Dimensions of Agile Maturity in CS Education

Dr. Ramadan Muawad[1]
R.Muowad@msn.com
College of Computing & Information Technology. Arab Academy

Dr. Ahmed El-Abbassy[2]
ahmed_elabbassy@yahoo.com
Higher Institution of Computer Science & Information System, Al-Shorouk Academy

Ahmed Gaber[3]
ahmedgaber_cs@hotmail.com
College of Computing & Information Technology, Arab Academy

Summary
Agile software development is becoming a matured, effective approach and has wide acceptance according to the recently published trends. Due to its success, agile practices have moved into other disciplines including Computing Education. Most of the computer science academic programs are currently rigid and use waterfall process model in delivery. Lightweight process framework like Agile is recommended to computer science education in order to improve quality and reacting to changes and industry requirements. This paper discusses and presents a framework to adopting and evaluating agile practices in computer science education.

Key words:
Software Engineering, Agile process, Computer Science Education, Learning Development.

1. Introduction

An important research work has been done in recent years to move software engineering practices into other domain/disciplines and especially into computing education. The rationale for this initiative is: 1) the commonalities between software development and teaching and learning process improvement. 2) the maturity and effectiveness of software engineering best practices.

Teaching and software development have a lot in common. Both are complex activities, both undergo a development life cycle, and we would like both to be of high quality [1, 5]. Figure 1 illustrates a high level correlation between software development and education [2]. In software development, the key Actors are: the programmer/ developer, the customer / client and collaboration result in software. In education, the key actors are: the teacher (acts as programmer in software development), the employer (acts as customer in software development) and collaboration result in qualified graduate/student (Student acts as software service/product).

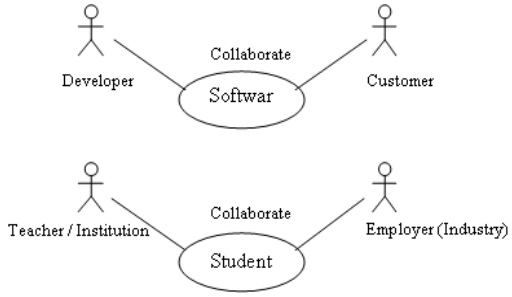

Fig.1 Analogy between Software Development and Education Process

It is clear that the institution of higher education should be considered as a firm delivering its own product/service: knowledge, skills, and attitudes necessary for students to acquire abilities for work and progress in professional area [3]. Therefore it is important to follow a process approach in the internal restructuring of the education institution in order to grow business performances, and its own competition on the education market.

In a first step towards the understanding of educational process improvement, the education process was correlated to CMMI practices [4] with the aim to propose a maturity model for computing education inspired by the capability maturity model (CMM) used in software engineering [1, 5]. Similar to CMM, a Computing Education Maturity Model (CEMM) was proposed to rate educational organizations according to their capability to deliver high quality education on a five level scale [5]. Furthermore, CEMM can be used in order to improve an institution's capability by implementing the best practices and organizational changes it describes. Application of a strict CMM in computing education raises the same issues and faces the same problems as in software development. The main criticism is following CMM implies the use of rigid waterfall process model with fixed scope.

This is why research in a second step was directed towards the new wave in software development with agile process and dynamic short cycles to meet the rapid changes in technology and business [2]. Agile software development is becoming a matured, effective approach and has wide acceptance according to the recently published trends [6, 7]. As illustrated in figure 2, agile development is rapidly becoming the norm. In a recent survey, 57.4% of surveyed organizations described their primary development method

Manuscript received October 5, 2010
Manuscript revised October 20, 2010

as Agile. Waterfall fell to third place in the 2010 survey, being preferred by only 13.1% of respondents.

