Curriculum Vitae

Nenad Medvidovic

5711 Verano Place
Irvine, CA 92612
U.S.A.
(714) 856-0543

Department of Information and Computer Science
University of California, Irvine
Irvine, CA 92697-3425
U.S.A.
(714) 824-3100
(714) 824-1715 (Fax)
neno@ics.uci.edu
http://www.ics.uci.edu/~neno/

1 Education

Ph.D. Candidate (expected graduation date: 12/98)
University of California, Irvine
Department of Information and Computer Science
Software Research Group
Advisor: Dr. Richard N. Taylor

Master of Science (6/95, Cumulative GPA: 4.00)
University of California, Irvine
Department of Information and Computer Science
Major Emphasis: Software

Bachelor of Science, *Summa Cum Laude* (8/92, Cumulative GPA: 4.00)
Arizona State University
Major: Computer Science

2 Professional Experience

9/94 - present — Graduate Student Researcher, Software Architecture Group
University of California, Irvine

9/93 - 8/94 — Teaching Assistant
ICS 52 - Systematic Software Construction
ICS 121 - Introduction to Software Engineering
ICS 141 - Programming Languages
University of California, Irvine

6/93 - 8/93 — Research Assistant, Software Engineering Group
Arizona State University

8/92 - 8/93 — Teaching Assistant
CSE 460/598 - Software Project Management and Development I
Arizona State University

3 Publications

Refereed Journal Articles

[1] Richard N. Taylor, Nenad Medvidovic, Kenneth M. Anderson, E. James Whitehead, Jr., Jason E. Robbins, Kari A. Nies, Peyman Oreizy, and Deborah L. Dubrow. “A Component- and Message-Based Architectural Style for GUI Software.” *IEEE Transactions on Software Engineering*, vol. 22, no. 6, pages 390-406 (June 1996). (A significant revision and extension of the ICSE17 paper below.)

[2] Nenad Medvidovic and Richard N. Taylor. “Exploiting Architectural Style to Develop a Family of Applications.” *IEE Proceedings Software Engineering*, vol. 144, no. 5-6, pages 237-248 (October-December 1997).

Refereed Conference Publications

[1] J. Blevins, D. Dubrow, N. Eickelmann, R. Grinter, N. Medvidovic, R. Reimer, J. Shaw, C. Turner, and G. Wong. “Report on the Software Project Management Technical Research Review.” In Proceedings of the California Software Symposium (CSS’95), pages 127-145, Irvine, CA, March 30, 1995.

[2] Richard N. Taylor, Nenad Medvidovic, Kenneth M. Anderson, E. James Whitehead Jr. and Jason E. Robbins. “A Component- and Message-Based Architectural Style for GUI Software.” In Proceedings of the Seventeenth International Conference on Software Engineering (ICSE17), pages 295-304, Seattle, WA, April 23-30, 1995.

[3] Nenad Medvidovic, Richard N. Taylor, and E. James Whitehead, Jr. “Formal Modeling of Software Architectures at Multiple Levels of Abstraction.” In Proceedings of the California Software Symposium 1996, pages 28-40, Los Angeles, CA, April 17, 1996.

[4] Nenad Medvidovic, Peyman Oreizy, Jason E. Robbins, and Richard N. Taylor. “Using Object-Oriented Typing to Support Architectural Design in the C2 Style.” In Proceedings of the Fourth ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE4), pages 24-32, San Francisco, CA, October 16-18, 1996. Also available as Technical Report UCI-ICS-96-6.

[5] Nenad Medvidovic, Peyman Oreizy, and Richard N. Taylor. “Reuse of Off-the-Shelf Components in C2-Style Architectures.” In Proceedings of the 1997 Symposium on Software Reusability (SSR’97), pages 190-198, Boston, MA, May 17-19, 1997. Also in Proceedings of the 1997 International Conference on Software Engineering (ICSE’97), pages 692-700, Boston, MA, May 17-23, 1997.

