Final report

Contract No. F61775-00-WE044

12th December 2001

Authors:
Prof. P. Nixon

Contact details:
Department of Computer and Information Sciences
The University of Strathclyde
Glasgow G1 1XH
Scotland
Email: Paddy.Nixon@cis.strath.ac.uk

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)* 14-03-2002	**2. REPORT TYPE** Final Report	**3. DATES COVERED** *(From – To)* 29 September 2000 - 29-Sep-01

4. TITLE AND SUBTITLE

An Investigation into Component Reconfigurability Management for Dependable Ad-Hoc Networking

5a. CONTRACT NUMBER F61775-00-WE044

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Professor Paddy Nixon

5d. PROJECT NUMBER

5d. TASK NUMBER

5e. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of Strathclyde
Livingstone Tower
26 Richmond Street
Glasgow G1 1XQ
United Kingdom

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

EOARD
PSC 802 BOX 14
FPO 09499-0014

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
SPC 00-4044

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This report results from a contract tasking University of Strathclyde to investigate requirements for robust reliable application communications in ad-hoc distributed mobile computer networks. Specifically, the contractor will investigate code mobility in ad-hoc collaborations and apply his insights into dynamic fault tolerant binding to the management of reliability. To implement and evaluate these issues, a heterogeneous, mobile, reconfigurable computer network will be assembled. This network will provide the required computational and physical mobility and will consist of hand-held computers and more powerful development laptop computers supported by a wireless Ethernet system (based on Lucent's WaveLan technology). Details of the deliverable are described below.

15. SUBJECT TERMS
EOARD, Distributed Computing, Fault Tolerant Computing, FPGA

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18, NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT UNCLAS	**b. ABSTRACT** UNCLAS	**c. THIS PAGE** UNCLAS	UL	22	Christopher Reuter, Ph. D.
					19b. TELEPHONE NUMBER *(Include area code)* +44 (0)20 7514 4474

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39-18

(1) In accordance with Defense Federal Acquisition Regulation 252.227-7036, Declaration of Technical Data Conformity (Jan 1997), All technical data delivered under this contract shall be accompanied by the following written declaration:

"The Contractor, University of Strathclyde, hereby declares that, to the best of its knowledge and belief, the technical data delivered herewith under Contract No. F61775-00-WE044 is complete, accurate, and complies with all requirements of the contract."

DATE: ______________________

Name and Title of Authorized Official:

____
(End of Clause)

(2) In accordance with the requirements in Federal Acquisition Regulation 52.227-13, Patent Rights-Acquisition by the U.S. Government (Jun 1989), CONTRACTOR WILL INCLUDE IN THE FINAL REPORT ONE OF THE FOLLOWING STATEMENTS:

~~(A) "Disclosures of all subject inventions as defined in FAR 52.227-13 have been reported in accordance with this clause."~~

~~Or,~~

(B) "I certify that there were no subject inventions to declare as defined in FAR 52.227-13, during the performance of this contract."

DATE: _______________________

Name and Title of Authorized Official: ___________________________________

In this report we detail the key architectural aspects for supporting ad hoc mobile software systems. We describe a computational model of context necessary for building adaptable systems, developed jointly between ourselves and some colleagues in the GLOSS project [http://www.gloss.cs.strath.ac.uk]. We then describe the implementation of a mobile version of this model that supports ad hoc interactions.

1. Introduction

In Anind Dey's definition, context is related to interaction between user and the application. Researchers take a user-centered approach in human computer interaction research. Task plays an important role in human computer interaction. From the interaction perspective, Joëlle Coutaz and Gaetan Rey at Université Joseph Fourier France have proposed a semi formal definition of context [Dey2001]. In their definition, user and task are first class entities and states are capitalized over time.

Given a set of users, U, a task, T, and two instants of observation, t0 and t, where t0 is the temporal reference for observations, the Context at t that relates to U for performing T, is the composition of the Situations observed between t0 and t that relate to U for performing T.

context ^{U,T} *(t) = COMPOSITION (situation* ^{U,T} *(t0), … , situation* ^{U,T} *(t))*
where situation ^{U,T} *(t) is the Situation at t that relates to U for performing T.*

The Situation, situation ^{U,T} *(t), is the set of the values observed at t of the peripheral state variables that relate to U for performing T, as well as their relations.*

Peripheral state variables denote the entities that are not central to U at t for performing T, but that may have an impact on T, now (i.e., at t) and/or in the future(i.e., at t+dt).

1.1 Computational Model of Context

From this semi formal definition, we introduce a computational model of context. The basic unit is contextor. A contextor models the relations between the peripheral variables of an observed context. For example Essentially a contextor gets input data, processes it then output some data. From this point of view, this is the same as widget in Context Toolkit. The difference is that they add something to deal with inaccurate sensors and faulty sensors. In their model, they add meta-data in to describe the quality of input data and add meta-data out to describe the quality of output data. For inaccurate sensor data, some researchers already included quality attributes in the sensor data in their projects. Daniel Salber has introduced similar ideas in a model but used a separate channel for sending meta-data rather than send data and meta-data together. The control in and control out are used to switch off faulty contextors. Other contextor can send control command to the control in channel of a contextor to switch it off. A contextor can send control command through its control out channel to switch off other contextors.

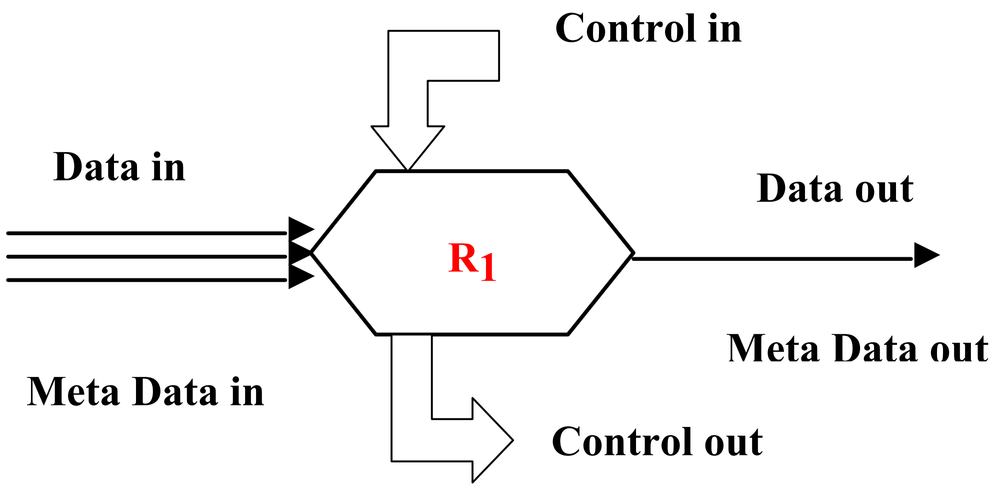


Figure 1. A graphical representation of a contextor

We also propose two methods to compose contextors to get other contexts: encapsulation and data channels connection. If data out channel of one context is of the same type of the data in channel of another contextor, then they can be connect together to get more advanced function. Encapsulation is also used to group contextors, but the purpose of grouping is to create a new contextor. From outside, the new contextor looks like a standard contextor, but hides the details of internal composition.

