Technology Transition and Safety-Critical Software Research

A. J. Kornecki, D. Gluch, R. Seker, Embry-Riddle Aeronautical University, Daytona Beach, FL
N. Kameli, CRM Guidant Corporation, St. Paul, MN

Keywords: software engineering, industry-academia collaboration, model-based development, tools

Abstract

This paper describes a six-year partnership between an academic department and a leading maker of software-dependent medical devices. Central to the collaboration is a campus research laboratory, sponsored by the industrial partner. The laboratory is a venue for software engineering graduate students, under faculty mentorship, to engage in applied and technology transfer-oriented research on safety-critical software technologies and practices. It provides a real-world learning environment that complements and enriches classroom experiences. The industry sponsor has direct access to the detailed results of the research and to well-prepared graduates who know their organization's technology, engineering practices, challenges, and culture. A hallmark of the laboratory is student teams working under conditions that reflect a real-world industry environment (structured processes, schedules, presentations, and professional work products). The investigations focus on the implications of model-based techniques in the software development process for safety-critical real-time systems. The applications of an information-theoretical methodology for discriminating between different architectural models and a study of design tools with automatic code generation functionality are presented. A principal objective of this work is an assessment of the role and applicability of these techniques, including the implications of their integration into an existing software development methodology and culture.

Introduction

Industry and academia can cooperate in a variety ways. An Industrial Advisory Board, composed of managers and engineers representing industries associated with a specific academic department often provides feedback about an academic program's direction as well as opportunities to explore joint projects, co-ops, internships, and employment opportunities for students and faculty. Occasional short-term projects tailored to the immediate needs of industry, allow faculty and students to become more familiar with the domain and contribute to an industrial partner's goals. Student co-ops and faculty internships facilitate understanding of industry needs. This enhanced understanding can be used to make curriculum and program adjustments and help ensure an academic program's currency and relevancy to the discipline [1].

Over the long-term, a project-based model of collaboration has its drawbacks due to the sporadic and deadline-driven nature of the relationships, which are difficult to manage in an academic setting. Timing of projects and their deliverables became major complications in managing resources within an academic environment. In most cases when a collaborating department has available resources, industry partners do not have an appropriate project or available funds. Similarly, when an industry partner has a prospective project, the faculty has been assigned to other academic responsibilities and do not have the time to devote to a new project.

There are multiple objectives and organizational structures for academic research. They can be partitioned into three (not necessarily disjoint) categories based upon their principal objectives as follows:

- Synergy Projects – These projects align with the goals of and contribute directly to the department and the Master of Software Engineering (MSE) program. The project outcomes provide value not only to a specific organization but also to the broad community (e.g. publications, conference presentations, workshops, etc.). There is no commercial product developed that requires long-term support.
- Customer Specific Research – These involve work that is aligned with specific customer interests and needs. They have a research component. Deliverables do not become part of a customer's product or development suites. For example, they are research investigation reports, feasibility prototypes, etc. The principal value in the effort is the knowledge and insights that are gained.

- Development Projects – These involve the development of a product (e.g. software, a process) according to schedules and contractual commitments. Deliverables in these efforts are ultimately used within the operations of the client or are part of a product that is sold or distributed by the client.

Long-Term Partnership Model

In mid-90's the Department of Computer and Software Engineering (CpSE) established a Software Center – a virtual organization reporting to the Department Chair dedicated to working with industry, government and other academic organizations in the area of software engineering research. The Software Center established a solid track record of industry sponsored small projects (Motorola, Lockheed Martin, Boeing, Volusia County). The primary purpose of the Software Center was to establish a place where students could apply good software engineering practices, under faculty supervision, on industry funded, real-world projects. The ultimate goal was to set-up a long-term collaboration with industry to allow for continued funding and enable effective long term planning and support for graduate students.

In 1997 the Cardiac Rhythm Management (CRM) of the Guidant Corporation with its main office located in St. Paul, MN, approached the ERAU CpSE Department to establish a long-term partnership, under the guidelines of the Software Center. Guidant is a leader in the design and development of cardiovascular medical products. These devices include critically important embedded software.

