AFRL-IF-RS-TR-2003-273
Final Technical Report
November 2003

PROTEUS: ASSESSMENT AND ADAPTATION THROUGH DYNAMIC ARCHITECTURE TECHNOLOGY

University of California

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. K510

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

STINFO FINAL REPORT

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2003-273 has been reviewed and is approved for publication

APPROVED: /s/

ELIZABETH S. KEAN
Project Engineer

FOR THE DIRECTOR: /s/

JAMES W. CUSACK, Chief
Information Systems Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	NOVEMBER 2003	Final Jun 00 – Oct 03

4. TITLE AND SUBTITLE
PROTEUS: ASSESSMENT AND ADAPTATION THROUGH DYNAMIC ARCHITECTURE TECHNOLOGY

5. FUNDING NUMBERS
C - F30602-00-2-0607
PE - 62302E
PR - DASA
TA - 00
WU - 09

6. AUTHOR(S)

Richard N. Taylor, David S. Rosenblum, David F. Redmiles, and
Andre van der Hoek

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

University of California, Irvine
Department of Information and Computer Science
Irvine California 92697-3425

8. PERFORMING ORGANIZATION REPORT NUMBER

N/A

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

Defense Advanced Research Projects Agency
3701 North Fairfax Drive
Arlington Virginia 22203-1714

AFRL/IFSA
525 Brooks Road
Rome New York 13441-4505

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

AFRL-IF-RS-TR-2003-273

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: Elizabeth S. Kean/IFSA/(315) 330-2601/ Elizabeth.Kean@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 Words)*

The Proteus project at the University of California, Irvine (UCI) has created means for effective software reuse and dynamic application adaptation. The project leveraged core ideas in loosely coupled decentralized software architectures and provided a foundation that advanced state-of-art architecture-based application engineering. Technology based on this architecture-centric approach for modeling, design, assessment, implementation, validation, configuration management, and run-time evolution of complex software systems was evaluated by DARPA and non-DARPA technology users.

14. SUBJECT TERMS
Dynamic Application Adaptation, Software Reuse, Architecture-Centric

15. NUMBER OF PAGES
18

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Table of Contents

Executive Summary

This document is the final report for the University of California, Irvine-led effort that proposed to build on previous software architecture research in effective software reuse and dynamic application adaptation to provide comprehensive support for application/component assessment, adaptation, and run-time change. The rest of this document describes the accomplishments made over a three year period as participants in the DARPA Dynamic Assembly of Systems for Adaptability, Dependability and Assurance (DASADA) program.

Introduction

The Proteus project at the University of California, Irvine (UCI) conducted software architecture research to create a means for effective software reuse and dynamic application adaptation. The project leveraged core ideas in loosely coupled decentralized software architectures and provided a foundation that advanced state-of-the-art architecture-based application engineering.

Architecture-based system development is central to the Proteus approach. A strict separation of an application into components (loci of computation) and event-based connectors (loci of communication):

- provides a demonstrated, effective basis for run-time dynamism: architectural models reside with the implementation, providing key resources for assessing, planning and effecting change;
- enables a variety of "wrapping" technologies to be used to adapt components for unanticipated users; and
- fosters use of run-time monitors, within connectors, to dynamically assess system functioning.

Utilizing this foundation, gauging and event-based notification technologies were developed in order to allow architecture descriptions to accurately represent and respond to dynamic change. For Proteus, gauges represent instruments of software measurement and evaluation. As such, gauges should aid software engineers in making decisions, throughout the software adaptation lifecycle. To this end, novel gauge technology was produced that channeled events to human engineers in order support enhanced awareness of system behavior.

