On a Partnership between Software Industry and Academia

Andrew J. Kornecki, Soheil Khajenoori, David Gluch
{andrew.kornecki,soheil.khajenoori,david.gluch}@erau.edu
Department of Computing
Embry Riddle Aeronautical University, Daytona Beach, FL
and
Nader Kameli
nader.kameli@guidant.com
CRM Guidant, St.Paul, MN

Abstract

This paper discusses a role for industry in software engineering education, specifically presenting a university-industry partnership between the Cardiac Rhythm Management (CRM) organization at the Guidant Corporation and Embry-Riddle Aeronautical University (ERAU). The focus of the partnership is technology transition. The partnership involves fostering students' professional development, providing students experience solving real-world problems, and exploring modern directions of software engineering. The critical component of the partnership is a student-oriented research laboratory. After discussing the background and history of the project, we focus on the partnership's accomplishments. These include facilitating the transition of graduates from student to employee by developing in them extended software engineering skills and in-depth understanding of the application domain.

1. Introduction

Safety is a system property that assures the system's operation will not endanger human life or adversely impact the environment. Any system controlling energy is an example of safety-critical system. Safety critical software-intensive systems are found in diverse human endeavors, from controlling a nuclear power station to pacing a human heart. Examples of such applications are found in military, aerospace, aviation, transportation, and medical industries. Most of these applications are embedded real-time systems requiring exceptional performance and high reliability. Their development requires rigorous development processes and an intimate knowledge of the application domain.

In most cases industry is interested in application-oriented activities that bring more immediate solutions, help in the implementation of new products, or improve the bottom line of everyday operations. Industry is competing for college graduates with the skills to meet the challenges of developing safety-critical software-intensive systems. It is difficult to find individuals with the requisite expertise and experience in the discipline of software engineering: its lifecycle, artifacts, and processes. Companies focusing on software-intensive safety-critical products understand the importance of having a steady influx of qualified personnel and reliable external source of technology assessment.

One way to assist in these personnel and technology assessment issues is to have a permanent academic partner familiar with the company domain and culture. The selection of such partner may depend on many factors. One of them is the physical proximity. It is much easier to co-operate with the partner located across the street for a steady stream of graduating students. However, most often the selection of partner depends on the academic program focus and qualifications that cannot be found locally. Such factors as the university's technology focus, environment and culture need also to be considered.

One of the ways to bridge the gap between industry and academia and improve the current situation is through close cooperation between academia and industry. This paper describes a program for such cooperation.

2. Past Collaboration Efforts

There are a variety of ways that industry and academia can cooperate. Many academic institutions have installed departmental or other academic unit-wide Industrial Advisory Boards. These groups, consisting of managers and engineers from industries closely associated with the academic organization, are one of the vehicles that provide feedback about an academic program's direction. Other typical forms of cooperation are occasional short-term projects, meeting the current needs of industry. Such projects allow faculty and students to become familiar with the domain and often contribute to the industrial partner's goals. Other vehicles of cooperation are student co-op and faculty internship programs. They facilitate understanding of industry needs and allow the university to make appropriate program corrections. All these approaches can be effective. However, they do not provide a complete solution.

In the mid-90's Embry Riddle Aeronautical University's computing department launched the Software Center. The Software Center was designed to provide a process-centered, measurement driven environment for educating future software engineers. We organized it with shared resources of the university and industry partners. It consisted of faculty, students, and industry collaborators engaged in projects organized to utilize software engineering's best practices. It was a resource center using faculty and students. It was dedicated to addressing selected software engineering problems, providing technology assessments and solutions to participating industrial partners. The Software Center provided feedback for the software engineering curriculum enhancement while providing support and solutions for the industry partners.

The initial concept was to attract a few committed industrial partners to sponsor the Center's operational budget. In three years of operations, the support was associated with a specific project. Even though specific projects brought value to our department by enabling faculty and students to work on realistic industry problems, the sporadic nature of the relationships was 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 the department had available resources, our industry partners did not have an appropriate project or available funds. When our industry partners had a prospective project, the faculty had already been assigned to other academic responsibilities.

3. Overview of the Strategic Cooperation

The specific partnership presented here is a long-term program composed of the following elements:

- Fellowships and other financial sponsorships for students and faculty.
- Ongoing internships for students to spend a summer or selected term at the industrial partner's research and development facilities.