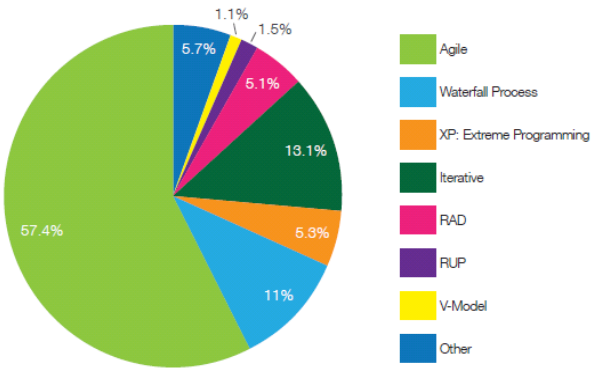

Fig.2 Software development Trends in 2010

The majority of reports from practitioners of agile development is positive and confirms the advantages of this approach. With Agile Software Development becoming more widely used, it is realized that adopting Agile within an academic setting is essential [18]. Introducing agile practices in education may be considered in all aspects of education process, i.e. the three key processes in education [2] which are: 1) Teaching (Teacher and support staff), 2) Evaluation (Examination and Marking), 3) Administration (infrastructure, and systems). The quality of the education product is directly related to quality of these three processes.

The rest of this work is structured as follows: Section 2 presents an overview of software agile methods. In section 3 a current situational analysis of Computer science teaching practices is discussed before describing a proposed "framework for evaluating agile principles in computer science education" in Section 4, and Section 5 describes the conclusion and future work on this topic.

2. Software Agile Methods

2.1 Agile versus plan driven methodology

The traditional Plan-driven approaches (such as Waterfall, PSP, or CMM-based methods) have been challenged in recent years by the emergence of the Agile methods (such as Extreme Programming, SCRUM and CRYSTAL) [8, 9]. Plan driven methodologies: focus heavily on process and way of doing things, requires a lot of documentation, time is spent on avoidable rework rather than value-added, aim at reducing cost by appropriately documenting to minutest detail so that the scope of error is reduced.

Agile methodologies: Focus is on reducing documentation; improving communication so that very little documentation is required and aim at reducing cost by reducing time spent on documentation and spend time in value added work. With Agile development, a project is divided into releases, each with its own requirements, design, build, and test activities.

The Plan-Driven and Agile methods both value the delivery of quality systems that meet stakeholders' needs, but they differ in strategies, not in goals.

2.2 Agile Manifesto & Principles

The agile manifesto defines four agile values as follows [10, 11, 12, 19, and 35]:

1) Individuals and interactions over processes and tools: Agile development is a human-centric approach that relies on people and enforces the interactions among them as a cornerstone in the definition of the agile software process.

2) Working software over comprehensive documentation: Agile approach is based on the iterative development model where early and frequent delivery of working software to the customer is crucial.

3) Customer collaboration over contract negotiation: Based on the agile manifesto, there must be significant and frequent interaction between the customers, developers, and all stakeholders of the project

4) Responding to change over following a plan: Basically, Agile is designed to be able to adapt to change. Products designs change throughout the project, and Agile helps to manage that change and keep everything under control. Hence an attitude of welcoming and embracing change should be maintained throughout the software development.

Twelve agile principles underlie the agile manifesto and define the core of what agile is. Agile principles are the essential characteristics that must be reflected in a process before it is considered Agile. Agile principles could be summarized as follows:

1) Customer satisfaction is #1 priority.
2) Delivery early, continuous value.
3) Embrace change and uncertainty.
4) Business and developers work together.
5) Provide motivated individuals the work environment and support they need to succeed.
6) Communicate face to face (co-locate if possible).
7) Best architectures, requirements, and designs emerge.
8) Trust the team to get job done.

