Change the Face of Software Engineering Education: a Field Report from Taiwan

Jonathan Lee[a], Yu Chin Cheng[b]

[a]*Department of Computer Science and Information Engineering, National Central University, Jhongli, Taiwan 320.*
E-Mail: yjlee@selab.csie.ncu.edu.tw
[b]*Department of Computer Science and Information Engineering, National Taipei University of Technology, Taipei, Taiwan 106.*
E-Mail: yccheng@csie.ntut.edu.tw

Abstract

Context: In Taiwan, the supply of software engineers provided by universities has suffered from both a quantity problem and a quality problem. An effort to change the software engineering education is in need.
Objective: The Software Engineering Consortium (SEC) of Taiwan sets its objective to increase the number of college graduates that are better prepared for filling software development and maintenance jobs.
Method: Four dysfunctions: avoidance of process, inattention to modeling, lack of awareness to software quality, and chasm between application domains and software engineering, of the current situation are identified. The effort to correct the dysfunctions involves design of a module-oriented software engineering curriculum, and organization of people, resource, and activities.
Results: In the academic years from 2003 to 2008, both the number of software engineering courses offered and the enrollment size increased significantly by a space of some 250 courses and 5000 enrollments, respectively.
Conclusion: The SEC effort to establishing software engineering modules has been received with enthusiasm by faculty members and students of the participating institutes. Inspired by the important foundational work such as SWEBOK and SE2004, we believe that the adopted strategy of identifying dysfunctions and then designing remedies to address these dysfunctions contributed significantly to the success of the SEC effort.

1. Introduction

Since the 1990s, Taiwan has established its position as a global supplier of Information and Communication Technology (ICT) hardware products [1]. The continuing success has resulted in a severe deviation from the global trend in ICT development. Specifically, the global trend shows a sustained growth of software and service over computer hardware. In 2008, the global ICT spending in software and service is 240% of computer hardware. On the other hand, Taiwan's software and service industry brought in a mere 4.2% of the computer hardware industry in revenue [2]. The deviation is considered as a potential risk by the government and it was decided that the software and service industry has much room to grow. The expected growth creates a strong demand on the supply of software engineers, which can be deemed as a "quantity" problem.

Despite the expected shortage, a recent government report on human resource projected an over-supply of a combination of Information Technology (IT) -related college graduates in the job market [3]. The projected average surplus is over twelve thousand per annum over the time span from 2005 to 2015. Given that graduates of IT-related programs constitute the main source of software engineers in this country, it can simply be concluded that the IT-related programs has failed to adequately prepare the graduates with core competence to work as software engineers, which can be viewed as a "quality" problem.

An initiative is set out for change to the current software engineering education to answer to the requests for more college graduates who are better qualified to work as software engineers. The change is effected in two steps. First, a reference curriculum is developed to increase the coverage of the knowledge areas defined in the Guide to the Software Engineering Body of Knowledge (SWEBOK) [4]. Second, measures are taken to encourage the universities to start new software engineering programs based on the reference curriculum and increase enrollment size.

Since only a change that occurs at the national level can possibly solve the shortage and competence problems and, before describing how the two steps are performed, it is helpful to understand one of the most frequently used mechanisms of effecting changes in Taiwan. Taiwan's success in ICT is due in no small part by a strong engagement of the government [1]. The engagement comes in the form of national-scale projects that involve both government agencies, organizations in the private sector, and universities;

Table 1: Software engineering courses offered in 2003.

course title	universities	courses
Introduction to Software Engineering	55	121
Advanced Software Engineering	4	5
Object-oriented Software Engineering	8	10
Software Quality Management	4	4
Software Engineering Environment	1	1
Software Project Management	8	17
Advanced Software Project Management	1	1
Capability Maturity Model Integration	0	0
Component-based Software Engineering	0	0
Software Architecture	0	0
Personal Software Process	0	0
Testing & Validation	0	0
Software Metrics	0	0
Workflow Software Engineering	0	0
Web Service Software Engineering	0	0
Total	81	159
Offer a single course	63	
Offer module program	5	

the objective is to make Taiwan competitive in per area concerned.

