A Maturity Model for Computing Education

Christof Lutteroth **Andrew Luxton-Reilly** **Gillian Dobbie** **John Hamer**

Department of Computer Science
The University of Auckland
38 Princes Street, Auckland 1142, New Zealand
Email: {lutteroth,andrew,gill,j.hamer}@cs.auckland.ac.nz

Abstract

We propose a maturity model for computing education which is inspired by the Capability Maturity Model (CMM) used in software engineering. Similar to CMM, the Computing Education Maturity Model (CEMM) can be used to rate educational organisations according to their capability to deliver high-quality education on a five level scale. Furthermore, CEMM can be used in order to improve an institution's capability by implementing the best practises and organisational changes it describes.

Keywords: Education, CMM, quality, maturity

1 Introduction

In this paper we draw an analogy between process improvement in software development and process improvement in computing education. 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, despite finding this difficult to measure.

In both domains, a main ingredient of success is good structure and the use of best practises, i.e. a process that helps us to structure and do things right. From a teaching perspective, the process of education is even more relevant than a development process for software engineers: a skilled software engineer may neglect good software development practises but still produce a good product at the end. The process by which software is developed is not directly visible in the quality of the end product. Teachers, however, can influence the end product of their work only indirectly. The actual learning outcomes are, ultimately, not up to them but up to the students. All a teacher can do is perform the process of teaching the best he can, and that is why the process is of such an importance: it is the only tool we have.

What exactly do we mean when we speak of a *process* for computing education? In our analogy, software development projects correspond to courses as they are taught, in contrast to a course as an abstract concept. Somehow a course has to be defined: what is taught, when is it taught, how is it taught and what is the measure of success (e.g. exam marks)? All these questions are very familiar to lecturers, and it helps to have most of them answered before the course is taught. Now, the same what, when and how—but of course with a different context—have to be answered for a software development project, and constitutes what is commonly understood as software development process. Software development is not the only analogy one can draw; a course is similar to other processes as well. However, in the context of computing education it seem particularly appropriate to refer to processes of software engineering.

If processes are so important for the quality of the product—and many disciplines agree in this matter—then we should spend time and effort on improving them. In manufacturing, for example, we find a long tradition of process management and process improvement. This is why meta-processes have been created that aim at improving the quality of the processes themselves. The point that we want to make is that for education, and in particular computing education, quality improvement concepts exist as well and can make a big difference for a teacher's success. In this paper we show how such a meta-process can be created for computing education along the lines of the Capability Maturity Model (CMM), a meta-process for software development.

We acknowledge that many factors affecting educational success are of a human nature: good teachers make a difference, and teaching and learning are greatly influenced by the personal interactions between teachers and students. Nevertheless, the benefits and support provided by a high-quality educational process should not be underestimated. Process can benefit both students and teachers. Furthermore, the maturity model for education which we propose in this paper should not be misunderstood as an attempt to dictate any particular approach to teaching. Our approach is descriptive rather than prescriptive, and the intention is to provide best practises and improvement strategies to support teachers in their work. The aim is essentially the same as for CMM: to narrow the variance of quality by applying common sense process management and quality improvement concepts.

We use CMM as an inspiration but not as a strict guideline. Thus, we are interested in the general insights and concepts, and not in the many details (such as the differences between the different versions of CMM and its successor CMMI). Education is covered in neither CMM nor CMMI, and we believe that the context of academia in contrast to the context of industry requires us to deal with CMM and related material in a constructive yet selective and critical manner. It is not possible to create a maturity model for education by strict analogy.

Section 2 motivates our maturity model. We present an overview of CMM in Section 3 before describing our approach in Section 4. Section 5 revisits a model of major factors in software engineering and transfers it to the domain of education. Section 6 provides an analysis and discussion of our work. In Section 7 we discuss related work, and Section 8 describes further work we intend to do on this topic.

Our conclusions are offered in Section 9.

2 Motivation

Reflective practise plays an essential role in improving an individual's professional activities (Schön 1987). As educators, we use reflective practise to examine our own teaching, and refine the approaches we use in the classroom. Harris (1998) reports that "effective teaching is linked to reflection, enquiry and continuous professional development and growth."

As professional educators, we should reflect on our processes and practise at a variety of levels. A teaching portfolio can be an excellent aid to reflective practise at an individual level. Course portfolios play a similar role for courses, assisting us to engage in reflective practise about courses. However, there is no comparable tool used to reflect upon process and practise at a departmental level.