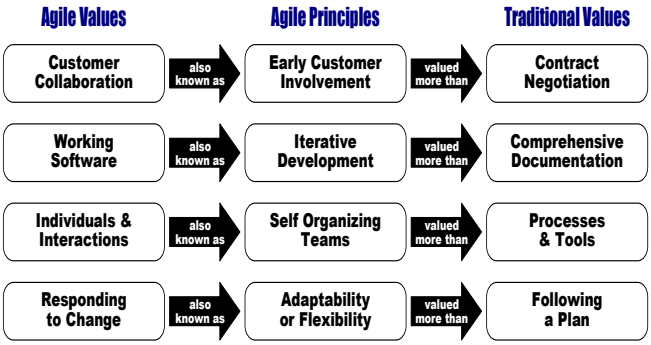

Fig.3

The use of agile principles and practices in software development is becoming a powerful force in today's workplace.

3. Current situation analysis of CS teaching practices

Industry complains that computer science graduates take at least one year to become productive once hired and there are several challenges to keep education current in the face of rapid change [13]. Also a decline is observed in student satisfaction and enrollment in computer science majors.

The reasons for such critical situation are multiple, but in this work, we will focus only on the reasons that are in direct conflict with the agile values.

3.1 The current Waterfall Teaching Model

Computer Science Academic programs are rigid and uses waterfall process model in delivery [2, 14]. In Most of the computer science curricula today, the topics are covered in their waterfall order specified by the existing prerequisite chains. The drawbacks of waterfall teaching model are as follows:

- Waterfall teaching will limit students' view of the complete education program and the type of engaging projects that the students can work on to enhance the learning process**.**
- Many important concepts and skills are scattered in many senior courses which cannot be taken earlier due to the strict course prerequisite requirements. As a result the instructors are limited in what kind of projects they can use to engage the students, and the students have limited opportunities in practicing the important skills.
- In many educational institutions courses and projects that emphasize Agile Software Development are minimal.
- A lack of basic programming skills reported by instructors of upper-division courses.
- Students are not exposed or have only limited exposure to the agile methods, and practices at the undergraduate levels of education

3.2 Knowledge Lag Problem

There is a lag between the knowledge scope of current computer science curricula and the expectations of the IT industry [2, 14, and 20]. In curriculum design, industry inputs are often missing and students graduate with little practical skills and no idea of industry expectations.

3.3 Knowledge Lag Problem

The planned Curricula based on instructor lecture are the dominant teaching method rather than increasing student participation and knowledge sharing [2, 20]. Consequently procedures are considered more important then outcome.
On the other hand teachers are not working as a project team; this affects the integrity of teaching contents and introduces inconsistencies in teaching as different instructors tried different approaches.

3.4 Adaptability and ability to change

Changes in Computer Science programs are not welcomed and Change is a bureaucratic process and on an average cycle time is 3-4 years [2, 20]. Also feedback from students and other stakeholders is not seriously considered. To improve this current situation we need to switch away from the waterfall teaching model and greatly shorten the current deep course prerequisite chains, and to advocate an in-depth lab-based computer science program with emphasis on the fundamental and recurring concepts and skills underpinning the modern computing technologies.

4. Agile Manifesto in CS Education

4.1 Mapping the Agile Manifesto to CS Education

Considering that the primary goal of a computing curriculum is to produce programmers and software engineers [15]. Therefore there is a need to learn to adapt to the ever- evolving nature of the field. Therefore adopting Agile within an academic setting is essential, and agile manifesto is correlated to computer science education as illustrated in Table 1[2, 20].

Table 1: Agile interpretation in CS Education Context

Agile Value	***Corally to CS Education***
Individuals and interactions over processes and tools	Students/Teachers over traditional processes and tools
Working software over comprehensive documentation	Iterative Teaching/Working projects over comprehensive documentation
Customer collaboration over contract negotiation	Interaction with industry plus academia over academia only
Responding to change over following a plan	Continuously add value / Responding to change and feedback rather than following a plan

