide-20030818/, August 2003.
16. T. R. Gruber. Ontolingua: A mechanism to support portable ontologies. Technical report, Stanford University, November 1992.