In the sequel, we examine the current situation of software engineering education through a survey of software engineering related courses offered in the higher education institutions in Taiwan in Section 2, propose a remedy to the four dysfunctions identified in the survey in Section 3, asses the effort for change by means of an evaluation model in Section 4, compare other software engineering education efforts in Section 5, and conclude with our future plan.

2. Dysfunctions of Software Engineering Education

We began with an analysis of software engineering courses offered in the academic year 2003 (Table 1). Data are collected from the web site of courses offering listing from all the universities in Taiwan hosted by National Taiwan Normal University.

A total of 150 universities were included in this survey. 63 out of 150 universities offered software engineering related (SE-related) courses, whilst 18 courses – difference between 81 and 63 – were offered as module programs by 5 universities, and a sum of 159 SE-related courses were offered. Two major problems with the software engineering education in Taiwan can be derived from the data in Table 1:

- One is that only some 3180 students – obtained from an estimation of 159 courses multiply 20 enrollments per course (assuming that 20 enrollment in each course) – took SE-related courses in the academic year of 2003, which attributes to the shortage of manpower of software engineers; and

- The other is that most of which, 121 out of 159, were "Introduction to Software Engineering", only 17 courses on "Software Project Management", and a small fraction – 10 courses – on "Object-Oriented Software Engineering", and 4 courses on "Software Quality Management". By scrutinizing these course offerings, we found that only a small chunk of knowledge on software engineering were conveyed in 2003; and that some of the key concepts in software engineering, such as "process" as well as "domain knowledge" were missing, and "software quality" as well as "modeling" were only addressed to a very limited degree, which explains why the majority of IT-related graduates are not well-equipped to work as software engineers.

As can be seen, there were problems related to both quantity and quality as of 2003. It is interesting to note that as early as 1987, an early call for software engineering education actions was made by L. Bernstein in a workshop held in Taiwan. However, the call largely went unheeded and no definitive resource was committed by the government.

As an attempt to alleviate the quantity and quality problems, we conclude the analysis by identifying four dysfunctions of software engineering education: avoidance of process, inattention to modeling, lack of awareness to software quality, and chasm between application domains and software engineering (see Figure 1). These four dysfunctions then serve as a basis for researching into which subjects of courses to be developed to further strengthen both the breadth and depth of software engineering knowledge, and thereby to increase the number of enrollment in software engineering courses.

The problems, analysis, and attributed dysfunctions are derived from the above analysis and experiences in collaborating with industry and government agencies. The experiences include both working on software related projects and serving on numerous software-related reviewing boards.

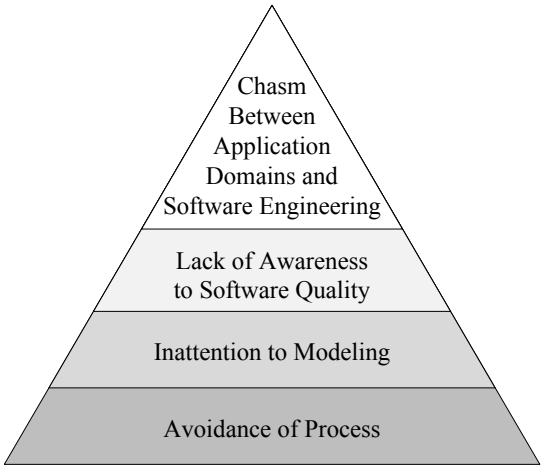


Figure 1: Four dysfunctions of software engineering education.

2.1. Avoidance of process

It is a common phenomenon among ICT product manufacturers that software is developed by individuals or by very small teams. Software usually exists either as an integrated part or as an accessory of the product; it is of small-scale (e.g., firmware and device drivers); reference implementations are available from suppliers and are used with little or no tailoring. Under these circumstances, software developers tend to work independently with little coordination. As a result, software process is often viewed as an overhead, that is, paper work that brings little benefit to justify the cost.

