IDEA GROUP PUBLISHING
701 E. Chocolate Avenue, Hershey PA 17033, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com
ITP4312

Conceiving Service-Based Architecture and Process for Quality Software Education

Kam Hou Vat
Faculty of Science & Technology, University of Macau, Macau, fstkhy@umac.mo

ABSTRACT

This paper describes the initiative to incorporate the practice of quality software education (QSE) into our undergraduate curriculum concerning the engineering of software. Specifically, we will expound the idea of component-based development, which is supported by the software industry's emerging consensus that components provide the kind of building blocks we need for developing today's complex systems. Particularly, the component-based technology asks of us the required portions of productivity, quality, and rapid construction of software artifacts. Consequently, our pedagogic approach to QSE focuses on designing and building a sensible service-based architecture characterized by objects of different services, which represent the cohesive collections of related functionality. We outline our QSE approach in terms of a service-based development process for both the solution construction and the components building, through which our students could learn to acquire their collaborative software engineering experience in the current practice of architected application development. The paper concludes by discussing how the constructivist's method of problem-based learning helps develop this QSE practice into our students' daily learning.

INTRODUCTION

The overall picture confronting today's enterprises could be characterized as this [2, 7, 8]: at the core is the installed base of existing information technology (IT) systems, which includes the legacy data and business logic. Around the edge are increasingly pro-active customers, to which the enterprise must offer an increasing quality of service through existing and new channels. In between, the enterprise is re-engineering its business processes, with a focus on knowing its customers better, and offering continuous improvement of its products and services. From an IT perspective, the legacy systems become surrounded by a matrix of go-between componentry providing services to support the changing business, with increased flexibility and reduced development times as compared with legacy systems. This often is the backdrop behind which most of our universities' undergraduate programs in Software Engineering have been running, despite the fact that it is not being consciously made known to our software graduates [12]. The quality software education (QSE) context we address is a component-based approach for developing enterprise systems with a large business composed of complex business processes. This paper describes the lessons we have learned experimenting and refining our QSE method on developing students as self-directed work teams of software professionals through practical projects of real-world requirements [26]. The important issues could be developed by considering the following "how-to's" [2, 6]: support increasingly adaptive businesses; capitalize on the rapid advances in component technology; deal with legacy systems; plan and build for reuse; prepare for quality issues; and retain a pragmatic focus in the face of increasing complexity. Collectively, they represent the drivers of change, worthy enough to secure a place in our discussion of quality software education.

DRIVERS OF CHANGE FOR QSE

Most businesses today are undergoing a period of rapid change, driven by trends such as business-process improvement and downsizing [7, 24]. In the past, the order of the day has been to re-organize the technology each time a business changes. Yet, the re-oriented consensus is to facilitate software solutions (technology) that adapt as the business adapts. This support for increasingly adaptive businesses is currently achieved through the reuse of business components [11], which are executable units of code that provide physical black-box encapsulation of related business services, accessed through a consistent, published interface that includes an interaction standard with other components. These business components support a process-based view of the business as it changes. Consequently, it is important to derive the business process models in order to provide a secure business foundation from which to develop component-based solutions, which are necessarily traceable back to originating business requirements. On the other hand, one of the main enabling technologies for component reuse is the standardization of distributed-object middleware [5], through which components can be moved around at execution time and deployed in a way that optimizes the technology in order to deliver the most business benefit. These advances in component technology have resulted in the movement toward separation of software applications from the increasingly heterogeneous technology platforms on which the services are deployed [3, 8]. It provides the potential for an application to be physically distributed so that it services the needs of the business and not the technology. We call this the service-based view of software construction, where components provide a method of packaging related services into pre-fabricated pieces of software from which solutions can be constructed. This service-based approach is also applicable in the area of legacy software where most development is about enhancing existing systems, providing new front-ends to established back-ends. It allows organizations to wrap the existing services into new offerings or products, so as to reuse their investments in existing packages, databases, and legacy systems within the context of component technology. Nevertheless, this is often a challenge requiring skilled and thorough design, taking into account such attributes as reliability, efficiency, usability, maintainability, testability, portability and the most essential reusability. In fact, it is worthy to point out that modern businesses today require applications that confer early user benefits at minimum cost, leveraging existing legacy systems where possible but not at the cost of maintainability, flexibility and reusability. Whether or not this spirit of pragmatism, could be incorporated into developing any software solution on a sound and evolving architectural base has been driving our initiative to introduce the concept of QSE to our curriculum of software development.