During the project, real world applications for DASADA developed technology were sought. To this end, a number of application platforms were utilized to test different parts of Proteus produced technology. DoD systems such as Air Force AWACS, Naval Fires Control, and GCCS-M evaluated Proteus technology for use in their domains. The Mission Data Systems (MDS) project at NASA's Jet Propulsion Laboratory successfully incorporated a number of Proteus tools and technologies to model software architectures

for future space probes for both ground and in-flight systems. Stakeholders in domains such as XML-based systems, distributed energy systems, testing and analysis, and product line architectures have utilized various aspects of Proteus produced technology during this project.

Significant advances in the areas of architecture description languages, product line architectures, domain specific software architectures, internet-wide event notification services, and dynamic architecture-based adaptable systems were achieved.

Participants:

Faculty:

Richard N. Taylor

David S. Rosenblum

David F. Redmiles

André van der Hoek

Graduate Students:

Adrita Bhor

Eric Dashofy

Joachim Feise

Roberto Silveira Silva Filho

John Georgas

Scott Hendrickson

Michael Kantor

Christian Luer

Jie Ren

Anita Sarma

Cleidson Ronald Botelho de Souza

Girish Suryanarayana

Staff Programmers:

Yuzo Kanomata

Kari Nies

Craig Snider

Visiting Scholars:

Alfonso Fuggetta, *Politecnico di Milano, Italy*

Degrees Awarded:

Michael Kantor, Ph.D. Computer Science
Awarded Winter 2001
Dissertation: *"Creating an Infrastructure for Ubiquitous Awareness"* [17]

Technical Approach

The objective of the DASADA program was to create automated self-adaptive systems. The Proteus project used an architecture centric approach, employing aspects of loosely coupled software architecture with emerging event-based technologies in order to meet the objective. This entailed the creation of an architecture description for an application that was accurate, up-to-date, and included with the actual in-field application. The target application was partitioned in the architectural description into manageable, loosely coupled components and connectors and executed using lightweight supporting frameworks. To manage each in-field system, each application was modeled as an executing instance of a model architecture utilizing events as the basis for internal and external information exchange. All variances and change to the application could then be modeled as changes to the configuration of the model architecture and applied as new instantiations of the new model.

In order for such a system to self-adapt, awareness of key parameters were transmitted through an awareness server to either the developers or the application itself. Having utilized events as the information exchange mechanisms internally in the application, tools to service such events became the basis for awareness services and for run-time application analysis. The events within the application could then be used to compare expected values with actual values, thus enabling the creation of gauging technologies.

Run-time in-field adaptation was developed through explicit architecture changes transmitted to the running application. From earlier research, loosely coupled systems that are described with explicit components and connectors have shown promise in run-time application evolution [25]**.** To manage this change effectively, the Proteus project developed mechanisms for configuration management of architecture descriptions. Specifically, diffing and merging technologies were created that specified what components and connectors to change. Utilizing such technology, changes to sets of similar applications became manageable as product line architectures. Proteus work in this area has contributed to the understanding of this domain.

Proteus technology topics used to meet the objective and test systems

In concrete terms, the Proteus project investigated architecture-based dynamic adaptable systems through:

- the creation of an extendable XML-based architecture description language
- development of architectural support frameworks and tools
- development of configuration management and product line management techniques for architectures
- creation of event-based awareness gauges
- development of event-based notification services

Each topic listed above was studied by surveying technologies available in the area, gaining new insights, and creating novel tools. Each tool was utilized by developers to address the needs of DASADA partners that emerged during the course of the project. Non-DASADA partners also contributed to the development through early tests and adoption of the technologies developed. Notable projects in which Proteus tools developed were utilized and/or evaluated include the following:

- Lockheed Martin Federal Systems AWACS Software Emulator
- US Navy SPAWAR Fires Control System
- US Navy SPAWAR Common Operational Picture (COP) data sharing
- JPL Mission Data Systems (MDS) Mars mission software

Accomplishments

Architecture Description Language

The architecture of a software system is a model, or abstraction, of that system. Architecture description languages (ADLs) provide meaningful representations for architectures to be specified and, ideally, to be implemented. In order to achieve dynamic self-adaptive application properties, an architecture description language supporting dynamic adaptability is needed.