They divided contextors into six categories: elementary, history, threshold, translation, fusion and abstraction.

An **elementary contextor** encapsulates physical sensors and has no data in channel.

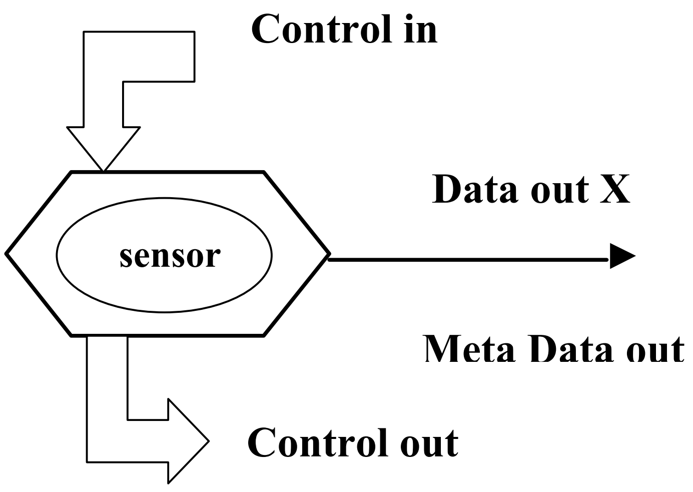


Figure 2. The elementary contextor

A **history contextor** saves the data and meta-in data that it receives from the input channel. It implements context data storage function.

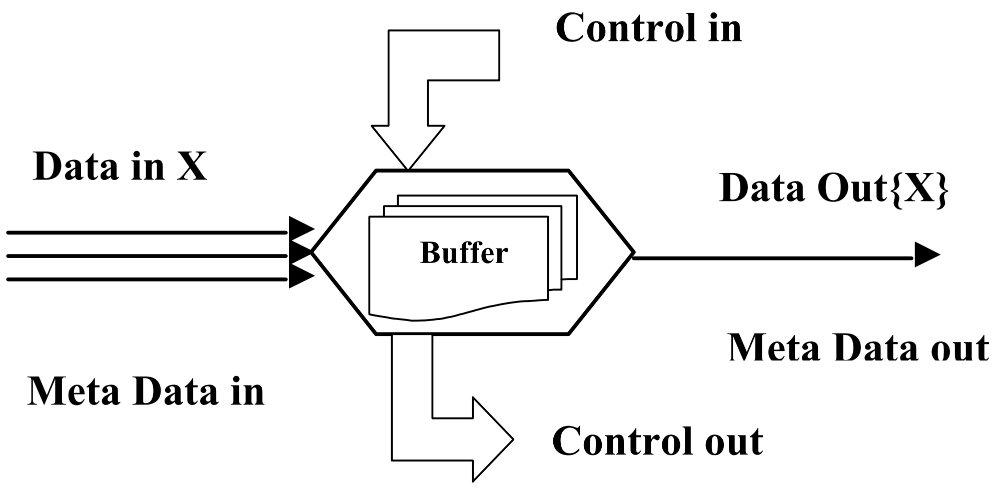


Figure 3. History contextor

A **threshold contextor** returns true if the data-in value satisfies a threshold condition, false otherwise.

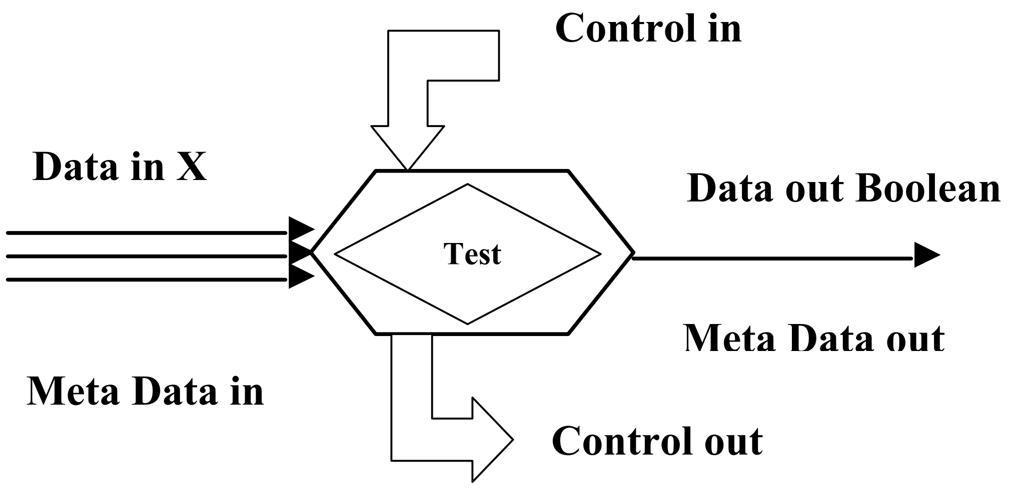


Figure 4. The Threshold contextor

A **translation contextor** performs type recasting but does not change the meaning, nor the level of abstraction of the values received on the data-in channel.

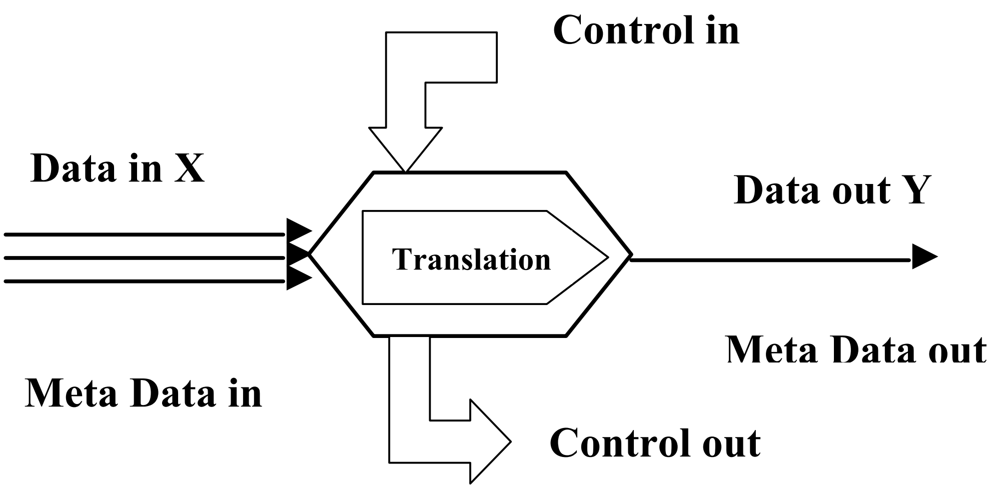


Figure 5. The Translation contextor

A **fusion contextor** has multiple data-in of the same type, each one with its own meta-data. The role of the fusion contextor is to produce a single data-out of the same type whose quality has been improved over that of the input data.

