

A Constituent-Centered Approach for Curriculum Assessment

Barbara D. Gannod^a, Gerald C. Gannod^a, and Mark R. Henderson^b

^a Division of Computing Studies

^b Dept. of Engineering

Arizona State University at the Polytechnic Campus

7001 E. Williams Field Rd, Mesa, AZ 85212, USA

E-mail: {bgannod, gannod, mark.henderson}@asu.edu

Abstract

The *Stakeholder Assessment and curriculum Gap Elimination* (SAGE) process [2] is meant to provide a methodical and repeatable approach for implementing continuous curriculum improvement. In the Spring of 2004, a curriculum gap assessment team undertook a critical step in the SAGE process. In particular, the team interviewed student interns, industry mentors, and faculty in order to identify gaps between the skills and knowledge that are needed by interns and fresh graduates to develop embedded systems, and what the Bachelor of Science in Engineering in Computer Systems Engineering (BSE program) curriculum provides through classroom instruction. Gaps were identified as being either course gaps, program gaps, or curriculum gaps. In addition, the committee examined the Bachelor of Science in Computer Systems (CST) degree offered by the Division of Computing Studies at ASU's Polytechnic campus. The ultimate goal of the gap analysis was to determine the effectiveness of the curriculum program supported by the Consortium for Embedded Systems and to identify ways that the program can be improved.

Introduction

The Consortium for Embedded Systems (CES) is a consortium consisting of Intel, Motorola, and Arizona State University that was formed to promote the development of a leading program in embedded systems via research and education [1]. The cornerstone of CES is the internship program that has led to over two hundred internships with Intel and Motorola since the inception of CES. As part of the efforts supported by CES, a number of curriculum projects meant to introduce new courses to Bachelor of Science in Engineering (BSE) in Computer Systems Engineering have been funded.

In the Spring of 2004, a CES curriculum gap assessment team interviewed student interns, industry mentors, and faculty in order to identify gaps between the skills and knowledge that are needed by interns and fresh graduates to develop embedded systems, and what the BSE program curriculum provides through classroom instruction [2]. Gaps were identified as being course gaps, program gaps, or curriculum gaps [3]. In addition, the committee examined the Bachelor of Science in Computer Systems (CST) degree offered by the Division of Computing Studies at ASU's Polytechnic campus. The ultimate goal of the gap analysis was to determine the effectiveness of the CES curriculum program and to identify ways that the program can be improved. The activities resulted in a report of approximately one hundred pages in which the committee identified three categories of conclusions and

recommendations that impact courses, degree programs, and curriculum content [4].

From the perspective of ABET, the activities that were performed correspond to retrieving feedback from program constituents. A number of outcomes resulted in performing this assessment including the development of a method for assessing coverage and gaps in desired knowledge within curricula via use of focus groups and identification of criteria determining impact of coverage and gaps upon receiving desired knowledge. This paper describes results of the gap analysis assessment from the viewpoint of content. Specifically, we describe highlights of student and industry mentor needs assessments and the identified coverage and gaps in critical skills for an embedded systems degree program.

Background

An original gap analysis was done in 2000 prior to the formation of the CES [5]. This analysis was done without the benefit of input from student interns experiencing the application of their coursework to real industrial needs. Instead, the original analysis was based solely on interviews with industry practitioners. Since the completion of the initial gap analysis, over 200 CES internships have been completed and significant new course content has been made available by the Computer Science and Engineering Department (CSE) and the Division of Computing Studies (CST) through the CES Curriculum Development Program.

The 2004 gap analysis was designed to focus on three key areas of information. The first goal was to use input from student interns and industrial mentors to identify the key skills required to succeed as an embedded systems intern or new graduate and to map these skills to the existing curriculum offerings to identify quality of coverage and size of gaps in the current curriculum. This information was then used to provide a status report summarizing the progress in delivering new course content to students. Finally, the report was used to deliver a set of recommendations to CES for improving the training of embedded systems students within ASU.