The Proteus project surveyed a number of first generation and XML-based ADLs [5] for features that promoted loosely coupled systems and ADL extensibility. Extensibility was sought because many originally capable first generation ADLs became limited due to formats that could not easily be changed. Because the domain of architecture-based self-adaptive systems was not known, prudence dictated that an extensible approach, a hallmark of XML-based ADLs, could encompass research findings without causing massive reengineering. Based on interaction with CMU, a common baseline ADL was

created to facilitate the specification of architectures for the DASADA community. Called xArch, this ADL allowed for simple extensibility and openness through the use of a basic XML schema and was adopted by the DASADA community for XML-based ADL descriptions.

Through early use, xArch was found to lack features that would allow for modular expansion into the realms of configuration management, explicit connector types, and product line architectures. The Proteus project sought to remedy these deficiencies by creating a set of XML schemas that augmented and extended xArch. Collectively this set was distinguished from other offspring of xArch by:

- Provisions for optionality and variance
- First class link objects
- Hierarchical arrangements for sub-architectures
- Explicit connectors
- Mapping of architectural models from conceptual models to in-field deployment

The Proteus project labeled these packages as xADL 2.0 and freely distributed it for review by DASADA and non-DASADA entities [5] (see xADL 2.0 website at http://www.isr.uci.edu/projects/xarchuci/)**.** The XML basis of xADL 2.0 also allowed for common open source tools to be used. This enabled interested parties to adopt xADL 2.0 at low cost. One such party was JPL MDS (see later section) which has gone forward and utilized xADL 2.0 on future Mars mission software.

Using xADL2.0, a variety of architectural styles were tested for effectiveness in different test applications involving different parameters. For instance, the C2-style was used for wrapping legacy systems such as GCCS-M for the US Navy SPAWAR COP data sharing demonstration. This led to the hypothesis that domain specific software architectures for distributed computing could be elaborated. To test this hypothesis, the PACE architectural style was created to model independent processes communicating over a variety of message formats. PACE models applications that interact in a distributed system as four-part machines that interact with other applications, collect information about other applications, collect personal state information, and control actions based on the collected information [33]. Full evaluation of PACE was not fully completed due to its development coming late in the project.

Architecture frameworks and support tools

In order for an ADL's model to actually be carried through from design to implementation, tools and frameworks must be in place that map the description to actual software entities. With dynamic architectures based on explicit connectors and components, frameworks must allow for dynamic properties to be preserved.

To address these issues, the Proteus project engaged in creating frameworks that enabled dynamic architectures and created tools within an integrated architecture development environment. First, an evaluation of existing frameworks and tools was conducted. Based on this investigation, dynamic architecture based on the C2-style presented desirable properties of dynamic change and event-based message passing but were tied to a specific style. Other candidates such as Meta-H and ACME Studio presented overhead constraints not conducive to lightweight packaging. Therefore, a new general event-based framework called c2.fw was created to support dynamic application.

c2.fw was designed to test middleware concerns by providing a means of switching to virtual styles of distributed program interaction. While event-based at its core, c2.fw could simulate RPC style interactions and peer-to-peer discovery through explicit connectors included in the distribution. Architecture style rules can be varied in c2.fw so that messaging can be tuned to the style constraints adopted in the ADL.

Based on c2.fw, core tools for run-time architecture change were created [5]. ArchEdit allows any xADL 2.0 architecture to be modified. It was based on a direct inclusion of all the XML schemas a given architecture specified. These schemas then became parameters that ArchEdit could access and vary. Apigen and xArchlib are a set of tools that can be applied to any xArch-based architecture. It allowed for schema extensions to be made available though a programmatic interface. AEM is a management component that will execute architecture change when directed to do so. As a whole, the Proteus project packaged tools together in an integrated architecture-based development environment called ArchStudio 3 (see website at http://www.isr.uci.edu/projects/archstudio/). The environment was deployed as a xADL 2.0 architecture with the tools listed above as components included in the structure of the overall application. Each tool could be added or subtracted from the system both before and during runtime by changing the architecture description.

