

AN OPEN ARCHITECTURE FOR PERVASIVE SYSTEMS

J.Indulska^{#a}, S.W Loke[#], A.Rakotonirainy[#], V. Witana[#], A. Zaslavsky^{#b}

[#]*CRC for Distributed Systems Technology*

Level 7, General Purpose,

The University of Queensland, QLD 4072, Australia

^a*School of Computer Science and Electrical Engineering*

The University of Queensland, QLD 4072, Australia

^b*School of Computer Science and Software Engineering*

Monash University, Caulfield VIC 3145, Australia

jaga@csee.uq.edu.au, swloke@dstc.monash.edu.au, andry@dstc.edu.au,

varuni@dstc.uts.edu.au, A.Zaslavsky@csse.monash.edu.au

Abstract

Recent advances in mobile devices create a need for computing architectures and applications which are able to react to environmental changes in order to adapt to the changing context of computation. To date insufficient attention has been paid to the issues of defining an open component-based architecture which is able to describe complex computational context and handle different types of adaptation for a variety of new and existing pervasive enterprise applications. In this paper an architecture for pervasive enterprise systems is proposed. The architecture uses a component based modelling paradigm and an event-based mechanism which provides significant flexibility in dynamic system configuration and adaptation. The architecture includes context management which captures descriptions of complex user, device and application context including enterprise roles and role policies, and allows easy extension by new types of context. The architecture provides an open approach to adaptation which allows easy extension with adaptation mechanisms. In addition, the coordination language used to coordinate system events provides the flexibility needed in pervasive computing applications to support dynamic reconfiguration and a variety of communication paradigms.¹

¹The work reported in this paper has been funded in part by the Co-operative Research Centre Program through the Department of Industry, Science and Tourism of the Commonwealth Government of Australia.

1. INTRODUCTION

Pervasive (ubiquitous) systems are built to support a large number of heterogeneous and ubiquitous devices which utilise heterogeneous networks and computing environments. The ubiquitous system is a system in which components, in spite of this heterogeneity, can react to environmental changes caused by mobility of users and/or devices by adapting their behaviour, and can be dynamically reconfigured (added or removed dynamically). Ubiquitous systems need to offer a variety of services to a diverse spectrum of entities in a dynamic system in which services can appear or disappear, run-time environments can change, failures may increase, and user roles and objectives may evolve (i.e. the context of user applications may change).

A major factor in pervasive systems is their ability to adapt to the context changes. Many adaptation problems (both application adaptation and environment adaptation, e.g. middleware adaptation) have been already addressed by existing research and industry approaches. There is, however, no architecture that is open, flexible and evolvable, and therefore allowing to use a variety of concepts in one architectural framework including (i) rich context description, (ii) generic approach to decisions about adaptability methods, (iii) dynamic definition of communication paradigms, (iv) dynamic reconfiguration of the architecture components, (v) incorporation of enterprise aspects like roles of participants and policies governing roles in the dynamic context management (the dynamic change of roles and policies needs to be recognised by the ubiquitous system in order to deliver appropriate services to the user. Most of the existing pervasive systems have been influenced by a specific type of adaptation or by the type of underlying technology, thereby resulting in a lack of generality (see Section 2).

In this paper we describe the m3 architecture and its Run-Time Environment (m3-RTE). The architecture supports adaptation and dynamic reconfiguration of the system. The proposed approach separates clearly the computation aspect from the component coordination aspect. These two aspects are often merged into a monolithic framework preventing flexibility and extensibility.

The structure of this paper is as follows. Section 2 characterises related work. Section 3 describes the design requirements and modelling concepts used in the proposed architecture. Section 4 characterises the architecture and describes the functionality of its main components and their interfaces. Section 5 discusses several issues related to the architec-

ture prototype including coordination of events, characterises the current status of the prototype and provides examples of applications adapted by the prototype architecture. Section 6 concludes the paper.