A SERVICE-BASED ARCHITECTURE FOR COMPONENT-BASED DEVELOPMENT

Our approach to component-based development, based on Allen and Frost [2], is evolutionary in nature. It is aimed at harnessing a service-based approach with effective object-oriented modeling to capitalize on the increasing power of the fast-developing component technology. It provides an overall design philosophy for realizing the vision of service-based reuse of components [2, 3, 8]. We call this philosophy the *service-based architecture* for component-based development [2, 9, 17, 20], which employs the concept of *service packages*, to facilitate a business-oriented modeling process. A service

This paper appears in *Issues and Trends of Information Technology Management in Contemporary Organizations,* the proceedings of the Information Resources Management Association International Conference.

package provides a set of services belonging to a single service category. The required services from a service package are provided through one or more service classes. This provides a business-oriented basis for modeling deployment of components using *service* (*component) packages*, which are implementation packages of objects, providing services through their interfaces. That way, components provide a means of packaging related objects together into pre-fabricated pieces of software. And service packages provide a mechanism for grouping those objects into units (in the form of components) that are cohesive to the needs of a particular set of services from which business solutions can be constructed. It is important to note that the service packages must be modeled in a way that makes the resulting components useful building blocks, simple to activate and inexpensive to administer. The level of granularity of a component can vary from large and complex to small and simple. In practice, large components have the greatest potential for reuse but are often not cohesive and may be difficult to assemble into solutions with other components. Small components are usually more cohesive but often need to be coupled with many other components to achieve significant reuse, resulting in excessive inter-component coupling. Clearly, settling on a good and useful level of granularity is a trade-off between these two extremes. Each organization should have an optimum level of granularity of component to best fit its own needs. Overall, the service-based architecture provides a framework for modeling that assists our efforts in QSE in a two-tiered process for architected applications development.

A SERVICE-BASED PROCESS FOR ARCHITECTED APPLICATIONS DEVELOPMENT

The term "process" as used in our pedagogic context for architected applications development, carries the connotation of process models designed to view the real world from the viewpoint of architectural software development [18, 23, 25]. This process could be considered as an abstract description of the software development activities within the service-based architecture. We are interested in a two-tiered approach to achieve this service-based methodology: the solution process, and the component process. The former is aimed at development of solutions, typically in terms of user services, to maximize reuse of existing services and provide early user value. The latter is aimed at developing components that provide commonly used business and/or data services across different departmental systems or for use by third parties. It is important to notice that we often need to use elements of both processes adapted to our own specific needs. Typically, the key driver of the solution process is a set of specific requirements to meet the needs of a business process. Various models are produced throughout the process, which evolve in detail as the process unfolds. Such models are often selected on a use-case by use-case basis for incremental development of user services. On the other hand, the generic business requirements that drive the component process may come from the need to reuse existing legacy assets, and the feedback from solution projects. An important part of the component process is to evolve the models so that they can be specialized and refined by solution builders to form an evolving set of components. The use of a process by a software development team should assist project management in numerous tasks. These include: identification and partitioning of work, identification of progress achieved, planning the staff resource profile, planning the requirement for physical resources, and provision of cost and time scale estimates for the work yet to be performed.