- Faculty and technical personnel exchange programs (the faculty may spend a summer or selected term at the industrial partner's facility and technical personnel from industrial partner may spend time at the university).
- A laboratory in support of the industrial partner's software engineering areas of interest; involving the following two steps:
    - to create an infrastructure at the university in support of specific industry principal application domain (with equipment, documentation, and matching configuration), and
    - to conduct related software engineering research in the areas of mutual interest.
- Transfer of solutions and technologies to the industrial partner.

The Mission statement for the project is*:* "To provide a challenging and rewarding environment for the development of students and faculty, one that enables important contributions to the professional community and the discipline."

The mission is realized in three areas: student development, faculty development, and contribution to the profession. The program consists of specific activities in each of these areas. In the area of student development, we include recruitment focused on the industry partner program, student-centered research, producing quality graduates educated in the practical aspects of the discipline of software engineering with a special emphasis on safety-critical systems, and student publications and presentations. Faculty development is realized by creating and expanding the body of knowledge in the practical application of science and technology to the field of safety-critical systems, providing an incubation environment to jump start junior software engineering faculty's research in the field of safety critical systems, improving teaching strategies and curricula, and improving faculty capabilities in technology exchange between academia and industry. The contribution to the profession includes publications of research findings s.

4. Implementation Details

The collaboration approach presented here is a joint venture between the Cardiac Rhythm Management (CRM) of the Guidant Corporation with its main office located in St.Paul, Minnesota, and Embry Riddle Aeronautical University (ERAU) located in Daytona Beach, FL. Guidant is a leader in development of cardiac devices. ERAU is a leading academic institution in aviation and aerospace education ranked between top engineering schools without doctoral programs according to the recent US News and World Report ranking. The computing programs at ERAU, currently affiliated with the Department of Computing in the College of Engineering, are producing students with bachelor degrees in Computer Science, Computer Engineering and Software Engineering, as well as master degrees in Software Engineering. The department is actively involved in software engineering education contributing to the identification of the software engineering body of knowledge and the related software engineering curricula [1].

The contact started in 1997 when two alumni from our graduate programs got positions with the Guidant/CRM and made significant impression on their management. It appears that the skills and knowledge of the graduates were at the time directly in sync with the company requirements. The appropriate elements of our student education included knowledge of the software technical project management (lifecycle and process management, metrics and measurement, estimation techniques, effort and defect tracking - in addition to the software engineering practices such as requirements engineering, software architecture and design, and a focus on safety critical real-time embedded systems [2]).

Guidant approached the computing department and, after several rounds of negotiations, including on-site visits by the faculty and industry representatives, an agreement was signed. Guidant committed a continuous level of funding to support building the infrastructure and the operations of a dedicated laboratory housed at ERAU. The university provided personnel (the student researchers and faculty mentors) primarily involved in building domain knowledge. Due to the selected topics and the direction of research, the students who worked in the laboratory gained significant experience in the company operation, knowing the products and industrial partner processes. In the long term the work has resulted in building software engineering expertise, with the special focus on safety critical systems in both students and faculty. It is important to note, that as per agreed mission statement, the university does not provide solutions to the industrial partner. The program is focused on providing benefits to the university, particularly to the computing programs and professional community that is interested in engineering software for safety critical systems. Figure 1 illustrates the structure of the Guidant Program at ERAU.