2. RELATED WORK

In Sumatra [1], adaptation refers to resource awareness, ability to react and privilege to use resources. Sumatra provides resource-aware mobile code that places computation and data in response to changes in the environment. Rover [14] combines architecture and language to ensure reliable operations and is similar to Sumatra in the sense that it allows loading of an object in the client to reduce traffic. It also offers non-blocking RPC for disconnected clients. Bayou [8] takes a different adaptability approach by exposing potentially stale data to the application when the network bandwidth is poor or down. Bayou is similar to Coda [21] in that respect. Coda is a distributed file system that supports disconnected operation for mobile computing.

Mobiware [3] and Odyssey [17] are more focused on adaptability measurement. Mobiware defines an adaptation technique for bandwidth changes based on a utility-fair curve. Odyssey's approach to adaptation is best characterised as application-aware adaptation. It is based on control system theory where output measurements are observed from the input reference waveform variation. The information about the change in availability of resources is delivered to the application which should adapt to the change.

PIMA, Aura, Limbo and Ensemble address issues about architecture as m3 does. Limbo [7] and Ensemble [19] are concerned with the mobile architecture and the programming model. Limbo offers an asynchronous programming model (tuple space) and defines an architecture for reporting and propagating QoS information that might be related to the system. Adaptability is achieved by filtering agents. Ensemble's architecture consists of a set of protocol stacks which are micro-protocol modules. Modules are stacked and re-stacked in a variety of ways to meet application demands. Adaptation is done transparently to the application. The lowest stack tries to adapt first. If it cannot, then it passes the notification to the next upper layer. This repeats until, eventually, the application itself has to reconfigure.

PIMA [2] is a framework that focuses on providing a device-independent model for pervasive applications that facilitates application design and deployment. It provides functionalities such as an application adaptation enabler, and dynamic application apportioning to services.

Aura [24] is a task-driven framework that helps the user to gather information about available services, selecting suitable services to carry out a tasks and binding them together. A Service Coordination Protocol controls adaptability mechanisms required when the environment changes.

Open-ORB [4] and dynamicTao [15] use reflection mechanisms to address dynamic configuration and adaptation of middleware. The two approaches make use of reflective CORBA ORB that gives program access to meta information that allow to inspect, evaluate and alter the current functionality and configuration of components.

As can be seen, most the above work addresses specific types of adaptability for networks, operating systems or middleware and hence caters for only specific kinds of contextual information. The cited works do not provide an open, evolvable architecture enabling the use of a broad range of concepts related to adaptation of enterprise applications. Furthermore, dynamic coordination of heterogeneous components, dynamic definition of communication paradigms and context awareness are not addressed in the existing solutions.

3. BASIC CONCEPTS

The design of the m3 architecture was driven by the following requirements:

(i) **Reactivity**: the architecture should allow easy programming of reactions to context changes. The changing context of a component determines its interactions with other components and the service it can offer. (ii) **Open dynamic composition**: the system must support inter-operation with open components such as legacy applications or new components. New components may exhibit new types of interactions which go beyond the currently common but very static and asymmetric Remote Procedure Call (RPC). Dynamic composition (plug and play) configuration and customisation of services is essential in a pervasive environment as lack of resources to carry out a task may require a partial/complete reconfiguration of services. (iii) **Enterprise focus**: As enterprise applications are complex and their complexity increases if they operate in pervasive environments, we need abstractions that capture the purpose, scope and policies of the system in terms of behaviour expected of the system by other entities within the enterprise.

To meet the above requirements the m3 architecture utilises three modelling concepts: components, their roles and the events to which components can react.

Event is an atomic modelling concept from which interaction protocols such as message passing, RPC, streams and other communication paradigms can be expressed. We assume that interactions between all components are event based.

In object-oriented terminology, a component is a set of interacting objects (or even a single object). Components can be composed of other components. Components describe their relationship with other components by means of interfaces. The interface of a component has a unique ID, role permissions, pre/post conditions and event type. An interface is a subset of the event interactions of a component.

m3 focuses on enterprise specifications which define the purpose, scope and policies for a system in terms of roles played, activities undertaken, and policy statements about the system. An enterprise role is an identifier for a behaviour (e.g. Director). It focuses on the position and responsibilities of a component within an overall system to carry out a task. A role is a placeholder that can be fulfilled by one or more interfaces. Roles provide abstractions required for enterprise modelling and allow dynamic addition/removal of roles during the lifetime of a component.

