TO : Professor Z.G. Gong. Head of DCIS - FST
FROM : Kam H. Vat, Department of CIS – FST
SUBJECT : Feedback on GE-Driven Revised Program Proposal in DCIS
DATE : January 13, 2010
CC : Professor C.L. Philip Chen, Dean of FST; Professor E.H. Wu, Associate Dean of FST

Thanks for your e-mails dated 2010JAN11 and 2010JAN13, requesting for feedback and suggestions for the DCIS Revised Program Proposal in the occasion of our University's introduction of General Education (GE) program in the year 2011. On studying the proposal concerning the suggested changes in our existing undergraduate program (Software Engineering), I have organized my thinking and feedback for your reference as follows:

- *General Feedback*: First of all, I applaud our university's initiative to introduce GE courses in our program broadening the horizon of students' knowledge and their preparation in a lifetime of profession in Software Engineering. The GE-driven process of program revision in our Software Engineering undergraduate program has given us faculty the opportunity to reflect on what we have been doing to empower student learning in the field of Software Engineering. It is really important for us to examine how best we could tailor the most suitable and compatible "Computing Curricula" in undergraduate education nourished by the GE elements, and administered by the Department of Computer and Information Science.

Essentially speaking, I believe there is an imminent need to have major revisions in the curricular offerings of our CIS programs. Currently, our undergraduate program, leading to the degree of Bachelor of Science in Software Engineering, is not in par with the substance expected in a Software Engineering curriculum as recommended by the SE2004 curriculum guideline (http://sites.computer.org/ccse/). Instead, it looks more like a Computer Science (CS) curriculum, but largely short of modern CS elements as recommended by the CS2008 Curriculum Update (http://www.acm.org/education/curricula-recommendations).

To fix the curriculum for a proper undergraduate program in Software Engineering (which is long overdue), and to keep going the intended direction in offering an undergraduate program in Computer Science, it is suggested that DCIS is committed to expand our programs into two specializations, Software Engineering, and Computer Science. We should not just change the title of the bachelor degree conferred from Software Engineering to Computer Science in an attempt to solve the problem. This is not the way it is supposed to be in a serious revision of our program. We are professionals accountable to sustain a quality college education enhancing the competitiveness of Macau in the global community.

Besides, we still have to account for what is entailed in the context of Information Science, as implied in the name of our department, CIS. We are obliged to interpret our perceived vision and mission of Information Science to the Macau public, or at least to the university community. If we are serious about offering courses in this area, it should be done from the perspective of a specialization, leading to the degree of Bachelor of Science in Information Science. This is indeed a challenge to be accomplished, as no such curriculum has ever been vigorously investigated under any ACM and/or IEEE task forces (www.acm.org/education/curric_vols/CC2005-

March06Final.pdf). In other words, we have no international guideline to reference in the design and assessment of the Information Science curriculum.

- *Specific Feedback: Item by item according to the proposal*

  Item 1 – FST as the name of Academic Unit;

  Item 2 – Degree Title: One way to resolve the potential struggle between Software Engineering and Computer Science is to offer specialization tracks, leading to the degree of *Bachelor of Computing with specialization in Computer Science*, and *Bachelor of Computing with specialization in Software Engineering*. It is also a practical idea to conceive the latter as a professional degree program, leading to a degree of *Bachelor of Engineering in Software Development*. Not the least, we should also include the degree of *Bachelor of Computing with specialization in Information Science*.

  Item 3 – Program Name: Our current undergraduate program name is *Software Engineering*, but this name is not reflected in the name of our department renamed from *Software Engineering* to *Computer and Information Science*, without any traceable records of rationales for justification under what reasons (this is not a good practice for such an important change). But it did happen. In fact, our program exists before the installation of our department. The implication is that the name of the department is not necessarily indicative of the program(s) offered; so, even though our department is called CIS, it does not matter that we do not offer any program in Information Science, or we just offer program in Software Engineering. The underlying assumption is that as a department, we are free to offer any suitable programs of study, as long as the programs are inclusively compatible with the department name.

  Item 4 – Major Title: Our current major title is *Software Engineering* because it is the program we offer. Now if we do resort to the specialization tracks of a) Software Engineering, b) Computer Science, and c) Information Science, then these could well become the major titles under our DCIS department; namely, our Major Titles include: *Software Engineering*, *Computer Science*, and *Information Science*.