Process

The gap analysis described above is a critical portion of a larger process developed by the assessment team. In particular, the *Stakeholder Assessment and curriculum Gap Elimination* (SAGE) process [2] is based on the engineering design process shown at the top of Figure 1. The purpose of SAGE is to provide a methodical and repeatable approach for implementing the

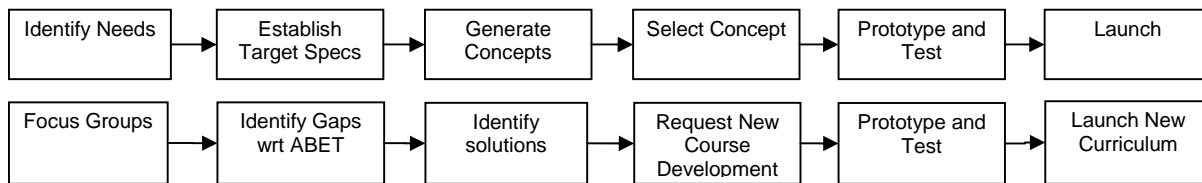


Figure 1 Engineering Design Process (top) and SAGE (bottom)

continuous curriculum improvement framework shown in Figure 1. The steps of the engineering design process are as follows:

1. Identify Needs – a requirements elicitation step whereby needs are identified using one or more of a wide variety of techniques
2. Establish Target Specifications – a requirements analysis step that formally defines identified requirements
3. Generate Concepts – a product conceptualization step that identifies potential solutions based on the target specifications
4. Select Concepts – a product selection step that narrows down potential solutions into those best suited to meet needs
5. Prototype and Test – a product development and testing step
6. Launch – a product release step

While the diagram shown in Figure 1 depicts a linear, waterfall-like process, the various steps can include feedback loops that allow for validation that concepts, products, prototypes, etc. meet needs and specifications identified in earlier stages of the process. Ultimately, the product of the SAGE process is an improved curriculum. We contrast this with the engineering design process, which can involve development of new products rather than improving and evolving existing ones. As such, the SAGE process assumes that a degree program has already been established and that objectives, outcomes, and topics for courses within the curriculum have been defined. Ideally, the program uses a standardized style for documenting course syllabi as would be exemplified by templates used for ABET accreditation.

The SAGE process uses the same engineering design steps in the assessment and improvement of curricula with the following modifications.

1. Focus Groups – requirements for a curriculum are elicited using focus groups made up of constituents from a number of stakeholder groups including students, faculty, and potential employers
2. Identify Gaps – gaps between existing courses and needs elicited from focus groups are identified; these gaps are categorized to identify importance
3. Identify Solutions – recommendations and potential solutions for improving curricula are suggested
4. Request New Course Development or Course Improvement – choose solutions (e.g., new course development or improvement of existing courses) that meet needs identified in Steps 1 and 2
5. Pilot Courses – offer courses in pilot offerings
6. Launch – establish pilot courses as fixtures in the degree program

While it is commonly expected that assessment of a curriculum using the SAGE process will be enacted solely within a department, there is opportunity for having independent assessment teams perform certain aspects of the process. The first

two steps of the SAGE process are steps that can be performed by an independent assessment team while steps 4 – 6 are expected to be enacted by a department. Step 3 is a shared step that must be performed both by an assessment team and the assessed department. This paper focuses primarily on the second step of the process.

Constituents

Figure 2 shows a diagram that depicts the relationships between the SAGE process and its primary constituents. A major contributing factor to the success of the SAGE process is the ability to establish partnerships with three different groups of constituencies:

- *Developers and users of current best practices.* While we believe that the SAGE approach has many benefits, it is important that it be evaluated against current best practices. In addition, the approach that is eventually developed, in order to be effective, should incorporate elements of those best practices. By establishing a relationship with this constituent group, we are able to gain important insight into practices that do and do not work, the context of those practices and their overall potential impacts.
- *Educators.* This group represents the target audience for our approach. While we have access locally to departments undergoing varying levels of reform ranging from new startup departments to mature departments looking to make improvements, it is of interest to contact and establish a strong working relationship with outside departments.
- *Constituents of target audience.* A major aspect of the SAGE process is the engagement of program constituents in order to validate that curriculum content is appropriate with respect to desired student outcomes. We have found that differences between different kinds of program constituents can yield data of varying quality. In order for our approach

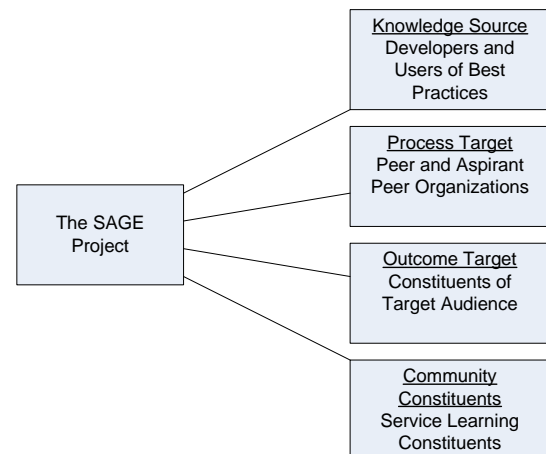


Figure 2 SAGE Constituents

to eventually gain widespread adoption we must develop an understanding of what kinds of assessment data gathering activities provide the most efficient use of constituent time and resources.