2.1 Individual reflection

Teaching computing is difficult. A large (and growing) body of research shows that individual staff are trying to improve student outcomes, but with limited success (Robins et al. 2003). Although those interested in CS Education may have developed more formal processes to evaluate and improve their own teaching performance, for most staff the cycle of teaching occurs on a more *ad hoc* basis.

Teaching staff generally have few processes in place to analyse and improve teaching performance. Student evaluations are reasonably common, but other formal processes are rare. One such process is the maintenance and evaluation of a teaching portfolio.

A teaching portfolio is an important tool used by reflective educators. It provides a means of exhibiting professional proficiency (i.e. it can be used for summative feedback), but is also frequently used to afford insight into an individual's professional practise (i.e. it can also be used for formative feedback) (Seldin 2004). The reflective practise exemplified by teaching portfolios is clearly focused at an individual level.

2.2 Course reflection

Although reflection on individual teaching practise plays an important role in the scholarship of teaching, reflection on the course itself is also valuable. A course portfolio can be used as a tool to reflect on a course and guide future improvement. Cerbin (1994) describes a course portfolio as consisting of four core elements:

1. a teaching statement;
2. an analysis of student learning;
3. an analysis of student feedback; and
4. a course summary.

Course portfolios are intended to form part of an ongoing process that critically analyses the course in detail. The reflection required for a course portfolio overlaps with that of a teaching portfolio, and the development of one portfolio can provide leverage for effective reflection on the other.

> *Ideally instructors should reflect upon every assignment, chapter, or lab experiment in their course. In reality, instructors may write down their personal reflections only once a week or for every major section of the course.* (Reeves et al. 1998)

Our experience indicates that course portfolios are not as widely used as teaching portfolios, and critical reflection at the level of courses is not as common as individual reflection.

2.3 Departmental reflection

The problems faced by computing educators are compounded when department processes fail to provide adequate support. Although reflective practise is often demonstrated at an individual level, and occasionally directed at courses, it is rare to see any critical reflection on departmental processes, although such processes have significant impact on both courses and individuals.

There is a need for a framework within which we can position current teaching practise at both an individual and departmental level. Such a framework would encourage educators to examine their own processes more critically and might suggest avenues for improvement at a strategic level.

Using a model of process maturity would act to support critical reflection at both an individual and departmental level. The reflection engendered by process analysis would complement the development of teaching portfolios and course portfolios. We intend to show here how the Capability Maturity Model used to describe software development processes can be adapted for use in the education domain, providing a framework for positioning current academic processes and reflecting upon current practise.

3 The Capability Maturity Model (CMM)

The Capability Maturity Model (CMM) (Institute 1995) was originally developed in the 1980s by the U.S. Department of Defence Software Engineering Institute (SEI) at Carnegie Mellon University as a method for objective evaluation of contractors for military software projects. It has been continuously revised since then.

There are numerous instances of large software systems suffering unexpected cost increases, schedule delays, and even complete failure. As a consequence, the U.S. military and other organisations were looking for a way to rate the reliability of the software development work a contractor could offer. The original CMM and its successors were, and are still, used for many government projects.

The idea behind CMM is that a high-quality process yields a high-quality product at the end. As a consequence, CMM aims at providing objective measures for the quality of software development processes and strategies for their improvement. CMM tries to define the key elements of an effective process and outlines how to improve suboptimal processes, i.e. the evolution from an "immature" process to a "mature, disciplined" one. It describes key practises for meeting goals for cost, schedule, functionality, and product quality. The CMM standard is relatively heavy-weight, being several hundred pages strong.

CMM ranks software developing organisations according to a hierarchy of five maturity levels, with the first being the least mature and the fifth being the most mature. The five levels are: *initial*, *repeatable*, *defined*, *managed*, and *optimising*. Each maturity level defines a certain capability of producing quality software, key process areas (KPAs) that state what is done in order to achieve the respective level of quality, and key practises which specify how it is done. A software developing organisation ranked at a certain maturity level can improve over time and reach the next level of maturity. However, a new level has to be well established before the next level can

be achieved, so that it is not possible to skip levels. This is because each level builds on the preceding ones and adds features to the process rather than replacing them.