  Item 5 – Specialization/Area of Concentration: It is suggested, as mentioned above, that DCIS should install the specialization options in at least three areas: *Software Engineering*, *Computer Science*, and *Information Science*, for good reasons, among which include the choices UM could offer to our prospective local students in terms of quality computing education, without going anywhere away from Macau. That is an important university vision and mission for Macau future generations.

  Item 6 – Rationale of Program Revision: As a committed instructor in college education in the area of Computing, I have borne in mind several essential rationales in this program revision:

  1. *Dilemmas of Current Program*: Our current undergraduate *Software Engineering* program was established more than 15 years ago, transplanted from a five-year engineering program in the Portuguese educational system. It is really time to review the substances of our program so as to provide the best possible training/education to our students in order to prepare their specialization in related professional areas such as industrial software development or academic computer

science research and development. It is also my hope that our program could eventually be accredited by international bodies, be it in the context of software engineering, computer science, information systems, information technology, or information science. But, we need to articulate our program philosophy and objectives pragmatically and clearly so as to convince both the local and the international communities that we have something really good to offer, be it from the perspectives of program context, content and structure, or from the angle of teaching and learning as well as assessment methods.

2. *Promise of Student-Centered GE Education*: Our current GE movement requires of each program to make room for the compulsory GE courses to broaden students' horizons in preparation for personal/professional development as a well-rounded person. This is part of the essence of a student-centered education. Thereby, we need to think about how to educate our students, not just in terms of knowledge or theories, but also there is a society in which they are preparing their journeys (into professional software practitioners, I mean), so that they must become an integral part of contributions to our community. In this light, our program must cultivate effective classroom teaching, which requires our professional commitment in the teaching act, founded on the planning and implementing of learning-centered educational practices and timely assessment of student performance.

3. *Issue of Professional Practice*: An important principle of the SE2004 curriculum guideline is that the education of all software engineering students must include student experiences with the professional practice of software engineering. It is expected that graduates of software engineering programs need to arrive in the workplace equipped to meet these challenges and to help evolve the software engineering discipline into a more professional and accepted state. Students, for example, need to understand the importance of professional conduct on the job and the ramifications of negligence. They also need to recognize that professional societies (ACM, IEEE, AITP, AIS world-renowned bodies) through their codes of ethics and established subgroups emphasizing professional practice, can provide a support network that enable them to stand up for what is ethically right. It seems, through my conversations with many of the past and current students, that our internship program offering summer jobs to students in Macau, can hardly provide opportunities of training genuine enough to come close to this expectation of professional practice. Many of the summer jobs are actually clerical in nature, instead of jobs providing opportunity to apply their learning in the field. We need to create opportunities of professional practice for our students, and recruit professional mentors in the field coaching our students, just as accounting students accumulating their experience from certified accountants in the field of auditing. It is believed that if we were to seek for international accreditation from recognized bodies for our *Software Engineering* program, we need to work out this issue of earning professional practice for students during their four-year education in our program.

Item 7 – Summary of Program Revision: Apart from the enumerated revisions proposed, I have the following additions underlining my specific concerns:

1. *To incorporate new general education courses into our current curriculum as recommended by the university.*

   Professional, legal and ethical issues are important elements in the overall curricula for computing disciplines, and must be integrated throughout the program of study. This context should be established at the onset and these matters should appear routinely in discussions and learning activities throughout the curriculum. The ACM Code of Ethics notes that "when designing or implementing systems, computing professionals must attempt to ensure that the products of their efforts will be used in socially responsible ways, will meet social needs, and will avoid harmful effects to health and welfare." Such Code goes on to provide an excellent framework for conduct that should be fostered beginning early in students' experiences. Unfortunately, in our undergraduate software engineering program, there has never been such a course installed to deal with this important concern for our student learning. It is suggested that we should propose a GE course in this specific area for the benefits of our students.

2. *To adjust the number of credits and the number of teaching hours for the major and elective courses, as well as final year projects. Contents of the courses would be streamlined or restructured possibly*.