5 . Discussion of Agile CS Education

5.1 Iterative Teaching/Working projects

The design of computer science curricula can benefit from three main agile practices as follows:
(a) An iterative teaching model [14, 16, 23].
An iterative teaching model is more effective than sequential/waterfall teaching. Computer science curriculum should early focus on the core expertise and master the basic hands-on problem-solving skills during their junior year. An example of experience piloting curriculum design based on the iterative model was developed at Pace University. A new lab-based overview course for computer science and modern information technologies was given to students who have just completed CS2 or the equivalent. The purpose of this course is to introduce the fundamental computer science concepts, methodologies and technologies underpinning the latest information technologies. With early the introduction of such course, the curriculum could be structured into three major iterations as illustrated in figure 3. The purpose of each iteration as follows:

- The 1st iteration: The first iteration covers in an early stage the important modern computing concepts with a simplified software framework project.
- The 2nd iteration: The second iteration consists of courses including data structures and algorithms, operating systems and architectures, networking, as well as many elective senior courses. With first iteration, the courses in iteration 2 can be taught in more flexible orders and with more depth and hands-on projects.
- The 3rd iteration: The 3rd iteration allows students to integrate the learned knowledge topics in problem solving and learn new knowledge/technologies with limited instructor assistance, which is important in developing life-long learning abilities.

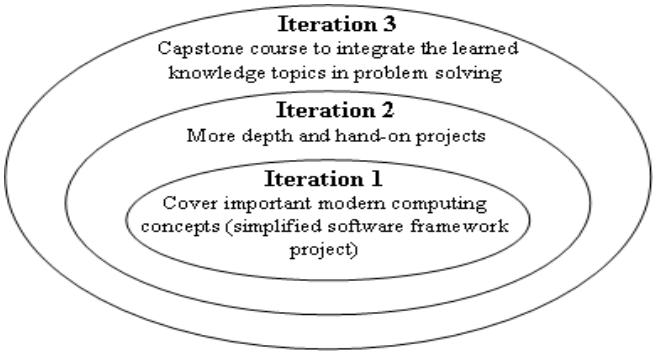

Fig. 4 an Example of Iterative Model

The iterative teaching model attracts the designers of recent Computer Science programs to deliver exciting programs for Computing that reflect enough technical material such that students can get some insight into career paths available to them, and also provide academic challenges to make courses attractive to top students [23].
(b) Working projects/ Student-driven projects.
Computer Science programs should be centered to prepare students to work as part of a team. With the iterative teaching model, it is possible to offer group projects early in first and second years and expanded in subsequent years [25, 28, 29, 30, 32, 33, and 36]. With such course projects students understand software process concepts, face problems such as scheduling, time management and planning combined with the course. They are also required to develop several types of documents. Such practices give students sufficient maturity and readiness for a more disciplined way to develop their programs.
(c) Emphasizing Agile Software Development in courses and projects
With Agile Software Development becoming more widely used, students must learn and understand the application of agile methods, principles and techniques. Software engineering courses and other courses that include a course project component are the best places to introduce and apply agile development methods [18, 19, 24, 25, 27, 31, 34, and 37].

5.2 Interaction and collaboration with industry

In an ideal world, review and changes to computing curricula should be driven solely by academic concerns for the needs of students. The process should be informed by industry accreditation processes and international best practice**,** it is important to explore current industry needs in order to suggest how to better prepare computer science graduates with the appropriate background that will enable a successful career. Agile Education is encouraging collaboration with industry and responding to market demands over syllabus and marks. Industry is an important source of practical problems, project ideas and technology trends. Many ideas are proposed for bridging the gap between computer science education and the IT industry as follows:

- Apprenticeship by immersion: in software engineering courses, real-world situations are imitated as closely as possible: a professional working environment, the client-supplier relationship, the application of a development baseline, the use of methods and associated tools, and cooperation within the team [13].
- Introducing a strong technological component to the curriculum. This normally comes in many different forms; prevalent among them is offering students courses in IT, work attitude and work

ethic, followed by a subsequent placement in industrial and commercial firms, where they get firsthand experience in real work environment [38].