In 1997 development of CMM was halted in favour of its successor, Capability Maturity Model Integration (CMMI) (Chrissis et al. 2003). CMMI provides a structured view of process improvement across an organisation, i.e. not just the organisational parts concerned with software development. It provides models for four different disciplines—Systems Engineering, Software Engineering, Integrated Product and Process Development, and Supplier Sourcing—and is intended to provide a framework for the integration of other models that might emerge. It further aims at creating appraisal and training products for process quality. For our education maturity model we solely refer to the much better known CMM, which provides all the necessary core concepts and ideas. In the following we will describe all the five maturity levels and summarise some of the criticism of CMM.

3.1 Initial development process

This level is the lowest possible and (tragically) the level most software developing companies fall into. It is also known as "ad hoc," "chaotic" or "heroic" level. It refers to a process which is informal and poorly controlled, and thus "chaotic." An organisation of this level does not provide a stable environment for developing and maintaining software, so constant changes of the process make it unpredictable, i.e. "ad hoc." The organisation's performance relies on the capabilities of individuals ("heroes"), who may do or not do their work well. Thus, the performance varies greatly with their innate skills, knowledge, and motivation. This all leads to unpredictable cost, schedule, functionality, and product quality.

3.2 Repeatable development process

Level two refers to an organisation in which a good performance is repeatable. A project management system is in place, and planning and management of new projects is based on experience with similar earlier ones. Thus, successful practises from those earlier projects can be repeated. Such an organisation has established policies for managing a software project and procedures to implement those policies, i.e. effective management processes for software projects are institutionalised. Key process areas of this level are management activities like requirements management, project planning, project tracking and oversight, quality assurance, and configuration management.

3.3 Defined development process

On level three, the process used in an organisation is standardised and documented. The organisation uses effective management as well as effective software engineering practises, and software engineering and management processes are integrated. The process is characterised and fairly well understood. Organisations on this level have formed a dedicated Software Engineering Process Group (SEPG) that takes care of all the process-related activities, i.e. process definition, adaption and development. Furthermore, such an organisation provides a training program about the process so that everybody can acquire the knowledge and skills required to fulfil the roles the process assigns to them. The standard process of an organisation can be tailored to the unique characteristics of a project, and the result of this adaption is called the project's defined software process.

In summary, this level adds engineering processes and organisational support for process management. Key process areas include: process focus, process definition, training program, integrated software management, software product engineering, intergroup coordination, and peer reviews.

3.4 Managed development process

On level four the products as well as the process are quantitatively controlled. This means that the process is instrumented with well-defined and consistent measurements, and there exist quantitative quality goals for both the software products and the process. An organisational measurement program exists that measures productivity and quality for all the important activities, and the surveyed data is collected in an organisation-wide software process database. Processes on this level are predictable because the process is measured and controlled so that it operates within measurable, well-defined limits. Thus, we ideally achieve a predictably high quality. The focus of this level is on product and process quality, and key process areas are quantitative process management and software quality management.

3.5 Optimising development process

On level five process improvement is institutionalised. The whole organisation is focused on continuous process improvement. The collected data on the effectiveness of a process is used to analyse the cost benefit of new technologies and proposed process changes. Furthermore, data is analysed for causes of defects, so that known types of defects can be prevented from recurring. Weaknesses are identified and respective improvements are undertaken proactively. Communication within the company ensures that innovation, once identified, is disseminated, and experience is exchanged also between projects. Key process areas are defect prevention, technology change management, and process change management.

3.6 Criticism

CMM has received a degree of criticism over the years. First of all, one has to observe that CMM has failed to take over the world, and commercially there are more successful methodologies, e.g. IBM's Rational Unified Process (Kruchten 2001), which try to provide a framework and guidelines for high-quality software development. CMM is neither a recipe nor guarantee for success: an organisation operating on level one may be more successful than one operating on a higher level, although this is considered less likely, especially for larger organisations. There is little validation of the cost savings provided by CMM below the fourth level since this is where quantitative measurement starts, and unfortunately there are only very few organisations on this level. The vast majority of software developing organisations is considered to be still on level one.

Many critics accuse CMM of having too much bureaucratic overhead, and it is therefore often thought to be only suited for organisations that exhibit a high degree of bureaucracy anyway, such as government agencies or large corporations. CMM may influence an organisation to focus on perfectly completed paperwork rather than on productive tasks like application development or sensitivity to client needs and the market. A highly-regulated process may stand in the way when entering a market with some kind of product is more important than functionality and high quality. The main criticism objects that CMM

promotes process over substance, i.e. predictability over the actual service provided to end users.