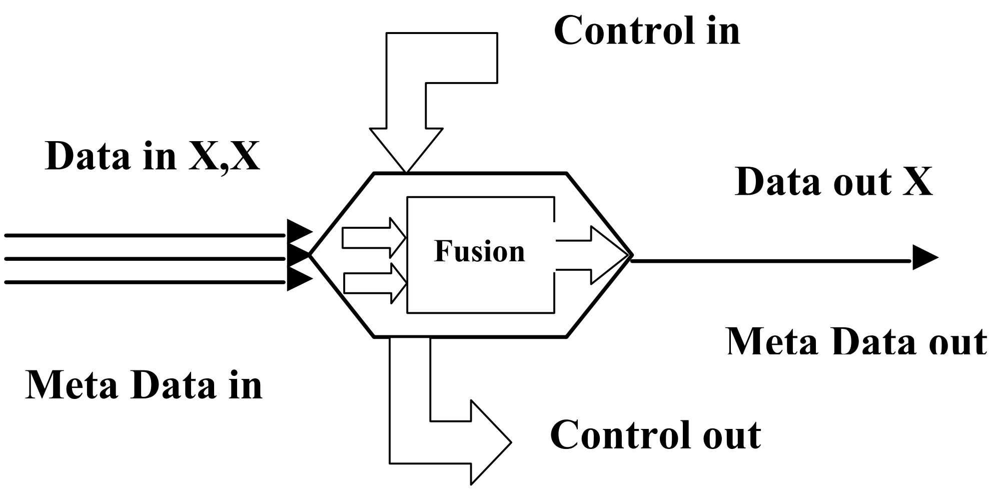


Figure 6. The fusion contextor

An **abstraction contextor** has multiple data-in. The role of the abstraction contextor is to produce a single data-out whose type is at a higher level of abstraction than that of the input data types.

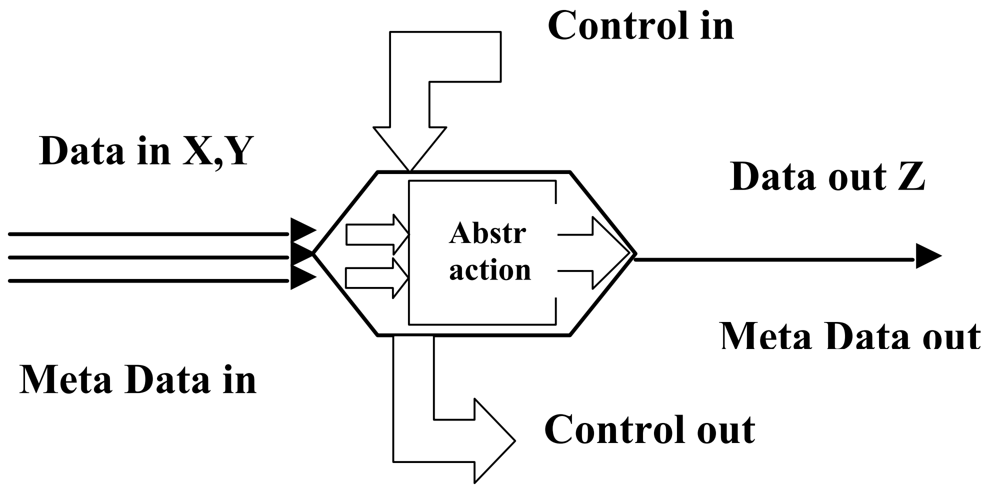


Figure 7. The abstraction contextor

2 Implementation of Mobile Contextor

A context-aware application usually makes use of some other software components that process context. Usually the binding to software components is hard coded into the application. Or when starting, the application consults some directory server to find concrete software component. After that the binding is fixed. This static assumption may not be true, however. During the lifetime of an application, the software component may fail or be enhanced with more functionality. Or the user may want the application to still function as usual when the user moves to a new place. And in previous chapter, we see a general semi formal model for context and the corresponding computational model. It seems a promising general model of context. In this chapter, we describe our initial efforts to implement mobile contextor to make context-aware applications function after moving. First we analyse the problems to be solved when applications move to new places.

2.1 Analysis

After the application moves, the application must void the bindings to the original contextors, locate new contextors and re-establish bindings to new contextors, or moves some of the contextors to the new place. The mechanism to deal with mobility problems of contextors is very similar to data space management in code mobility [FuggettaPicco1998]. It depends on whether contextor can be moved, how contextor is bound and applications requirements.

According to whether contextor can be transferred or not, they can be divided into two kinds: transferable or not transferable. For example, a elementary contextor to detect temperature is not transferable. A translation contextor that translate temperature in Celsius into temperature in Farenheit is transferable. Among transferable contextor, contextor can be marked fixed or free. The difference is that it is unreasonable to move fixed transferable contextor although it can be transferred. For example, a softwarte contextor that makes use of a huge file is marked fixed. So totally there are three types of contextor: free transferrable, fixed transferrable, fixed not transferable.

Contextors are small software components. Context-aware applications may bind to some contextors. Some contextors may further bind to other contextors. These bindings can be divided into three categories: by identifier, by value and by type. Contextor bound by identifier means that the contextor is uniquely identified and cannot replaced by other contextors. Contextor bound by value means that the type and the content of the data out channel of the contextor remains the same after migration. Contextor bound by type means that the type of the data out channel of the contextor remains the same after migration.

When an application moves, the bindings or references to contextors at the application must be modified accordingly in order to for the application to work at the destination. Mechanisms dealing with modification of reference to contextors can be divided into the following categories:
1. by move : The contextor moves with the application and the reference does not change.
2. by copy : The duplicate of the contextor moves with the application. The original contextor stays at the source. The application changes the reference to use the duplicate of the contextor.
3. by network reference: The contextor does not move with the application. The application still uses the contextor at the source and the reference is modified to become a network reference to the original contextor.
4. by rebind: The contextor does not move. The application finds a new contextor of the same type at the destination and modifies the reference to use the new contextor.

Alfonso Fuggetta et al. list the possible mechanisms to deal with data space management problems in code mobility in the following table [FuggettaPicco1998].

	Free Transferrable	Fixed Transferrable	Fixed Not Transferrable
By Identifier	By move (Network reference)	Network reference	Network reference
By Value	By copy (By move, Network reference)	By copy (Network reference)	(Network reference)
By Type	Re-binding (Network reference, By copy, By move)	Re-binding (Network reference, By copy)	Re-binding (Network reference)

Table 1

2.2 Implementation

Above we see the basic mechanisms to deal with rebinding problems, however, how can we make use of these mechanisms in the implementation? When we move some contextors with the application, these contextors may rely on other contextors. So we may have to move still some other contextors and/or change the bindings to these contextors. The FarGo project at Israel Institute of Technology [HolderBen-Shaul1999a] [HolderBen-Shaul1999b] provides dynamic component relocation while the application is running. It implemented the above basic mechanisms. FarGo is Java-based programming environment for the development of mobile-component-based distributed applications. In FarGo an applications consists of complets. A complet closure is a collection of local objects. All references among these objects are local references. Complets are interconnected via inter-complet references. There are five types of complet references to other complets: Link, Pull, Duplicate, Stamp and Bi-directional Pull. Each reference type has different requirement on the relocation activity of the target complet after the source complet moves. With respect to relocation, Link means that there is no special relation between the source and the target complets. Pull means that when the source complet moves, it pulls the target complet with it. Duplicate means that when the source complet moves, a copy of the target complet is brought with the source complet. Stamp means that the source complet find a local instance of the target complet at the destination. Bi-directional Pull means that when the source complet moves, it pulls the target complet with it and vice versa.