Because of the open nature of xADL 2.0, the Proteus project supported a number of OTS tool integrations. Among these was an integration with Microsoft's Visio that enabled a visual representation of an actual architecture to be made [26]. The first AWACS software emulator created by Proteus used this integration to replace the buggy one-off visual interface that Lockheed Martin Federal Systems used for their emulator. Another OTS integration was the creation of a plug-in for IBM's Eclipse program development environment. This integration allowed a developer to look at an abstract architecture specification, implement it, and link the developed object code as an architectural instance of the abstract model.

Awareness and Message Tracing technologies

In order for systems to self-adapt, vital system information must be transmitted to entities that decide what changes should be instituted in the system. During the Proteus project, an approach to creating monitors, gauges, and actuators using event-based service

monitors was conducted along the lines of awareness research. The primary tool developed was CASSIUS, a tool that can identify event patterns and sequences of interest from even-based architectures. Because CASSIUS accepted different event sources, it was used to evaluate different messaging technologies and characterizing the cost/benefits for such message-based technologies as Elvin, SOAP, and Sienna as well as other distributed technologies such as CORBA [8]. This characterization allows for event services to be selected that best map to the focus of those needing information.

Based on the event flow from applications, characterizing the messages for critical sequences becomes important to understand application execution. CASSIUS served as a gauge for sequences that could then be filtered and used as triggers to change actuators, but temporal analysis of application events was out of its scope. To understand how application entered critical states in even-based architectures, a message tracing and analysis tool was constructed called MTAT. MTAT traces event sequences in the non-deterministic loosely coupled event-based architectures as temporally related actions. While subject to uncertainty in exact causal chains, MTAT did deliver to developers feedback on areas of unexpected program execution [14].

Architecture Configuration Management technologies

In the Proteus project, architectures are carried throughout the software life-cycle as descriptions packaged with the application. Thus each running program carries an accurate up-to-date model. How application models differ and relate to each other has been a large complex problem area. To attempt to handle this complexity, application of Configuration Management technologies (CM) to identify models as objects distinguished by different describable changes was conducted.

An initial approach was to map CM tools to the architecture domain. This was done early in the Proteus project through the integration of the Menage CM tool with xADL 2.0. While handling architecture descriptions as software source documents does work, interior graph-like structures within XML-based ADLs called for more refined tools to accurately describe differences and to assist in merging similar architectures. To this end, ArchDiff and ArchMerge [36] were created as tools in the ArchStudio 3.0 design environment. These tools allowed for remote changes to be transmitted to the target application and applied using the AEM on board the application. The ability for diffing and merging to be utilized for run-time application change indicated that a compositional environment in which CM and deployment should be addressed.

A comprehensive analysis of ongoing efforts into developing composition environments was conducted in order to create an increased understanding of the nature of those environments [22]. Key questions asked were “What is a deployable component?, ” “Which characteristics must a component model exhibit such that components built using that component model can be effectively deployed and composed?”, and “Which basic features should a composition environment exhibit?” The outcome of this study was

exploration of a number of closely related, but individually targeted composition environments. TWICS (Two-Way Integrated Configuration management and Software deployment), an early example of a configuration management system specifically designed to address component-based software development and evolution, was developed [32], as well as Palintir which focused on the CM concept of workspace as the environment wherein evolution and change of systems occurred [29].

Utilizing insight gained through working with CM-based environments, an approach to Product Line Architectures was conducted on a number of issues. A service utilization study was conducted [15]. This study identified the concepts of optionality and variance as key metrics.