   Although Software Engineering initially emerged as an area within Computer Science, it has begun to develop as a discipline unto itself. Originally, the term software engineering was introduced to reflect the applications of traditional ideas from engineering to the problems of building software. As software engineering matured over the past decades, the scope of its challenge became clearer. In addition to its computer science foundations, software engineering also involves human processes that, by their nature, are harder to formalize than are the logical abstractions of computer science. Experience with software engineering courses within computer science curricula showed that such courses can teach students about the field of software engineering, but usually do not succeed at teaching them how to do software engineering in the field. Many experts conclude that the latter goal requires a range of coursework and applied project experience that goes beyond what can be added to a computer science curriculum. In the context of our program, there is only one course in the junior year, called *SFTW 300 Software Psychology*, to deal with human aspects in software development. It is thus largely insufficient in terms of students' preparation in the whole curriculum of software engineering. It is suggested that *SFTW300* as a compulsory course be renamed *Human Aspects of Software Engineering*, and this course should be assigned 4 instead of 3 credits to provide students more hours to experience the *doing* rather than the *listening* of software engineering in context. Meanwhile, to allow existing learning of human-computer interaction (HCI) in *SFTW300*, it is suggested to create a new junior compulsory course named *Human-Computer Interaction Fundamentals* because this is a field that is growing very maturely in the context of both Software Engineering and Computer Science today, and it must not be ignored.

3. *To modify the list of our currently offered courses, by adding, merging or eliminating some existing courses wherever appropriate.*

It is understood that our core computing courses in the current degree program are yet to be constructively aligned to strengthen student learning. Typically, in a well-conceived Computer Science (CS) curriculum, we should have the CS 1-2-3 coursework installed to provide a common core body of knowledge (BoK) in computing to our students. The course sequence is typically designed such that students progress in knowledge, proficiency and professional maturity. Such courses represent the basic competencies expected of our students. For example, the progression of software engineering topics across CS 1-2-3 is often positioned as:

- Originating in CS 1, where there is an emphasis on using a cyclic approach for program development by iterating through design, coding, and testing program modules. Complemented by algorithm analysis, students are encouraged to think abstractly about problems and to begin developing processes for decomposing problems into organized parts. Encouraging clear documentation, good naming conventions and consistent secure coding style contribute to a discipline approach to writing programs.
- Continuing in CS 2, where greater emphasis is placed on abstraction and sound software design principles, engaging students in the development of secure software components that could solve a wide range of related problems. By their very nature, these components must be well-documented to support reuse. Thereby, students are expected to write assertions such as preconditions and post-conditions describing each component behavior, thereby, encouraging students to think deeply about a simple problem before coding. After coding, the components must be well-tested, and therefore the use of test plans and test drivers are practiced. These activities reinforce the notion of constructing software from well-defined, independent pieces and complement the study of using existing library classes and APIs in software solutions.
- Concluding in CS 3, where students are asked to step beyond the programmer role and take a broader view of software development; to consider its lifecycle from problem description to maintenance. Here students have experience with analysis and design of medium-sized systems. Standard modeling tools are introduced and the students complete the phases of analysis, design, implementation and testing of a medium-sized team project that includes documents such as UML diagrams or CRC cards in addition to test plans. Oftentimes, students consider design patterns and write applications using data structures and templates.

To empower our students' learning along the above CS 1-2-3 sequence (suggested by ACM Computing Curriculum 2008 update) represents the necessary efforts we must exert to ensure that our students should acquire the essential computing competencies throughout their first two years of study. However, in our current program, we are yet to design a coherent sequence of freshman and sophomore courses to achieve this purpose. Consequently, many of the students coming to *SFTW 241 Programming Languages Architecture (I)*, in the second semester of their sophomore year, are often found to be largely under-prepared in such competencies. It is convinced that the essential question to ask is this: What do students need to know to derive maximum benefit from their computing educational experience?

Thereby, it is suggested that in the revised program proposal we could strengthen SFTW241 and its sequel course SFTW342 along the CS 1-2-3 scheme so that instead of allowing SFTW342 to be an elective course, it should be made another compulsory

course, to make a stronghold two-course sequence to build up students' programming competencies. It is also suggested that the two courses be renamed as *SFTW241 Object-Oriented Programming Fundamentals*, and *SFTW342 Object-Oriented Programming Practice*. Besides, it is important to follow up these core courses with elective major courses like *Aspect-Oriented Programming*, and *Programming the World Wide Web*, that could be test-run under the auspices of our Special Topics in CIS courses one in junior year and the other in senior year.