The partnership agreement that was established between the organizations provided the Department of Computing with dedicated testing equipment and heart simulator, pacemaker, and testing software in addition to continuing financial support [2]. The funding has been used to support students and faculty working on research projects expanding knowledge in the area of safety critical software architecture, model based verification, and automated code generation. The laboratory operations are:

- Problem Oriented - result driven investigations of contemporary software technology issues with a focus on advancing the state of the art and practice, especially for safety-critical software systems
- Student Oriented - learning experiences for graduate students in investigating contemporary software technology issues with a focus on knowledge exchange and technology transition

Broad Research Areas

Dependability covers diverse aspects of a system's software and computer hardware including reliability, availability, safety, and security. The faculty and students of the Department of Computer and Software Engineering have been investigating problems involved in the development and operation of high dependability software-intensive systems [3]. They have been researching engineering techniques, tools, and practices for ensuring confidence that these systems will deliver the services demanded of them. The four focus areas of research are: safety, model-based engineering, quality, and component-based engineering.

These dependability areas include specialized software engineering research and skill development with an emphasis on medical and aerospace domains. The medical areas have included work with the work with the Guidant Corporation mentioned earlier. In addition, ERAU has a history of involvement with industry and government agencies in the analysis and synthesis of safety related issues in aviation (including operation, flight safety, aircraft safety design), including research for the FAA, and a proven ability to work with and deliver value to industrial partners.

The core team is comprised of faculty members with strong software and computer engineering credentials and industrial research and development experience. While the current focus is on aviation/aerospace and medical industries, it is recognized that similar challenges in high dependability and safety-critical systems exist in a diverse variety of other industries, such as transportation and nuclear power plants. In addition, ensuring high dependability involves every element of an operational system (software, hardware, materials, physical behavior and properties, etc.). Consequently, part of the research is to generalize results and address multi-disciplinary aspects. These broadened research topics include the following:

- Proper techniques and tools in high dependability systems development
- Appropriate quality models in development of system components
- Systems/Software engineering quality models
- Disciplined practices (e.g. Personal and Team Software Processes (PSP/TSP) and other best practices)
- Techniques and tools for problem or defect prevention
- Techniques and tools for early problem and defect identification
- Techniques and tools for reduction in the cost of quality while maintaining or improving product quality
- Improvement in product development cycle efficiency and effectiveness
- Quality improvement in system/software processes

Lab Operations

The structure of the Guidant lab work involves elements of Personal and Team Software Process (PSP/TSP) that are used for planning, tracking, and estimation [4]. The organization of the laboratory includes a number of defined roles for students and faculty: Program Lead, Faculty Mentor, Student Researcher, and Lab Administrator. The responsibilities for these roles are described below.

Program Lead:

The principal role of the Program Lead is to assure the continuity of research and alignment with the Program Vision and Mission and to maintain the operational framework of the lab. The specific tasks include:

- Selection of the Lab personnel (in a coordination with the Department of Computer and Software Engineering graduate faculty and CRM/Guidant representatives)
- Supervision of everyday activities of Lab personnel and track the progress of the work
- Development of the Operational Plan in collaboration with the sponsor on yearly basis
- Participation in administrative team meetings and selected technical meetings and presentations.
- Serving as Faculty Mentor to assist students in their research
- Managing Lab resources in coordination with the Department Computer and Software Engineering Chair
- Reporting and maintaining contact with the Sponsor’s Technical Liaison
- Helping to select candidates for CRM/Guidant summer internship and employment interviews
- Coordinating visits between the CRM/Guidant and ERAU
- Identifying additional projects and sabbatical opportunities at the Sponsor’s site for the faculty
- Evaluating the quality of research results and performance of the personnel
- Presenting and promoting the Program initiatives inside and outside the university

Faculty Mentor:

Supervision and guidance of the students in research is the principal responsibility of the Faculty Mentor, in addition to:

- Developing the specific research tasks proposals contributing to the Operational Plan in a coordination with the Program Lead
- Participating in technical project meetings
- Participating in Sponsor’s visits
- Contributing to and supervising creation of the deliverables as specified in the Operational Plan

Student Researcher:

Students work on assigned topics and record their effort, product defects, and engineering observations. The work mimics a software engineering organization’s operational environment with planning, reviews, tracking, etc. The specific tasks include:

- Preparing a detailed plan for research activities
- Actively participating in technical and administrative project meetings
- Producing deliverables as specified in the research task plan and present the research results
- Coordinate Lab operations with Lab Administrators