- *Community Constituents.* Service learning opportunities have many benefits for both students and the community. Challenges of effective service learning include determining what the appropriate outcomes should be (e.g., validation of the learning objectives of a given project) and how to assess the effectiveness of a given project (e.g., verification of the effectiveness of the project in meeting educational objectives). One of our future activities will be to engage with constituents in the community to study how to assess and evolve programs that incorporate service learning.

The establishment of relationships with the aforementioned groups is critical for further refinement of the SAGE steps. Input and feedback from these groups is used to provide direction in the further development of the SAGE process.

The remainder of this paper discusses how data, insights, and comments from different *constituent* groups, or stakeholders, were used to assess a pair of programs at Arizona State University.

Approach

This section describes the approach that was used to collect and analyze data for the curriculum gap analysis and assessment (step 2 of the SAGE process).

Student Focus Groups

The gap analysis committee conducted two 1.5 hour interview sessions with 5 - 6 member student groups. The procedure was to introduce a goal to the group and then gather information and comments on course topics and skills that were most useful for their internships. They were asked to categorize both technical and non-technical skills that were found useful and to identify specific courses where those skills were learned. They were also asked to identify skills that were not taught in courses that they believed were needed for their internships and to summarize the value of the internship upon their overall educational experience.

In order to ensure equal participation by all students, the Affinity Process was used [6]. The Affinity Process helps organize a large set of items into a smaller set of related items in an environment that allows all participants to feel free to contribute equally.

The results consisted of several grouped post-it notes, each with a student-generated summary header card. The individual post-it note topics will be called secondary items and the header card summary topics will be called primary topics. Multi-voting was used to rank the resulting primary topics.

Mentor and Faculty Focus Groups

Two additional focus groups were convened, one exclusively with intern industry mentors and one exclusively CSE faculty. We invited all participants to identify curriculum outcomes for two groups of engineers: interns and new college graduates. Fifteen mentors and four faculty participated in these sessions.

A mentor focus group was convened with 6 intern mentors from Intel and 9 from Motorola. Instead of the Affinity Process, we asked the mentors the following questions: *What skills are needed for interns to be successful in their internship responsibilities?* and *What skills are needed for new graduates?* Scribes recorded the mentor responses for the mentor focus group. The separate faculty focus group resulted in responses that turned out to be more general than those of the mentors and, therefore,

were more useful in confirming the student and mentor concerns rather than itemizing them for a ranked analysis.

Analysis

After the focus group meetings were completed, the assessment team performed an analysis to determine the gaps between desired knowledge identified by students and mentors and the programs offered by the Department of Computer Science and Engineering and the Division of Computing Studies. The analysis was performed by creating cross-reference tables showing the relationship between student and mentor identified needs and courses in the CSE and CST programs. These tables enabled identification where gaps existed as well as those courses in the programs that contribute to embedded systems desired knowledge. Conversely, these tables also allowed the team to identify those aspects of the curriculum that did not contribute to embedded systems knowledge. Furthermore, by doing the gap analysis in this way, the team was able to identify the impact of current degree programs by examining the percentage of topics covered in core courses versus technical electives.

Gaps

Our study revealed three different target areas for making changes: curriculum as a whole, individual courses, and programs themselves.

Curriculum Gaps. The first type of gap we identified, we call a curriculum gap. In particular, if a topic is determined to be necessary but does not exist in the current curriculum, that topic is identified as a curriculum gap. When the current curriculum does not meet the needs of its constituents, perhaps the most obvious way to address the problem is to design new courses that provide coverage of crucial topics currently absent from the curriculum. For example, our study found that the use of scripting languages was considered a necessary skill for success in the development of embedded systems. Currently, the BSE curriculum does not provide coverage of scripting languages in any courses. To address this curriculum gap, a natural solution is to introduce a course that gives practical experience with several scripting languages.

Course Gaps. Sometimes the introduction of new courses is necessary to provide coverage of important concepts and skills. However, often times the material that is missing is related to courses that already exist in the curriculum. Because technology is so rapidly changing, courses can quickly become "out of date." Thus, a second type of gap, what we call a course gap, exists. In particular, if courses in the current curriculum state that a desired topic is addressed, but constituents report knowledge in the area is not appropriate, the topic is a course gap.