4. *To update our courses syllabus in order to catch up with the latest development in both academic and practical industries.*

▪ *Adopt an outcomes-based approach*: Our current CIS undergraduate degree program is yet to articulate clearly the intended learning outcomes (ILOs) for our students at the end of their study. Indeed, an ILO is a statement describing what and how a student is expected to learn after exposure to teaching. ILOs apply at the institutional level, as a statement of what the graduates of the university are supposed to be able to do; at the degree program level, as a statement of what graduates from particular degree programs should be able to do; and at the course level, as a statement of what students should be able to do at the completion of a given course. It should be noted that graduate attributes as articulated by the university can provide useful guidelines for designing program outcomes, which, in turn, are addressed by the outcomes of specific courses under the program. Oftentimes, it is important to stipulate the kind of knowledge to be learned, declarative or functioning, and to use a verb and a context that indicates clearly the level at which it is to be learned and how the performance is to be assessed.

▪ *Install a departmental assessment program*: Our current CIS undergraduate degree program needs badly a well-crafted assessment program to ensure the quality of our student learning. Such an assessment program, long overdue in our program, should ensure a meshing of goals, instruction, and assessment. Simply put, to ensure that every student has the opportunity to reach the required level of proficiency in each area that has been identified, several major tasks on our part, must be accomplished:

a) A list of the basic competencies for all students must be developed and approved by the University;
b) These competencies must be described in terms that are measurable and demonstrable;
c) A comprehensive plan must be developed to make sure that the basic competencies are learned and reinforced throughout the entire time a student is enrolled;
d) Each disciplinary area responsible for a portion of the core curriculum must describe and include in the goals of each of their courses the appropriate learning outcomes that it introduces or reinforces;
e) For these courses a consistent set of assessment techniques and instruments must be developed;

Thereby, a curriculum must be developed sequentially, beginning with our institutional statement of goals and ending with the assessment of each student before and after graduation. As we move through the design process, from defining program goals to developing course goals and then unit-by-unit objectives, the statements become increasingly specific. The design of each course, the selection of instructional methods, and student assessment will be based on these statements. The process of moving from a statement of goals to deciding on and implementing a program and related individual

courses to the curriculum requires careful planning. If, for example, computer programming skills are identified as a basic competency that each student in Software Engineering majors, must have by graduation, programming with specific computer languages must be initially taught and then reinforced, and no student should be able to graduate without receiving appropriate instruction and practice in this skill. Courses must also be analyzed to identify where this skill is introduced and then reinforced, and the curriculum must be structured so that every student has the opportunity to acquire computer programming skills. Once all key competencies are determined, a curriculum committee might use a basic competency checklist (mostly a matrix) to assign specific competencies to individual courses or other formal learning experiences. The same checklist could also describe the level at which the competency will be taught, indicating in which courses the competency will be introduced, used, further developed, and assessed. If we are responsible for one of the required courses in our program, one of our major responsibilities will be determining which of the basic competencies can be taught or reinforced within it. Developing fundamental competencies such as teamwork, creative problem solving, collaborative project development, can be an integral part of most courses. Most importantly, under this assessment program, no student should be able to pass through our curriculum without having the opportunity to learn and use each of the identified core competencies.

Item 8 – Comparison between original study plan and revised study plan/curriculum:

Regarding the four points enumerated here in the proposal, I am not so sure what they mean because there are no explicit items singled out for comparison here. Namely, no specific data have been shown to support the claims which are very vague, too. I would surely suggest the use of total credit hours as a starting point, and then the schedule of courses taken semester by semester as another comparison, followed by the rationales of why such arrangements.

For example, if it is expected that a typical student is taking not more than five courses per semester (as against the six to eight courses currently in our Software Engineering program), and each course occupies four contact hours, then it is estimated that out of 5 x 8 hours per weeks on campus, an average student would spend 20 contact hours in classes and 20 hours left doing studies on campus, say in the library or other co-curricular activities. This is a good supply of hours to participate in group-based project work compared to the current situation, in which an average student could hardly make time to do out-of-class project work given the many hours (close to 40) in attending lectures or labs.