4. M3 ARCHITECTURE

The m3 architecture is organised into three layers as shown in Figure 1 and characterised in the following subsections: *(i)* **Coordination layer**: a coordination manager that executes a coordination script written in MEADL (Mobile Enterprise Architecture Description Language). *(ii)* **Dedicated manager** layer consisting of three fundamental parts of pervasive computing environments. They are the Context, Adaptation and Policy managers. *(iii)* **Distributed Service layer**: existing services such as notification, security and network protocols.

4.1. COORDINATION LAYER

The need for increased programmer productivity, rapid development for dynamic changes within complex systems is the main motivation that led to the development of coordination languages and models. Coordination-based methods provide a clean separation between individual software components and their interaction within the overall software organisation. This separation makes large applications more tractable, more configurable, supports global analysis, and enhances reuse of software components.

There exist Architecture Description Languages (ADL) which are modelling notations to support architecture-based development [16]. Their

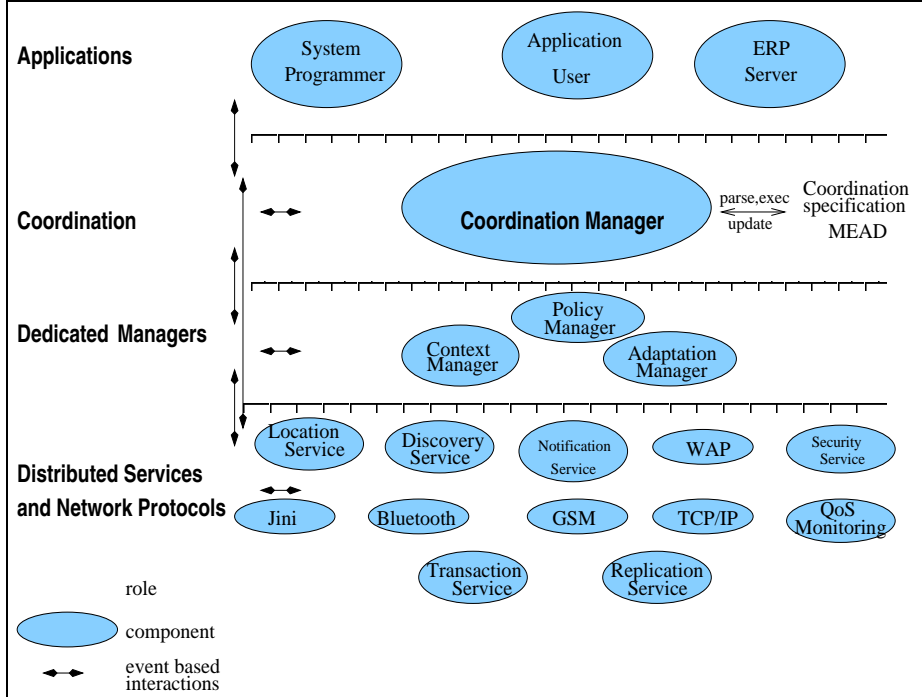


Figure 1 m3 Architecture

scalability, in a large scale system and pervasive environments, are yet to be proven [20]. In addition, these languages do not cater for dynamic change of communication paradigms between components.

The coordination layer is the core of the m3 architecture. The coordination focuses on concepts necessary to manage the implications of mobility for enterprise-wide activities. The coordination consists of two parts (*i*) specifying with MEADL the interaction between enterprise roles and (*ii*) executing MEADL scripts by a coordination manager.

Coordination specification with MEADL. MEADL specifies the coordination of events. It is a rule-based coordination script that separates the behaviour's specification (functionalities) of components from the communication between components. MEADL specifications provide means to dynamically configure interactions between components, to change dynamically the communication paradigm and to provide session management. We take a pragmatic (not based on formal description techniques) approach to specify coordination. The script offers two main advantages: (*i*) interaction between components can be changed

dynamically, and (ii) coordinated components can be added or removed dynamically.