Crucial topics in the desired curriculum can be introduced by replacing outdated material in existing courses and by providing greater depth of coverage of important topics that are currently covered "superficially" in a course. For example, our study showed that although all CSE majors are required to take a junior-level course in computer architecture, interns did not have the desired skills in the area of computer architecture. Upon further examination, we discovered that the gaps were in modern computer architectures and in depth of coverage. Introduction of a new course is unnecessary. Rather, making changes to the existing course will address the root of the problem.

Program Gaps. Updating existing courses to include desired knowledge and skills, and introducing new courses that cover topics previously absent will have no effect if the constraints of the program do not allow students to be exposed to the courses in

which changes occur. The third type of gap, which we term a program gap, addresses this issue. If courses in the current curriculum address a desired topic, but students cannot take the courses due to inflexibility in program constraints, the topic is identified as being a program gap.

The only courses students are guaranteed to take are those required by the program. Courses that are electives may not be taken, especially if the program has little or no room for electives. Many engineering programs are extremely prescriptive. They feel that in order to meet accreditation standards there is little room for flexibility. Consequently, if the program does not require desired knowledge and skills, it is very unlikely that students will be exposed to it. For example, computer networking was identified as a crucial topic for graduates of the CSE program. A general course in computer networks exists, and the CES funded development of a course on network processors. Neither of these courses is required. Students can take only two elective courses (6 credit hours), and students involved in the internship program typically use all elective credits for internships. Thus, many CSE students and virtually all interns do not take the networking courses. The only way to rectify this problem is to make changes to the program requirements themselves. This could take on different forms such as freeing up more hours for technical electives or by making networks a required course.

Making room in an already full program for everything that is desired is extremely challenging. A program may need to make drastic changes in its requirements to eliminate the gaps between what is and what should be. Program gaps certainly present the hardest change to make.

Findings

In this section we present information regarding the data that was collected from student and mentor focus groups. Specifically, this section serves two purposes. First, important topical areas in Embedded Systems are identified. Second, an example of how the data was analyzed in order to drive curriculum improvement is presented. In particular, the data was cross-referenced with CSE and CST courses and then analyzed to determine where gaps exist in the Bachelors in Science and Engineering (BSE) offered in the CSE department and the Bachelors of Science in Computer Systems (CST) offered in the Division of Computing Studies.

Student Needs Assessment

Data was obtained by asking interns to identify (in rank order) those major topics or skills (and refinements of them) that they believed would have most helped them in their internships but were not learned in their course work. As such, a desired program from the student perspective was identified (e.g., student interviews targeted desired knowledge). In order to assess desired knowledge, we cross-referenced the responses received against current CSE and CST program offerings. Furthermore, we cross-referenced the responses against the mentor concerns (see the next section.)

In the interview session, the students identified eighteen major topics. These topics were then ranked according to importance. Those with a high ranking indicate those areas that were of most importance and, from the perspective of the students, required more significant treatment in the curriculum. The top eight topics as well as the percentage of student votes for the topic are as follows: C/C++ and other non-Java Languages (15%); Operating Systems and Linux (14%); Device Drivers (13%); Networking and Network Programming (12%); Hardware (11%); Architecture (9%); Using Linux and Scripting (5%); and Embedded Systems (5%).

A sample of an analysis of one of the topic areas is shown in Table 1. Specifically, the table shows data regarding the area ranked highest by the students that were interviewed: C/C++ and other non-Java languages. The discussions of this topic area centered on the need to provide a stronger foundation for programming in C and C++. Currently, Java is used as the primary educational languages in programs offered in the CSE and CST units. The label on the far left of the table gives the major topic area and its rank votes in parentheses. The rows of the table (enumerated in increasing order starting with "a.") are the minor topic areas or skills that were identified as being both necessary and missing from the BSE program. The first column of this table indicates whether the minor topic area was a concern also identified by industry mentors, while columns two and three indicate courses in the BSE and CST programs that address the topic, respectively. All topic areas were analyzed in the same format. Courses in columns two and three in bold font are required courses and all other courses are technical electives.