However, as ICT products are getting more integrated and end-user oriented, for instance, in the cases of cell phones and personal data assistants, software becomes too complex to be tackled by individuals in a time-efficient manner. While capable as individuals, ICT software engineers know little to work as a team; thus, individual productivity does not scale well where teamwork is required.

ICT product software development is not alone in suffering from avoidance of process. In recent years, both of the authors participated in reviewing project proposals for government funding to subsidize the implementation of Capability Maturity Model Integration (CMMI) [5] by software organizations. The ultimate goal was for the local companies to obtain a CMMI maturity level rating and become convincing suppliers in the outsourcing market. In reviewing the proposals, it is not uncommon to find that many of

Table 2: A multi-track curriculum for software engineering module.

	Software engineering methodology	**Software process management**	**Software formal method**
Introductory level	• Introduction to software engineering † • Programming with personal software process † • Object-oriented software engineering † • Software testing and verification	• Software quality management †	• Software engineering mathematics
Intermediate level	• Advanced software engineering • Design patterns • Software architecture † • Model-driven architecture	• Introduction to the team software process (TSP) • Capability maturity model integration (CMMI) †	• Formal methods in software engineering † • Advanced probability and statistics for software engineering
Advanced level	• Software development: case study • Agent-based software engineering • Software engineering of workflow programs • Web-based software engineering † • Software engineering for embedded systems • Software/hardware co-design	• Software metrics † • Software project management and economics † • Empirical software engineering	• Software engineering with computational intelligence

† These course materials are published as open courseware in OpenCourseWare Consortium.

the organizations are unable to produce a description of their current software development and maintenance processes. The avoidance of process can lead to the check-list mentality in the preparation for an appraisal [6].

While the dysfunction can have many causes, as far as software engineering education is concerned, the IT-related departments has not been paying enough attention to courses on software process. Indeed, many people have traditionally viewed software development and maintenance as an art performance rather than an engineering practice and emphasized on innovations and inventions over processes and disciplines. A popular view among people equates developing software with writing program code. While this is not incorrect from a programming language point of view, in equating the two the engineering aspects are completely omitted and software development and maintenance easily becomes "hacking" performed by individuals. To help turning things around, software process and process improvement should receive the right amount of attention.

2.2. Inattention to modeling

Reducing software development and maintenance to coding means bypassing artifacts created during the various stages of the process and looking at software at the programming language level. In the software development parlance, this reduction implies (1) using programming languages as the only way to model the target systems; and (2) skipping some or all of requirements models, domain models, design models and test models.

The former usually leads easily to the trap of limiting oneself to the scope and expressiveness power of what a language can provide, and thus is prone to getting bogged down to the bugging and debugging vicious circle, hampering the ability to escalate to a top-down view of the whole system. The latter makes the visualization of the target system nearly impossible since it breaks the layers of abstractions, which are highly effective for reducing a complex problem to a sequence of increasingly formalized models in a manageable way.

2.3. Lack of awareness to software quality

There is a general lack of concern regarding software quality both in software systems and in the development process. Commonly, students confuse testing with debugging. Most of the programs that the students write while in school are only debugged but not tested, and rarely run for more than a few times. There is no time for testing. The debugging-as-testing and

no-time-for-testing mentalities carry over when graduates enter the job market. The symptoms resurface repeatedly, for example, when developers are pressed by a time-to-market deadline to ship the product.

2.4. Chasm between application domains and software engineering

By scanning through the courses offered in 2003 in Table 1, we can find that none of the courses emphasized on bridging the domain knowledge gap to better prepare our students for the application domains they may encounter in their careers as software engineers.

In a software development project, software engineers must effectively communicate with stakeholder such as users, customers and domain experts in order to capture the right requirements, and develop an ability to grasp domain knowledge in a more effective and efficient manner. Courses that provide training in this regard can also be helpful for non-IT students.