4 The Computing Education Maturity Model (CEMM)

CMM is used to rate the maturity of software developing organisations. Our proposed Computing Education Maturity Model (CEMM) aims to do the same for educational organisations. The basic building block of CEMM, however, is the course. This is possible because courses, as an abstract entity in an educational organisation's curriculum, are (ideally) well-specified and (usually) adhere to fixed time and cost constraints. Whenever the course is held, these invariants will typically stay the same. Consequently, the course unit is a stable enough concept to be rated. In contrast, software development projects are more often unique events without such invariants and thus the only stable concept to be rated in CMM remains the organisation itself.

Like CMM, CEMM rates an education process according to a hierarchy of five levels. We retain the names of the levels, and the generic concepts and ideas of each level are essentially the same, transferring them into the new domain. On the course level, the absence or presence of a process for a course and its characteristics determine the maturity level of that course. On a higher organisational level, maturity is determined by the maturity of individual courses and the way in which all those processes affecting the quality of the organisation are standardised and integrated. In the following sections we will discuss the different maturity levels.

4.1 Initial education process

The initial level of maturity is like the one in CMM: informal and poorly controlled. This means that the process is neither understood nor reflected upon. A typical cause for this are changes of requirements, as would be the case for a new course that has never been held before, or for a lecturer who decides to change large parts or all of a course. Such drastic changes invariably lead to instability and frequently result in a rather chaotic and stressful environment. Course preparation and material is made on demand, i.e. *ad-hoc*, and because of the instability there is hardly any reviewing at all. Time pressure also plays a role here, with the last lecture slides often being created the hour before the lecture starts. Unsurprisingly, this results in a lot of mistakes in material like handouts and slides. For reflective activities like evaluation of feedback or grades there is usually no time left. The reuse of the created material is low and constrained to the personal teaching activity of the lecturer. Like in CMM, the success of courses depends on the skills and initiative of "hero lecturers," who are able to perform despite the lack of structural support.

4.2 Repeatable education process

On level two, courses can be taught again as they were taught before, and previous successes can be repeated. Planning and implementation of courses are based on previous experience, and the plan for a course is used to track its progress. The educational organisation has established policies for important aspects concerning the management of courses, and procedures to implement those policies. For example, there should be policies about helping students to catch up with missed lectures, on how students can seek assistance, and how cheating is dealt with. Courses have well-defined prerequisites, requirements and intended outcomes that fit well with the other courses offered. Course materials like slides, assignments, lab sheets and exam questions exist and can be reused not only by the lecturer who created them but also by others. A transition is made from personal resources to resources that can be shared with others. Changes to course material are controlled; i.e. checks are made to ensure they are in harmony with the course plan and other constraints, and managed with a version control system.

4.3 Defined education process

On level three, the course concepts and materials have been improved in several iterations, so that they have reached a certain level of stability. Courses are well-documented and come with a rich collection of material, for the students as well as for the lecturers. There exist scripts or textbooks for all the topics covered in a course, which potentially give students the possibility of achieving the aims of a course solely by guided self-study. There exist sets of slides and other materials that may be appropriate for a course, such as source code examples. Students can find learning material and important course information on a well-defined location on the Internet. If material cannot be made available online (e.g. textbooks), references and instructions are provided on how it can be acquired by students. For the lecturers there exist collections of proven lab exercises, assignments and exam questions. Material is of high quality and can be reused by different lecturers. It is accessible in a shared repository which is an official organisational resource, and maintained collectively by all lecturers. The quality of the material is ensured by the systematic application of reviewing techniques, e.g. peer reviews. Lecturers teaching (or having taught) the same or related courses communicate and work together and organise themselves in working groups.

4.4 Managed education process

On level four, the educational organisation has established a measurement program. Data about each course—assignment, exam and teaching evaluation results—are collected and stored in a central database. The data is fine-grained and not unnecessarily aggregated, e.g. the data set will contain scores for the individual items of a test. Statistical methods are applied on a regular basis in order to verify course practises and manage quality, e.g. controls check whether the variance of grades is within a certain range. Intervention policies are defined and triggered when controlled quality parameters do not fall within well-defined acceptable limits. For example, a lecturer may contact a student who fails an assignment, to propose a meeting and offer advice.