Interpretation of the table is as follows. If a topic is covered by a required course (bold face), then the need is being met and a gap exists only in the depth of coverage (a course gap). If coverage of the topic is in a technical elective, the topic is not guaranteed to be taken by students (a program gap). The BSE degree allows only 6 credits of technical electives. Students enrolled in an internship receive 3 credits of technical electives per semester (up to 6 credits of internship can be used on the program of study). The typical case is that students involved in the CES internship program effectively can take no other technical electives. Finally, if no course covers the topic then no opportunity exists for students to be exposed to the topic (a curriculum gap). In the full report, the table is several pages long and addresses each of the topical areas identified by students in a manner similar to that shown in Table 1.

With respect to data specifically found in Table 1, a number of issues were identified. First, C and C++ were identified as the primary languages in which students felt they lacked adequate training. The importance of this topic was reinforced by information received from industry mentors. Note item I.h - Standard Template Library. The item suggests that the topic would be useful as identified by both interns and mentors, but that the topic is not identified as being covered in either program. This constitutes a Curriculum Gap (e.g., the content does not currently exist).

Overall, the topics that are identified by students (and expressed as important by industry mentors) fall under two broad categories: skills necessary to do day to day work and modernization of architectures and technologies used to illustrate foundational concepts. From the standpoint of foundational concepts, both the BSE and CST programs provide coverage while in general, the CST program covers more of the skills necessary to perform day to day work. For example, one of the major topic areas (VII. Using Linux and Scripting) has nearly full coverage in the CST program but nearly no coverage in the BSE program.

With regards to the areas ranked in the top seven, we found better coverage of topics by the CST program in a pair of areas: networking and scripting. We qualify this by stating that coverage does not mean superior depth but rather breadth. Note that the breadth of coverage indicates coverage of both fundamentals and practical skills. By and large this points to the differentiation between the programs that is part of the mission statements of the departments.

Another issue regarding curriculum coverage that was identified in our analysis was in regards to required courses versus technical

Table 1. Excerpt from Student Needs Assessment Table

	Industry Mentor Concern	CSE Program	CST Program
I. C/C++ and other non-Java languages (18)	a. Fundamentals of C/C++ programming	✓	240 , 326 , 494
	b. Pointers in C/C++	✓	240 , 494 RT, 326 , 494
	c. Object Oriented Programming in C++	✓	240 , 326
	d. inheritance	✓	240 , 326
	e. Advanced C/C++ programming	✓	494 RT, 326 , 494
	f. Pointers to functions (C/c++)	✓	494 RT
	g. Understanding C code/low level use of a high level language	✓	494
	h. Standard template library	✓	
	i. Writing to virtual/physical memory in C/C++	✓	494 RT, 494
	j. Visual Basic		326
	k. Writing GUI		

electives. For the most part, most skills that are covered in the BSE program occur in courses that are not required (e.g., technical electives). In general, since technical electives are not required, exposure to the topics covered in them is potentially missed. In many computer engineering programs, availability of technical electives is limited, with the BSE in the CSE department being included. As a result, in most cases it is unlikely that students would acquire, by graduation, the skills identified as necessary by both the student and mentor focus groups.

Mentor Needs Assessment

As mentioned earlier, industry mentors were asked to identify knowledge and skills that they expect interns and graduates from a BSE program to have in order to excel in their work environment.

The data for the mentor needs assessment was collected and organized in a manner different than that of the student data due to the fact that the mentors did not have a strong knowledge of the structure of the BSE degree program. Instead, topical areas were identified by the assessment team after a brief consultation with a CES committee of faculty and industrial mentors. The following eight major areas were identified: Programming Languages; Tools; Technologies; Problem solving, Trouble Shooting, and Debugging; Real time Concepts; Reliability and Fault Tolerance; and Teaming and Communication. The mentors were asked for refinements of each as well as an opinion of whether the skills were appropriate for interns or new BSE graduates. In this way, we could identify which topics were appropriate for lower division or early upper division, and which topics were

appropriate for upper division only. By allowing open discussion via refinement, any issue with biasing identification of topics was mitigated.

Table 2 shows an excerpt of the mentor needs assessment from the full report. The table is structured as follows. Row one identifies the major topic. Row two identifies refinements of the major topic. Rows three and four indicate whether given topics were identified as necessary for interns or new graduates¹, respectively. Finally, rows five and six cross reference courses that provide coverage of the topics.

Interpretation of the table is similar to that of the student needs assessment table in the following sense. If a topic is covered by some required course (bold face), then the need is being met and a gap exists only in the depth of coverage (a course gap). Another potential gap exists if the course is a technical elective and is not taken by a student (a program gap). Specifically, the topic is only guaranteed to be covered if the course covering the topic is required. Otherwise, coverage is not guaranteed (either because the topic is not covered in a core course or because the course is a technical elective, of which students have limited access to due to the constraints of the BSE program).