Parallel to the four identified dysfunctions in software engineering education in Taiwan is a problem found in many similar settings in which competing for resources and lacking of coordination are viewed as a common phenomena but with a slightly different variation of causes. At the present time, software engineering is viewed as an area of computer science and must compete with other IT areas in terms of student enrollment in classes, faculty quota and teaching resources. It is interesting to note that similar observations were made by a recent survey of master's programs in software engineering [7].

3. A Proposed Remedy

A four-year fund at the national-level is secured to support the effort for solving problem of the shortage of software engineers by correcting the four identified dysfunctions. The three principal components of the effort include a plan to organize people and activities, a focused yet highly adaptable software engineering curriculum schema, and a strategy for allocating resources.

3.1. Organization of people and activities

Currently, there are more than one hundred and fifty higher educational institutions in Taiwan. To maintain a focus on the objectives, to coordinate

the efforts, and to ensure cost-effective resource allocation, the *Software Engineering Consortium at www.sec.org.tw* (SEC) was founded to organize the participating institutes.

The planning and reviewing board is responsible for developing the curriculum schema, setting the resource allocation policies, reviewing project proposals and monitoring project executions. The board members include experienced educators, researchers and practitioners from academia, industry, and government. Resources are allocated to participating institutes through competitive projects. Two types of projects are defined. Institutes applying for the individual course project concentrate on improving the teaching of a specific fundamental course such as Introduction to Software Engineering. Institutes applying for the module curriculum project aim at building a more comprehensive software engineering program adapted from the SEC curriculum schema.

The participating institutes are organized into a number of clusters based on their strengths and regions. In particular, a number of universities with a larger faculty size (e.g., over three faculty members in software engineering) are invited to submit a project proposal to become a *cluster center*. To qualify as a cluster center, an institution either must have established a software engineering module program or is in the process of establishing one. A cluster center is assigned the responsibilities of developing common teaching materials to be shared by consortium members, recruiting cluster partners, and organizing and hosting teaching workshops. In contrast, the cluster partners are institutes that are less well-endowed but show clear commitment to software engineering education. Their responsibilities include offering the basic core courses to their students and participating in consortium and cluster activities.

3.2. A tailorable software engineering courses framework

The SEC favors a curriculum suitable for tailoring into an *elective module program* over a comprehensive undergraduate degree program [8]. An elective module program is defined as a relative compact collection of core courses in a discipline outside of the student's major. An elective module program in the Taiwanese universities comprises anywhere from six to ten 3-credit courses. A student completing the requirements of an elective module program is conferred a certificate at graduation.

An elective module program has a number of advantages. First of all, it can be set up in a relatively short period of time, typically from six to

twelve months. In contrast, the approval process for starting an undergraduate degree program can take two to four years. Secondly, since a module program is more feasible to students outside IT-related departments to elect and can be more effective in helping non-IT majors to cross the domain-and-software-engineering chasm. Thirdly, a module program can be elected by both undergraduate and graduate students, therefore producing more better qualified software engineers. Finally, the module program serves as the interim stage of establishing comprehensive undergraduate and graduate programs in software engineering.

The main disadvantage of an elective module program for software engineering is its limited coverage of the knowledge areas in the SWEBOK. In developing the module-oriented curriculum, materials are drawn from the standard references [9, 4, 10, 11]. The identified subjects are allocated into a software engineering courses matrix with the horizontal dimension of three knowledge areas: software engineering methodology, software process management, and software formal methods; and with the vertical dimension of three levels: introductory, intermediate, and advanced. In Table 2, twenty-four courses populate the nine cells of the matrix, of which ten courses are equipped with courseware materials and can be found in the OpenCourseware Consortium web site under the path of members/affiliate society/Taiwan/SEC.

A detailed account on the selection of the courses is beyond the scope of this paper. The following briefly described course selections to fix the dysfunctions related to process, modeling, and quality.