Events belong to roles, and therefore, MEADL specifies the coordination of roles. Roles can be fulfilled by users, ERP (Enterprise Resource Planing) servers, network protocols or dedicated managers. A role's description is the interface of tasks that a component can provide. A role's duty is described in a set of input/output events.

The m3 architecture is a reactive architecture and this is reflected in the use of reaction rule syntax `on event` as first class construct of MEADL. `on event $R_j : E_i$` waits for the occurrence of a set of events E_i from a defined set of roles R_j . The value of E_i parameters is checked with a `WHERE` clause. The relevant context is also checked with `in context` or `out context` clauses.

Code 4.1 MEADL EBNF syntax

```

rule      : "on" pexpr [guard] "{" actions "}"
pexpr     : "event" role "(" [ident(","ident)*] | "(" pexpr ")"
           | pexpr op pexpr
guard     : "not"* ( "in" | "out" ) "context" string
actions   : [action(";"action)*]
role      : ident ":" ident
op        : "where" | "and" | "or" | "eq" | "neq"
action    : ( "call" | "emit" ) ident "(" [expr (","expr)*] ")"

```

The MEADL syntax is summarised in Code 4.1. An example of a MEADL sentence is shown in Code 4.2. Code 4.2 specifies a rule that waits for the occurrence of event `http:get(balance, account)` from the role `Director`. If the current context is secure and the balance is more than 10,000 then it emits a new event featuring the new balance then calls a function `foo`.

Code 4.2 Example of MEADL specification

```

ON EVENT Director:http:get(balance, account)
  WHERE (balance > 10,000)
  IN CONTEXT 'secure_environment'
{
  EMIT employee_chat_line('DSTC has',balance);
  CALL foo(account, 23);
}

```

To allow rules to affect other rules (a form of reflection) and to allow dynamic change of the above reaction rule, the body of the reaction rule can use the following methods: (*I*) create a reaction rule, (*ii*) delete a reaction rule, (*iii*) map an event to a set of operations, (*iv*) inhibit a reaction rule, (*v*) reactivate an inhibited reaction rule.

Coordination manager. The coordination manager is an engine that executes a MEADL script. MEADL scripts are transformed into XML [23]. Then, the coordination manager coordinates the events based on the generated XML specification. The coordination manager coordinates components by sending/receiving events. Using XML is a provision for the future use of different coordination mechanism (e.g. workflow engine) and language independent coordination.

4.2. DEDICATED MANAGERS LAYER

The middle layer of the architecture is composed of three dedicated managers which provide the three major functionalities. The three managers are context, adaptation and policy managers. The Context Manager (CM) provides awareness of the environment and feeds the Adaptation Manager(AM) with context information in order to enable selection of an appropriate adaptation method. The Policy Manager(PM) ensures that the overall goals of the roles involved are maintained despite adaptation and that a change of roles is followed by appropriate adaptation. The use of these managers is invisible for the applications running in m3-RTE. They are autonomous components with well defined interfaces that interact together to provide the appropriate level of services to applications. The coordination manager has an impact on the way the three dedicated managers interact.

Context Manager (CM). Context management in pervasive systems must contain at least three separate but complementary functionalities. They are: (*i*) context sensing (gathering contextual information) from variety of hardware and software sensors, (*ii*) interpretation of context information gathered from various sensors (may involve complex processing and resolution of conflicting context information), (*iii*) context description represented in a generic way.

Context sensing and interpretation depends on the type of context and is reasonably well understood for various types of context information. In our architecture we, therefore, concentrate on the final context description which is used to evaluate context changes in order to make adaptation decisions. The CM is responsible for gathering context descriptions and making it available to the other entities. The CM enables

consistent addition, deletion and modification of the context (value, attribute and relationship between attributes) and the detection of context changes. The CM represents context using RDF (Resource Description Framework) graphs. RDF enables the definition of schemata which describe the gathered context as well as allowing the use of relationship operators between attributes. The use of RDF enables us to use the work of the W3C CC/PP [5] WG which is defining a vocabulary for describing device capabilities as well as protocols to transmit these device profiles from the client. We are extending the CC/PP vocabulary to include user and application context.