Table 2 shows the Real-time concepts topic that was discussed by mentors. In this topic area, CSE 494 Real-time Embedded Systems, CSE 430 Operating Systems, and CSE 330 Computer Organization and Architecture achieve coverage for the BSE program. It must be noted that the coverage is primarily achieved in a technical elective and that basic issues of semaphores, pipelining, and branching are covered in required courses. In the CST program, coverage of these concepts is achieved in three required courses: CST 364 Computer Architecture, CST 386 Operating Systems Principles, and CST 420 Foundations of Distributed Web-Based Applications in Java (threads and processes only).

As was the case in the student needs assessment, Table 2 exhibits the characteristic that in the BSE program, coverage is minimally achieved in required courses and the remainder falls in technical electives. At ASU, technical electives have very limited enrollment for BSE majors. The example in Table 2 is consistent with many of the topics in the mentor needs assessment table; while coverage is achieved for some number of topic areas, they are primarily covered by technical electives, which effectively limits exposure to BSE majors. The CST program does a little better in covering topics; additionally, the CST program achieves coverage in required courses, thus assuring exposure to students.

In examining the data regarding whether students should have certain knowledge prior to internships and prior to graduation, it was noted that many courses that provide coverage are at the 400-level. While no concrete conclusion can be drawn, it does provide some motivation for analyzing whether some topics can be moved earlier in the degree programs so as to better prepare interns for internships in Embedded Systems.

Coverage by Current Courses

In addition to analyzing the data on a topic by topic basis, the committee performed a course by course analysis. Content from course offerings in both CSE and CST was compared to the topics identified as important to interns and mentors. The current course content was taken from the official departmental ABET syllabi that are used for accreditation purposes.

¹ Topics needed by interns are assumed to be needed by graduates.

Table 2. Excerpt of Mentor Needs Assessment Table

		<i>Interns</i>	<i>Graduates</i>	<i>CSE Program</i>	<i>CST Program</i>
V. Real-time concepts	a. Latency issues and effects		✓	494 RT	
	b. Soft and hard real-time		✓	494 RT	
	c. Pre-emption and interrupts	✓		430, 494 RT	386
	d. Limitations of real-time		✓	494 RT	386
	e. Threads and processes	✓		445, 494 RT	386, 420
	f. Exception handling & semaphores	✓		430, 494 RT	386
	g. Architecture		✓	494 RT	
	h. Pipelining		✓	330	364
	i. Branching	✓		330	364

CSE Coverage. For the CSE program, we examined each required technical course as well as courses taken as technical electives. For each specific topic listed on a course syllabus, we recorded the following information (in tabular form): whether students learned about the topic in their formal education and found it useful for their internship; whether the topic was identified by interns as needed, but not taught; and whether the industry mentors found the topic to be necessary for interns and graduates of the programs. The course coverage tables can be found in the full report [4].

Analysis of the course coverage tables led to several interesting observations and conclusions. First, in some cases course topics were identified as being useful to students in their internships, and these same topics were not identified as "gaps" by interns or their mentors. This indicates that topics are being covered appropriately, and that these courses are not current targets for change.

On the other hand, some courses were not considered to be useful to students in their internships. Different reasons were identified for this: either the topic was truly not found useful, the topic was a pre-requisite to some other useful topic and thus not considered, or the students never took the course and consequently could not identify the topics as useful or not. If a required course was not identified as useful and faculty and industry mentors did not identify it as necessary for interns or graduates, that course is a candidate for elimination from the program requirements. If course content was not identified as useful but was listed as needed by interns, this indicates that the content is offered in the curriculum but the students are not getting exposure to it (Program Gap). Elimination of less crucial courses can free up space to take critical courses. For example, an advanced course on Programming Languages (a second course beyond a more traditional course in programming language concepts) is required by the CSE program. However, Computer Networks is not

currently a required course. The Programming Languages course was not identified as useful to the students or as being desired by the industry mentors or faculty; whereas Computer Networks was one of the most needed and desired topics by students, mentors, and faculty alike.

A third observation is that in several cases topics were identified as being both useful to students in their internships as well as needed but not taught. This indicated that the general topic exists, to some extent, in the curriculum, but the students found that coverage of the topic was not adequately deep or that it needed updating (Course Gap). If mentors also identified the topics as necessary, it carries a strong recommendation for updating course content addressing those areas. For example, the course on Computer Organization and Architecture was identified as a candidate for updating to include more coverage of modern architectures and technologies.