In order to put the notion of process into perspective, several courses on process and process improvement are included. For example, Programming with Personal Software Process (PSP) [12] is included to create a focus on discipline and quality at the level of the individual programmers. At the intermediate level, Capability Maturity Model Integration (CMMI) [10] is included since it addresses process and quality concerns at the organization level. Note that CMMI has become the *de facto* process improvement standard in industry and government in Taiwan. To provide a prism for viewing into the target systems and analyzing the real world application domains in a higher level, courses covering abstractions such as the notion of objects, agents, and services injected into this curriculum, including Object-Oriented Software Engineering, Design Patterns, Agent-Based Software Engineering, and Web-Based Software Engineering. To raise the awareness of quality, courses such as Software Testing and Verification, Software Quality Manage-

Table 3: A Summary of SE-related Course Offerings from 2003 to 2008.

	2003		2004		2005	
course	**univ.**	**courses**	**univ.**	**courses**	**univ.**	**courses**
Intro. to SE	55	121	70	159	83	223
Advanced SE	4	5	5	5	5	5
OOSE	8	10	9	13	16	27
Software Quality Management	4	4	3	4	9	13
SE Environment	1	1	1	1	1	1
Software Project Management	8	17	11	15	18	37
Adv. Software Project Management	1	1	1	1	2	8
CMMI	0	0	3	5	1	1
Component-based SE	0	0	2	2	1	1
Software Architecture	0	0	0	0	0	0
PSP	0	0	5	7	6	6
Testing & Validation	0	0	1	1	0	0
Software Metrics	0	0	3	4	2	3
Workflow SE	0	0	0	0	0	0
Web Service SE	0	0	0	0	1	1
Total	81	159	114	217	145	326
Offer a single course	63		76		86	
SEC-Sponsored individual course	0		54		45	
Offer module program	5		7		12	
SEC-Sponsored module program	0		0		6	

	2006		2007		2008	
course	**univ.**	**courses**	**univ.**	**courses**	**univ.**	**courses**
Intro. to SE	86	261	95	268	87	274
Advanced SE	5	5	5	5	5	5
OOSE	19	36	19	30	18	26
Software Quality Management	7	13	8	13	11	22
SE Environment	0	0	2	2	2	3
Software Project Management	24	47	21	49	26	48
Adv. Software Project Management	3	6	1	1	1	1
CMMI	1	1	7	9	2	2
Component-based SE	0	0	2	2	1	1
Software Architecture	5	6	4	4	4	4
PSP	9	10	6	7	6	6
Testing & Validation	2	8	2	2	3	6
Software Metrics	2	2	2	2	2	2
Workflow SE	1	1	0	0	1	1
Web Service SE	4	5	3	5	3	5
Total	169	402	177	400	172	406
Offer a single course	88		101		92	
SEC-Sponsored individual course	47		10		0	
Offer module program	20		21		22	
SEC-Sponsored module program	14		10		0	

ment, Software Architecture, and Software Measurement are included in the curriculum.

3.3. Tracks and instantiation examples

The SEC members are given much freedom in tailoring the curriculum to meet their own developmental objectives. To this end, three generic tracks are identified: the *research track*, the *pragmatic track* and the *professional track*.

The research track is intended for research-oriented institutes, where the primary issue is to stay at the frontier of software engineering research, including development of new methods, tools, theoretical studies, etc. The pragmatic track is for institutes emphasizing software development using matured methods and technologies. Notably, the pragmatic track seeks to further enhance a close collaboration between industry and academia that has been cited as one of the most significant reasons behind the success in Taiwan's ICT industry [1]. The professional track serves people already on job. For example, with government subsidy, the Software Engineering Association of Taiwan (SEAT) has run a number of highly successful short courses of software testing based on the corresponding consortium course. The short course is a good example of the collaborated effort among industry, academia, and government for improving software development practices and quality.

4. Evaluation of the Effort for Change

In order to assess the results of the effort for change, we have designed an evaluation model with three criteria: the number of courses offered, the number of universities that offer SE-related courses, and the number of universities that offer two or more SE-related courses to measure the impact of proliferation, especially, on the module programs after the closure of this project. We have applied this model to both the universities sponsored by SEC and to all the universities in Taiwan. Data are collected from the same data source as Table 1.