- Modifying computer science curriculum to provide more emphasis on negotiation skills, time management, cultural differences, outsource management, in addition to a strong technical background [26].
- Liaising with the relevant industries to receive industrial knowledge to augment the classroom lectures.
- In student projects encourage student interaction with external customers or assign students to tutor/instructor with sufficient domain and programming knowledge Projects based on need as defined by stakeholders [17, 37].
- Attending local and international workshops on the latest IT innovations with the intension of transferring same knowledge to their students.
- University should encourage students towards registration for certifications in IT.
- Implement faculty improvement programs to upgrade their caliber and learn new technologies based on suggestions of leading software industrialists.

5.3 Collaboration among the major players in the education process

Agile practices focus on Teacher/Student productivity and value Competence and Collaboration over Compliance and Competition. Agile education encourages student-centered active, collaborative, cooperative learning over lecture-only approaches. Collaboration must be considered among the major players in the education process: student-student, student-teacher and teacher-teacher.
The instructional approach for improvement in undergraduate education involves the following principles [20, 39]: (a) Focus on collaborative learning early in the CS curriculum through engaging students in collaborative learning experiences through team-based problem solving, project planning, pair programming, and other agile software development practices; (b) Encourage frequent interaction between students and faculty; (c) Develop mutual cooperation among students; (d) Provide frequent active learning exercises; (e) Provide prompt feedback for student reflection; (f) Assisting teachers to be agile and develop mutual cooperation among teachers: Organisational skills sessions, Opportunities to work collaboratively, Teachers – focus on purpose.

5.4 Continuously add value/Responding to feedback

Content has its own life cycle and it needs to be improved. There should be well established mechanisms that allow to continuously adding value to programs based student feedback and changes in technology Constantly Updated Programs Modern content based on up-to-date theoretical foundations Classmates who are working in today's business environment professionals sharing their real-life experience. A program structure that encourages the sharing of insights Classmates compare different cultural/national/organizational approaches to the same problem Instructors actively involved in the field they teach Constant student feedback. Acting on Student Feedback: In response to student feedback, the instructors try to react promptly and visibly. Whether the particular student feedback relates to course curriculum issues, the coverage of technical content or the state and needs of a given team project, in preparation for a class session we consider if it is appropriate to adjust the order or the content of what is covered in order to increase the learning benefit for students.

6. Evaluation Framework For Agile CS Education

6.1 Evaluation model

The proposed evaluation framework measures how a Computer Science education process fulfills the agile values described in Sections 4. For this purpose, the framework provides measurements for the four postulates presented in Section 4. These postulates (Pi, i=1...4) are expressed as the assessment of the two sub-postulates (Pi.1, Pi.2). The measure of each postulate is defined as the difference between the measures of the related sub-postulates as follows:

$$m(Pi) = m(Pi.1) - m(Pi.2) \text{ where } i = 1..4$$

For example, Postulate 1 (P1) - Value Students/Teachers over traditional processes and tools, it's measured by calculating the difference between the measure of how the process values Students/Teachers and their interactions (P1.1) and the measure of how it values the process and the tools (P1.2). Both the sub-postulate encouraged by the agile principles (positive sub-postulate: Pi.1) and the other sub-postulate (negative sub-postulate: Pi.2) are measured in a scale of 0 to 10 as follows:

$m(Pi.x) = (\sum$ rate of related attributes) mapped to scale of 10,

Where x=1, 2

Table 3: Evaluation Framework for Agile CS Education Process

P1	Students/Teachers over traditional processes and Tools		
P1.1	Value the Students/Teachers	P1.2	Value the process and the tools
Attribute	description	Attribute	description
1	Adjusted Curricula	1	Planned Curricula
2	student participation and knowledge sharing	2	a standard method of delivering the material
3	cooperative teachers	3	teachers are isolated
P2	Iterative Teaching/Working projects over comprehensive documentation		
P2.1	Value Iterative Teaching	P2.2	Value an exhaustive documentation
Attribute	description	Attribute	description
1	iterative teaching model	1	waterfall teaching model
2	Early Student-driven projects	2	Courses are well documented
3	Agile practices in capstone courses	3	Waterfall practices in capstone courses
P3	Interaction and collaboration with industry plus academia over academia only		
P3.1	Value the collaboration with the IT industry	P3.1	Value the contractual negotiation
Attribute	description	Attribute	description
1	Curricula with strong technological component	1	industry inputs are often missing
2	High interaction with local industry	2	Low Interaction with Local Industry
3	Apprenticeship by immersion	3	
4	In student projects encourage student interaction with external customers	4	
5	encourage students towards registration for certifications in IT	5	
P4	Continuously add value/Responding to feedback rather than following a plan		
P4.1	Value the answer to change	P4.1	Value the monitoring of a plan
Attribute	description	Attribute	description
1	Change cycle time is shorter and based on iterations	1	Change on an average cycle time of 3-4 years
2	Evolution and change is recommended to be considered during iterations	2	Only high priority changes can be introduced during the cycle
3	ability to adapt to different learning styles and change the delivery	3	It defines a detailed plan for each year and does not accept change due to feedback
4		4	It defines a detailed plan for each year, which can be modified based on student feedback
5	Changes in IT/student feedback can be introduced during each iteration	**5**	It defines no planning whatsoever
6	being receptive and responsive to changes	**6**	adhering to a specific schedule

6.2 How to use the model

Like CEMM [5], the model is proposed to rate educational organizations according to their capability to deliver high quality education according to agile best practices.
As indicated in 6.1
Therefore, each agile principle might obtain a measure of -10 – in case both sub-postulates take the worst value. If the measure is a value close to 10, it means that the process is significantly value agile practices. If the measure is of 0 or close to 0, it means that the process does not significantly value the positive attribute over the negative, which means that the Agile Manifesto postulate is not completely satisfied, or in other words there is a balance between agile practices and current practices. The result can be represented as a process capability profile using Kiviat chart.
A Kiviat chart (sometimes called a radar chart) is used to present the evaluation of computer science process against the proposed agile practices and values. A kiviat chart is a graphical method of displaying multivariable data in the quantitative variables on axes starting from the same point. In the presented examples, the chart is composed of 8 axes extending from a central point. The axes correspond to the 8

Manuscript received October 5, 2010
Manuscript revised October 20, 2010

variables used in the evaluation model: P1.1, P1.2, P2.1, P2.2, P3.1, P3.2, P4.1, and P4.2.

Each axis is scaled according to the lowest and highest value of its associated variable as proposed in table 2.

As illustrated in figure xx, typical patterns for different situations are presented;
Figure 4.a: An agile centric computer education process.
Figure 4.b: A waterfall centric computer science education process.
Figure 4.c: A balanced agile/waterfall computer science education process.
Figure 4.d: A non collaborative computer science education process.
Figure 4.e: A missing industry input computer science education process.

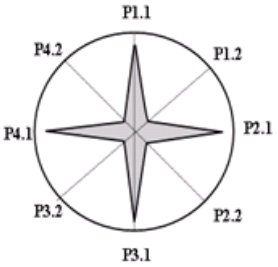

Fig. 4.a: An agile centric CS education process
- Collaborative, Iterative, replying to industry needs, adaptable

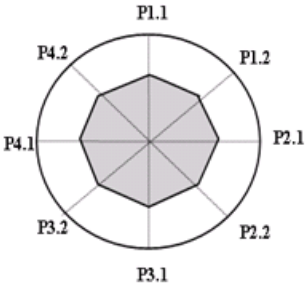

Fig. 4.c A balanced CS education process
- Agile and water fall practices are balanced

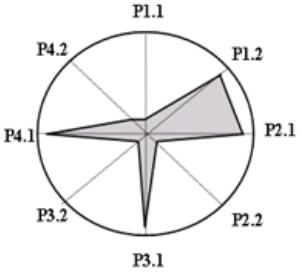

Fig. 4.d A non collaborative CS education process
- Agile but requires improvement regarding collaboration

