WCRE General Chair's Message**:**

On the Meeting of Software Architecture and Reverse Engineering

Elliot Chikofsky, WCRE 2005 General Chair
Engineering Management & Integration Inc.
& Northeastern University (Boston)

This year, the 12th Working Conference on Reverse Engineering (WCRE) is meeting in conjunction with the 5th IEEE/IFIP Working Conference on Software Architecture (WICSA) in Pittsburgh at Carnegie Mellon University. However, that linking phrase "in conjunction with" does not do justice to the vision, discussions, creativity, and effort that have brought us here.

In its simplest form, putting together two conferences can be limited to booking them into the same hotel on the same week and hoping for some interaction. But, in the case of WCRE and WICSA, that would hardly have met the need. Our goal is to get these two software engineering communities:

- **to interact,**
- **to learn each other's points of view,**
- **to understand how the forward and reverse views of architecture could best complement each other in the future, and**
- **to discover many new areas where collaborative projects could advance both fields.**

To this end, the programs of WICSA and WCRE have been integrated to encourage joint discussions and communication. After the two conferences conducted separate selections of papers and developed their own program plans, the Chairs put those plans together by adjusting the sequence and the juxtaposition of topics to maximize value for attendees, to have clear session choices by people's topics of interest, and to encourage strong and meaningful interaction between the communities.

Special thanks go to Robert Nord of Software Engineering Institute (WICSA General Chair); Nenad Medvidovic of University of Southern California and Rene Krikhaar of Vrije Universiteit Amsterdam (WICSA Program Chairs); Andrea de Lucia of Universita' di Salerno and Susan Sim of University of California Irvine (WCRE Program Chairs), as well as the program committees and other chairs of both conferences, for their dedication, their vision, and their flexibility. Their hard work has led to an integrated program that will allow us to make a valuable joint contribution to advancing software and systems engineering, while being true to the purposes, ideals, and standards of both the WCRE and WICSA working conferences.

Why these two conferences?

WCRE continues to be the premier research conference on the theory and practice of recovering information from existing software and systems. WCRE explores innovative methods of extracting the many kinds of information that can be recovered from software, software engineering documents, and systems artifacts, and examines innovative ways of using this information in system renovation and program understanding. WCRE is truly a "working conference" and a joint meeting place of industrial and academic researchers. Unlike most conferences, where paper speeches and talking heads at the front of the room provide little opportunity for interaction, WCRE is ***focused*** on discussion and interchange. Paper presentations are strictly limited to 20 minutes, and we've found that WCRE participants often communicate much more solid information in 20 minutes than speakers in other meetings do in an hour. After each set of papers on related topics, there is an open discussion on the topic area, often comparing and contrasting findings, considering related areas of exploration, or suggesting new research approaches.

Our field advances best by interaction. Out of previous WCREs have come cooperative research projects and new industry-academic relationships. Joint research efforts have been formed and people who did not know they were working along similar lines are now sharing expertise and insight. Even in an era of tight travel budgets and extra selectivity in conference attendance, WCRE can be readily justified by what you can get out of it. Papers presented at WCRE have been among the most-cited reverse engineering and reengineering sources, but even more valuable is WCRE's technical interchange among leaders of the field from all over the world -- something you miss if you just read the proceedings after the fact.

WICSA has a similar role, in its part of the software engineering community, as the leading gathering of software architecture researchers and practitioners. This year marks the 10th anniversaries of the first ISAW workshop (Seattle 1995) and of the software architectures theme issues of both IEEE Software magazine and IEEE Transactions on Software Engineering. At this 5th WICSA, the community will reflect on the past decade of software architecture work and identify and discuss the key challenges facing software architects and the research agenda for the field. WICSA includes topical working sessions, where attendees explore and consider emerging techniques, challenges and opportunities, practical experience and new directions, aided by web-based Wiki technology for collaboration among the participants. This year, WICSA will publish its work as a post-proceedings, incorporating both its working session results and its interaction with WCRE.

Architecture - Forward and Reverse

WCRE 2005's theme, "Recovering and Reclaiming Architecture", highlights the role of reverse engineering in learning about the reality of architecture in practice in our systems. Having software architecture and reverse engineering meet is not new. In fact, it has been a fundamental concept over the last 15 years.