With respect to mobile contextors, we can add an attribute about reference type in a reference object of an application. When an object moves to the destination, the receiving portal modifies the reference and gets resources according to the reference type. We propose providing two methods in the agent: beforemove and afterarrive. In beforemove method, the application saves the binding relation between the application and the contextors and sends itself and contextors to the remote site. After

In the implementation of mobile contextor, we use Migrants from Trinity College Dublin as mobile agent implementation. A mobile agent application usually consists of two parts: the portal and the mobile agent. The portal runs on each machine all the time and is responsible for sending and receiving mobile agents. The application was a ping application to check whether a list of machine is alive or not. There are two contextors. The data output of the first contextor is a list of machine. The second contextor connects to the first contextor and uses "ping" command to check whether these machines are on or not. The application was implemented as an agent. There are two implementations of mobile contextors. In one implementation, the application makes use of the original contextor by network reference. In the second implementation, the application moves all the contextors with it and restarts all the contextors at the destination.

2.3 Conclusion

The implementation shows that it is possible to move the application with corresponding contextors to a new place and still runs there. However, it does not consider reuse. In an instrumented environment, the environment can provide some facility to process context. And the applications in that environment can shared these facilities provide by the environment.

3 Conclusion, discussion and opening issues

Up to date, most of context-aware research is limited to isolated indoor environment. Typical examples include meeting rooms, design room, office buildings and museum rooms. Many scenarios have been identified and act as test bed for context-aware research. Researchers also proposed various supporting infrastructures that focus on these environments. But when the user move between different separated spaces, there is no software support for bridging this. Little research has been done in outdoors environment. Tourist guide is an exception. However, tourist guides usually only considers location. They rarely make use of other context. But there are many other outdoor human activities that can be supported by context-aware research. Correspondingly there are very few context-aware infrastructures for outdoor environment. In order to provide coherent interactions and responses in an ad hoc outdoor, and by definition mobile, world we need to use context to support the decision process.

In some projects for example the TEA project, the software support for context-aware applications was provided solely by software on the computing device. In other projects, the infrastructure in environment provides the software support. These are the two extremes. It seems that none of the above software support make use of the mobile device and the infrastructure at the same time.

REFERENCES

[Brown1996] Brown, Peter J. (1996b). The Stick-e Document: A framework for creating context-aware applications. In the *Proceedings of the Electronic Publishing '96*, pp. 259-272, Laxenburg, Austria, IFIP. September 1996.

[BrownJohn1997] Brown, Peter J., John D. Bovey and Xian Chen (1997). Context-aware applications: From the laboratory to the marketplace. *IEEE Personal Communications* 4(5): pp. 58-64. October 1997.

[BrumittMeyers2000] Brumitt, Barry L., Brian Meyers, Jon Krumm, Amanda Kern and Steve Shafer (2000). EasyLiving: Technologies for Intelligent Environments. In the *Proceedings of the 2nd International Symposium on Handheld and Ubiquitous Computing (HUC2K*), pp. 12-27, Bristol, UK, Springer-Verlag. September 25-27, 2000.

[CastroMuntz2000] Paul Castro and Richard Muntz (2000). Managing Context for Smart Spaces. *IEEE Personal Communications,* 7(5), 44-46.

[ChenKotz2000] Guanling Chen and David Kotz. A Survey of Context-Aware Mobile Computing Research. Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College, November, 2000.

[CoolTown2000] http://www.cooltown.hp.com

[DaviesMitchell1998] Davies, N., Mitchell, K., Cheverest, K., Blair, G. (1998). Developing a Context Sensitive Tourist Guide, First Workshop on Human Computer Interaction with Mobile Devices, GIST Technical Report G98-1. <http://www.dcs.gla.ac.uk/~johnson/papers/mobile/HCIMD1.html>

[Dey2000] Anind K. Dey. Providing Architectural Support for Building Context-Aware Applications. PhD dissertation. Georgia Institute of Technology, November 2000.

[DeySalber2001] Dey, A. K., Salber, D., Abowd, G. D. (2001). A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-Computer Interactio*n, *1*6, xxx-xxx. Vol.16 2001

[FranklinFlaschbart1998] Franklin, David and Joshua Flaschbart (1998). All gadget and no representation makes jack a dull environment. In the *Proceedings of the AAAI 1998 Spring Symposium on Intelligent Environments (AAAI Technical Report SS-98-02*), pp. 155-160, Palo Alto, CA, AAAI Press. March 23-25, 1998.

[GellersenBeigl2000] Hans-W. Gellersen, Michael Beigl and Albrecht Schmidt, "Sensor-based Context-Awareness for Situated Computing", Workshop on Software Engineering and Pervasive Computing SEWPC00 at ICSE 2000, Limerick, Ireland, June 2000.

[HealeyPicard1998] Healey, J.; Picard, R.W (1998). Startlecam: A Cybernetic Wearable Camera. 2nd. International Symposium on Wearable Computers, Pittsburgh, Pennsylvania, 19-20 October, 1998, pp.42-49.

[HolderBen-Shaul1999a] O. Holder, I. Ben-Shaul and H. Gazit, System Support for Dynamic Layout of Distributed Applications. Proceedings of the 19th International Conference on Distributed Computing Systems (ICDCS'99), Austin, TX, USA, May 1999, pp 403-411.

[HolderBen-Shaul1999b] O. Holder, I. Ben-Shaul and H. Gazit, Dynamic Layout of Distributed Applications in FarGo. Proceedings of the 21st International Conference on Software Engineering(ICSE'99), Los Angeles, CA, USA, May 1999, pp 163-173.
[FuggettaPicco1998] Alfonso Fuggetta, Gian Pietro Picco, Giovanni Vigna. Understanding Code Mobility. IEEE TRANSACTIONS ON SOFTWARE ENGINEERING Vol. 24, No. 5; MAY 1998, pp. 342-361

[HongLanday2001] Hong, J., & Landay, J. A. (2001). An infrastructure approach to context-aware computing. *Human-Computer Interactio*n, *1*6, xxx-xxx. Vol.16 2001

[HullNeaves1997] Hull, Richard, Philip Neaves and James Bedford-Roberts (1997). Towards situated computing. In the *Proceedings of the 1st International Symposium on Wearable Computers (ISWC'97*), pp. 146-153, Cambridge, MA, IEEE. October 13-14, 1997.

[Ipina2000] Building Components for a Distributed Sentient Framework with Python and CORBA. Diego López de Ipiña. *Proceedings of the 8th International Python Conference*, Arlington, VA, USA. 24-27 January, 2000

[KindbergBarton2001] Tim Kindberg, John Barton, Jeff Morgan, Gene Becker, Debbie Caswell, Philippe Debaty, Gita Gopal, Marcos Frid, Venky Krishnan, Howard Morris, John Schettino, Bill Serra, Mirjana Spasojevic. People, places, things: web presence for the real world. Internet Systems and Applications Laboratory Hewlett-Packard Laboratories.
http://cooltown.hp.com/dev/wpapers/WebPres/WebPresence.asp

[LamminFlynn1994] Lamming, M. and Flynn, M. (1994). Forget-me-not: Intimate Computing in Support of Human Memory, in proceedings of FRIEND 21: International Symposium on Next Generation Human Interfaces,Tokyo,1994 , pp. 125-128.