Adaptation Manager (AM). As the mobile environment is dynamic, devices may switch from one kind of network to another, users may move causing changes in terms of communication, security, device capabilities and reliability (context changes). The role of AM is to make decisions about adaptability if context changes and to install/invoke appropriate adaptability methods. The AM is able to incorporate various types of adaptation.

The AM concept is based on **Event Condition Action (ECA)** rules. **Event** is an event that might trigger the **Action**. **Conditions** are related to the event and **Action** is a reference to an adaptation action. Such rules support multimedia (continuous) and operational (discrete) adaptation and decouple the rules from the actual implementation of the adaptation. This approach also supports the specification of application aware adaptation [17] and application transparent adaptation [21, 3] by respectively giving the control and implementation of the (A)ction of ECA rule to the application or to the environment.

Policy Manager (PM). The Reference Model for Open Distributed Processing (RM-ODP) Enterprise Language [13] comprises concepts, rules and structures for the specification of a community, where a community comprises objects (human, software, or combinations thereof) formed to meet an objective. The Enterprise Language focuses on the purpose, scope and policies of the community, where an enterprise policy is a set of rules with a particular purpose related to the community's purpose. We see such an approach as providing useful concepts for high-level modelling of a distributed system in general and pervasive systems in particular. The goals of roles can change due to a lack of resources required to achieve a task, for example. This implies that policies can be altered and changed dynamically depending on the context. The policy manager ensures that policy specifications described as *Obligation, Prohibition and Permission* associated with enterprise roles are respected

in pervasive environment. The policy manager enables consistent addition, deletion and modification of policies. The Policy Manager is also responsible for maintaining consistency and resolving conflicts between role's objectives.

4.3. SERVICE LAYER

A dynamic and open environment is characterised by the frequent appearance of new services and communication protocols. A unified mechanism is required to access services and communication protocols. This layer wraps services and protocols to fit into the m3 modelling requirements. The wrapping allows upper layers to interact with the service layer using the event-based paradigm. This layer provides: *(i)* at least the notification and discovery services. All the services fulfill a role, accessed as components using event notification service. The discovery service communicates via the notification service to the rest of the m3-RTE environment. *(ii)* Universal interface to the upper layers that enables the use of communication protocols such as SMTP, HTTP, TCP/IP or security protocol transparently. The current protocol independent interface is similar to the Linda tuple space model [12] and evolves towards the SOAP protocol [22] to achieve greater openness.

5. ARCHITECTURE PROTOTYPE

Several of the described concepts have been already tested in the prototype m3-RTE environment. The goal of the m3-RTE prototype is to create a test bed run-time environment for the current and future concepts in pervasive computing. This section discusses the event notification service used in the prototype coordination manager, prototypes of main architecture components (context, adaptability and policy managers) and examples of application adaptations.

5.1. M3 MANAGERS

Coordination Manager. Event notification is the building block of m3-RTE. Messages are sent to an intermediate system to be forwarded to a set of recipients. The benefits are reduced dependencies between components and as a result greater flexibility. We use the DSTC Elvin [6] notification system. Elvin messages are routed to one or more locations, based entirely on the content of each message. Elvin uses a client-server architecture for delivering notifications. Clients establish sessions with an Elvin server process and are then able to send notifications for delivery, or to register to receive notifications sent by other components. Clients can act, anonymously, as both producers and con-

sumers of information within the same session. The Elvin subscription language allows to express complex regular expressions

In m3-RTE, the coordination manager executes MEADL specifications by transforming them into XML which in turn is parsed and transformed into Elvin subscriptions using Python [18]. Elvin's data is encoded in SOAP 1.1 [22] to provide for the use of different communication paradigms in the future.

Context Manager. Currently, the context manager is able to manage device descriptions in an RDF (Resource Description Framework) graph. In addition, our previous work on both application context (application requirements and its current state) and user context are currently being incorporated into the m3 context manager [9]. Context information also includes location information for users and devices.

Adaptation Manager. Several adaptability methods have been prototyped including insertion of variety of filters, migration of objects from resource poor sites to the static network and sending the result of the operation back to the object's client, restriction of an object interface (e.g. allowing access to subset of operations only) [10], adaptation of Web content [11].