[6] Nenad Medvidovic and Richard N. Taylor. “A Framework for Classifying and Comparing Architecture Description Languages.” In Proceedings of the Sixth European Software Engineering Conference together with the Fifth ACM SIGSOFT Symposium on the Foundations of Software Engineering, pages 60-76, Zurich, Switzerland, September 22-25, 1997.

[7] Nenad Medvidovic and David S. Rosenblum. “Domains of Concern in Software Architectures and Architecture Description Languages.” In Proceedings of the USENIX Conference on Domain-Specific Languages, pages 199-212, Santa Barbara, CA, October 15-17, 1997.

[8] Jason E. Robbins, Nenad Medvidovic, David F. Redmiles, and David S. Rosenblum. “Integrating Architecture Description Languages with a Standard Design Method.” To appear in Proceedings of the 20th International Conference on Software Engineering (ICSE’98), April 19-25, 1998, Kyoto, Japan. Also available as Technical Report, UCI-ICS-97-35.

[9] Peyman Oreizy, Nenad Medvidovic, and Richard N. Taylor. “Architecture-Based Runtime Software Evolution.” To appear in Proceedings of the 20th International Conference on Software Engineering (ICSE’98), April 19-25, 1998, Kyoto, Japan. Also available as Technical Report, UCI-ICS-97-39.

Refereed Workshop Publications

[1] E. James Whitehead, Jr., Jason E. Robbins, Nenad Medvidovic, and Richard N. Taylor. “Software Architecture: Foundation of a Software Component Marketplace.” In David Garlan, ed., Proceedings of the First International Workshop on Architectures for Software Systems, pages 276-282, Seattle, WA, April 24-25, 1995.

[2] Nenad Medvidovic. “ADLs and Dynamic Architecture Changes.” In Alexander L. Wolf, ed., Proceedings of the Second International Software Architecture Workshop (ISAW-2), pages 24-27, San Francisco, CA, October 14-15, 1996.

[3] Nenad Medvidovic and Richard N. Taylor. “Reusing Off-the-Shelf Components to Develop a Family of Applications in the C2 Architectural Style.” In Proceedings of the International Workshop on Development and Evolution of Software Architectures for Product Families, Las Navas del Marqués, Ávila, Spain, November 18-19, 1996.

[4] Peyman Oreizy, Nenad Medvidovic, Richard N. Taylor, and David S. Rosenblum. “Software Architecture and Component Technologies: Bridging the Gap.” In Proceedings of the Workshop on Compositional Software Architectures, Monterey, CA, January 6-8, 1998.

[5] Richard N. Taylor, Nenad Medvidovic, and Peyman Oreizy. “Architectural Implications of Common Operator Interfaces.” To appear in Proceedings of the Ground Systems Architectures Workshop (GSAW 98), El Segundo, CA, February 25-27, 1998.

Current Workshop Submissions

[1] Nenad Medvidovic, Richard N. Taylor, and David S. Rosenblum. “An Architecture-Based Approach to Software Evolution.” Submitted for publication, January 1998.

Non-Refereed Publications

[1] Jason E. Robbins, E. James Whitehead Jr., Nenad Medvidovic, and Richard N. Taylor. “A Software Architecture Design Environment for Chiron-2 Style Architectures.” Arcadia Technical Report UCI-95-01, University of California, Irvine, January 1995.

[2] Nenad Medvidovic. “Formal Definition of the Chiron-2 Software Architectural Style.” Technical Report UCI-ICS-95-24, Department of Information and Computer Science, University of California, Irvine, August 1995.

[3] Nenad Medvidovic and Richard N. Taylor. “Reuse of Off-the-Shelf Constraint Solvers in C2-Style Architectures.” Technical Report UCI-ICS-96-28, Department of Information and Computer Science, University of California, Irvine, July 1996.