[LeonhardiKubach1999] Leonhardi, A.; Kubach, U.; Rothermel, K.; Fritz, A. (1999). Virtual information towers - a metaphor for intuitive, location-aware information access in a mobile. 3rd International Symposium on Wearable Computers, San Francisco, California, 18-19 October, 1999, pp. 15-20.

[Long1996] Long, S., et al. (1996). Rapid Prototyping of Mobile Context-aware Applications: The Cyberguide Case Study. 2nd ACM International Conference on Mobile Computing and Networking (MobiCom'96) 1996 November 10-12, 1996.

[Nelson1998] Giles John Nelson. Context-Aware and Location Systems. PhD dissertation. University of Cambridge. January 1998.

[NewmanClark1999] Neill J. Newman and Adrian F. Clark (1999). Sulawesi: A wearable application integration framework. In the *Proceedings of the 3rd International Symposium on Wearable Computers (ISWC '99*), pp. 170-171, San Francisco, CA, IEEE. October 20-21, 1999.

[PascoeRyan1998] Pascoe, J, Ryan, N.S., Morse, D.R. (1998). Human Computer Giraffe Interaction - HCI in the Field. Workshop on Human Computer Interaction with Mobile Devices.

[PascoeRyan1999] Pascoe, J., Ryan, N.S. and Morse, D.R. (1999). Issues in Developing Context-Aware Computing. Proceedings of the International Symposium on Handheld and Ubiquitous Computing (Karlsruhe, Germany, Sept. 1999), Springer-Verlag, pp. 208-221.

[Rey2001] G. Rey. *Systèmes Interactifs Sensibles au Context*e. Ecole doctorale de Mathématiques et Informatique, DEA d'Informatique, Systèmes et Communications, Université Joseph Fourier et Institut National Polytechnique de Grenoble, June, 2001, 84 pages.

[Rhodes1997] Rhodes, B.J. (1997). The wearable remembrance agent: a system for augmented memory. 1st International Symposium on Wearable Computers, Cambridge, Massachusetts, October 13-14, 1997, pp.123-128.

[RoddenCheverst1998] Rodden, Tom, Keith Cheverst, Nigel Davies and Alan Dix (1998). Exploiting context in HCI design for mobile systems. In the *Workshop on Human Computer Interaction with Mobile Device*s, Glasgow, Scotland. May 21-23, 1998.

[RyanPascoe1997] Ryan, N., Pascoe, J., Morse, D. (1997). Enhanced Reality Fieldwork: the Context-Aware Archaeological Assistant. Gaffney, V., van Leusen, M., Exxon, S. (eds.) Computer Applications in Archaeology.

[RyanPascoe1998] Ryan, Nick, Jason Pascoe and David Morse (1998). *Enhanced reality fieldwork: the context-aware archaeological assistan*t. Computer Applications and Quantitative Methods in Archaeology. V. Gaffney, M. van Leusen and S. Exxon, Editors. Oxford.

[SamulowitzMichahelles2001] Adaptive Interaction for Enabling Pervasive Services, M. Samulowitz, F. Michahelles, C. Linnhoff-Popien. 2nd ACM International Workshop on Data Engineering for Wireless and Mobile Access (MobiDE01), Santa Barbara, California, USA, May 2001.

[Schilit1995] Schilit, W.N. (1995). System Architecture for Context-Aware Mobile Computing, Ph.D. Thesis, Columbia University.

[SchilitAdams1994] Schilit, Bill N., Norman I. Adams and Roy Want (1994). Context-aware computing applications. In the *Proceedings of the 1st International Workshop on Mobile Computing Systems and Application*s, pp. 85-90,
Santa Cruz, CA, IEEE. December 8-9, 1994.

[SchilitTheimer1994] Schilit, Bill N. and Marvin M. Theimer (1994). Disseminating active map information to mobile hosts. *IEEE Network* 8(5): pp. 22-32. September/October 1994.

[SchmidtAidoo1999] Schmidt, Albrecht, Kofi Asante Aidoo, Antti Takaluoma, Urpo Tuomela, Kristof Van Laerhoven and Walter Van de Velde (1999). Advanced interaction in context. In the *Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing (HUC '99)*, pp. 89-101, Karlsruhe, Germany, Springer-Verlag. September 27-29, 1999.

[WangGarlan2000] Task Driven Computing, Zhenyu Wang and David Garlan, Workshop on Software Engineering for Wearable and Pervasive Computing, ICSE 2000.

[Want1995] Want, R. et al. (1995). An Overview of the PARCTAB Ubiquitous Computing Environment. IEEE Personal Communications, vol 2, no 6, Dec 1995, pp. 28-43.

[WantHopper1992] Want, R., Hopper, A., Falcao, V., Gibbons, J. (1992). The Active Badge Location System. ACM Transactions on Information Systems 10(1) pp. 91-102.

[WardJones1997] Ward, Andy, Alan Jones and Andy Hopper (1997). A new location technique for the active office. *IEEE Personal Communications* 4(5): pp. 42-47. October 1997.

[Winograd2001] Terry Winograd. Architectures for Context.
Human-Computer Interaction journal, Special Issue of Human-Computer Interaction, Volume 16, 2001

[YangYang1999] Jie Yang; Weiyi Yang; Denecke, M.; Waibel, A. (1999). Smart sight: a tourist assistant system. 3rd International Symposium on Wearable Computers, San Francisco, California, 18-19 October, 1999, pp. 73-78.

HIGH LEVEL REQUIREMENTS AND ARCHITECTURE

Prof. Patrick Nixon
Computer Science Department
The University of Strathclyde

This report introduces a high level architecture for a system to support ad hoc connections within a CORBA/Java compliant environment. The report outlines the progress made in defining the architecture based on policies and events which will be used to integrate robustness and fault tolerance features into a an ad-hoc mobile code environment.

1.1. Mobility Architecture

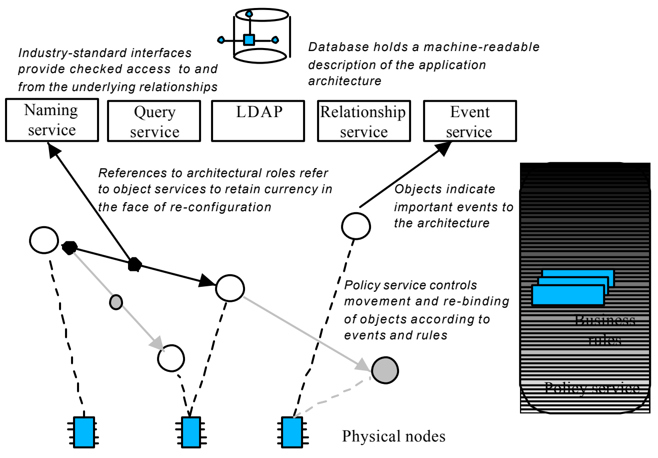
The mobility architecture's task is to ensure that mobile computations can transfer from node to node successfully and continue executing their tasks. What objects end up being migrated is dictated by the decision of the policy service and event service.