Item 9 – Transitional arrangement for students following the original study plan:

We ought to figure out some transition plans for students to choose by themselves. There are many possibilities here given the flexibility allowed to *draw the line*.

Item 10 – Revised Program Curriculum (Study Plan):

After reviewing the revised study plan, I have the following feedback:

1) Technically, the plan has integrated the GE requirements into the overall curriculum. But, I am wondering what the revised plan has achieved for student learning. I could still see a schedule that is loading our students with seven to nine courses per semester with the exceptions of the freshman year and sophomore year. What is our philosophy of curriculum design, taking the current dilemmas into account in this revised study plan?

2) Regarding the specific courses under my teaching assignment, I notice that SFTW241 and SFTW342 are gone, and there appear only one elective *Programming Languages* course to be offered in the first semester of the junior year. It is understood that in the original curriculum, SFTW241 is supposed to be the core language preparation for students if they want to follow up the study in the subsequent SFTW341 Compiler Construction, together with the concurrent SFTW342 (sequel to SFTW241). In SFTW241, we study the languages of C++ and Java, based on students' prior knowledge in ANSI C acquired in SFTW120 Programming Science. Now in the revised program, students are to take the *Compiler Construction* course in the second semester of sophomore year, with only one semester of ANSI C programming training in the first semester of freshman year. Yet, the current SFTW341 Compiler Construction course requires of each student the specific knowledge of Java and object-oriented programming. So, the question is: Where should our students go to acquire the pre-requisite knowledge to do study in the *Compiler Construction* course in the revised study plan? This is an indication of design inconsistency, whose solution is suggested in my earlier comments related to Item 7 (4) concerning the installation of a departmental assessment program. Such an assessment program should constantly detect such inconsistency in shaping our curriculum to enable student learning. Hence, it is strongly suggested that in the revised program, the course *Programming Languages* should be included as early as the first semester in the second year and expanded into a two-course sequence as suggested in Item 7 (3) under the course titles of *Object-Oriented Programming Fundamentals*, and *Object-Oriented Programming Practice*.

Item 11 – Structure of the Proposed Program:

Given the table of course and credits requirements for students to fulfill before graduation, I am quite reluctant to accept this before we could iron out some specific philosophy in shaping our curriculum. Technically, it is not appropriate to comment on this arrangement until DCIS has a clear idea of where we would like our students to achieve at the end of their four years of study. Please refer to my comments in Item 7 (4) coupled with my perspectives below:

1. A learner-centered education is marked by a quality learning experience.
For some time since 2007, we have been tackling the idea of providing a student-centered education in FST. The student-centered context focuses attention on learning: what the student is learning, how the student is learning, conditions under which the student is learning, whether the student is retaining and applying the learning, and how current learning positions the student for future learning. The student is an important part of the equation. Indeed, we make the distinction between student-centered instruction and teacher-centered instruction as a way of indication that the spotlight has moved from teacher teaching to student learning. When instruction is student-centered, the action focuses on what students (not teachers) are doing. Since the instructional action now features students, the student-centered orientation accepts, cultivates, and builds on the ultimate responsibility students have for learning. Teachers cannot do it for students. They may set the stage, so to speak, and help out during rehearsals, but then it is up to students to perform, and when they do learn, it is the student, not the teacher, who should receive accolades.

I personally prefer the term *learner-centered* instead of *student-centered* although both implies a focus on student needs. Yet, being student-centered carries the connotation that might give rise to the idea of education as a product, with the student as the customer and

the role of the faculty as one of serving and satisfying the customer. Faculty members resist the student-as-customer metaphor for some very good reasons. When the product is education, especially in a four-year undergraduate degree program, the customer cannot always be right; there is no money-back guarantee, and tuition dollars do not "buy" the desired grades. Nonetheless, the learner-centered approach orients to the idea of "product quality" constructively. Being learner-centered is not about cowering in the competitive academic marketplace. It is not about kowtowing to student demands for easy options and is not about an ethically irresponsible diminution of academic standards in an attempt to placate "shoppers" who may opt to purchase educational products elsewhere. It is about creating climates in classes and on campus that advance learning outcomes. It is an orientation that advocates for more, not less learning. It is about offering a quality learning experience.