Lab Administrator

In addition to conducting research the students are owners of the Lab and as such they share the following support functions:

- Computing Equipment Administration (accounts, passwords, software installations, licensing, coordination with Information technology department, backups, connections, printers, scanners, etc)
- Lab Operations (maintaining meeting schedules, keeping track of printed documents and books, phones, coordinating with cleaning personnel, non-disclosure agreements, assigning space, resolving conflicts, helping with preparation of our sponsor visits)
- Web Administration (website maintenance, maintaining electronic copies of project documents and procedures, collecting all students effort data, maintaining public relations)

Each Faculty Mentor and their student researchers constitute a team to wo rk on a specific project. Each team keeps regular scheduled hours for work, discussion, technical meetings, consultation, and feedback. Each team is responsible for research outcomes resulting in a presentation and end of term report. On a weekly basis all lab personnel convenes a formal administrative meeting where status reports, progress made, risks, action items, and critical issues are discussed. Occasionally, technical issues are presented and feedback is solicited from other teams. The administrative meeting minutes, reflecting the status of the work are shared with the industry liaison. The partial results and pre-presentation materials are available to team members over an internal secure network. At the end of the term all projects reports are compiled into a semester report shared with the sponsor.

Current Work

It is noteworthy that the European Commission project SafeAir (www.safeair.org), focusing on avionic system development environment, proposes an approach to provide a seamless process path linking the system level modeling tools with automatic code generation tools and rigorous system verification based on formal model checking. These tools are used by the aviation industry and are of interest to safety critical developers in other domain experts seeking to explore the results of avionics research. A lot of work has been accomplished in the area of verification of systems based on synchronous language concepts. European organizations have used this concept to certify aircraft avionics. Recent research includes reports on tools that allow developers to verify the correctness of translations from models to C to assembly. The research attempts to explore these approaches and give student researchers the opportunity to become familiar with concepts that could be very useful in their future work.

As noted earlier, work in the Guidant Lab involves research in the general domain of software development for safety critical systems. Specifically, student researchers have been exploring architectural solutions, techniques, and tools supporting a model-based development methodology that embodies an integrated design and analysis paradigm. Integrated analysis and design can facilitate development of “correct by design” solutions. These approaches often use formal (analyzable) representations for the design and incorporate automated code generation. The efforts in pursuing this research direction have been organized into two tasks with dedicated teams responsible for each task, while sharing the information and promoting inter-team communication. We present these two tasks briefly describing their research objective, approach, and results.

Software Architectures for Safety Critical Systems Assessment using Graph Theoretical Analysis.

Research Objective

The objective of research has been to study a formalism extending the Knowledge Centered Architectural Design Process (KCAD), investigated earlier, for assessing safety concerns in software architecture for safety critical systems. The goal of the proposed formalism has been to make an attempt to forecast the performance of the target system, based on its architecture and the heuristic knowledge about the system (due to unavailability of the actual system). The hypothesis is that applying Graph-Theoretical Formalism allows us to determine some of the quality attributes of selected architectural solution. The approach will enable comparison of different architectural solutions for a software system [5, 6].

Research Approach

The proposed analysis technique starts with representing the proposed architectural solutions in terms of strongly connected Control Flow Graphs (CFGs). The arcs in the CFGs are labeled as functions of execution times for the components (or combination of components) hidden along the arcs as well the ancestor node. Figure 1 presents an example of modeling a specific architectural solution as a CFG. The Shannon metric values are then calculated for both of the candidate architectural solutions. When the arcs in the CFGs are labeled to represent connectivity rather than the execution time, the coupling in the architectural solutions can be assessed. The prediction of quality attributes at an architectural level has serious limitations, which do not allow their assessment. The output of this framework is envisioned to help the architect make decisions about the possible performance and coupling characteristic, which may be imposed by different architectural solutions. The appropriate interpretation of the Shannon metrics is possible only by comparing with a set of historical data, which currently do not exist. However, the proposed framework can be utilized to make relative comparisons between different architectural solutions. When the historical data exist, it can provide the upper and lower prediction limits of the metric for a particular quality attribute.