The structure of a generic migratory distributed application is displayed in figure 1. It shows how each component (also referred to as an autonomous object) can exist on a different node and yet remain in communication with other components of the application. Here the term node refers to a single processor, typically a single host. The circles represent components, while the dashed lines represent the various network references, or simply links, between them. The components are capable of being migrated to another node yet still carry on its computation, and more importantly its role in the computation. This is the goal for a real world system but prior to building such a system it is necessary to describe a disciplined abstract structure.

The complete structure of the architecture can be encapsuated in a logical database. It represents a rational method to store such a configuration because it is well structured, secure and retains data integrity automatically. The logical database can be implemented as a single, distributed or federated database, and we use the central database term solely to convey the methodology to be used in implementing such a system.

Different interfaces to the database allow access to the data. These interfaces are in the form of the standard services such as those specified by the OMG, RM-ODP, ODMG, and IETF. Due to the fact that they are stateless they represent logical filters to all data. For consistency we assume the definitions of the OMG for discussions of services named below.

The relationship service and naming service will assist migrating components to rebind to well specified or generic services. For example, should a migrating component wish to avail of a standard printing service on a destination host then the relationship service will provide the appropriate resource name. If the docking component has special 'secure' privileges then the relationship service may specify a secure output device such as the manager's printer while 'normal' components are referred to a standard office printer. Using this evaluated resource name a concrete binding to the particular resource is returned from the naming service. The Event Service [4] allows objects to communicate using decoupled event-based semantics. The Query Service provides an interface in much the same way as SQL provides an interface to databases.



Industry-standard interfaces provide checked access to and from the underlying relationships
Database holds a machine-readable description of the application architecture
Naming service
Query service
LDAP
Relationship service
Event service
References to architectural roles refer to object services to retain currency in the face of re-configuration
Objects indicate important events to the architecture
Policy service controls movement and re-binding of objects according to events and rules
Business rules
Policy service
Physical nodes

Fig. 1. A Managed Architecture Mobile Distributed Applications

All the services see the same data in the database. Their purpose is to give the component a view of just the data needed to provide the bindings between resource names and their object references. This provides a seperation of data from the interfaces used to access it. Migrating components use the service interfaces to reconfigure their resource bindings. Each migration is different and can have different consequences. Use of standards allow the architecture to evolve as more interfaces become available over time and therefore, such an open system has the potential to do for migratory applications what OMG's CORBA did for remote object invocation.

1.2. Proposed migration technique

The research community has two broad notions in regard to mobile computations. Cardelli [9] states that mobile agents are meant to be completely self-contained. They do not communicate remotely with other agents, rather they move to some location and communicate locally when they get there. Agents are generally perceived to have an intelligence aspect which is lacking in simple mobile object systems. Mobile objects have their course set out from the start. They exhibit no independent judgement and can be though of as a drone. These methods represent opposite ends of the control spectrum. One having complete autonomy the other having none. We propose a mid grained approach where the key goal is **control and not constraint**.

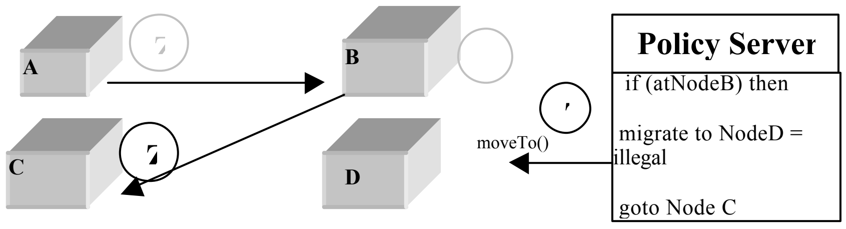


Fig. 2. Migration Technique

An important part of the architecture resides in the policy management unit. Each component, which is capable of migrating, has a 'moveTo()' method for such a purpose. The component itself can not call this method. Instead the policy unit invokes it, providing it with a destination and any other necessary parameters. The policy manager can dictate for example, whether it will allow a particular component access to its node or even whether an object can leave its node. Such decisions are dynamic and complex, and the design of a comprehensive policy manager is outside the scope of this paper. Were the component able to invoke moveTo() itself it would exibit the primary trait of an autonomous agent. Such autonomy can be simulated within the architecture by the use of a null policy which simply defers to requests of the requesting object. Policies are the key to adaptibility, as they also allow a range of different design choices to be facilitated in a single implementation. Policies even enable for design decisions to be revised post implementation without the necessarily having to change the algorithmic solution.

1.2.1. State Space Access

The Name Service in OrbixWeb [4] provides a standard OMG service. The format it saves its data in is proprietary. We have implemented a stateless name service which saves its data in an oracle database. This procedure allows other services to view the same data and alter it accordingly without causing conflicts which would arise were the name service stateful. Another advantage is the ease of use from a management point of view. It separates the concerns of users and administrators.

1.2.2. Architectural Pointers

Central to a stable migration is the abstract concept we refer to as an *architectural pointer*: a reference to an object specified relative to the architecture database by way of one or more of the services made available by the architecture. The pointer essentially encapsulates a query against the database together with the object resulting from the last time the query executed.

The power of architectural pointers comes from the use of the database. By combining architectural, configuration and placement data within a single logical structure, and allowing different views through the services, it is possible to identify roles in a very flexible manner. It also allows objects to reference roles directly, rather than purely retaining links to the objects which happened to fulfill those roles at any point in time. Finally, by acting as event consumers, they allow the policy component to re-configure the application and propagate these changes (in the form of pointer invalidations) directly to all affected objects.

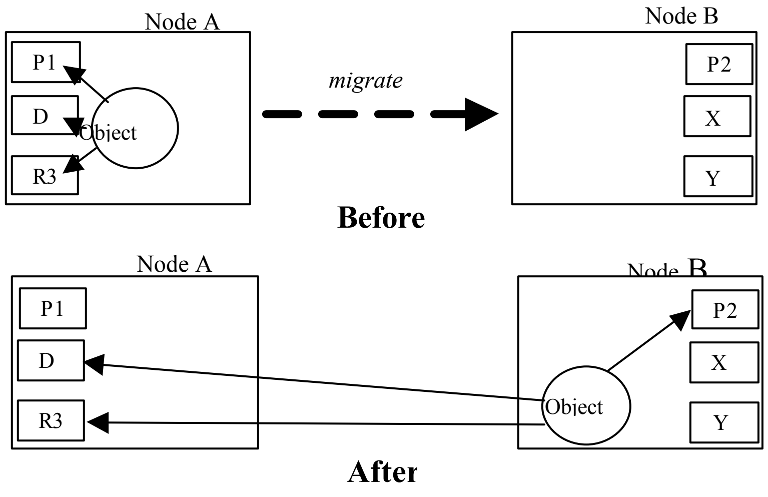


Fig. 3. Architectural Pointers

As an example of the utility of architectural pointers consider an object using both shared user database (D) and a printer (P1), with the constraint that the object always uses the same database irrespective of location while it's printer is defined as the printer on the local node. Architecturally this may be specified by identifying the user database explicitly by name and the printer in relation to the local node (figure 3 - before). If the object is migrated (figure 3 - after) then the database reference will remain valid but the printer reference will be invalidated, and when re-evaluated will re-bind to the printer on the object's new node (P2). The invalidation and re-binding is performed by the migration run-time system.