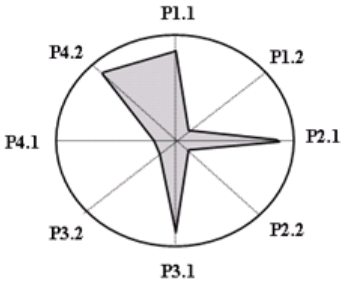

Fig. 4.f A non-adaptable CS education process
- Agile but requires improvement regarding accepting changes

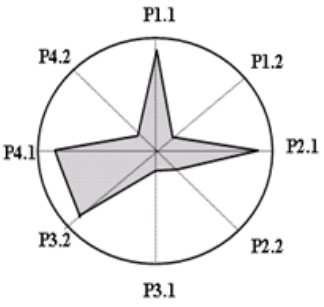

Fig. 4.e A missing industry input CS education process
- Agile but requires improvement regarding collaboration with industry

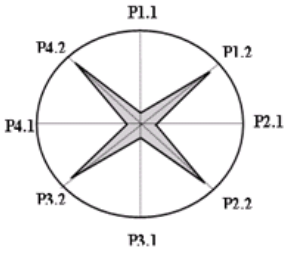

Fig. 4.b: A waterfall centric CS education process
- non Collaborative, Iterative, missing industry input, non-adaptable

7. Conclusion and future work

This paper summarizes the agile best practices applied to CS education. It focuses on the urgent need to apply agile framework to CS education system in order to improve quality and reacting to changes and industry requirements. The paper also tries to relate the agile CS education practices to the four agile values described in the agile manifesto.

The main contribution of this paper is proposing a basic model to rate and evaluate educational organizations according to their capability to deliver high quality education according to agile best practices. This model is inspired from an evaluation framework used to assess agile software methodologies [21]. The model can also be used to organize the improvement effort based on the institution priorities.

The future work includes comparing standard CS curriculum such as ACM/IEEE CC 2001 with the aspects related to curriculum design in the proposed model. In addition, we plan to extend the model by introducing weighting factors for the attributes used to measure the agility in order to reflect the relative importance of the attributes.

Referances

[1] Petros K. Dounos and George A. Bohoris, (2007), "Exploring the interconnection of known TQM process improvement initiatives in Higher education with key CMMI concepts, 10th QMOD Conference, Helsingborg, Swede

[2] Dr. Venkatesh Kamat, (2008), "experience of using *Agile* in the education *process*", the *Agile* Goa conference.

[3] Zoran L., Ljiljana R., Boza N., (2007), "Information System Implementation Based on Process Approach at Higher Education Institutions", Proceedings of Computer Science and IT Education Conference.

[4] SEI-Software Engineering Institute, (2006), CMMI (Capability Maturity Model Integrated) for Development, Version 1.2, Staged Representation.

[5] Christof L., et. Al., (2007), "A Maturity Model for Computing Education", Ninth Australasian Computing Education Conference, Ballarat, Victoria, Australia.

[6] Technology Management Resource for Business Leaders, (2010), The 2010 Software Development Trends-Survey Results.

[7] Dr. Dobb's Global Developer Technogaphics Survey, Forrester research, Inc., Q3 2009.

[8] Sutap Chatterjee, (2010), "The Waterfall That Won't Go Away", ACM SIGSOFT Software Engineering Notes, Volume 35 Number 1

[9] Li Jiang, and Armin Eberlein, (2008), "Towards A Framework for Understanding the Relationships between Classical Software Engineering and Agile Methodologies", APSO'08, Leipzig, Germany.

[10] Beck, K., et.al, Manifesto for Agile Software Development, http://agilemanifesto.org/

[11] Agile Alliance, http://www.agilealliance.org/.

[12] Decan Whelan, (2008), "Agile Adoption & Adaptation Framework, Whelan & Associates Inc.