In the original reverse engineering and reengineering taxonomy project[i] (1988-1990), James Cross of Auburn University and I found that an acceptable definition of reverse engineering required context. It was not sufficient to identify reverse engineering and its related terms. They had to be placed in a framework that was based on an adequate definition of "forward engineering". The adjective "forward" was needed to distinguish the kinds of creative problem solving, design, architectural, and build activities of the typical development process from the analysis and discovery activities of reverse engineering.

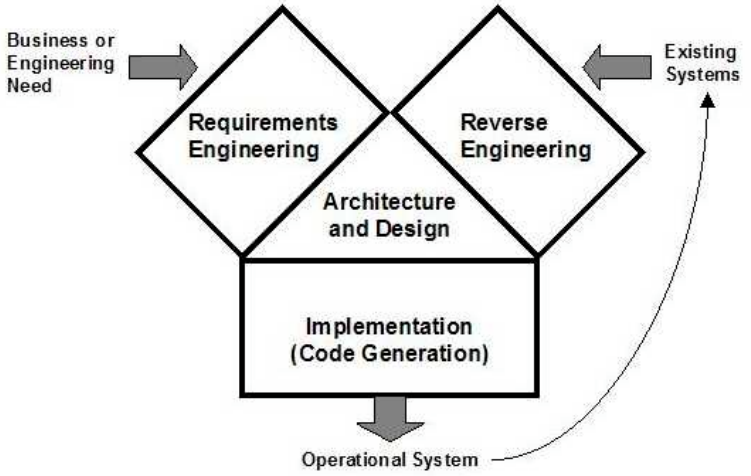

Figure A

In the early 1990s at Index Technology, I used a model like the one in Figure A to explain the how engineering and development processes were related and how they were supported through the Excelerator products and similar repository-based CASE environments. Architecture and design information provides the connecting link between the business or engineering requirements and the system implementation.

As shown in Figure B, the left side of the model is the traditional forward engineering path. The automation of architecture frameworks and design activities through integrated tools, a support environment (IDE or CASE), and a comprehensive repository, enables an automated software development process. When combined with automated approaches to both code generation and system testing, this path yields a powerful and very versatile capability to field a solution from requirements through to the build and deploy activities. This was the mainstay of the CASE movement from the middle of the 1980s through the middle of the 1990s.