1.3. Event Service

On examination of a typical room in an office, depending on the level of granularity you choose, there can be many thousands of different things going on all around. Every time someone or something moves can be classified as a 'happening' or an 'event'. Typically in an office environment we are only interested in the events that interact on the human level. These are occurrences which we can see and observe. People moving from one room to another, doors opening and closing, a printer finished printing, a kettle boiled, these all fall into events which happen or which we cause to happen.

Even in a relatively small building and even at this level of granularity there are still immense amounts of information relating to the full set of events. Each object which raises an event wants all interested parties to know that this event has occurred. The simplest manner to deal with this is to broadcast to all who are listening. However, when examining all the objects which make up the average building, this will inevitably be to much information. What results is an 'event storm' where all objects are trying to tell every other object about it's condition. This leads to total congestion on the communication medium with the effect of nothing, or very little, getting done.

What is required is a system which will manage these events and control the flow of event notification only to those objects who require it. This is achieved by firstly categorising objects into sources and sinks. An object is an *event source* when it generates events. Therefore, a door opening is a type of event so the door object is an *event source*. An *event sink* is an object which is interested in particular events. An example here might be the security object, which is interested in a the door object being opened. It will want to check whether the room should be accessible and whether the person entering it has permission to do so. For the *event sink* to be informed of such events it needs to register with the *event source* requesting that it send it an such occurrences.

1.3.1. Event Service Advertising

As mentioned previously an object which produces events are referred to as *event sources*. Prior to the issuing of events objects have first to advertise their intention to publish events.

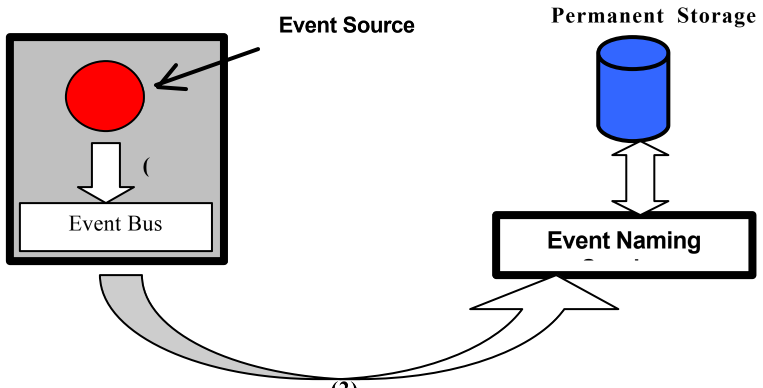


Fig. 4. Event Service Advertising(2)

The first step in advertising an will be a call to the *Advertise(name, other info)* method to indicate to the event bus it readiness to produce events. A naming system must be adopted whereby the events can be uniquely identify throughout the system. To this end a proprietary naming scheme will be used for the naming of the events.

An example URI that could uniquely identify an event might look like the following, *ie.tcd.Oreilly.floor2.Simon.enterEvent*. This would uniquely identify the event, but also give the location to where the information on the event is found and what protocol is to be used.

The event bus is the support mechanism for production and consumption of events. In this model an event bus will be located on each device. The Advertise method passes the event bus the name of the event, which is the identifier discussed earlier. It also passes other information such as owner event source.

The next step is for the local event bus to forward this information, along with location of the event source, onto the Event Naming. This is the central repository for named events. It is a look up mechanism which *event sinks* use to discover an *event source*. After the Naming Service receives the information, it inserts it into its database thereby adding the event source to the list of other sources. Once this is completed the event source is considered as been advertised.

1.3.2. Event Subscription

As with most event services clients are required to, in some way, subscribe to events that they are interested in. This narrows the scope of the events on the overall system. It also prevents event storming by only sending the notification of the event to the parties which have shown an interest in that event. Clients in this event service are required to subscribe to an event using the full name of the event

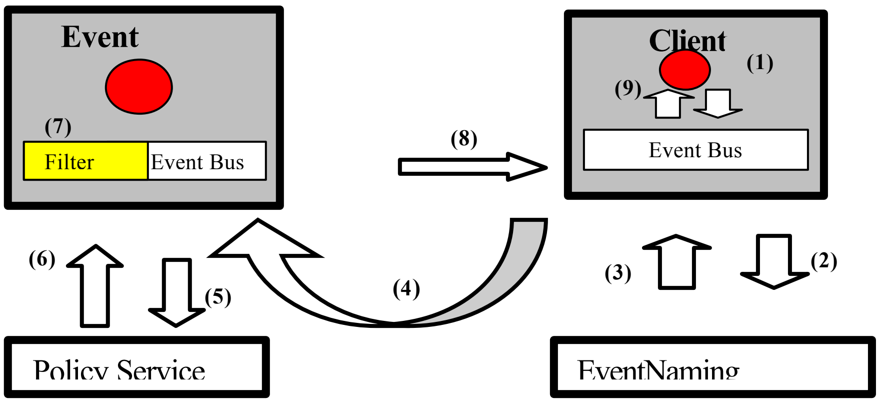


Fig. 5. Subscribing to Events

Figure 5 shows the series of steps required for a client to subscribe to an event. There are four distinct stages to subscribing; the client making the request, finding the location of the event bus which support the event source, subscribing to the event bus and the installation of the filter. The filter in this case is acquired from the policy service which is discussed in Section 3.3. Suffice to say that the filter will ensure that event sinks will only receive notice of the event if certain eventualities prove true.

An event sink can unsubscribe from event sources. It does so in a similar, but reverse order, fashion as event subscription.

1.3.3. Event Notification

Figure 6 shows the manner in which the event bus informs registered objects. The event source notifies it local event bus of a new event and passes the parameters associated with this instance of the event. The job of the event bus is to then to notify the interested parties of the event.

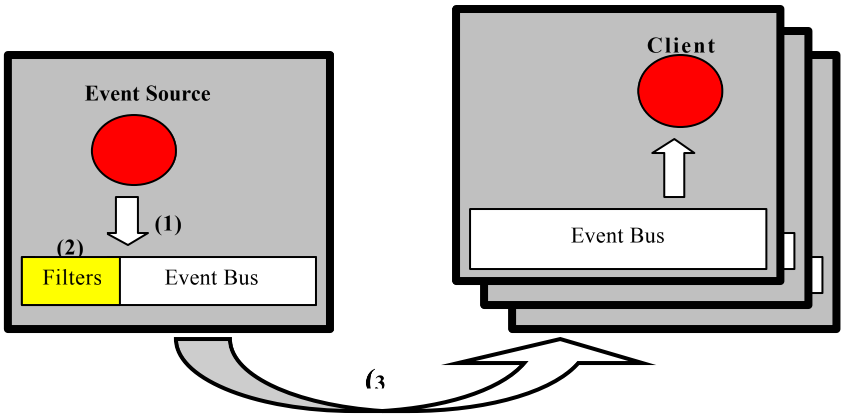


Fig. 6. Event Notification

1.4. Policy Service

All formal organisations have policies, which are defined as 'the plans of an organisation to meet its goals'. Policies, on a more abstract level, represent a capability or a capacity to do something.