4.5 Optimising education process

On level five, changes to the process are carefully managed and reflect best practice as informed by the community of scholars engaged in education research. Qualitative and quantitative research methods are used to gain insight into the process and lead to improvements in the process itself.

Statistical methods are used in order to control and change the process itself. Changes of the process are carefully managed, i.e. identification and implementation of process improvements is institutionalised. This may be, for example, in the form of process improvement meetings at the end of each course iteration. Statistical methods can be of great help in, for example, the selection of exam questions and

analysis of course results. Consequently, improvement meetings need to discuss descriptive statistics that illustrate the characteristics of the collected data effectively. Defects have to be identified and analysed, so that people get to know what went wrong and why. Knowledge about defects is used to prevent recurrences, and knowledge about weaknesses used for proactive process improvement.

5 Sneed's Square revisited

Every industrial software project has some key factors that have to be balanced against each other by means of careful project management. Sneed's "devil's square," shown in Figure 1, illustrates this balancing act with a simple model of how these factors influence each other.

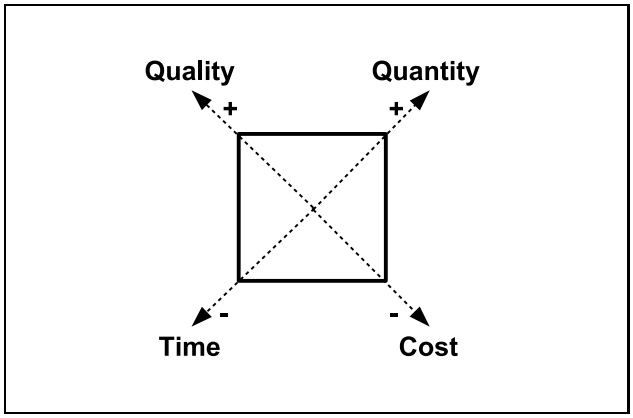

Figure 1: Sneed's "devil's square"

Four factors are considered:

1. the *quality* of the software that is developed;
2. the *quantity* of the software, i.e. the amount of functionality covered by it;
3. the *time* required to develop the software; and
4. the development *cost*.

In the figure these factors are represented as axes pointing outward. On the top two axes the outward orientation represents positive change, and on the bottom two the outward orientation represents negative change. The values for each factor are given by a square, with the area of the square (or, for better illustration, the circumference) being the productivity of the developing organisation. The productivity is considered invariant over the project duration. The square can be thought of as a thread that is looped around movable pins on the axes. If we want to increase the quality or quantity of the developed software, then the pins on the quality or quantity axes are pulled outward. Since the pins are constrained by the thread, which has a constant length, this will inevitably pull the pins for time and cost into the square: if we want to increase quantity or quality, then development will take longer and/or cost more. Hence the devil's square models the antagonistic effects of the four factors.

When teaching a course we face a balancing act similar to that of the devil's square, and, as in software development, changes to the different factors have to be managed carefully in order to preserve the structural integrity of the course. We illustrate this by means of a similar model, the education square, which is depicted in Figure 2. The education square models four factors:

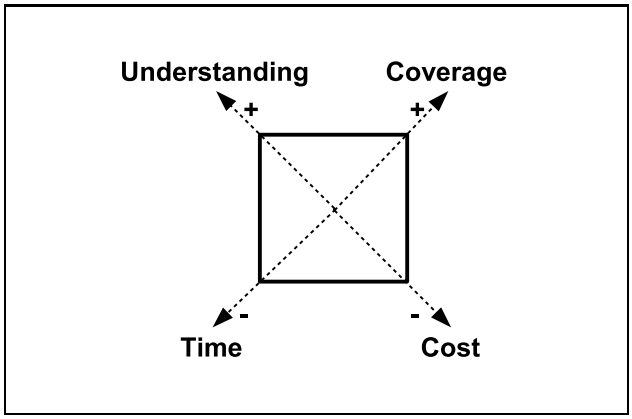

Figure 2: Education square (similar to Sneed's "devil's square")

1. the breadth of *coverage* of important topics in a course (quantity);
2. the students' depth of *understanding* of the taught subject in the course (quality);
3. the *time* required to develop the material for the course, teach the course and for students to learn the material; and
4. the *cost* of the course for both the lecturers and students.