A final observation is that in a few cases course topics were not identified by students as useful or needed, however those same topics were identified as desired by industry mentors. This occurred for courses available only as technical electives. Students did not identify them as useful because they likely did not take them. Students did not identify them as needed because the particular internship position they held did not make use of them. Leaving these courses as technical electives and not targeting them as program requirements seems appropriate at this time; however, it is desirable to free up more credit hours for technical electives so more students can be exposed to these important topics.

CST Coverage. The primary focus of the gap analysis was on the CSE program because students from that program were the only students accepted for CES internships at the time of the study. As such, the interns interviewed had not taken CST courses and could not comment on their content or applicability. To determine CST coverage, we constructed tables similar to those mentioned for CSE except that instead of identifying whether students found topics in the courses useful, we presented a mapping between CSE courses and equivalent CST courses (many lower division courses are essentially replicated in both departments). In cases where a CST course is equivalent to a CSE course, we assume that if students found the CSE course useful that the CST equivalent would be equally useful. When no CSE equivalent existed, we simply tied course topics to student and mentor stated needs.

Analysis of the CST coverage tables revealed that a few CST courses with no CSE equivalent covered topics identified by students and mentors. For example, CST offers a course on Shell Scripting and Programming with UNIX. Although this course exists, students in the BSE program cannot use the course to fulfill program requirements. In addition, CSE offers courses with no CST equivalent, but CST students are not able to enroll in CSE classes. Finally, some CST courses considered "equivalent" to CSE courses place more emphasis on desired topics. For example, the CST course on programming languages spends about 12 weeks on C/C++ (the student's highest ranked topic), whereas the CSE programming languages course spends only 5 weeks on C/C++. These observations identify target areas where the two programs can work together to provide more complete coverage of the primary areas without having to spend additional resources to duplicate courses.

Gaps

The data presented in the Student Needs Assessment Table, Mentor Needs Assessment Table, and Course Offerings tables were used to determine Embedded Systems gaps.

Table 3 identifies the gaps found in both the CSE (BSE degree) and CST (Embedded Systems concentration) Departments. Topics are listed from most important to least important (as identified by students). A check mark indicates that the Gap specified by the corresponding column exists in the topic specified by the corresponding row. A “†” indicates that we do not have enough data to conclude that the gap exists. This occurs for a single topic: Practical experience with project management. We expect students to acquire such skills during a senior capstone experience. Thus students doing an internship before graduation do not have this experience. Additional information from graduates of the program is needed to determine if the capstone experience is adequately preparing them in this area.

In cases where a topic has a single gap, the target for closing the gap is fairly well-defined. If the gap is a course gap, the associated courses need to be updated. If the gap is a program gap, changes in the program requirements (required courses, number of technical electives) must be made before the gap can be closed. If the topic has a curriculum gap, new content must be introduced via restructuring required courses or through adding new technical elective courses. Note, these changes are not easy to make, but we can target where changes must occur.

In some cases, topics were identified as having multiple gaps. This is due to the fact that there are several approaches for addressing coverage of the topic. For example, C/C++ is identified as having all three types of gaps in the CSE program. This is because the gap could be closed through the pursuit of a variety of avenues. Currently, CSE's Programming Languages course is the only required course with coverage of C/C++. The course provides only a basic 5 week introduction to C/C++ and thus is a course gap. In addition, courses offered in CST (e.g., Embedded C Programming) and some technical elective courses in CSE (e.g., SW/HW Interface for Embedded Systems) require C/C++, but these courses are essentially unavailable to students; CST courses cannot be used on the CSE program of study, and even CSE elective courses have low subscription by target students due to the limited number of technical electives on the program of study. For these reasons, C/C++ was also identified as a program gap. Finally, in order to adequately prepare students for careers in which the majority of their programming may be done in C/C++, a few weeks of exposure and even one entire course may not be adequate (students that plan to do software development in Java typically have no less than 3 semesters of Java experience). Thus, C/C++ has been identified as an Embedded Systems Curriculum Gap as well.

In general, a topic that has both a CSE program and CSE curriculum gap is identified as such because an adequate course covering the topic exists in the CST program but is not available to CSE students due to lack of coordination between the programs. If a topic has a CSE program gap but does not have a CSE curriculum gap, an adequate elective course exists in CSE but is unavailable to students due to the limited number of technical electives on the program of study. If a topic has both a CSE Course and Program Gap, a technical elective course (unavailable to students) exists in the curriculum, but the course needs updating to emphasize more secondary topics of importance.