Policy Manager. The Policy Manager extends context information by adding descriptions of roles and policies associated with roles. Our Policy Manager prototype is in an early stage of development and is currently able to manage simple policies such as Permission and Prohibition. We use concepts from MEADL to specify policy rules.

5.2. APPLICATION EXAMPLE

One of the applications adapted by m3-RTE is the Tickertape application. This application was used to test the architecture support for component coordination, management of a variety of context descriptions and a variety of adaptations methods.

Tickertape is an application that gives visual scrolling display of Elvin (event) traffic. It displays notifications to which the user subscribed. In the prototype we use it for chat-style messages.

We use m3-RTE to adapt the tickertape user interface rendering and the communication protocol according to the device capabilities and user preferences provided by the CM. The CM has an abstract description of the class of devices such as Palm, Solaris Ultra10, Nokia7110 that contains their capabilities (e.g. WAP 1.0, WAP 1.2 [25], touch-screen, Netscape ...). The AM has the adaptation rules that select a required component to exchange Elvin messages with the appropriate device (e.g. selection of proxies, gateways). The MEADL script executed by the

Coordination Manager forwards a tickertape connection request to the AM. The list of subscribed channels for roles is known by the CM from the user profile. The content of Elvin messages to be forwarded and the communication protocols to be used are chosen by the AM. As a result Tickertape was adapted to several different device types.

Another application built to operate in m3-RTE is an adaptive email client/server application [9]. The application is built as a set of cooperating components each implementing a particular functionality of the e-mail service. The application can be dynamically adapted to context changes in three different ways:

(*i*) a variety of filters (attachment compression, reduction of image attachments, format conversion) can be inserted between the email client and the email server, (*ii*) interfaces available to the client can be restricted (e.g. only short messages can be read), (*iii*) some components of the email client (e.g. search the user's mailbox for a specific pattern) can be migrated from the client device if there is not enough processing power in the device.

The openness of the m3 architecture is demonstrated by the fact that we only added less than ten lines of code into the MEADL and AM to plug a new component that enables tickertape viewing on different devices. m3-RTE also demonstrates (*i*) a universal approach to adaptation which can support a large set of adaptation mechanisms, (*ii*) dynamic definition of event coordination between the context manager and the adaptability manager, and (*iii*) component based applications.

6. CONCLUSION

This paper describes an open architecture used as a testbed to build adaptable enterprise applications for pervasive environments. It leverages heterogeneous devices, existing mobility related standards and protocols. We have identified context, adaptation and policy management as fundamental elements of pervasive computing applications, and shown how our architecture integrates these three notions. The use of a role-based component model and MEADL coordination gives the m3 architecture the ability to provide dynamic plug and play, and yet effectively coordinate enterprise components. The use of notification service as the main communication paradigm within the m3 architecture enables a loose form of integration of components (or roles), permitting architectural reconfigurability as reactions to the ever-changing contexts of applications or occurrences of (a)synchronous events in a pervasive environment. Work is being carried out on both an extension to the pro-

prototype and on further adaptive applications in order to fully test the concepts introduced by the m3 architecture.