[4] Neno Medvidovic. “A Classification and Comparison Framework for Software Architecture Description Languages.” Technical Report UCI-ICS-97-02, Department of Information and Computer Science, University of California, Irvine, February 1997.

[5] Nenad Medvidovic and David S. Rosenblum. “Assessing the Suitability of a Standard Design Method for Modeling Software Architectures.” Technical Report, UCI-ICS-98-02, Department of Information and Computer Science, University of California, Irvine, February 1998.

4 Formal Presentations

[1] *A Component- and Message-Based Architectural Style for GUI Software.* Technology Presentation Track, 17th International Conference on Software Engineering (ICSE17), Seattle, WA, April 1995.

[2] *Chiron-2: A Component- and Message-Based Architectural Style for GUI Software.* Bay Area Roundtable (BART), Palo Alto, CA, June 1995.

[3] *Chiron-2: A Component- and Message-Based Architectural Style for GUI Software.* Sun Microsystems Laboratories, Palo Alto, CA, June 1995.

[4] *Formal Modeling of Software Architectures at Multiple Levels of Abstraction.* California Software Symposium, Los Angeles, CA, April 1996.

[5] *What's New with C2?* Arcadia Research Meeting, Portland, OR, August 1996.

[6] *ADLs and Dynamic Architecture Changes.* Second International Software Architecture Workshop (ISAW-2), San Francisco, CA, October 1996.

[7] *Using Object-Oriented Typing to Support Architectural Design in the C2 Style.* Fourth ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE4), San Francisco, CA, October 1996.

[8] *Reusing Off-the-Shelf Components to Develop a Family of Applications in the C2 Architectural Style.* International Workshop on Development and Evolution of Software Architectures for Product Families, Las Navas del Marqués, Ávila, Spain, November 1996.

[9] *A Classification and Comparison Framework for Software Architecture Description Languages.* Second EDCS Architecture/Generation Workshop, Santa Fe, NM, April 1997.

[10] *Reuse of Off-the-Shelf Components in C2-Style Architectures.* 1997 Symposium on Software Reusability (SSR'97) and 1997 International Conference on Software Engineering (ICSE'97), joint session on Software Reuse, Boston, MA, May 1997.

[11] *A Framework for Classifying and Comparing Architecture Description Languages.* Sixth European Software Engineering Conference together with the Fifth ACM SIGSOFT Symposium on the Foundations of Software Engineering, Zurich, Switzerland, September 1997.

[12] *Domains of Concern in Software Architectures and Architecture Description Languages.* USENIX Conference on Domain-Specific Languages, Santa Barbara, CA, October 1997.

[13] *A Style-Based Approach to the Design and Evolution of Adaptable Software.* University of Texas, Austin, TX, November 1997.

5 Professional Activities

Reviewer

- 1996 Conference on User Interface Software and Technology (UIST’96)
- 1997 International Conference on Software Engineering (ICSE’97)
- IEEE Transactions on Software Engineering, 1997

6 Professional Associations

- Association for Computing Machinery (ACM)
- ACM Special Interest Group on Software Engineering (SIGSOFT)

7 Honors, Awards, Fellowships

1989-92	Dean’s Honor List
1990-92	Arizona Regents International Student Scholarship
1990-93	Golden Key National Honor Society
1991-93	UPE National Honor Society for the computing sciences
1992	Moeur Award, Arizona State University (top student in the graduating class)
1993-94	ICS Departmental Fellowship

8 Research Interests

My research interests are in the field of software engineering. A common theme, and long-term goal, of my research is *development and evolution of adaptable, large-scale software systems*. Practitioners have traditionally faced many problems with achieving support for reuse, interchange, reconfiguration, extension, and scaling of software modules and/or systems when using existing development approaches. These problems are often the result of poor understanding of a system's overall architecture, unintended dependencies among its modules, decisions that are made too early in the development process, and so forth. Existing techniques that are intended to remedy these problems (e.g., separation of concerns or isolation of change) are only partially adequate in the case of development with pre-existing, large, multi-lingual components that originate from multiple sources.