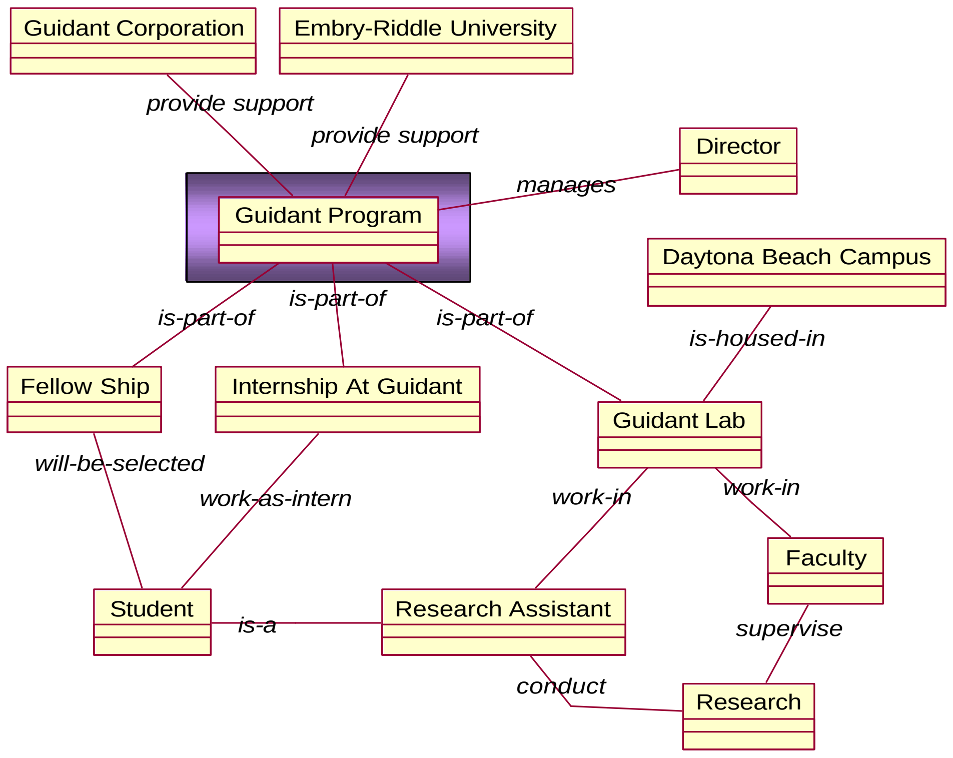

Figure 1

The specific topics addressed during the last three years of the Guidant Program operation were architecture for safety critical systems and model based verification.

The laboratory operations are realized on two orthogonal planes. They are:

- <u>Problem Oriented</u> - result driven investigation of contemporary software technology issues (with focus on advancing state of the art and practice)

- Student Oriented - providing learning experiences for graduate students while investigating contemporary software technology issues (with focus on knowledge exchange and technology transition)

Historically, most of entry-level software engineers are assigned to the tail end activities of software engineering life cycle, such as testing or maintenance. This is primarily to learn the product and gain experience with a company's operations. It is not an easy task to mold a fresh college graduate into the industrial environment. However, entry-level software engineers have a better chance of participating in other life cycle activities when:

- They are familiar with the technical aspects of software engineering project management in addition to other skills such design and coding, and
- They are familiar with the domain, the product concept, and the used hardware/software platforms

The ERAU students meet the first condition. In their graduate software engineering program student focus on practical aspects of software development, using Personal and Team Software Processes proposed by Watts Humphrey [3,4] (http://www.sei.cmu.edu/tsp/). The second condition was made possible by creation of a dedicated laboratory working on real projects. The industrial partner provides software engineering challenges or problems that they have once faced and solved, in a particular way, through their product development process. However, it is critical to note that, even though the research projects focus is provided by the industrial partner, the research team will attempt to solve the problem in a generic form as oppose to being specific to the partners case. This is due to the important mission of the program to provide benefit to the community of safety critical system developers as oppose to any direct benefit to the partner.

The research philosophy is student-centered and explorative. To enhance the learning opportunity, it is the students who carry out the research in the lab. The faculty members serve as mentors by providing guidance, insight, and a conceptual framework for conducting the research, and assessing and providing feedback. Such learning must be marked by strong self-direction, willingness to take risks, and integration of the learning that life teaches outside of academic environment – "a journey of exploration that corrects its course as it proceeds" [5]. In addition, the faculty mentors have advisory roles in overseeing the projects from the perspective of achieving overall goals of the program and managerial liaison with the industry partner. The faculty mentors have control over hiring students. They also define the research direction of the program and provide continuity, building upon the knowledge gained in the previous years. A designated industry representative is the technical liaison with the laboratory serving as an advisor and a required "sanity check".