Take the example of a typical building, be it a shop, an office or a home environment, policies are in force. Offices and shops typically have a security guard with specific instructions from a building manager. They will only allow certain authorised individuals into sensitive areas of the premises or they may refuse others access altogether. In this particular case the person attempting to enter is referred to as the policy subject while the security guard is the policy target. The target will have actions available which are to be accessed by the subject. In this example the only action available might be *enter*. If there is a policy constraint associated with the target object, for example *"are underage"* then the *enter* action may not be accessible to a person who is declared to be underage.

The primary purpose of the policy service is to model these rules in such a way that they are specific, unambiguous and fulfil their purpose.

To model a building for policy needs, a tree of zones is mapped out. Figure 7 shows the O'Reilly Institute in Trinity mapped out into constituent zones. The fixed elements which a building contains (refer to section 3) will be assigned to specific zones while the mobile elements have the ability to move between zones. Zones and objects have policies associated with them. For all elements in the environment a policy object will be associated with it.

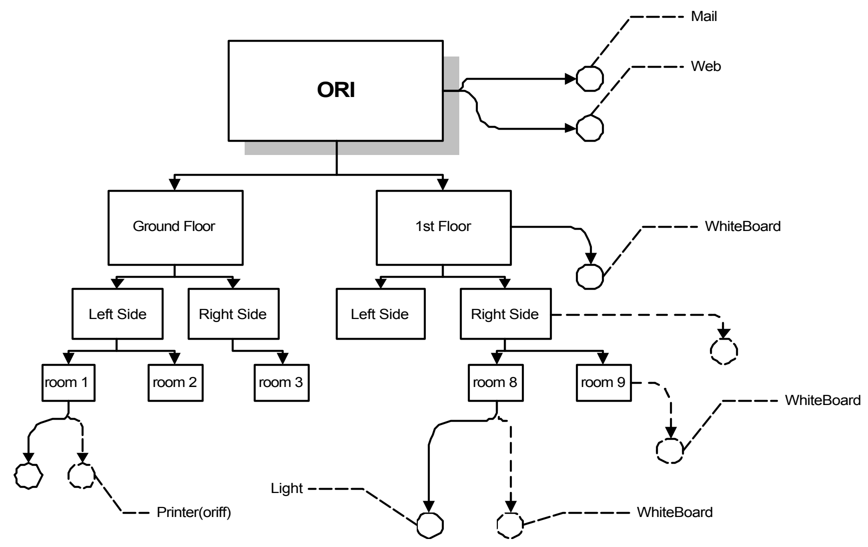


Fig. 7. Zone tree

As mentioned in the example above policies have certain elements associated with them. A full definition is provided by Sloman. According to Sloman [2] a policy can be defined by five criteria.

?? **Modality:** A policy may be authorisation or obligation. In both cases the modality can be positive of negative. Positive authorisation (A+) is a flag which indicates that if a certain action occurs the perpetrator will be permitted to carry on. For example a system administration person attempting to enter a server room would be allowed. Negative authorisation (forbidding) provides the opposite effect so that a policy could be installed to withhold access for all non system administration personnel unless they are accompanied by an authorised person.

Positive obligation (O+) ensures that if a particular action takes place then this policy dictates a course of action which must be followed. An example could be that if the non sys-admin person was in the server room then that presence requires a log entry in a security database. Negative obligation provides a deterring mechanism

?? **Policy Subject**: This attributes defines the user objects to whom the policy applies. The policy subject can be a specific user, a group of users, or a list of users and groups.

PS: Simon
PS: All
PS: Under, Post, Paddy

The policy subject is classified as the producer of an event. The full relationship between events and policies is discussed at the end of the section.

?? **Policy Target Object**: it defines the objects at which the policy is directed. It can also describe a list of objects or a zone. The target object is the consumer of events.

Printer_oriff
Mail_Server
Door_G51

?? **Policy Action**: This attributes defines the method to which the policy applies. This represents a specific method of the policy object. These methods are available to the policy subject providing that no constraints are in place. On the pretence that no constraints are in place, the policy subject can invoke those methods on the target object.

Print()
Setup()
Send_Mail(Mail)

?? **Policy Constraints**: Constraints serve as our way of setting rules. They give us control over the policy subject's actions. The basic structure is a set of Boolean methods that must all return. Failure to do this will prevent access to the target's functionality. The conditions are very specific to the object. No constraints means free access. This constraint can be another policy. In that case the policy in question have to be respected to fulfil the present one. The following scenario gives an insight as to how constraints work. Simon's office door has a door opening mechanism. The door is the target object. Associated with this is it's action *openDoor*. Also associated is the constraint *Access(UserID)*. As Simon (policy subject) approaches the door an event is generated. The *openDoor* action is not immediately available because a constraint is in place. The UserID is passed to the policy, it returns true as Simon has full access so now *openDoor* is invoked and the door sweeps open. All this information is encapsulated in a policy object.

Cond (Job_Name)	Allow to print only a certain job
Cond (Time)	Allow only at certain time

?? **Owner**: The Owner attribute specifies the owner of the policy. It is used for allowing changes to the policy and for accounting or requests such as all the retrieval of all the policies of a specific user. By default the owner of a policy is the application that owns the object, but the administrator can always modify policies.

Owner: Admin
Owner: Printing_Service
Owner: Paddy

Policies, events and system management are all tightly related. The event service generates the events, the policy service defines the capabilities of all entities/objects and the management service carries out the required tasks.

Next stages

The netx stage sof the work are to develop an implemntation of migrant and disconnected objects based on the above and then to integrate the existing CORBA trader based rebinding services to provide update and sychonisation. This will then be used as the foundation to empiraclly investigate this approach in ad-hoc networking scenarios.

References

[1] Trinity College Library, New Library Building, Architectural Brief
available from http://www.tcd.ie/Library/Local/Newlib/Brief/

[2] J.D. Moffett, M.S. Sloman, "The Representation of Policies as System Objects", SIGOIS Bulletin, Vol. 12 No.2, pp 171-184.

[3] Jini Technology Executive Overview, Sun Microsystems, Inc.
available from http://www.sun.com/jini/overview/

[4] Michael H. Coen, "The future of human-computer interaction, or how I learned to stop worrying and love my intelligent room", IEEE Intelligent Systems, March/April 1999.

[5] Michael C. Mozer, "An intelligent environment must be adaptive", IEEE Intelligent Systems, March/April 1999.
[6] James L.Flanagan, "Autodirective sound capture: towards smarter conference rooms", IEEE Intelligent Systems, March/April 1999.

[7] Frank Olken, Hans-Arno Jacobsen, Chuch McPartland, Mary Ann Piette, Mary F. Anderson, "Object Lessons Learned from a Distributed System for Remote Building Monitoring and Operation", ACM SIGPLAN, Vol 33, No. 10, October 1998.

[8] Sean M. Dorward, Rob Pike, David Leo Presotto, Dennis M. Ritchie, Howard W. Trickey, and Philip Winterbottom, The Inferno Operating System, Lucent Technologies.

[9] Luca Cardelli, "Mobile Computation", Microsoft Research. *available from* http://www.luca.demon.co.uk