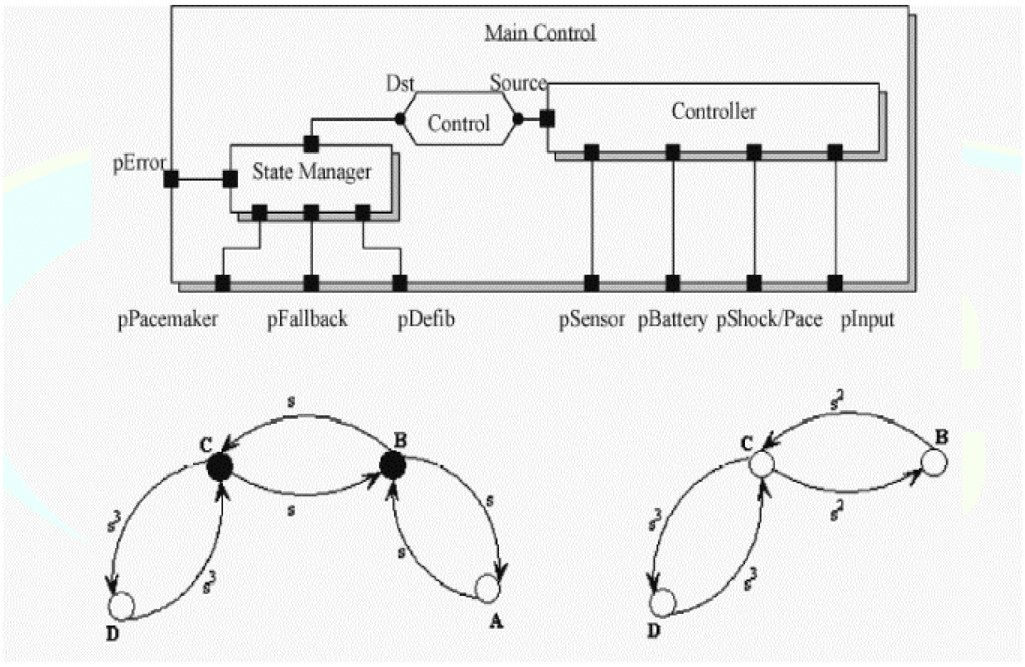

Figure 1 – Graph Theory Representation

The current work included conducting a case study to implement the specific architectural solution collecting the quality attributes data, building the graph model, collecting the metrics, and analyzing the data. Future work on this framework may include applying it to different systems and different architectural styles in order to collect enough data points to establish the "prediction level" for assessment of quality attributes of the system. Another research direction for the future will be to compare this technique with Petri nets and other analysis techniques such as Bayesian networks.

Results

Two architectural solutions, main program/subroutine and microkernel, were proposed to be used by KCAD. Both of the solutions were represented as CFGs with their arcs labeled as functions of module execution times. The proposed framework concluded in favor of using main program subroutine style. The team prototyped both of the solutions in MS Visual C++ and measured the average execution times through multiple runs of both systems. The implementation of these two solutions confirmed what the theoretical framework originally suggested. The results propose possible usability of the graph theoretical framework for aiding the system architect in discriminating between different architectural solutions. Taking this viewpoint, we envision the proposed framework being helpful in making decisions related to modifications to be made at the design phase.

Automated Code Generation in a Model-Based Software Development Paradigm

Research Objective

The main objective of this research is to investigate the feasibility of automated code generation (ACG) techniques and tools for safety-critical software development using case studies. The data compiled throughout the case studies highlight important characteristics of model-based software development methodologies and the automated code generation tools that support them. Specifically, these data relate to the engineering challenges, skills, and effort associated with ACG practices and technologies. The data can be used to determine whether ACG is a viable approach for the creation of safety-critical systems and whether it can provide improved efficiency and effectiveness in design, verification, and validation. The project's case studies involved Model-Based Software Engineering (MBSE) practices that incorporate integrated analysis and design iterations throughout the development process [7,8]. Figure 2 presents the conceptual path of software development based on a a model-based approach. Central to this approach is the nexus of models that is the focus of the design and anal ysis activities.