The mechanism of the model is the same as the devil's square. The area within the square is the productivity of the department. In order to increase breadth of coverage of material and depth of understanding, with the same productivity, there must be more time and/or more cost. If it is possible to increase the productivity of a department, perhaps when moving up maturity levels, it is possible to increase the breadth of coverage and depth of understanding without increasing the time and cost input.

What the education square does show us is that in order to improve the maturity of a department we have to use the experiences of each course iteration to incrementally change this balance for the better. Such changes, however, should not be made thoughtlessly but have to be controlled and managed well.

6 Analysis and discussion

In this section we address these questions:

- Will CEMM defined for computing education suffer from the same shortcomings as the CMM used in the software industry?
- How will the CEMM be used in practise?
- What is the cost of introducing and maintaining CEMM at Department level?

6.1 Possible shortcomings in CEMM

The main criticism levelled at the CMM is the overhead in following the process. One overhead is documentation preparation, and another the measurements that must be taken.

In computing education, it is recognised as good practise to handout a course outline at the beginning of a course and submit a course audit at the end of the course. The course outline enables students to gain an understanding of what the course is about and what is expected from them. Completing the course audit allows the lecturer to reflect on what

they have done in the course, and document the success of teaching techniques they used in the course. The latter information is useful to anyone who takes on lecturing the course the following year or anyone who intends to use the same techniques in their own courses. Course material such as slides, course handbooks, exams and tests, and text books, which are already designed for and used for courses would be included in the documentation. In a well organised department, all this material is kept in a content management system. The documentation required in the CEMM fits naturally with the education process as it is already undertaken in many computing departments.

Measurement in the software industry is fraught with problems. For example, how do we measure productivity, how do we measure the size of a system, and how do we compare the measurements. In addition to this, measurement might not be accepted by employees as they may have fears that the surveyed data might be used against them. In computing education, there are measurements that we deal with on a regular basis. We grade students, we have incoming and outgoing GPAs, we have pass rates. Collecting these measurements are a basic requirement in computing education. These measurements can be used in CEMM.

6.2 CEMM in practise

What are the benefits to individuals and organisations from using CEMM?

Some lecturers and departments operate at level one. Individual lecturers prepare new material each year a course is taught and do not reflect on or learn lessons from previous years. At the department level, teaching is distributed on an ad-hoc basis with no reflection on previous years.

In our experience, individual lecturers and departments are more likely to be operating at level two. Lecturers place course outlines, lectures, assignments and lab specifications on the web. This resource is available for use the following year. Departments ask lecturers what courses they would like to lecture and typically the same person will lecture the same material each year.

What is required to lift individuals and departments to level three? A process by which the material is collected in a repository in a format such that it could be used by anyone in the department. The resource would need to be seen to be "owned" by the department rather than by the individual lecturer.

In order for individuals and departments to move to level four, it is necessary to measure and collect the measurements. Some of this may be going on now, but to be at this level it is necessary to have a process that collects this information. Many Universities currently have the tools in place to collect this information, but they often do not utilise these tools to their full potential.

To get to level five there must be a process that uses this information for improving the teaching within the department, and the promotion of departmental reflection.

There are small gains to be made for the individuals lecturer. They gain from:

- reusing material that they prepare;
- being able to measure and document the improvement in their teaching practise;
- maintaining statistics that can be used when applying for tenure, grants and promotion;
- sharing of practise with other lecturers in the department.

There are larger gains to be made for the departments. They gain from:

- reusing material and best practise between courses;
- being able to measure and document the improvement in their teaching practise;
- maintaining statistics that can be used at faculty level;
- sharing of best practise across the department.

6.3 Cost of CEMM

What is the cost of moving from level to level for a department?

The cost is mainly one of cultural change. Computing education has advanced a long way over the past twenty years, from chalk-and-talk through overhead transparencies to the web. As we pointed out above, much of the documentation that is needed is already prepared and many of the measurements are already taken, so the implementation of the process is simply one of using the material that is available in the management and maintenance of the process.

However, in many departments there is still a feeling of ownership with respect to not only the material prepared for courses but also for courses themselves. Convincing staff to give up this ownership to the department could require quite a large shift.

7 Related work

The idea of a CMM for education is not new. For example, Jalote (2003) points out a general strategy for its realisation as a means to overcome the lack of quality standards in the education sector. He addresses the need but says hardly anything about how such a model should look like.

