EXAMINING THE ROLE OF ITERATION IN THE COMPLETE LIFECYCLE OF A TRACEABILITY-ORIENTED SOFTWARE DEVELOPMENT PROCESS

STUDENT POSTER ABSTRACT

Stefan C. Christov, Borislava I. Simidchieva, Department of Computer Science, State University of New York, College at Brockport, Brockport, NY 14420
Advisor: Dr. Sandeep Mitra, Associate Professor

Software engineering methodologies have received considerable attention in the past few years, due to the ongoing "software crisis" presented in the Standish Group's CHAOS Report. Royce's Waterfall Model correctly identifies the phases of software development (requirements capture, problem analysis, design, implementation, and testing), but it does not facilitate iteration between phases. Over the past several years numerous methodologies have been developed, which attempt to rectify this deficiency (e.g. the Spiral model, the Rational Unified Process (RUP), and Agile methodologies like eXtreme Programming (XP)). XP significantly differs from RUP's prescriptive approach by underemphasizing the role of explicit analysis and design artifacts, and focusing instead on techniques that quickly achieve a "working system." XP requires incremental augmentation of the features of a system--primarily by working at the code level, a.k.a. "refactoring."

A recent IEEE Computer (September 2005) article illustrates how iteration is an inevitable aspect of software development and talks about the different kinds of "rework." Such rework requires refactoring at the implementation level. However, the article (like most current methodologies) does not discuss techniques for embracing iteration during earlier phases, focusing only on artifacts created during these phases.

This poster will present a case study of a Journal Maintenance System that has a database at the backend and a graphical user interface (GUI) at the front end. The case study was developed using the Software Engineering Effectiveness Model (SEEM) presented at CCSCNE 2005 by Dr. Sandeep Mitra, who won the Best Paper award. SEEM emphasizes traceability, and presents explicit, easily-understandable guidelines and techniques that enable the systematic mapping from technology-independent early
phase models to *technology-specific* later phase models, ensuring precision and unambiguity. These techniques include *design patterns* and *coding idioms*.

The poster illustrates how SEEM was used to develop the analysis model of a particular use case of the case study. The case study in this poster shows how SEEM allows (and even encourages) easy and productive rework even at the upper layers of abstraction. Using SEEM results in high-quality analysis, design and implementation models, with easily discernable connections between them. Moreover, thanks to "early iteration", implementation models (i.e., working code) are of a higher quality right from the beginning. "Evolutionary" rework (i.e., adding new functionality) is also easier, and can be clearly tracked to determine its "invasive" and "non-invasive" aspects.

---

**PARALLEL ALGORITHMS ON CONNEX-LIKE MACHINES**

**STUDENT POSTER ABSTRACT**

*Alejandro Giacometti*

*Department of Computer Science*

*Saint Anselm College*

*Manchester, NH, 03102*

*603 – 264 – 4123*

*agiacome@anselm.edu*

*Advisor: Assistant Professor Mihaela Malita*

Complex problems like image processing function approximation, pattern recognition, weather prediction and neural network applications require computationally heavy array manipulation. The main tailback when handling arrays is their size. Many of these practical applications tackle this problem by making use of parallel programming. The power of multiple single instruction processing units is very advantageous.

The CONNEX ARCHITECTURE is a new type of Single Instruction Multiple Data (SIMD) parallel architecture. It consists of thousands of processors that compute single instructions in parallel. The first CONNEX chip - CA1024 - is under development by a *start-up company in Silicon Valley: Connex Technology* (http://www.connextechnology.com).

---

* Copyright © 2006 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.