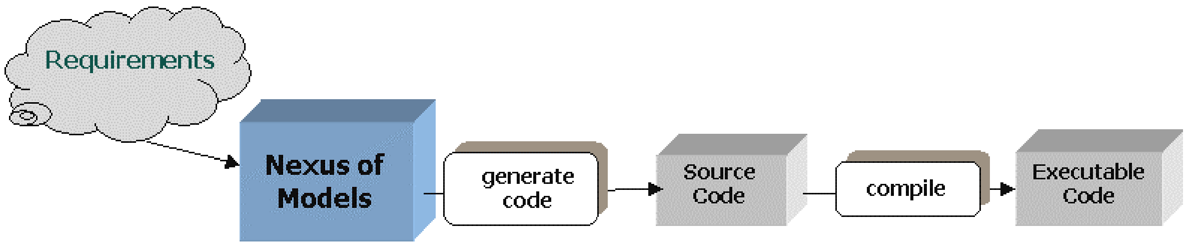

Figure 2 – Model-Based Software Development

Research Approach

Various methodologies and automatic code generation tools, supporting different design approaches, were used in the case studies. Characteristics that identify tools showing promise for safety-critical systems development as well as other factors (e.g. technology foundations, market acceptance) were used in the selection process [9]. The tools selected were Statemate, which uses a structured design approach, Rhapsody, which uses UML 1.4, and Tau, which uses UML 2.0. The research effort was divided into two phases.

The first phase was a learning phase intended to reduce the bias associated with the learning curve of the methods and tools. This phase included a controlled learning experience that involved the development of simple designs using ACG techniques and tools. During this phase, time data and engineering observations were collected. The principal objectives of this phase were to ensure a solid level of competence in the tools and techniques, to gain insight into the learning process by capturing a variety of metrics and observations (e.g. time, unusual or noteworthy difficulties, idiosyncrasies, etc.) that can be used to guide the learning process and support subsequent development efforts, and to define an instrumented study for subsequent phases of the project. The focus was on the efficiencies that may be realized (e.g. lines of code and memory required), the difficulties and limitations (e.g. what percentage of the total code must be manually generated?) in the code generation process, and the issues associated with early analyses of operational and quality attributes of the design. In addition, the impact of systematic design approaches (e.g. architectural styles, views, etc.) on the process was investigated. For example, issues such as whether a specific architectural or design approach is required for a tool and how much can the design vary from these approaches. The issue of integrating new designs with a legacy system was also explored.

In the second phase, each of the researchers continued with the same tool to complete a more complex task: an anti-lock brake controller (a real-time safety-critical system). The research also addressed aspects of a tool that are important in creating an implementation for a safety-critical environment. These aspects included: affects of the tool on the implementation of a common architecture, the trade-offs associated with using one methodology as opposed to another or one tool as opposed to another, the impact of a tool on engineering practices important in safety-critical design, and the tool's support of fault tolerant constructs. Through exploration of such aspects, a repository of data was created.

Results

The results of these efforts included observations of the engineering development process and specific tool outputs. With the tools, the process changed significantly. Originally, a development process similar to those used in hand coding a system was considered. However, an integrated analysis and design approach that is supported by a tool's analysis capabilities can significantly reduce, if not eliminate, phases such as design review and code review. While potentially providing these reductions, a tool significantly constrains a user to a specific process and design approach.

Observations about tool outputs provided insight into some of the challenges associated with ACG techniques. For example, although the implementations were generated from a common architecture, the generated code varied from tool to tool. Among the tools there were differences of more than an order of magnitude in number of lines of code. In addition, each implementation of a subsystem involving interacting concurrent state machines and specified in UML statecharts, exhibited differing behaviors.

Conclusions

The partnership and the on-campus laboratory have provided exceptional opportunities for graduate students engaged in the program. These benefits include financial as well as professional growth experiences. They have published the results of their work and have provided value technical data and results on software engineering practices and technologies to our industry sponsor. In meeting the challenges of the real-world learning environment, graduates have a solid foundation for professional success. The benefits to the university are apparent. The collaboration model is attractive to other industry partners whose interests include high dependability systems and a desire to enhance their software engineering teams through a mutually beneficial industry-academic relationship.

The ongoing work of the laboratory is technology-transition oriented research, especially considering innovative practices and technologies representative of the maturation of software engineering into an engineering discipline. These efforts encompass modeling and analysis approaches particularly relevant to high dependability and safety-critical real-time applications.

Acknowledgement

Over six years many individuals made the program development and operations possible. The authors would like to acknowledge CRM Guidant managers: John Schmidt, Wesley McDermott, and Ramin Mojdeh, ERAU faculty: Iraj Hirmanpour and Soheil Khajenoori, as well as over 25 ERAU graduate software engineering students who over the last six years have participated in the program.