Table 3 lists the number of courses offered, the number of universities that offer SE-related courses, and the number of universities that offer two or more SE-related courses in the time span from 2003 to 2008. We can see clearly that there is a significant improvement measured by the three criteria:

- the growth of the total number of courses offered has been rising at about 36 percent per year from 2003 to 2006, and then has flatlined at about 400 courses since 2006;

- since 2003, the number of universities that offer SE-related courses was up 46 percent from 61 in 2003 to 92 in 2008; and

- the number of universities that offer two or more SE-related courses tripled from 18 in 2003 to 80 in 2008 which accounts for the dramatic increase of the number of universities offering module programs from 5 to 22.

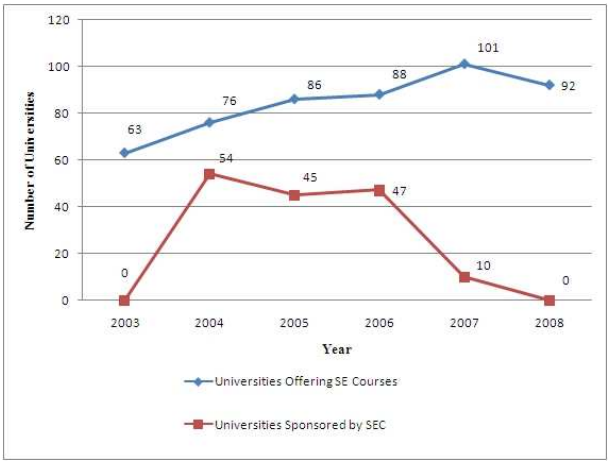
To put into perspective the impact of proliferation of this plan for change (see Table 3), we take the universities sponsored by SEC as a baseline to measure the degree of improvement by means of the total number of universities that offer SE-related courses and the total number of universities that offer module programs.

Since 2003, referring to Figure 2(a), the number of universities that offer SE-related courses was up 46 percent from 61 in 2003 to 92 in 2008; meanwhile, the number of universities sponsored by SEC dropped from the peak of 54 to 10 in the time span from 2004 to 2007.

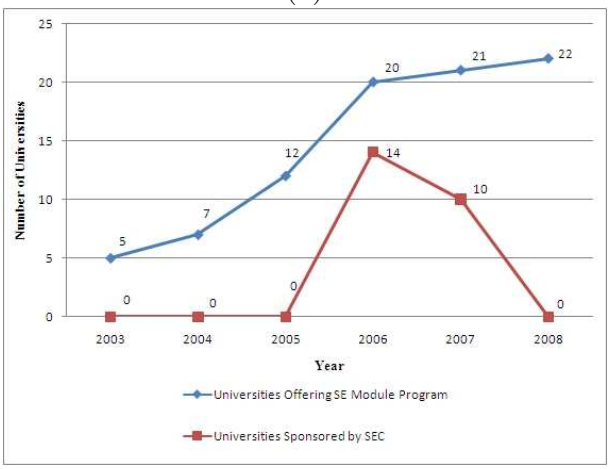
In Figure 2(b), it is clear that more universities, exploding by a factor of nearly five, have developed the capacity from offering single courses to offering a comprehensive module program based on the SEC curriculum models; on the other hand, the universities sponsored by SEC account only for a small fraction of the total share.

5. Other Software Engineering Education Efforts

We have surveyed the recent publications in the CSEET, ICSE education track and other sources. Since the body of literature is vast, we shall focus our scope to three specific software engineering education efforts, the Graduate Software Engineering 2009 (GSwE2009) [13], the Top SE program of Japan [14] and the KAIST-CMU Master of Software Engineering (MSE) program of Korea [15]. As with the SEC effort reported in this paper, all of the surveyed efforts are either directly or indirectly influenced by SWEBOK [4], SE2004 [8], and the SEI graduate curriculum [16]. In addition, the latter two efforts are included for the following reasons: (1) both are in the same geographical



(a)



(b)

Figure 2: Impact of Proliferation

region as Taiwan and traditionally strong in ICT products manufacturing and export; and (2) both programs received government subsidy.