Embry Riddle submits a three-year proposal indicating overall research direction, required level of funding, and expected outcomes. The industry acceptance of the proposal in principal is indication of a commitment to the continuous level of funding, given that the program progress and achievements are consistent with the objectives of the industrial partner. This process eliminates the need and effort for every year proposal development and provides basis for three year planning. But, continuation of funding is contingent on the assessment and satisfaction of the industrial partner of the progress and the program achievements. At the beginning of each academic year, the ERAU develops and submits a yearly plan of operation. The plan details the research projects, methods, personnel, techniques and deliverables. Once the industrial partner approves the plan, it is executed throughout the year. To track the project progress and interact with the research teams, the industry liaison, often with accompanying technical or managerial representatives, visits the lab at least twice during the academic year, typically at the beginning and the end of the planning cycle. Furthermore, to keep the interaction alive during each year, the academic team participates in the technical seminars organized at the industry site. In addition, to further enhance the relationship, practical knowledge and skills of students and faculty, some of the students spend summer internships at the industry site. And thus far, one faculty has completed and another has just started an additional sabbatical assignment sponsored by the Guidant.

The tasks assigned in the laboratory are executed using software engineering conventions. Each task is planned and a detailed record is kept during task execution. For example, one of the early areas of research was related to the acquisition, retaining, and replicating knowledge in a software engineering domain. The particular application was in the area of software understanding. The Software companies often face the following issue: given the requirement documents and the source code what is the best way to have new software engineer understand the software. By *understanding* we mean to have enough expertise to answer questions about the software operation as well as to be able to propose modifications and improvements.

The logistics of collaboration have included regular visits at the university and the industry sites, planning discussions, regular e-mail exchanges, weekly minutes, teleconferences, presentations on technical meetings, one-to-one contacts between student researchers and industry representatives, etc. These means worked well over the last four years despite the physical distance between Minnesota and Florida.

5. Accomplishments

The first year of the program was geared toward gaining overall knowledge and familiarity with different aspects of software intensive applications in the safety critical systems domain. Guidant equipment, product, processes, problems, and insights where made available to the research team. These served as representative instances of the domain, focusing on the critical elements. For example, one of our early projects involved understanding the implicit and explicit relationship between the requirements for an early generation pacemaker product and its associated test suites. The research team analyzed (reverse engineered) portions of the test suite, and attempted to understand the requirements. This was accomplished in part by contrasting the requirements against the test suites and identifying potential new test cases. We also explored issues of reliability assessment and certification [6].

Subsequent years have allowed us to embark into more substantial activities. We ported one of the testing subsystems to different compiler platform and analyzed possible architectural solutions for testing software. We developed a prototype implementation of software architecture for testing real-time embedded safety critical systems enhancing capabilities of the existing testing software. In the course of work the team did extensive research on identifying software requirements for testing software system to effectively test safety critical devices or products. The first phase of the research resulted in creating software requirements, identifying software architecture for the test system and design documents. Subsequently, to validate the resulted architecture and to ensure that it satisfies the desired properties and attributes the team selected to develop prototype test system and implement and evaluate the architecture in the prototype. Implementation platforms were evaluated and selected. A prototype of the testing software architecture was implemented, using an incremental development process to construct and test each build and then integrate with the previous build. The specific outcomes of the work were clear identification of requirements of testing software and good understanding of the selected architecture. The students gained familiarity and skills with tools and process for software construction and advanced their knowledge about testing software: requirements, architecture, design and implementation.

5.1. Knowledge Patterns

The temporary nature of student employment graduating in two-year cycle required detailed planning and assurance that there are always “old” workers in the lab who could help newcomers with the knowledge of the lab operation and domain. This resulted in the need to explore the techniques and mechanism for knowledge capturing artifacts.

In the process of software understanding we have identified three basic elements. The first element is related to the question *WHAT?* Analysis of the software requirements documentation, coupled with the familiarity of the system domain, allows one to address this question. The second element is related to the question *HOW?* Here, the most useful is the analysis of a source code and the knowledge of the applicable hardware platform. The most difficult is the third element. It is related to question *WHY?* To answer this question one needs to get rather thorough familiarity with both: the application domain and the software.