2. *Outcomes-based assessment is the essence of a learner-centered education.*
The practice of outcomes-based assessment (OBA) is not a new movement, and it is related to an educational model in which curriculum and pedagogy and assessment are all focused on student learning outcomes. It is an educational process that fosters continuous attention to student learning and promotes institutional accountability based on student learning. Operationally speaking, the OBA model emphasizes such important practices as: Faculty publicly articulating assessment information in advance of instruction; students being able to direct their learning efforts to clear expectations; and students' progress and acquisition of learning being determined by evidence demonstrated in achieving the learning outcomes. So, the key component in the OBA model is outcomes which inform curriculum, teaching and assessment. In this light, it is convinced that if we are truly embracing a learner-centered model of undergraduate education, outcomes-based assessment is the necessary essence not to be ignored. One of the most important conclusions about the effect of outcomes on student learning comes from the studies of John Biggs (2007). Biggs found that student achieve deep learning when they have outcomes on which to focus. If students do not know what is important to focus on in their studies, they try to cover all the information, so they skim, they cram, and they stay on the surface. If they have a priority or focus, they are able to dig, to expand, and to achieve depth of understanding.

I agree with Linda Suskie (2009) that assessment for student learning should constitute an ongoing process of learner-centered activities including: a) establishing clear, measurable expected outcomes of student learning; 2) ensuring that students have sufficient opportunities to achieve those outcomes; 3) systematically gathering, analyzing, and interpreting evidence to determine how well student learning matches the expectations; and 4) using the resulting information to understand and improve student learning. Tellingly, the four steps enumerated do not represent a once-and-done process, but a continuous four-step cycle. Namely, in the fourth step, assessment results are used to review and possibly revise approaches to the other three steps, and the cycle begins anew. Today, assessment is often viewed as part of an integrated, collaborative learning experience. It is convinced that students learn better when their college experiences are not collections of isolated courses and activities but are purposefully designed as coherent, integrated learning experiences in which courses and out-of-class experiences build on and reinforce one another. Understandably, when students can see connections among their learning experiences, their learning is expected to be deeper and more lasting. Still, the question we need to ask is: How coherent is our current undergraduate CIS degree program to produce integrated quality learning experiences for our students?

Biggs, J. & Tang, C. (2007). *Teaching for quality learning at university, 3e.* Buckingham, Great Britain: Open University Press.
Suskie, L. (2009). *Assessing student learning: A common sense guide, 2e.* San Francisco, CA: Jossey-Bass.

3. *Keeping evidence of student learning is an institutional responsibility.*
As an act of public accountability, our CIS programs must cultivate effective classroom teaching, which requires our professional commitment in the scholarship of teaching and learning, founded on the planning and implementing of learner-centered instructional activities and timely assessment of student performance. In practice, the application of the OBA model of education must be demonstrated through an approach to student learning founded on a variety of knowledge building processes (relevant to software engineering and computer science), and that teaching should encourage students to work actively towards understanding within a framework of personal responsibility and institutional freedom. So, keeping a portfolio of student activities and performance starting from right after his or her admission becomes important. We should be interested not just in the grade earned at the end of each semester's work in a specific course, but also the full record of academic work performed preferably with both peer-based and instructor-based comments gathered from some retrievable coursework evidence (such as documents or work products in electronic form) to be included in the student's portfolio as one of his or her lifelong records in learning, subject to the authentication and keeping by the university. Surely, it is, in one sense, the student's resume in the making, but realistically it is a person, a professional in the making, indeed.

4. *Support Personal Development Planning (PDP) and Portfolio Building*
PDP is a structured and supported process undertaken by an individual to reflect upon their own learning, performance and/or achievement and to plan for their personal, educational and career development. In an academic setting, it is expected that all students must be offered the opportunity to engage in the processes of PDP at each and every stage of their degree program. The primary objective for PDP is to improve the capacity of individual students to: understand what they have learned; how that learning was acquired; and take responsibility for their own future learning through the processes of planning and reflection. In so doing, students may be expected to become more effective, independent and confident self-directed learners. Moreover, students should be able to improve their general skills for study and career management, to articulate personal goals, and to evaluate progress towards their achievement. Consequently, through the introduction of PDP in higher education, students may begin to respond to their educational experience in different ways, becoming less passive and adopting a more critical and evaluative approach.