Technology Transfer of Proteus technologies

Lockheed Martin Federal Systems AWACS

The AWACS family of systems is the preeminent airborne command and control system in the world. Originally developed in the 1970's, AWACS is scheduled to continue in service until 2025. The U.S. Air Force anticipates that large aspects of the mission software will change in planned upgrades (such as Block 40/45). Current understanding of the functions of the mission software is limited. Original system requirements (such as a 30 second fail-over/system restart time) have remained unmet because the complexity of the application has outpaced the currently used engineering models. Changes and updates to new capabilities also present engineering challenges to the AWACS program. The current software technology in use requires new mission software to be tested on a software emulator that was originally designed as a one-off prototype. This added cost because each new test emulation must first be encoded as new sequences of operations that are written in the core of the emulation program and then complied before any emulation could be run.

To facilitate an increase in the speed of development and to develop knowledge of the legacy system, Proteus technologies were engaged to:

- model the mission software on the emulator
- model the emulator
- analyze event sequences in the emulator to discover properties of the mission software
- facilitate rapid, low-cost experiments in new technologies and methodologies.

To do this, the Proteus project carefully studied the software emulator and rapidly reengineered it as an architectural model represented in xADL 2.0. Awareness technologies were embedded in the simulator to discover key event patterns and

sequences. Using the Visio integration, an architecture-based model of both the emulator and the underlying AWACS architecture was created that matched the properties of all tested simulation runs of the original emulator. The model was constructed to match the exact style developed by the Lockheed Martin engineers in modeling AWACS. This new architecture-based model allowed for new parameters to be described, compiled, and run on the new simulator in a matter of hours as opposed to the weeks which was common in the old simulator.

Further refinement of the AWACS model was conducted in order for scenarios of run-time mission system adaptability to be examined. The goal was to produce a system between 300-400 distributed components and connectors, performing realistic evolutions in less than 60 seconds. Evolution management tools (AEM) along with support frameworks (c2.fw) were extended to meet demands for a full scale Proteus injection into AWACS. Experimentation continued with the employment of CM technologies that allowed for a notional AWACS simulator to be constructed. This notional simulator tested simulated run-time change capabilities not currently in the AWACS system but sought in Block 40/45 updates. These results led to a DARPA proposal by Lockheed Martin to implement Proteus and other DASADA technology on the Block 40/45 update to AWACS.

US Navy SPAWAR Naval Fires Control

Early in the DASADA program, the Naval Fires Control project examined Proteus technology for injection into the project. Application of awareness technology was conducted along with architecture modeling using xADL 2.0. A detailed examination of the software system indicated that the Naval Fires Control would not benefit from dynamic adaptability as its functions were by fiat designed to be static nonetheless, the examination of Naval Fires Control led to a refinement of Proteus wrapper technology. Naval Fires Control was modeled as a complete component that operated through a defined interface through a special set of components in order allow for event-based message flow.

US Navy SPAWAR Common Operation Picture

Another application of Proteus technologies included a multi-national collaborative project on Common Operational Picture (COP) interoperability for coalition forces. The U.S. effort was led by the Space and Naval Warfare Systems Center (SPAWAR), San Diego. The intent was to support the dynamic exchange of sanitized COP service data between multiple countries and the rapid creation of coalition COP architectures.

The goal was to create a general approach to COP interoperability and adaptability while not replacing or recreating existing national systems. The intent was to develop the information architecture, dynamic adaptation services, and “wrapper” technology around

the existing systems to allow the dynamic exchange of COP elements and thereby enable rapid creation of coalition COPs.

In support of this effort, a demonstration program was developed that simulated a French-US coalition. This prototype consisted of U.S. and French command and control systems, each with their own display. The architecture of each command and control system was modeled in xADL 2.0 and a coalition was formed dynamically between the two independently running command and control systems by applying variants to the executing architectures which allowed for data sharing. Special data filtering components ensured that only select data was shared between countries.

Jet Propulsion Laboratory Mars Smart Lander

JPL examined DASADA technology in order to increase understanding of in-flight mission software and to alleviate the high cost of in-flight mission software change. Current changes typically cost $1 million per in-flight change.