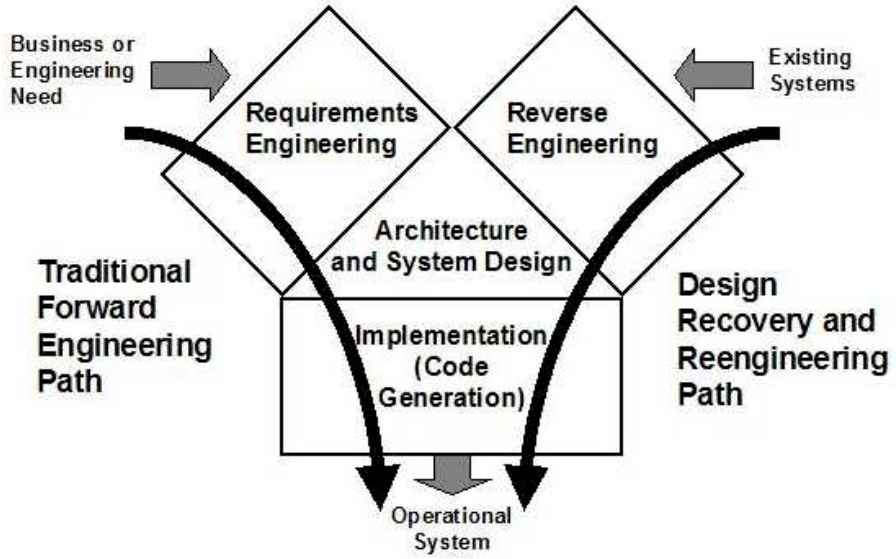

Figure B

The right side of Figure B identifies the new path opened by the addition of practical reverse engineering capabilities into CASE environments around 1990. Artifacts and design information from existing systems could be captured and reflected as architecture and system design models stored in the repository. Analysis tools could then be applied to detect flaws and issues of existing systems and assist in planning their modification. This "Design Recovery and Reengineering" path enables the automation of software and systems reengineering through the same architecture and design tools used in forward engineering. This is true to the 1990 taxonomy definition of reengineering -- some degree of reverse engineering to a more abstract level of design, followed by some amount of forward engineering to yield the new system result. This allows us to detect flaws and issues in existing systems and make changes to improve or replace parts of the whole as necessary. This is the reengineering approach, enabled by reverse engineering and CASE environments, that carried us through the 1990s and was so critical in the IT industry meeting and besting the challenge of the Year 2000 software crisis.

But, this model also has a third path, as illustrated in Figure C. It is this path which should be a focus of discussion as the WICSA and WCRE communities meet together at Pittsburgh. Forward engineering yields the designs and plans of the software or system architecture. Reverse engineering can tell us what the architecture of the existing system really is. How well do these match? What will a gap analysis reveal? Does the reality of the running system meet the architectural vision of the framework it was

intended to follow? Did the practical issues of implementation force compromises that the planned architecture did not expect? We finally have the tools and techniques that enable us to do this kind of cross comparison between architectural intent and architectural execution. What are we likely to find?

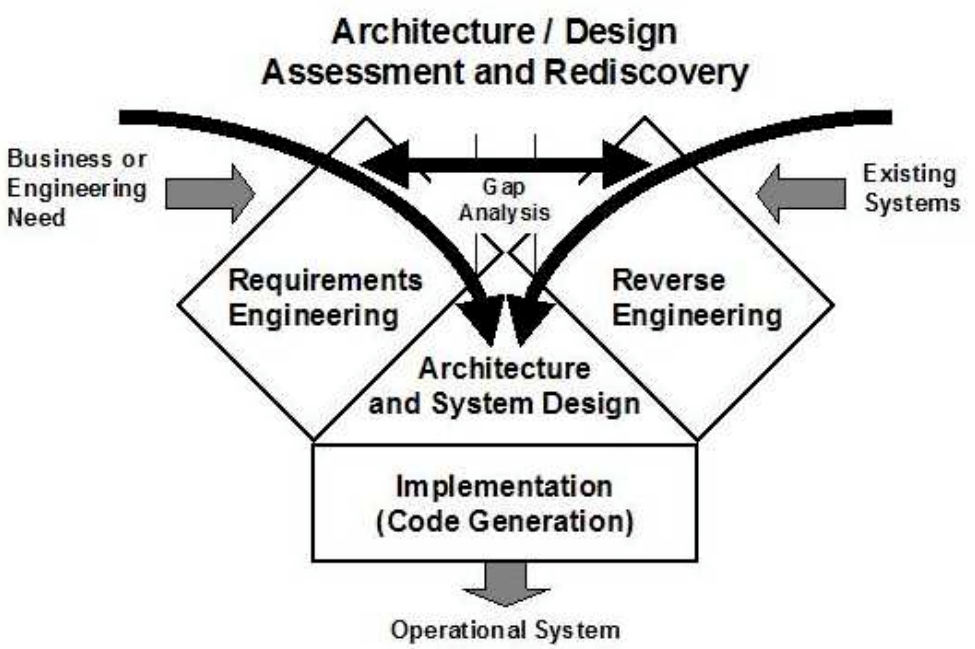

Figure C

I predict that we will learn a lot about software architecture that we didn't know we did not know. In database design, we learn that the as-built database usually has many paths and secondary access opportunities that were not the designer's original intent. These additional physical paths arise naturally as an indirect result of the forward engineering design processes we use. Making use of these paths often allows us to capitalize on reporting and analysis opportunities that can enrich and enhance the database's value to its users. I expect no less from the architecture information recovered from existing systems.

These are topics that we should consider as the software architecture community members of 2005 meets with their counterparts in the reverse engineering community of 2005. The discussions should open some new vistas for us all and, at the same time, renew the opportunities and vision that were enabled by the CASE and reengineering communities of the 1970s, 80s, and 90s.

[i] Chikofsky and Cross (1990), "Reverse Engineering and Design Recovery: A Taxonomy", IEEE Software magazine, January 1990.