An e-learning Maturity Model (eMM) (Marshall & Mitchell 2004, 2003, 2002) was proposed which is based on the Software Process Improvement and Capability dEtermination (SPICE) model (ISO/IEC 1998). SPICE can be seen as the ISO answer to SEI's CMM, and uses basically the same five levels of maturity plus an additional level zero for processes that are performed incompletely or not at all. It groups process activities into five areas, and eMM defines five corresponding areas for the domain of e-learning. Benchmarks and best practises for e-learning were examined and used in order to compile a set of practises that would fit into the five areas. Unlike CMM, the maturity levels are used for the evaluation of each practise and not just the whole organisation. The result is a framework for e-learning maturity evaluation, which has been applied to the e-learning systems of several New Zealand universities (Marshall & Mitchell 2005). While we believe that the aims of eMM with respect to an educational organisation are similar to the ones of CEMM, its domain is a different one and thus not entirely suitable for our purpose. However, some practises, e.g. from the process areas evaluation and organisation, could be reasonably applied in CEMM as well.

Neuhauser (2004, 2005) presents the Online Course Design Maturity Model (OCDMM), which is also a maturity model for e-learning, based on CMM. This model describes different states of adoption of e-learning technology, which form five maturity levels. The difference between the levels is mainly the degree to which e-learning technology is successfully

used. Like eMM, OCDMM targets specifically the domain of e-learning and is very different from our approach. Only some aspects of OCDMM are of direct relevance to CEMM, e.g. the proposed measures for the success of a course.

Kajko-Mattsson et al. (2001) propose the Corrective Maintenance Maturity Model (CM^3) for Maintainers Education and Training, a maturity model for workforce development of software maintenance engineers. Their maturity model is based on two educational processes from industry and several generic process models, among them CMM. It defines three levels of maturity: initial, defined and optimal. This maturity model looks at continuing education in an industry environment, which is radically different from that of an educational organisation. In contrast to our domain, education is not the main focus of the context of CM^3. Therefore, even though both maturity models deal with education, they are hardly comparable.

Frailey & Mason (2002) describe how the Software Engineering Body of Knowledge (SWEBOK) (Abran et al. 2004) was used in order to direct workforce development at a company and curriculum planning at a university. SWEBOK is an attempt of the IEEE to categorise and summarise the knowledge a software engineer should have after four years of practise. The common use of SWEBOK enabled the company to find appropriate university courses for their employees, and the collaboration resulted in the creation of certificate programs that were inspired in their structure by CMM. While this study is not directly related to our work, it shows that standards like SWEBOK and CMM can be useful for educational organisations as well as industry.

People CMM (Curtis et al. 2001) is a maturity model for workforce management and development that was created by the SEI. It addresses the importance of continuing education for the improvement of a software organisation's maturity according to CMM. It describes "best practises" from human resources, knowledge management and organisational development, fitting them into the hierarchy of the five CMM maturity levels. Although the model is tailored to the industry context, its practises for motivation, communication and the realisation of potential synergies within an organisation may inspire similar practises in the educational context. Motivation of students is a major factor for education, as can be communication among students, between students and lecturers, and among lecturers. The practises described in our model rely on collaboration between the staff and, on a higher level, between the organisational units of an organisation. This is a basis on which further synergies may evolve. Fostering a "culture of excellence" is one of the aims of People CMM, and also something that we as educators hope to instill into the students we teach.

To the best of our knowledge, CMM has not been applied to the general domain of teaching computing, with the exception of the work presented here. As discussed above, other maturity models for education either focus on professional development in an industrial context, or on e-learning as a very particular sub-domain. Neither of these approaches are applicable to CS education in general, but may inspire analogous best practises.

8 Future work

In this paper we have proposed CEMM. The next step is to validate the proposed model. We will do this by introducing the process into a computing department and collecting data at both the individual lecturer and department level. Computing departments are technically savvy, and we believe it may be more challenging to introduce an education maturity model to other departments. The next step is to work with another department and analyse whether the CEMM would be a good fit or whether it needs to be changed. Ultimately, we want to undertake a university-wide study, analyse the model at this level and study its effect.