References

- [1] Acharya, A. Ranganathan, M., Saltz, J. "A language for Resource-Aware Mobile Programs" *Mobile Object Systems: Towards the Programmable Internet, pages 111-130. Springer-Verlag, April 1997. Lecture Notes in Computer Science No. 1222.*
- [2] Banavar, G., Beck, J., Gluzberg, E., E., Munson, J., Sussman, J. and Zkowski, D. "Challenges: An application Model for Pervasive Computing" *6th Proc annual Intl. Conference on Mobile Computing and Networking MOBICOM 2000, Boston August 2000*
- [3] Bianchi, G., Campbell, A.T, Liao, R. "On Utility-Fair Adaptive Services in Wireless Networks" *Proc of the 6th Intl Workshop on QoS IEEE/IFIP IWQOS'98 Napa Valley CA, May 1998*
- [4] Blair, G., Blair, L., Issarny, V., Tuma, P., Zarras, A., The Role of Software Architecture in Constraining Adaptation in Component-Based Middleware Platforms. *Middleware 2000 Proc LNCS 1795 - IFIP/ACM NY, USA, April 2000*
- [5] Composite Capabilities/Preference Profiles CC/PP - W3C - <http://www.w3.org/Mobile/CCPP/>
- [6] Arnold, D., Segall, B., Boot, J., Bond, A., Lloyd, M. and Kaplan, S. Discourse with Disposable Computers: How and why you will talk to your tomatoes, Usenix Workshop on Embedded Systems (ES99), Cambridge Massachusetts, March 1999 also <http://elvin.dstc.edu.au/>
- [7] Davies, N., Friday, A., Wade, S. and Blair, G. "A Distributed Systems Platform for Mobile Computing" *ACM Mobile Networks and Applications (MONET), Special Issue on Protocols and Software Paradigms of Mobile Networks, Volume 3, Number 2, August 1998, pp143-156*
- [8] Demers, A., Petersen, K., Spreitzer, M., Terry, D., Theimer, M., Welch, B. "The Bayou Architecture: Support for Data Sharing among Mobile Users" *Proceedings of the Workshop on Mobile Computing Systems and Applications, Santa Cruz, California, December 1994, pages 2-7.*
- [9] Indulska J., Bond A., Gallagher M., "Support for Mobile Computing in Open Distributed Systems", *Proc. of the IEEE Region Ten Conference, Delhi, India, December 1998.*
- [10] Gallagher, M "A nomadic Computing Architecture for Open Distributed Computing", Ph.D Thesis - University of Queensland, Submitted September 2000.

- [11] Henricksen, K. and Indulska. J., "Adapting the Web Interface: An Adaptive Web Browser", Proceedings Australasian User Interface Conference 2001, *Australian Computer Science Communications, Volume 23, Number 5, 2001.*
- [12] Wyckoff, P., McLaughry, S. W., Lehman, T. J. and Ford, D. A. "TSpaces", *IBM Systems Journal, August 1998* also <http://www.almaden.ibm.com/cs/TSpaces/>
- [13] Information Technology - Open Distributed Processing - Reference Model - Enterprise Language (ISO/IEC 15414 — ITU-T Recommendation X.911) July 1999
- [14] Joseph A., Kaashoek F. "Building reliable mobile-aware applications using the Rover toolkit" *MOBICOM '96. Proceedings of the second annual international conference on Mobile computing and networking, pages 117-129'*
- [15] Kon, F. et al Monitoring, Security, and Dynamic Configuration with dynamicTAO Reflective ORB *Middleware 2000 Proc LNCS 1795 - IFIP/ACM NY, USA, April 2000*
- [16] Medvidovic N, Taylor "A Framework for Classifying and Comparing Architecture Description Language" *Proc Software engineering Notes, ESEC/FSE'96 - LNCS Vol 22 numer 6 November 1997*
- [17] Noble, B., Satyanarayanan, M., Narayanan, D. Filton J.E, Flinn J., Walker K. , "Agile Application Aware Adaptation for Mobility" *16th ACM Symposium on Operating System Principles 1997*
- [18] Python programming Language <http://www.python.org>
- [19] Renesse, v-R. Birman, K., Hayden, M., Vaysburd, A., Karr, D. "Building Adaptive systems using Ensemble" *Cornell University Technical Report, TR97-1638, July 1997.*
- [20] Rakotonirainy A., Bond A., Indulska, J., Leonard, D. SCAF: A simple Component Architecture Framework. *Technology of Object-Oriented Languages and systems TOOLS 33 - June 2000 - IEEE Computer Society - Mont St Michel France*
- [21] Satyanarayanan, M. The Coda Distributed File System *Braam, P. J. Linux Journal, 50 June 1998*
- [22] Simple Object Access Protocol (SOAP) 1.1 <http://www.w3.org/TR/SOAP/>
- [23] Extensible Markup Language (XML) 1.0 <http://www.w3.org/XML/>
- [24] Want, Z. and Garlan D., "Task-Driven Computing". *Technical Report, CMU-CS-00-154, School of Computer Science CMU May 2000*
- [25] Wireless Application Protocol - WAP Forum Specifications <http://www.wapforum.com/what/technical.htm>