JPL's Mission Data Systems division (MDS) has examined xADL 2.0 along with supporting tools for managing configurations and deploying them. To date MDS has adopted xADL 2.0 for use on upcoming Mar missions including the Mars Smart Lander Mission, scheduled to launch in 2009. MDS has also examined Proteus architecture technology for use in the context of ground systems. Heavy feedback has led to continued refinement of architecture support tools and architecture design environments such as ArchStudio 3.

Key benefits to JPL of Proteus technologies were:

- flexible, extensible architecture descriptions through xADL 2.0.
- the ability to manage in-flight on-the-fly program changes as concise architecture change description.
- ability to manage complex collections of variant architectures and configurations.

MDS has continuously evolved Proteus tools and integrated them at low cost to their standard engineering tools. Continued use of Proteus technology will take place throughout this decade.

Technical Publications

Research supported by this contract has led to 2 technical journal publications [16, 23], 19 conference publications [1, 5, 6, 8, 10-15, 18, 20, 21, 26, 27, 29, 33, 34, 36], 12 workshop publications [2-4, 7, 9, 19, 24, 28, 30-32, 35], a technical report [22] and a doctoral dissertation [17].

References

[1] Baker, A., Oh Navarro, E., and Hoek, A.v.d. Problems and Programmers: An Educational Software Engineering Card Game. In *Proceedings of the Twenty-fifth International Conference on Software Engineering (ICSE 2003).* p. 614–619, Portland, OR, May, 2003.

[2] Dashofy, E., Hoek, A.v.d., and Taylor, R.N. Towards Architecture-Based Self-Healing Systems. In *Proceedings of the First ACM SIGSOFT Workshop on Self-Healing Systems.* p. 21-26, ACM. Charleston, South Carolina, November 18-19, 2002. <http://portal.acm.org/citation.cfm?id=582128.582133>.

[3] Dashofy, E.M. Issues in Generating Data Bindings for an XML Schema-Based Language. In *Proceedings of the Workshop on XML Technologies in Software Engineering (XSE 2001).* Toronto, Canada, May 15, 2001.

[4] Dashofy, E.M. and Hoek, A.v.d. Representing Product Family Architectures in an Extensible Architecture Description Language. In *Proceedings of the International Workshop on Product Family Engineering (PFE-4).* p. 330-341, October, 2001.

[5] Dashofy, E.M., Hoek, A.v.d., and Taylor, R.N. A Highly-Extensible, XML-Based Architecture Description Language. In *Proceedings of the Working IEEE/IFIP Conference on Software Architecture (WICSA 2001).* Amsterdam, The Netherlands, August 28-31, 2001. <http://www.ics.uci.edu/~edashofy/papers/wicsa2001.pdf>.

[6] Dashofy, E.M., Hoek, A.v.d., and Taylor, R.N. An Infrastructure for the Rapid Development of XML-based Architecture Description Languages. In *Proceedings of the 24th International Conference on Software Engineering (ICSE 2002).* p. 266-276, ACM. Orlando, Florida, May, 2002.

[7] De Souza, C.R.B., Basaveswara, S.D., Kantor, M., and Redmiles, D.F. Lessons Learned using Event Notification Servers to Support Awareness. In *Proceedings of the Human Computer Interaction Consortium Workshop 2002.* Fraser, CO, January 2002, 2002.

[8] De Souza, C.R.B., Basaveswara, S.D., and Redmiles, D.F. Using Event Notification Servers to Support Application Awareness. In *Proceedings of the IASTED International Conference on Software Engineering and Applications.* p. 691-697, Cambridge, MA, November, 2002.

[9] De Souza, C.R.B., Basaveswara, S.D., and Redmiles, D.F. Supporting Global Software Development with Event Notification Servers. In *Proceedings of the Workshop on Global Software Development - International Conference on Software Engineering (ICSE 2002).* p. 9-13, Orlando, FL, May 19-25, 2002.