In the situation with fluctuating a workforce and new student employees, it is imperative to have good documentation. Granted, all mature organizations have an appropriate amount of documentation for young engineers to study. The idea has been, however, to capture various aspects of the work in a standard and easy to understand format. To support the process of understanding we have used the concept of patterns. Patterns originated in the architectural domain [7]. A pattern is an abstraction from a concrete form that may reoccur in a specific context. Patterns constitute a template for problem-solving documentation, which, for a given context and the problem, describe a proven solution. Patterns are particularly applicable for re-use and knowledge retention, since the context and problems reoccur and the workforce changes. The patterns describe practical solutions while may also identify good practice and processes. Having such Knowledge Pattern Library we may provide a base for better understanding the domain, tasks, products, organization, and the company operations. Patterns may help capture domain expertise documenting design/solution decisions and rationale. They convey expert insight to novices reusing wisdom and experience of master practitioners. Additionally, they also constitute shared vocabulary for the discussion. Of course, the patterns must not to be treated as all solving silver bullet.

After analysis of the pattern applicability, we decided to create a template meeting our project needs. Using the presented format we recorded few selected elements of the project in both the organizational and technical domain. Significant work was spent on capturing the software system understanding. The software system we have analyzed was a testing system. The testing process is structured into chunks defined as “features”. To test one of the features it is required a significant knowledge of domain, the product, test station hardware and software. A preparation of the first Knowledge Pattern Template (KPT) for a testing feature was a lengthy task. It has been shown, however, that the subsequent students, learning from the information captured and documented in KPT, were capable to develop not only understanding of the specific feature but also create new templates for completely different features. This process was significantly shorter, compared to our first experiment of understanding the features.

5.2. Software Architecture

In the last two years, one of the main research trust of the lab has been on software architecture. Over the course of last year the team engaged in Knowledge-Based Assessment of Safety Concerns in the Software Architecture. An objective of this activity has been to evaluate the emerging discipline of software architecture for its application in development of safety critical systems. Software architecture is the stepping-stone between the requirements and the design stages. There is an evident contrast between these phases of software development. The *requirements* are concerned with the determination of the information, processing, and the characteristics of that information and processing needed by the user of the system. The *architecture* is concerned with the selection of architectural elements, their interactions, and the constraints on those elements and their interactions necessary to provide a framework in which to satisfy the requirements and serve as a basis for design. The *design* is concerned with the modularization and detailed interfaces of the design elements, their algorithms and procedures, and the data types needed to support the architecture and to satisfy the requirements.

As mentioned previously, the first architecture research project focused on identifying appropriate software architecture for testing safety critical software systems. The tasks conducted by the team included:

- identification and assessment of architectures for software intensive safety critical systems,
- identification of the features and metrics for software architectures (as related to safety needs and requirements for safety critical systems),
- development of an appropriate architecture that meets the requirements of a testing environment for safety critical systems,
- architectural trade-off analysis for safety critical systems, and
- development of a prototype to demonstrate the proposed architecture for testing safety critical systems.

The most recent software architecture research project has concentrated on investigating a mechanism to increase product safety and minimizing the risk of failure. The research has been focusing on identifying techniques and mechanism for more effective deign and assessment of safety critical systems. The long-term goal of this research project is to design and develop a knowledge-centered framework for effective detection and prevention of defect in software architecture, design and implementation of safety critical systems. The architecture research team objectives have been developing tools and techniques for reducing cost of quality while maintaining or improving product field quality. Their goal is to achieve the objectives by focusing on early defect detection and resolution through better assessment techniques and tools and defect prevention through providing insight and tools for architecting and designing software for safety critical systems [8].

5.3 Model-Based Verification

The objective of the Model Based Verification (MBV) effort is to investigate software model checking practices and technologies and to explore their role as part of verification and validation processes. This work is based upon the approaches presented in [9]. The focus of the work is on identifying defects early in the development cycle. In a model-centered approach, models are the basis for automated code generation. Verification and validation techniques are used on the models and the code to assure the quality of the software. The investigation focus has been on acquiring insight into the potential for model checking, both as an autonomous verification practice and as a coordinated activity with testing, to provide greater efficiency and effectiveness in testing processes and increased assurance in the quality of software. It explores the technical factors and practice issues, considering the effectiveness of the techniques, the process implications, and the expertise required to implement the approach.