   In the UK, the PDP initiative is supported by a Web-based personal and professional development planning tool, called the RAPID (Recording Academic, Professional & Individual Development) Progress File (http://rapid.lboro.ac.uk). It enables registered users to input and maintain information on a password-protected database. The RAPID Progress File has two main components: PACE and SPEED. The former component acts as the record of achievement. Here, students maintain their personal details, a record of their achievements (including qualifications), a summary of relevant work experiences and a personal statement that they are encouraged to keep up to date. The information in PACE is downloadable to enable a Curriculum Vitae (CV) to be produced from the data stored. The SPEED component of RAPID contains up to 60 separate skills compatible with the professional competence requirements of the respective professional institution. Each skill is presented in the same format. The template offers the student four

statements, ranging from a fairly low level to a high level of competence in the skill. Students are expected to self-audit their competence in a broad range of the skills offered and to record evidence to support the claim of competence that is made. In particular, students are encouraged to engage in the SPEED skill development process that involves action planning to develop their skills, reviewing and reflecting upon the development activities undertaken and documenting evidence of competence gained. This process mirrors a similar process required for the completion of most competence-based professional development programs.

In fact, when students are engaged in consistent production of evidence to demonstrate progress completion of different sets of learning outcomes for a course, for a major, or as a comprehensive summative assessment for meeting graduation requirements, the mission of any software tool, such as RAPID Progress File, in the UK example, is to facilitate the organizing of such evidence in the form of portfolios as a reflective way of assessing student learning. Simply put, portfolios are collections of student evidence accompanied by a rationale for the contents and by student reflections on the learning illustrated by the evidence. Portfolios are considered best when they are planned and purposeful and contain evidence of student efforts, progress, and achievement.

5. *Develop Effective Educational Practices suitable for assessing students' cumulative learning throughout the undergraduate curriculum of study*
   To engage our students in their pursuit of an excellent education at UM, effective educational practices must be developed and deployed to help them accomplish their essential learning outcomes expected of our UM graduates. Thereby, it is important that our school or department in which the curriculum is offered makes sure that each learning experience successfully meets the instructional goals that have been assigned to it. One of the most problematic areas perceptible is that many core competencies, despite being vital to the curriculum, are often not the primary foci of the courses delivered. Take active learning as an example, which is well known for its positive influence in our student learning; however, the conventional transmissive model of education is still the dominant mode of course delivery and the basis for course evaluation. We are still more concerned with providing instructions than with producing student learning. Elaborated below are some examples of effective educational practices considered of importance to our college students, and widely practiced in many a university campus around the globe, but they are yet to be fully realized in our program (AACU, 2007, p.53-54):

   - **First-Year Seminars and Experiences**
     Many schools now build into the curriculum first-year seminars or other programs that bring small groups of students together with faculty or staff on a regular basis. Typically, such first-year experiences place a strong emphasis on critical inquiry, frequent writing, information literacy, collaborative learning, and other skills that develop students' intellectual and practical competencies. First-year seminars typically can involve students with cutting-edge questions in scholarship and with faculty members' own research.

   - **Common Intellectual Experiences**
     Today, the older idea of a core curriculum has evolved into a variety of modern forms such as a set of required common courses, or a vertically organized general education program that includes advanced integrative studies and/or required participation in a learning community. These programs often combine broad themes,

such as technology and society, or global interdependence, with a variety of curricular and co-curricular options for students.

- **Learning Communities**
  The key goals for learning communities are to encourage integration of learning across courses and to involve students with big questions that matter beyond the classroom. Students take two or more linked courses as a group and work closely with one another and with their professors. Many learning communities explore a common topic and/or common readings through the lenses of different disciplines. Some deliberately link liberal arts and professional courses; others feature service learning.

- **Writing-Intensive Courses**
  These courses emphasize writing at all levels of instruction and across the curriculum, including final-year projects. Students are encouraged to produce and revise various forms of writing for different audiences in different disciplines. The effectiveness of this repeated practice across the curriculum could lead to parallel efforts in such areas as quantitative reasoning, oral communication, information literacy, and on some campuses, ethical inquiry.