[10] De Souza, C.R.B., Penix, J., Sierhuis, M., and Redmiles, D.F. Analysis of Work Practices of a Collaborative Software Development Team. In *Proceedings of the First International Symposium on Empirical Software Engineering (ISESE 2002).* Nara, Japan, October 3-4, 2002.

[11] De Souza, C.R.B., Oliveira, H.L.R., da Rocha, C.R.P., Gonçalves, K.M., and Redmiles, D.F. Using Critiquing Systems for Inconsistency Detection in Software Engineering Models. In *Proceedings of the 15th International Conference on Software Engineering and Knowledge Engineering (SEKE 2003).* San Francisco, CA, July 1-3, 2003.

[12] De Souza, C.R.B., Redmiles, D.F., Mark, G., Penix, J., and Sierhuis, M. Management of Interdependencies in Collaborative Software Development: A Field Study. In *Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering (ISESE 2003).* Rome, Italy, September 30 - October 1, 2003.

[13] Garg, A., Critchlow, M., Chen, P., Van der Westhuizen, C., and Hoek, A.v.d. An Environment for Managing Evolving Product Line Architectures. In *Proceedings of the IEEE International Conference on Software Maintenance (ICSM 2003).* Amsterdam, The Netherlands, September 22-26, 2003.

[14] Hendrickson, S.A., Dashofy, E.M., Bhor, A., Taylor, R.N., Li, S., and Nguyen, N. An Approach for Tracing and Understanding Asynchronous Systems. In *Proceedings of the 18th IEEE International Conference on Automated Software Engineering (ASE 2003).* Montreal, Canada, October 6-10, 2003. To appear.

[15] Hoek, A.v.d., Dincel, E., and Medvidovic, N. Using Service Utilization Metrics to Assess the Structure of Product Line Architectures. In *Proceedings of the 9th International Software Metrics Symposium (METRICS 2003).* Sydney, Australia, September 3-5, 2003.

[16] Hoek, A.v.d. and Wolf, A.L. Software Release Management for Component-Based Software. *Software - Practice and Experience.* 33, p. 77-98, 2003.

[17] Kantor, M. *Creating an Infrastructure for Ubiquitous Awareness.* Thesis. Information and Computer Science, University of California, Irvine, 2001.

[18] Kantor, M. and Redmiles, D. Creating an Infrastructure for Ubiquitous Awareness. In *Proceedings of the Eighth IFIP TC 13 Conference on Human-Computer Interaction (INTERACT 2001).* p. 431-438, Tokyo, Japan, July, 2001. <http://www.ics.uci.edu/~mkantor/dissertation/CASS_final_sub2.pdf>.

[19] Lingen, R.v.d. and Hoek, A.v.d. Dissecting Configuration Management Policies. In *Software Configuration Management: ICSE Workshops SCM 2001 and SCM 2003 Selected Papers*. p. 177-190, 2003.

[20] Lüer, C. and Rosenblum, D.S. Wren - An Environment for Component-Based Development. In *Proceedings of the Joint 8th European Software Engineering Conference (ESEC) and 9th ACM Sigsoft International Symposium on the Foundations of Software Engineering (FSE-9).* Vienna, Austria, September 10-14, 2001.

[21] Lüer, C., Rosenblum, D.S., and Hoek, A.v.d. The Evolution of Software Evolvability. In *Proceedings of the International Workshop on Principles of Software Evolution (IWPSE 2001).* p. 131-134, Vienna, Austria, September, 2001.

[22] Lüer, C. and Hoek, A.v.d. *Composition Environments for Deployable Software Components.* Information and Computer Science, University of California, Irvine, Technical Report UCI-ICS-02-18, August, 2002.