From a technical viewpoint [10, 13, 14], a process should assist in such areas as identification of preconditions required before each activity is started, specification of the products and deliverables required from each activity, techniques that may be used during each activity, and experience gained from earlier work. Clearly, building and refining generic models is an important aspect of component-based development where we want to leverage model reuse more than code reuse. Service packages provide a means of structuring a project in terms of architectural context and allow us to build on and capitalize on the very best work of others. A service package can be effectively employed in a component process to contain a generic model, which can be refined and extended to meet the specific needs of a solution process. The model solution space evolves to contain more detail as a project moves through the iterations of the process. Eventually, portions of the model are mature enough to be transformed into code. The tested code represents the model at its most detailed level of abstraction. As for deliverables, they are simply views of a maturing model. In practice, the service-based process for architected applications development is very much an adaptive process that can be tuned and customized to specific organizational needs. Checkpoints can also be built into the process to help evolve it. This includes documenting the lessons learned so that others can avoid making the same mistakes.

THE PBL MODEL OF INVESTIGATION

It is understood that collaborative project work [1, 4, 15, 16, 21, 22] is recognized as having many educational and social benefits, in particular providing students with opportunities for active learning. However, teaching, directing and managing group project work is not an easy process. This is because projects are often: *expensive* demanding considerable supervision and technical resources; and *complex* combining design, human communication, human-computer interaction, and technology to satisfy objectives ranging from consolidation of technical skills through provoking insight into organizational practice, teamwork and professional issues, to inculcating academic discipline and presentation skills. In preparing our students to get started with group-based project work to prepare for their future journey as software professionals, we have adopted problem-based learning (PBL) [1, 4] as our pedagogic approach. The notion of PBL is based on the premise that students learn more effectively when they are presented with a problem to solve rather than just being given instruction [15, 21]. Pedagogically, students have to identify and search for the knowledge they need to approach the problem. When applied in the course setting, PBL could be decomposed into several stages of activities [16, 22], which help develop in students, self-directed learning and problem-solving skills while they interact, discuss and share relevant knowledge and experience. At the *problem analysis stage*, students, divided into small groups and assigned a facilitator, are respectively presented a problem without any instruction given. They generate ideas about possible solutions to the problem based on what they already know. They then define what they need to know by identifying the key learning issues and formulate an action plan to tackle the problem. During the *information gathering stage,* a period of self-directed learning follows. Students are responsible for searching for relevant information. They are largely engaged in just-in-time learning as they are seeking for information when their need to know is greatest. Arriving at the *synthesis stage* after a specified period of time, students reconvene and reassess the problem based on their newly acquired knowledge. They become their own experts to teach one another in the group; they use their learning to re-examine the problem. In the process, they are constructing knowledge by anchoring their new findings on their existing knowledge base. In the *abstraction stage,* once the students feel that the problem task has been successfully completed, they discuss the problem in relation to similar and dissimilar problems in order to form generalizations. Finally, at the *reflection stage,* students review their problem-solving process through conducting a self- or peer-evaluation. This phase is meant to help students' meta-cognitive ability as they discuss the process and reflect on their newly acquired knowledge.

Essentially, PBL revolves around a focal problem, group work, feedback, class discussion, skill development and final reporting. The instructor's role is to organize and pilot this cycle of activity, guiding, probing and supporting students' initiatives along the way so as to empower them to be responsible in their own learning. Meanwhile, it is

important that PBL students are taught how to work in teams and positively experience the team process because the team skills they acquire are applicable throughout their future careers. The PBL team process we experienced, requires each team composed of 3-5 students, to be assigned a supervisor (instructor) and a client if applicable. The client's role is to clarify the project, and to resolve ambiguities as they arise, whereas the supervisor's is to guide, motivate and provide feedback to the team. Also, one of the team members is designated the team leader for the duration of the project, whose role is to coordinate the team activities, and to ensure effective team communications. The leader also has to interface with the supervisor, arrange meetings with clients when necessary, and facilitate meeting through setting agendas, taking minutes, and allocating tasks. Each team member has to help set the team goals, accomplish tasks assigned, meet deadlines, attend team meetings and take a turn editing a document to be submitted at the end of each major stage of project development.