GSwE2009 [13] got started in 2007 using the SEI curriculum [16], SWEBOK [4] and SE2004 [8] as the foundation. GSwE2009 has evolved according to the experiences obtained from 28 MS programs throughout mainly US but with participating schools in Canada and UK as well [7]. GSwE2009 has gained formal sponsorship from The IEEE Computer Society and the Association for Computing Machinery (ACM). The SEC curriculum design, which was started in 2004, shares many commonalities with GSwE2009, including emphasis on an identified core and flexibilities for the adopting institutes to tailor. In continuing the effort of SEC, GSwE2009 constitutes a good source of reference, including the specification of defined capabilities of the graduates and the emphasis on integration with domain (e.g., telecommunications, finance, medical, etc.) with software engineering.

The Top SE program of Japan aims to bridge the gap between academia and industry by educating software professionals and graduate students into what is called the "superarchitect": software professionals with capability to model, apply tools to practical problems, adapt to new technologies and tools, and to promote the technologies and tools [14]. Top SE has produced 87 graduates in its first four years from 2007 to 2010. Note that the program by itself does not offer a degree.

The KAIST-CMU MSE program is a master-degree program jointly offered by KAIST of Korea and Carnegie Mellon University [15]. An applicant to program is expected to have two or more years of experience in industrial software development. The most recent experience report indicated that 47 graduates (out of 60 students initially enrolled) completed the degree program from 2006 to 2009. The vast majority of graduates were able to get good software development jobs in the Korean industry. The curriculum design follows a core-elective-studio-master thesis structure: the student is required to take five core courses, two to three elective courses, one software studio project, and lastly the master thesis. The joint offer provides a unique experience for the students, though it has been indicated that cultural issues should be considered as the graduates will eventually work in Korea.

In comparison, the SEC curriculum focuses on reforming SE curriculum both in undergraduate and graduate levels with the main objective of creating as many SE-ready graduates as possible. Thus, by its nature, the design of the SEC curriculum lacks the kind of Top SE's focus on tools and the application domain, nor can it enforce a coherent set of core courses such as

in the KAIST-CMU MSE program. On the other hand, the SEC curriculum adopting institutions choose their own focus. In particular, it is worth mentioning that many domain-specific education reform programs in Taiwan have begun to incorporate some courses of the SEC curriculum into their program, especially software process, software design and software testing. Programs like Top SE and KAIST-CMU MSE can be seen as an effort towards a small scale focus group in which application domain and use of tool can reap best results. Furthermore, the software studio course in the KAIST-CMU MSE program is obviously a good idea to follow, although finding enough industrial partners/sponsors to provide the project subjects can be a challenge to the SEC consortium members.

In summary, the following items constitute what makes the SEC curriculum unique. (1) The SEC effort seeks to incite software engineering education in Taiwan at the national level, whereas Japans Top SE and Koreas KAIST-CMU MSE program operate at the institutional level. (2) Undergraduate and graduate students without professional experience constitute the main enrollment body. (3) The SEC curriculum emphasizes a module-oriented program with certificate rather than a degree program, which has a low entry barrier for most universities in the SEC to produce enough graduates to satisfy national level demand in a short amount of time. (4) Less and incoherent coverage of SWEBOK; unlike KAIST-CMU and Top SE where core courses are mandatory and fixed, the SEC curriculum allows its members to tailor the program. (5) Like the programs in Japan and Korea, government subsidy was used very differently. In our case, spending the government money wisely becomes a challenge since many universities are involved, which is why we explain in some detail on the organization of people and activities (Section 3.1). We believe the SEC experience could be useful to other regions and countries seeking to revitalize software engineering education.

6. Concluding Remarks

Having just completed the first four-year funding cycle, the SEC effort to establishing software engineering modules has been received with enthusiasm by faculty members and students of the participating institutes. The endeavors have already created a number of new phenomena. Institutes are setting up software engineering modules or offering core software engineering courses and many students are signing up; more faculty positions are open up to people specialized in software engineering; and above all, there is an

increasingly strong sense of community among people teaching, researching, and practicing software engineering.