[13] Vincent Ribaud and Philippe Saliou, (2008), "A few elements in software development engineering education", Workshop on the Roles of Student Projects and Work Experience in Undergraduate and Taught Postgraduate Programmes - CSEET 2008, United States.

[14] Lixin Tao and Li-Chiou Chen, (2010), "A Hands-On Overview Course for Computer Science and Modern Information Technologies", Proceedings of Student-Faculty Research Day, CSIS, Pace University.

[15] Matthias F., et.al., (2004), "The Structure and Interpretation of the Computer Science Curriculum", Journal of Functional Programming.

[16] Duben, Naugler, and Surendran, (2004), "Agile Computing Curricula", Proc ISECON 2004, v21.

[17] Sue De Vincentis, Agile Education: Student-driven knowledge production, Australian Council for Educational Leaders

[18] Shvetha Soundararajan, James D. Arthur and Amine Chigani, (2010), "Understanding the Tenets of Agile Software Engineering:Lecturing, Exploration and Critical Thinking", Computers and Society (cs.CY)

[19] David F. Rico and Hasan H. Sayani, (2009), Use of Agile Methods in Software Engineering Education, Agile Conference, AGILE '09.

[20] John C. Stewart, et.al., (2009), "Evaluating Agile Principles in Active and Cooperative Learning", Proceedings of Student-Faculty Research Day, CSIS, Pace University.

[21] Karla Mendes Calo, Elsa Estevez, Pablo Fillottrani, (2010), "A Quantitative Framework for the Evaluation of Agile Methodologies", JCS&T Vol. 10 No. 2.

[22] Asif Qumer, (2006), " Measuring Agility and Adoptability of Agile Methods: A 4-Dimensional Analytical Tool", IADIS International Conference Applied Computing

[23] Tim Bell, Peter Andreae, and Lynn Lambert, (2010), "Computer Science in New Zealand High Schools", the Twelfth Australasian Computing Education Conference (ACE2010), Brisbane, Australia.

[24] Thomas Reichlmayr, (2003), " The Agile Approach in an Undergraduate Software Engineering Course Project", 33rd ASEE/IEEE Frontiers in Education Conference, 2003 IEEE.

[25] Andrew D. H. Chow and Mike Joy, (2004), "Shifting the Focus from Methodologies to Techniques", HE Academy for Information and Computer Sciences.

[26] Chris B. Simmons and *Lakisha L. Simmons*, (2010), "Gaps in the Computer Science Curriculum: An Exploratory Study of Industry Professionals", Consortium for Computing Sciences in Colleges.

[27] Bill Davey, Technology in Education: An Agile Systems Approach , Proceedings of Informing Science & IT Education Conference (InSITE) 2010

[28] Richard L. Upchurch and Judith E. Sims-Knight, (1997), "Integrating Software Process in Computer Science Curriculum", Frontiers in Education Conference, 27th Annual Conference.

[29] Deepak Dahiya, (2010), "Teaching Software Engineering: A Practical Approach", ACM SIGSOFT Software Engineering Notes, V35 Number 2

[30] Brian R. von Konsky and Jim Ivins, (2008), "Assessing the Capability and Maturity of Capstone Software Engineering Projects", Proc. Tenth Australasian Computing Education Conference, Wollongong, Australia.

Ahmed El-Abbassy: Received the Ph.D. Degree Computer Science from ENSAE, France, 1979, now is a professor in the Department of Computer Science, El-Shorouk Academy and consultant to Ministry of Health, Egypt, His research interest includes: Software Engineering, Operating Systems and Information Management.

Ramadan Muawad: Chief Professor of computer science and Information System department, Arab Academy for Science, Technology and Maritime Transport.

Ahmed Gaber: Received a bachelor degree in Computer Science with Excellent grade with honor from Al-Shorouk Academy, Egypt 2006, now is an IT Analyst at Agility Logistics Co.