The main hypothesis of my research is that an explicit architectural focus can remedy many of these difficulties and enable flexible construction and evolution of large systems. I have therefore centered on *software architectures* as a key to developing techniques, tools, and methodologies for engineering flexible, large-scale software. Architectures present a high-level view of a system, enabling developers to abstract away the unnecessary details and focus on the "big picture." Another key is their explicit treatment of software connectors, which separate communication issues from computation in a system. However, existing architecture research has thus far largely failed to take advantage of this potential for adaptability for two reasons:

- Connectors are often not treated explicitly or, when they are, they are too rigid and do not accommodate modification of their attached components easily; and
- The architecture community has to date chosen to focus on issues other than flexible design and evolution: formalism, analysis, and system generation.

My research presents an architecture-based approach to achieving software adaptability. It expands the traditional notions of evolution (e.g., modularity, typing) for use in architectures, and introduces explicit, flexible connectors. I have designed an architecture description language (ADL) that encompasses these concepts; I am currently implementing its supporting toolset. While I have thus far exploited the benefits of a particular architectural style, C2, I intend to demonstrate that many of the results are style- and domain-independent.

This research is also closely related to two other areas: *component-based software development* and *software reuse*. Engineering large-scale software systems is fundamentally different from programming in the small. A programming language statement becomes inadequate as the unit of development. Instead, *components* must become software building blocks. Component-based development of software has become an area of intense research, resulting in several component interoperability models, e.g., CORBA and ActiveX, and wide adoption of certain software design methodologies, such as object-oriented (OO) design. Software architecture researchers have by and large also adopted a component-centered view of software development and evolution.

In the context of the C2 project, I have participated in the development of an OO class framework used for prototyping and implementing C2 components and connectors, as well as a number of example applications that used the framework. This experience has sensitized me to the fact that adopting a component-based approach is, by itself, not enough. Repeatedly engineering components from scratch is likely to become prohibitively expensive; it also fails to exploit the greatest potential benefit of component-based development: reuse. Architectural styles have the potential for controlling the scope of OTS reuse and providing structure for it. Styles typically reflect and leverage key properties of an application domain and recurring patterns of application design within that domain. To adequately support reuse, a style must also provide a balance, where its rules are strong enough to make reuse tractable but broad enough to enable integration of legacy components. I have thus far conducted or participated in a series of exercises intended to investigate reuse issues in the context of the C2 style. Drawing from this experience, I intend to further study what properties are likely to make a given style well suited for supporting OTS reuse.

As part of my longer-term research plans, I also intend to continue investigating several other questions that remain unanswered. A subset of these is given below.

- What techniques can be employed effectively for refining an architecture across levels of abstraction and for mapping that architecture to an implementation?
- What is the role of ADLs in architecture-based runtime software evolution? What ADL primitives are needed to support such evolution?
- What are the issues in runtime insertion, removal, and replacement of connectors? This problem is further compounded by the potentially complex mappings of an architecture to its implementation. A single connector at the architectural level may be distributed across several implemented modules. Extracting such a connector in the implementation and removing or replacing it is a difficult and still open research problem.
- What is the role of architectures and ADLs in the construction of application families?
- How can explicit architectural models aid component discovery and extraction, and the construction of component adaptors and wrappers?

Finally, I intend to further pursue the work I have begun on understanding the nature of software architectures and ADLs and clarifying their roles in the software engineering lifecycle. Despite much ongoing research in architectures and ADLs, consensus is lacking on several fundamental issues, including:

- What problems can software architectures adequately address?
- How can those problems be resolved most effectively?
- What role should ADLs play in addressing those problems? and
- What is the relationship between requirements, which are in the problem domain, and architecture, which is in the solution domain? A related question is, what is the exact relationship between architecture and design?