Current SEC effort has been recognized as a success and further fund appropriation is underway. In the next stage, the objectives are to encourage the infusion of software engineering ingredients into four basic courses offered in computer science, including Introduction to Programming, Object-Oriented Programming, Data Structures, and Algorithms, to explore the possibility of integrating open source software engineering tools with four SE-related subjects: Software Testing, Configuration Management, Project Management, and Issues/Defects Tracking, and to build a platform for initiating a collaboration program with ICT industry sector in Taiwan to co-offer SE-related courses in the university.

References

[1] A. L. Dahl, A. Lopez-Claros, The impact of information and communication technologies on the economic competitiveness and social development of taiwan, Tech. rep., Global Information Technology Report 2005-2006, in Leveraging ICT for Development, edited by S. Dutta, A. Lopez-Claros, I. Mia, Palgrave Macmillan (March 2006).

[2] The 2008 Year Book of the Information Services Industry (In Chinese), III-MIC, III-0453-T702(97), 2008, available at http://www.itis.org.tw/pubinfo-detail.screen?industry=1&ctgy=9&pubid=58471516.

[3] Y.-M. Lou, W.-T. Chao, S.-C. Fan, An Analysis of Taiwans Science and Technology Human Resource Supply and Demand for 2005 2015, Council for Economic Planning and Development, Taiwan, ROC, ISBNG986-00-5419-3, 2006.

[4] SWEBOK Project Web Site, available at http://www.swebok.org.

[5] Software Engineering Institute (SEI) CMMI Web Site, available at http://www.sei.cmu.edu/cmmi/.

[6] D. M. Ahern, J. Armstrong, A. Clouse, J. Ferguson, W. Hayes, K. Nidiffer, CMMI SCAMPI Distilled: Appraisals for Process Improvement, Addison-Wesley, 2005.

[7] A. Pyster, R. Turner, D. Henry, K. Lasfer, L. Bernstein, Master's degrees in software engineering: An analysis of 28 university programs, IEEE Software 26 (5) (2009) 94–101. doi:http://doi.ieeecomputersociety.org/10.1109/MS.2009.133.

[8] Software Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering, 2004, available at http://sites.computer.org/ccse/.

[9] P. Bourque, R. Dupuis, A. Abran, J. W. Moore, L. Tripp, The guide to the software engineering body of knowledge, IEEE Software 16 (6) (1999) 35–44.

[10] D. M. Ahern, A. Clouse, R. Turner, CMMI Distilled: A Practical Introduction to Integrated Process Improvement, Addison-Wesley, 2003.

[11] M. B. Chrissis, M. Konrad, S. Shrum, CMMI : Guidelines for Process Integration and Product Improvement, Addison-Wesley, 2003.

[12] W. S. Humphrey, A Discipline for Software Engineering, Addison-Wesley, 1995.

[13] Graduate Software Engineering 2009: Curriculum Guidelines for Graduate Degree Programs in Software Engineering, available at http://www.gswe2009.org/.

[14] S. Honiden, Y. Tahara, N. Yoshioka, K. Taguchi, H. Washizaki, Top se: Educating superarchitects who can apply software engineering tools to practical development in japan, in: ICSE '07: Proceedings of the 29th international conference on Software Engineering, IEEE Computer Society, Washington, DC, USA, 2007, pp. 708–718. doi:http://dx.doi.org/10.1109/ICSE.2007.89.

[15] S. Kang, I.-Y. Ko, J. Baik, H. Choi, D. Lee, Kaist-cmu mse program - the past and the future, in: CSEET '10: Proceedings of the 2010 23rd IEEE Conference on Software Engineering Education and Training, IEEE Computer Society, Washington, DC, USA, 2010, pp. 49–56. doi:http://dx.doi.org/10.1109/CSEET.2010.24.

[16] G. Ford, SEI report on graduate software engineering education, SEI (CMU/SEI-91-TR-2), 1991, available at http://www.sei.cmu.edu/library/abstracts/reports/91tr002.cfm.