Table 3. Gap Summary

Topic	CSE Course Gap	CSE Program Gap	CSE Curriculum Gap	CST Course Gap	CST Program Gap	CST Curriculum Gap
C/C++ programming	✓	✓	✓			✓
Practical experience with Linux		✓	✓			
Operating Systems Internals		✓				✓
Device Drivers	✓			✓		✓
General Networking		✓				
Applied Networking	✓	✓				
Advanced Hardware	✓			✓		
Modern Systems Architecture	✓			✓		
Scripting Languages		✓	✓			
Emb. Sys. Design and Development		✓				✓
Practical use of compilers and tools	✓			✓		
Experience with S/W Eng. best practices	✓			✓		
Business aspects			✓			✓
Experience w/ proj. management	†			†		
Information Tech.		✓	✓			
Reliability and Fault Tolerance		✓	✓			✓

Conclusion

We have described the *Stakeholder Assessment and Gap Elimination* (SAGE) process as an approach for continuous curriculum improvement. In addition, we have provided an example of an approach for carrying out the second step of the process, “Identify Gaps,” that was used for curriculum assessment of the CSE and CST departments at Arizona State University. We have categorized the Embedded Systems Gaps in the CSE and CST departments as Course Gaps, Program Gaps, and Curriculum Gaps. We believe that Course Gaps are the least problematic gaps to address (short term) and will have a high impact on interns and graduates of the programs. Thus, the initial focus should be on addressing the Course Gaps.

To date, the majority of the CES curriculum funding has been directed toward Curriculum Gaps. Unfortunately, efforts to close the Curriculum Gaps have had very low impact on BSE undergraduates due to Program Gaps. Until program changes are made, elective courses with desired content will continue to have a low impact on BSE interns and graduates. Finally, efforts should be made to provide a bridge between the CSE and CST

departments to better tailor a degree program that emphasizes embedded systems.

References

- [1] Huey, B., Prital, D., Dannenberg, D., and Robertson, J., "An Arizona Ecosystem for Embedded Systems," *In Proc. Of the 2001 IEEE International Performance, Computing, and Communications Conference*, April 2001, pp. 131-134.
- [2] Gannod, B. D., Gannod, G. C., and Henderson, M. R., "Development and Utilization of a Process for Incorporating Constituent Feedback Into Curriculum Improvement," *In Proceedings of the 2005 American Society for Engineering Education Annual Conference & Exposition*, June 2005.
- [3] Gannod, B. D., Gannod, G. C., and Henderson, M. R., "Course, Program, and Curriculum Gaps: Assessing Curricula for Targeted Change", abstract accepted to the 2005 Frontiers in Education Conference.
- [4] B. Gannod, G. Gannod, M. Henderson, S. Coleman, and L. Karam. Analysis and Recommendations on the State of the CES Curriculum Program: A Curriculum Gap Analysis. Technical report, Arizona State University, June 2004.
- [5] G. Gannod, F. Golshani, S. Panchanathan, J. Robertson, and C. Lipari. Embedded Systems Program at Arizona State University. Technical report, Arizona State University, August 2000.
- [6] J. Kawakita. The original kj method. Technical report, Tokyo: Kawakita Research Institute, 1991.

Author Biographies

BARBARA D. GANNOD is an assistant professor in the Division of Computing Studies at Arizona State University at the Polytechnic Campus. She completed B.S. degrees in Mathematics and Computer Science and a Secondary Education degree at Calvin College in 1992. She received her Ph.D. in Computer Science from Michigan State University in 1997. Her research interests include engineering education and high-performance computing.

GERALD C. GANNOD is an assistant professor in the Division of Computing Studies at Arizona State University at the Polytechnic Campus and an Affiliate Assistant Professor in the Department of Computer Science and Engineering at Arizona State University in Tempe. He received the MS('94) and PhD('98) degrees in Computer Science from Michigan State University. His research interests include software product lines, software reverse engineering, formal methods for software development, software architecture, and software for embedded systems. He is a recipient of a 2002 NSF CAREER Award.

MARK R. HENDERSON is a professor of Industrial Engineering at Arizona State University in Tempe, AZ. He received the Ph.D. in mechanical engineering from Purdue University. Henderson was named a Presidential Young Investigator from 1985-90. He directs the Global Engineering Design Team for undergrads, the Nomadic Design Academy summer study abroad and is also co-director of the InnovationSpace, a transdisciplinary entrepreneurial course for undergrads.