Further, PBL students are made aware of the difficulties in teamwork throughout the project period. These include setting realistic project goals, carefully allocating tasks to team members, managing time, and communicating and managing shared group documents. Teams have regular meetings to which they invite their supervisor, and in which they organize themselves to manage the project. Students are often reminded of setting appropriate agendas before meeting, assigning enough time to the agenda items during meeting, restating the decisions made at the meeting, and converting decisions into action items after meeting. They are also advised on clearly separating the social and work aspects in meetings, and assessing each meeting for doing it better next time. Moreover, it is suggested that teams plan their project around major deadlines of individuals in the team thereby acknowledging the other commitments team members may involve. Deadlines represent the milestones set down for the PBL students to submit project documents and to receive evaluation. Each member is assessed by the project supervisor and the team peers. The supervisor's evaluation is based on what each team member adds to the meetings and what the instructor perceives each member's contributions to the team to be. The peers' evaluation is based on a confidential rating sheet, to be completed by each team member at the end of each major phase of the project. This rating sheet should include each team member's contribution for that phase with explanatory comments. And the overall project assessment is made up of the group grade and the individual grade. The former is the same for each group member and is based on the quality of the documents produced and the product developed. The individual component is based on the quality of the student's contribution to the documents and the product, their participation in group-meetings, their commitment to the team process, and their professional attitude developed.

CONCLUSION

It is experienced that the conventional approach to education remains the instructivist one, in which knowledge is perceived to flow from experts to novices. This transmissive view of learning is most evident in the emphasis on lectures, in the use of textbooks to prescribe reading, and in the nature of tutorials and assessment methods. Yet, this approach might deny students the opportunity to apply their learning to dynamic situations such as quality software development through team-based collaboration. We question the transferability of the instructivist learning and ask how much of that which is assigned to academic learning ever gets applied to actual scenarios, when there is such a rapid surge in knowledge commonly associated with the birth of the "Information Age." Actually, the content product of learning is assuming a less important role relative to the process of learning as the life of information content shortens and the need for continual learning increases. In designing the service-based process for QSE to be injected into our courses for software engineering, we have tried to reorient towards a meaningful direction by reducing the obsession with knowledge reproduction. And PBL represents one such relief from the constructivist pedagogy. Greening [15] describes it as a vehicle for encouraging student ownership of the learning activities. There is an emphasis on contextualization of the learning scenario, providing a basis for later transference, and learning is accompanied by reflection as an important meta-cognitive exercise; for example, assessing whether a project should be approached by a solution process or a component process. Also, the implementation of PBL is done via group-based work, reflecting the constructivist focus on the value of negotiated meaning [19]. More importantly, it is unconfined by discipline boundaries, encouraging an integrative approach to learning, which is based on requirements of the problem as perceived by the learners themselves. In conclusion, this paper has described our approach of quality software education through the service-based architecture for component-based development and the corresponding solution and component processes for architected applications development. We have also explained how to incorporate this QSE-based practice into our curriculum through the PBL pedagogy. Practically, when technology meets pedagogy, we agree that the education of component-based development should start off with the ability to build individual components efficiently. Then it will evolve through efficient construction of component-based solutions in new domains, efficient adaptation of existing solutions to new problems, and efficient evolution of installed solutions by people with limited technical knowledge. Finally, it will achieve the efficient integration and evolution of sets of solutions. The real challenge is to derive a coherent set of principles that will bring the whole of system development, including technology, infrastructure, distributed system architecture, methodology, and project management, into a single component-centric whole.

REFERENCES

[1] Albanese, M., and Mitchell, S., " Problem-Based Learning: A Review of Literature on Its Outcomes and Implementation Issues," Academic Medicine, Vol. 68, No. 1, 1993, pp. 52-81.

[2] Allen, P., and Frost, S., *Component-Based Development for Enterprise Systems: Applying the SELECT Perspective*, Cambridge University Press, 1998.

[3] Anderson, B., and Dyson, P., "Reuse Requires Architecture," In L. Barroca, J. Hall, and P. Hall (Eds.), Software Architectures: Advances and Applications, Springer-Verlag, Berlin Heidelberg, 2000, pp. 87-99.