- **Collaborative Assignments and Projects**
  Collaborative learning combines two key goals: learning to work and solve problems in the company of others, and sharpening one's own understanding by listening seriously to the insights of others, especially those with different backgrounds and life experiences. Approaches range from forming study groups within a course, to team-based assignments and writing, to cooperative projects and research.

- **Undergraduate Research**
  With support from different international research communities, many a faculty member is encouraged to reshape his or her courses to connect key concepts and questions with students' early and active involvement in systematic investigation and research. The goal is to involve students with actively contested questions, empirical observation, cutting-edge technologies, and the sense of excitement that comes from working to answer important questions. These educational practices are part of a broader movement to provide research experiences for students in all disciplines.

- **Diversity/Global Learning**
  Many colleges and universities now emphasize courses and programs that help students explore cultures, life experiences, and worldviews different from their own. These studies often explore difficult differences such as racial, ethnic, and gender inequality, or continuing struggles around the globe for human rights, freedom, and power. Frequently, intercultural studies are augmented by experiential learning in the community and/or by study abroad.

- **Service Learning, Community-Based Learning**
  In these programs, field-based experiential learning with community partners is an instructional strategy – and often a required part of the course. The idea is to give students direct experience with issues they are studying in the curriculum and with ongoing efforts to analyze and solve problems in the community. These programs model the idea that giving something back to the community is an important college

outcome, and that working with community partners is good preparation for citizenship, work, and life.

- **Internships**
  Internships are another increasingly common form of experiential learning. The idea is to provide students with direct experience in a work setting – usually related to their career interests – and to give them the benefit of supervision and coaching from professionals in the field. If the internship is taken for course credit, students should complete a project or paper that is approved by a faculty member.

- **Capstone Courses and Projects**
  These courses require students nearing the end of their college years, to create a project of some sort that integrates and applies what they have learned. The project might be a research paper, a performance, a portfolio of best work, or an exhibit of professional accomplishments. Capstone courses, representing culminating experiences, are offered both in departmental programs and, increasingly, in general education programs as well.

AACU (2007). *College Learning for the New Global Century*: A Report from the National Leadership Council for Liberal Education and America's Promise (LEAP). Washington, DC: Association of American Colleges and Universities (AACU).

Item 12 – Course description of each new/revise course:

According to my comments above, the following courses need to be updated for the revised program of study:

Course Code	Current Title	Revised Title
SFTW 241	Programming Languages Architecture (I)	Object-Oriented Programming Fundamentals
SFTW342	Programming Languages Architecture (II)	Object-Oriented Programming Practice
SFTW300	Software Psychology	Human Aspects of Software Engineering

And the new course *Human-Computer Interaction Fundamentals* must be included, as shown already in the revised curriculum. The revised syllabi of these courses are prepared in a separate document to be submitted.

Finally, it is my conviction that the curriculum is the heart of a student's college experience. It is a university's primary means of helping students develop in directions valued by its faculty. In today's Macau, we are being urged to assess especially carefully the quality of our curricula. We as faculty are responding to this challenge as a practical means of both attracting and retaining more students and ensuring their success and producing high-quality outcomes for Macau. Thereby, a curriculum should be based on a carefully thought-out philosophy of education and should be clearly connected to our institution's stated mission. Any curricula mission statement and written curricula goals (intended student learning outcomes or results) should articulate curricula purpose and aims – what graduates should know and be able to do and those attitudes and values our faculty believes are appropriate to well-educated men and women. These goals and their more specific objectives must be described in considerable detail and in behavioral language that will permit designing the curriculum and assessing its degree of achievement (its actual outcomes).

With that note, please kindly accept, once again, my sincere thanks for the opportunities to share with you my comments and findings in this brief memo in the occasion of our university's data gathering for our input in the revised study plan for our DCIS undergraduate degree programs.

All the best!

Kam Hou Vat
Department of Computer & Information Science
Faculty of Science & Technology
University of Macau, Macau
http://www.fst.umac.mo/en/staff/fstkhv.html
e-mail: fstkhv@umac.mo