In the program, we have applied the MBV methodology and a specific tool, the Symbolic Model Verifier SMV, on a portion of the requirements for a pacemaker. This effort also defined a process, used metrics to compile costs, and captured engineering observations and rationale associated with model checking techniques [10]. In this work we evaluated the processes and skills required to use this MBV. We also explored different notations and their applicability in building SMV models as well as issues relating to translating state machine representations into SMV models [11]. These efforts are aimed at facilitating the transition of the processes and technologies into routine software engineering practice.

6. Conclusions

The collaboration has had a positive effect on both partners. The university has a venue for research and financial support for that research. These provide exceptional learning experiences as well as scholarships and stipends for our graduate students. The program also provides incentives for faculty to engage in applied research, publish research results, and contribute to the solution of industry problems. The results of the lab activities are also used in the classroom, where the students present their work and course content is enhanced with insights and examples from the research. These activities have had significant impact on graduate classes in the Master of Software Engineering program including Software Architecture, Software Quality, and Software Safety. The support for this claim can be found in the course evaluations. For example, the data on software architecture course evaluation demonstrate evidence of instructor's increased performance in the term following the research ("excellent" evaluations increased from 12% to 55%)

One of the assumptions for the program was that the laboratory activities must not provide deliverables directly applicable to day-to-day operations of the industrial partner. In addition, the technical reports produced in the laboratory are available and accessible to other organizations directly from the Lab website (http://erau.computing.edu/guidantlab).

While the sponsor provides the technical challenge, the research conducted in the lab and the published results address solutions to the generic problems in safety critical software systems. For other activities like specific projects supporting the sponsor's business operation (e.g. training, reports, travel) we receive separate funding. The direct involvement of students and faculty through summer internships and mutual visits and hiring some of the Lab graduates, are part of the technology transfer component of the program. The industrial partner has ready access to graduates, specifically to evaluate them as potential employees. Over the last three years eight out of 16 students who worked in the Lab have joined Guidant/CRM, and the others have taken employment with companies such as Lockheed Martin, Boeing, and Carrier.

7. References:

[1] D.Bagert, T.Hilburn, G.Hislop, M.Lutz, M.McCracken, S.Mengel, Guidelines for Software Engineering Education, Version 1.0 (CMU/SEI-99-TR-032, ESC-TR-99-002). Pittsburgh, PA, SEI, Carnegie Mellon University, 1999, http://www.sei.cmu.edu/pub/documents/99.reports/pdf/99tr032.pdf

[2] S.Khajenoori, "Process-Oriented Software Education," IEEE Software, No. 6, November 1994.

[3] W. Humphrey, Introduction to the Personal Software Process, Addison Wesley, 1997

[4] W. Humphrey, Introduction to the Team Software Process, Addison Wesley, 2000

[5] P.B.Vaill, Learning As A Way Of Being: Strategies for Survival in a World of Permanent White Water, Jossey-Bass, A Wiley Company, 1996

[6] A.Kornecki, S.Khajenoori, W.Thabet, H.Li, J.Chapman, "Reliability Assessment through Certification Activities", Fast Abstracts and Industry Practices Proceeding of the International Symposium on Software Reliability, ISSRE'99, Boca Raton, FL, November 1999, pp 41-42

[7] C.Alexander, *A Pattern Language: Towns, Buildings, Construction*; Oxford University Press, 1977

[8] Khajenoori, L.Prem, K.Stevens, B.Kang, N.Kameli, "Knowledge Centered Assessment Patterns: An Effective Tool For Assessing Safety Concerns in Software Architecture," accepted for printing, Journal of Systems and Software, 2003

[9] D.Gluch, C.Weinstock, Model-Based Verification: A Technology for Dependable System Upgrade (CMU/SEI-98-TR-009, ADA354756). Pittsburgh, PA, SEI, Carnegie Mellon University, 1998, http://www.sei.cmu.edu/pub/documents/98.reports/pdf/98tr009.pdf

[10] K.Myers, K.Dionne J.Cruz, V.Vijay, S.Dunlap, D.Gluch, "The Practical use of Model Checking in Software Development," Proceedings of the IEEE Southeastcon 2002, April 5 –7, 2002, pp. 21-27.

[11] H.Li, A.Kornecki, D.Gluch, "SM2SMV - A Tool for Facilitating Dependable Software Requirements Analysis Using Model Checking", Fast Abstract in Proceeding of International Conference on Dependable Systems and Network, New York, NY, June 2000, pp B.36-37