References

1. Hilburn, T., Humphrey, W., "The Impending Changes in Software Education", IEEE Software, September / October 2002
2. Kornecki, A., Khajenoori, S., Gluch, D., Kameli, N., "On a Partnership between Software Industry and Academia", Proceedings of the Conference on Software Engineering Education and Training CSEE&T 2003, March 2003, pp. 60-69
3. Zalewski, J., Ehrenberger, W., Saglietti, F., Gorski, J., Kornecki, A., "Safety of Computer Control Systems: Challenges and Results in Software Development", Annual Reviews in Control, Vol. 27, No. 1, 2003, pp. 23
4. Hilburn, T., Humphrey, W., "Teaching Teamwork", IEEE Software, September/ October 2002
5. Seker, R., Tanik, M. M., "Component-Based Software Modeling Based on Noisy Information Channels," Proceedings of the International Conference on Computer, Communication, and Control Technologies CCCT'03, Orlando, Fl, July/August 2003, pp. 144-149

6. Seker, R., Aktunc, O., Ozaydin, B., Jololian, L., Tanik, M., "Pervasive Shannon Metrics and Component-Based Software," ACIS International Journal of Computer & Information Science, Vol.5, March 2004, pp. 52-62
7. Liu, H., Gluch, D., "A Proposal for Introducing Model Checking into an Undergraduate Software Engineering Curriculum," The Journal of Computing Sciences in Colleges, Vol. 18, No.2, December 2002, 259-270
8. Gluch, D., et al., "Model-Based Verification: An Engineering Practice," CMU/SEI-2002-TR-021, Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, October 2002
9. Crawford, L., Erwin, J., Grimaldi, S., Mitra, S., Kornecki, A., Gluch, D., "A Study of Automatic Code Generation for Safety-Critical Software: Preliminary Report", fast Abstract in Proceedings of 8th IEEE International Symposium on High Assurance Systems Engineering, HASE'04, ISSN 1530-2059, March 2004, pp. 287-288

Biographes:

Andrew J. Kornecki, Ph.D., Professor in the Department of Computer and Software Engineering at Embry-Riddle Aeronautical University, 600 S. Clyde Morris, Daytona Beach, FL 32114. E-mail: andrew.kornecki@erau.edu, (386) 266-6888.

Dr. Kornecki's research and teaching interest include: modeling and simulation, real-time systems, performance analysis, and software safety with the results published in journals and conference proceedings. He served as a Visiting Scientist at the Federal Aviation Administration and National Academy of Sciences Committee on Aging Avionics in Military Aircraft. Recently he has been engaged in FAA sponsored research on testing and certification and assessment of development tools for safety critical real-time systems.

David P. Gluch, Ph.D., Professor in the Department of Computer and Software Engineering at Embry-Riddle Aeronautical University, 600 S. Clyde Morris, Daytona Beach, FL 32114. E-mail: david.gluch@erau.edu (386) 266-6455.

Dr. Gluch's research interests are technologies and practices for model-based software engineering of complex systems, with a focus on high dependability performance-critical systems analysis and verification. Prior to joining Embry-Riddle, he was a senior member of the technical staff at the SEI participating in the development and transition of innovative software engineering practices and technologies. His professional experience includes research and development in fault-tolerant computer, fly-by-wire aircraft control, and Space Shuttle software modeling, resulting in numerous technical reports and professional articles.

Remzi Seker, Ph.D., Assistant Professor in the Department of Computer and Software Engineering at Embry-Riddle Aeronautical University, 600 S. Clyde Morris, Daytona Beach, FL 32114. E-mail: remzi.seker@erau.edu, (386) 226-8658.

Dr. Seker's research interests are component based software, software architectures and metrics, and high assurance systems. Most recently, he has been working on the assessment of software architectures and an FAA sponsored project on safety.

Nader Kameli, MSCS, Manager of Software Engineering & Independent Requirements Verification at Cardiac Rhythm Management, Guidant Corporation, St.Paul, MN. E-mail: nader.kameli@guidant.com, (651) 582-7283.

Mr. Kameli has over twenty years of experience in the software industry participating in product development using full life cycle engineering. He is a member of IEEE and INCOSE. He is on the advisory board of four universities contributing to the development of Software Engineering programs at the undergraduate and graduate level.