[23] Medvidovic, N., Dashofy, E.M., and Taylor, R.N. The Role of Middleware in Architecture-Based Software Development. *International Journal of Software Engineering and Knowledge Engineering.* 2003. To appear.

[24] Muccini, H. and Hoek, A.v.d. Towards Testing Product Line Architectures. In *Proceedings of the International Workshop on Test and Analysis of Component Based Systems (TACoS 2003).* p. 111–121, Warsaw, Poland, April, 2003.

[25] Oreizy, P. and Taylor, R.N. On the Role of Software Architectures in Runtime System Reconfiguration. *IEE Proceedings - Software Engineering.* 145(5), p. 137-145, October, 1998.

[26] Ren, J. and Taylor, R.N. Incorporating Off-The-Shelf Components with Event-based Integration. In *Proceedings of the ISCA 12th International Conference on Intelligent and Adaptive Systems and Software Engineering (IASSE-2003).* p. 188-191, San Francisco, CA, July 9-11, 2003.

[27] Ren, J. and Taylor, R.N. Visualizing Software Architecture with Off-The-Shelf Components. In *Proceedings of the Fifteenth International Conference on Software Engineering and Knowledge Engineering.* p. 132-141, San Francisco, CA, July 1-3, 2003.

[28] Sarma, A. and Hoek, A.v.d. Palantír: Increasing Awareness in Distributed Software Development. In *Proceedings of the ICSE 2002 International Workshop on Global Software Development (IWGSD 2002).* Orlando, Florida, May, 2002.

[29] Sarma, A., Noroozi, Z., and Hoek, A.v.d. Palantír: Raising Awareness among Configuration Management Workspaces. In *Proceedings of the Twenty-Fifth International Conference on Software Engineering (ICSE 2003).* p. 444–453, Portland, Oregon, May, 2003. <http://www.ics.uci.edu/~andre/research/papers/ICSE2003-1.pdf>.

[30] Silva Filho, R.S., Slabyak, M., and Redmiles, D.F. A Web-based Infrastructure for Awareness based on Events. In *Proceedings of the Workshop on Network*

Services for Groupware at CSCW 2002. New Orleans, LA, November 16-20, 2002. <http://awareness.ics.uci.edu/~rsilvafi/papers/Workshops/CSCW2002-workshop.pdf>.

[31] Silva Filho, R.S., De Souza, C.R.B., and Redmiles, D.F. The Design of a Configurable, Extensible and Dynamic Notification Service. In *Proceedings of the Second International Workshop on Distributed Event-Based Systems (DEBS '03) (SIGMOD/PODS 2003).* San Diego, CA, June 8, 2003. <http://www.eecg.toronto.edu/debs03/papers/silva_filho_etal_debs03.pdf>.

[32] Sowrirajan, S. and Hoek, A.v.d. Managing the Evolution of Distributed and Interrelated Components. In *Proceedings of the 11th Workshop on Software Configuration Management (SCM-11).* Portland, Oregon, May, 2003.

[33] Suryanarayana, G. and Taylor, R.N. A Decentralized Algorithm for Coordinating Independent Peers: An Initial Examination. In *Proceedings of the Tenth International Conference on Cooperative Information Systems (CoopIS).* p. 213-229, Irvine, California, October 30 - November 1, 2002.

[34] Taylor, R.N. Moving On: Software Engineering Paradigms for the 21st Century. In *Proceedings of the Working Conference on Complex and Dynamic Systems Architectures.* Brisbane, Australia, December 12-14, 2001.

[35] Taylor, R.N. and Dashofy, E.M. *Function follows Form: Architecture and 21st Century Software Engineering.* Software Design and Productivity Working Group, Report, December 13-14, 2002.

[36] Westhuizen, C.V.d. and Hoek, A.v.d. Understanding and Propagating Architectural Changes. In *Proceedings of the Third Working IEEE/IFIP Conference on Software Architecture 2002 (WICSA 3).* p. 95-109, Montreal, Canada, August, 2002.