9 Conclusion

We described a maturity model for computing education that is based on the well-known Capability Maturity Model created by SEI. Although our approach is new and has not yet been validated, the model incorporates best practises which are either based on common sense or have been successfully applied to other domains with similar motivations. Other best practises are supported by CS education literature, or are founded on our personal experience as lecturers. We find that certain key practises of CMM like rigorous documentation and use of a measurement program seem even better suited for our purpose, as they occur naturally in the context of education. This mitigates the negative impact of bureaucratic overhead on our model, the main criticism about CMM. We plan to continue the research on CEMM and further elaborate and validate this maturity model.

References

Abran, A., Bourque, P., Dupuis, R. & Moore, J. (2004), *Guide to the Software Engineering Body of Knowledge-SWEBOK*, IEEE Press.

Cerbin, W. (1994), 'The course portfolio as a tool for continuous improvement of teaching and learning', *Journal on Excellence in College Teaching* **5**(1), 95–105.

Chrissis, M., Broekman, B., Shrum, S. & Konrad, M. (2003), *CMMI: Guidelines for Process Integration and Product Improvement*, Addison-Wesley Professional.

Curtis, B., Hefley, W. E. & Miller, S. A. (2001), *People Capability Maturity Model (P-CMM) Version 2.0*, Addison-Wesley Professional.

Frailey, D. & Mason, J. (2002), Using SWEBOK for education programs in industry and academia, *in* 'CSEE&T'02: Proceedings of the 15rd Conference on Software Engineering Education and Training', IEEE Press, pp. 6–10.

Harris, A. (1998), 'Effective teaching: a review of the literature', *School Leadership & Management* **18**(2), 169–183.

Institute, S. E. (1995), *The Capability Maturity Model: Guidelines for Improving the Software Process*, Addison-Wesley Professional.

ISO/IEC (1998), Software process assessment, Technical Report TR 15504:1998, ISOSPICE.

Jalote, P. (2003), 'Needed: a capability maturity model for engineering education', *The Economic Times India* .

Kajko-Mattsson, M., Forssander, S. & Olsson, U. (2001), Corrective maintenance maturity model (CM^3): maintainer's education and training, *in* 'ICSE'01: Proceedings of the 23rd International Conference on Software Engineering', IEEE Press, pp. 610–619.

Kruchten, P. (2001), What is the rational unified process?, Technical report, Rational Software Canada.

Marshall, S. & Mitchell, G. (2002), An e-learning maturity model, *in* 'Proceedings of EDUCAUSE'02: 19th Annual Conference of the Australian Society for Computers in Learning in Tertiary Education'.

Marshall, S. & Mitchell, G. (2003), Potential indicators of e-learning process capability, *in* 'Proceedings of EDUCAUSE'03: 20th Annual Conference of the Australian Society for Computers in Learning in Tertiary Education'.

Marshall, S. & Mitchell, G. (2004), Applying SPICE to e-learning: an e-learning maturity model?, *in* 'ACE'04: Proceedings of the Sixth Conference on Australasian Computing Education', Australian Computer Society, Inc., Darlinghurst, Australia, Australia, pp. 185–191.

Marshall, S. & Mitchell, G. (2005), E-learning process maturity in the New Zealand tertiary sector, *in* 'Proceedings of EDUCAUSE'05: 22th Annual Conference of the Australian Society for Computers in Learning in Tertiary Education'.

Neuhauser, C. (2004), 'A maturity model: Does it provide a path for online course design?', *The Journal of Interactive Online Learning* **3**(1).

Neuhauser, C. (2005), A five-step maturity model for on-line course design, *in* 'Proceedings of the 19th Annual Conference on Distance Teaching and Learning'.

Reeves, J., Hugo, K., Heussner, R., Hala, A., Sarlioghu, B., Bialek, S. & Courter, S. (1998), Course portfolios: A systematic mechanism to document teaching and learning, *in* 'Proceedings of the 28th ASEE/IEEE Frontiers in Education Conference', IEEE, Tempe, AZ.

Robins, A., Rountree, J. & Rountree, N. (2003), 'Learning and teaching programming: A review and discussion', *Journal of Computer Science Education* **13**(2), 137–172.

Schön, D. A. (1987), *Educating the reflective practitioner: toward a new design for teaching and learning in the professions*, Jossey-Bass, San Francisco.

Seldin, P. (2004), *The teaching portfolio: a practical guide to improved performance and promotion/tenure decisions*, 3rd edn, Anker Pub. Co., Bolton, Mass.