[4] Barrows, H., " How to Design a Problem-Based Curriculum for the Pre-Clinical Years," New York: Springer, 1985.

[5] Berstein, p., "Middleware: A Model for Distributed System Services," Comm. ACM, Vol. 39, No. 2, Feb. 1996, pp. 86-98.

[6] Braude, E.J., *Software Engineering: An Object-Oriented Perspective*. John Wiley & Sons, Inc., 2001.

[7] Cook, M.A., *Building Enterprise Information Architectures: Reengineering Information Systems*, Prentice Hall PTR, 1996.

[8] Cook, S., "Architectural Standards, Processes and Patterns for Enterprise Systems," In L. Barroca, J. Hall, and P. Hall (Eds.), Software Architectures: Advances and Applications, Springer-Verlag, Berlin Heidelberg, 2000, pp. 179-190.

[9] Cox, B. *Object-Oriented Programming: An Evolutionary Approach*. Reading, MA: Addison-Wesley, 1986.

[10] DeMarco, T. *Controlling Software Projects*. Eaglewood Cliffs, NJ: Yourdon Press, 1982.

[11] Eeles, P., "Business Component Development," In L. Barroca, J. Hall, and P. Hall (Eds.), Software Architectures: Advances and Applications, Springer-Verlag, Berlin Heidelberg, 2000, pp. 27-59.

[12] Favela, J., and Pena-Mora, F., "An Experience in Collaborative Software Engineering Education," IEEE Software, Vol. 18, No. 2, March/April 2001, pp. 47-53.

[13] Gartner Group, *Best Practices in Application Development Project Management*, Part 2, SPA-650-1293. ADM Research Note, March 20, 1996.

[14] Gartner Group, "Rapid Application Development, Part 2: Organizing for Success," Inside Gartner Group This Week, June 7, 1995.

[15] Greening, T., "Emerging Constructivist Forces in Computer Science Education: Shaping a New Future?" In T. Greening (Ed.), Computer Science Education in the 21st Century, Springer 2000, pp. 47-80.
[16] Greening, T., "Scaffolding for Success in Problem-Based Learning," Medical Education Online, 3(4) 1998, pp.1-15. http://www.utmb.edu/meo/
[17] Jacobson, I., Christerson, M., Jonsson, P.M., and Overgaard, G. *Object-Oriented Software Engineering: A Use Case Driven Approach*. Reading, MA: Addison-Wesley, 1992.
[18] Microsoft Corporation. *Microsoft Solutions Framework: Reference Guide, Version 2.0*, 1996.
[19] Perkins, D.N., "What constructivism demands of the learners?" In T.M. Duffy & D.H. Jonassen (Eds.), Constructivism and the Technology of Instruction: A Conversation (pp. 161-165). Hillsdale, NJ: Erlbaum 1992.
[20] Repenning, A., Ioannidou, A, Payton, M, et al., "Using Components for Rapid Distributed Software Development," IEEE Software, Vol. 18, No. 2, March/April 2001, pp. 38-45.
[21] Ryan, G., "Student Perceptions about Self-directed Learning in a Professional Course Implementing Problem-Based Learning," Studies in Higher Education, Vol. 18, 1993, pp. 53-63.
[22] Savery, J.R., and Duffy, T.M., " Problem-Based Learning: An Instructional Model and its Constructivist Framework," Educational Technology, 35(5) 1995, pp. 31-38.
[23] Stapleton, J., *DSDM: Dynamic Systems Development Method – The Method in Practice*, Addison Wesley, 1997.
[24] Umar, A., *Application (Re)Engineering: Building Web-Based Applications and Dealing with Legacies*, Prentice Hall PTR, 1997.
[25] UML. *Unified Modeling Language Version 1.0.* Santa Clara, CA: UML Partners, 1997.
[26] Vat, K.H., "REAL: Towards a WWW-Enabled Course Support Environment for Active Learning," Faculty Technical Report, FST/SE-1999-01, University of Macau, Macau.