A PERCEPTION OF THE SOFTWARE PROCESS MODEL

Albert H. S. Scott
Hong Kong Institute of Vocational Education

Index: Perception; software; product; process; students; quality attributes

Abstract: A student's perception of a software development process model appears to be a series of milestones when documentation is produced to satisfy academic requirements. Certain academic and industrial practices may influence students to place more emphasis on the software product than the process model.

The proposal presented is to establish a more structured approach to project supervision to ensure a student performs quality planning, which includes the identification of appropriate software product quality attributes. Such an approach may ensure that sufficient attention is given to the software development process model to ensure that desired software product quality attributes are achieved.

INTRODUCTION

A student, who seeks vocational training, rather than undergraduate studies, normally has an aptitude for doing practical tasks. A major piece of work that computing students undertake is an individual software development project. The development of good quality software requires the adoption of a disciplined approach to all the tasks involved. This approach is in the form of a chosen methodology based upon a particular software development process model.

DEVELOPMENT PROCESS MODELS AND METHODOLOGIES

Sommerville (1995) has identified the following generic software development process models:

- The waterfall model - separate and distinct phases of specification and development
- Evolutionary development - specification and development are interleaved
- Formal transformation - a mathematical system model which is formally transformed to an implementation
- Reuse-based development - a system is assembled from existing components

Common methodologies include:

- Structured Systems Analysis & Design (SSADM) which is based on the waterfall model (Goodland and Slater, 1995).
- Mechanism for component reuse in Languages and Environments are: Borland Delphi; Microsoft Visual Basic; Microsoft Windows: CORBA and Java (Bennett et al., 1999).
- Rapid Applications Development (RAD) is a rapid linear sequential model based on component construction (Pressman, 1997).

- Unified Software Development Process for: visualizing, specifying, constructing and documenting object-oriented systems (Booch et al.,1998).

These methodologies/methods/software provide help in: analyzing a situation; defining the problem; designing the solution and in the subsequent implementation.

EXPERIENCE OF PROJECT SUPERVISION

A recurring experience in the supervision of software development projects in vocational education is the apparent lack of importance attached, by the students, to the use of a software development process model. Student projects should address a significant business or technical problem and it must be feasible to complete the project within the allocated time. Many possible realistic projects require some application domain experience which students lack. Thus many student projects are more implementation-oriented dealing with more general applications which are more easily understood. The major focus for the student therefore is the implementation issues. Students having the necessary practical skills can develop a tangible software product without compliance to any software development process model. Whereas adhering to a software development process model, which necessitates the use of abstractions, such as architectural design and process design, to ensure the achievement of certain software product quality attributes is much less tangible.

DEFINING SOFTWARE PRODUCT QUALITY ATTRIBUTES

Definitions of some software product quality attributes are given below:

Maintainability - it should be possible for the software to evolve to meet changing requirements

- Dependability - the software should not cause physical or economic damage in the event of failure
- Efficiency - the software should not make wasteful use of system resources
- Usability - the software should have an appropriate user interface and documentation

STUDENT PERCEPTION OF A PROCESS MODEL

A typical student's perception of a software development process model appears to be a series of milestones at which documentation is produced and presentations made to satisfy academic requirements. The documentation can take the form of:

- Initial Report - The student has to: demonstrate an understanding of the problem; to have completed sufficient preparatory work; to have a plan of work.
- Interim Report - This includes the analysis, requirements, initial design, prototypes etc.,
- Final Report - A documented system in terms of requirements, analysis, design, implementation, testing and systems manual and user guide.

The contents of the first two reports are frequently the only tangible visualization of the software product that can be seen and the correlation between these and the actual development work is really unknown.

The perception held by students may be attributed to a number of academic factors and industrial practices, which are listed below:

- academic assessment is based on the quality of the documentation;
- academic and industrial supervisors attach great significant to a fully functional product;
- graduates attached great importance to meeting deadlines;
- Small to Medium Enterprises (SMEs) place great significant in commissioning new software systems on time.

How each of these factors might lead to a lack of importance being attached to software development process models is given below:

academic assessment is based on the quality of the documentation

Whilst this statement is true, a common practice is the production of good quality documentation solely for the purposes of assessment **and not** as a design solution or a record of actual work done. In other words the documentation is not a natural byproduct of following a software development process model.

academic and industrial supervisors attach great significant to a fully functional product

It is a natural tendency to try and fulfil a software specification and satisfy the 'client' with the required working system. The drawback here is that 'slippage' in following a software process model may result in inherent defects in a delivered system. Practitioners may consider that additional functionality is advantageous to a 'client' and any defects can be corrected when a problem arises. Thus the software will be unavailable whilst the defects are removed. However with safety-critical software systems, such as air traffic control, the achievement of certain quality attributes such as reliability and response time is very important. However the achievement of some quality attributes may conflict with the achievement of other quality attributes e.g. the more functionality added to software the more difficult it may be to achieve or maintain a high level of reliability. A table provided by Vliet (1993) illustrates the trade-offs between quality factors.

graduates attached great importance to meeting deadlines

Informal feedback from graduates is that, the delivery of new systems is **the** priority and little emphasis is placed on documentation, if even produced. The rationale for this practice is that the life expectancy for systems is relatively short.

Small to Medium Enterprises (SMEs) place great significant in commissioning new software systems on time

It is a natural desire for all enterprises to have new software systems delivered on time. In the context of Hong Kong it has had a greater significance for SMEs who have always operated in a very competitive environment with very short time horizons. However if SMEs are to achieve more maturity in the process model they follow to develop software then they need to better manage their requirements and quality assurance. The requirements include setting the desired quality attributes at the commencement of each project.

The perception held by students of process models can have major consequences for industry. Graduates will develop software systems, which do not have the necessary quality attributes. The absence of the necessary quality attributes means a system does not fulfill the requirements. This may result in an organization committing additional time and resources to overcome the shortcomings of the delivered system. Systems might need to undergo considerable reworking to

achieve a state in which it is acceptable for use. In extreme cases the software may be scrapped altogether.

TEACHING AND LEARNING DOMAIN

Within the teaching and learning domain there is an established strategy in the conduct of software development project supervision. The learning objectives normally state that a student should apply their knowledge and skills to implement a solution of a reasonably complex problem set. The established strategy has strengths in terms of: monitoring progress, by regular meetings with the project supervisor : quality assurance, with students timetabled for project laboratory classes each week when teaching staff are available for general advice assistance. Quality assurance in this context is really to ensure that students are taking a disciplined approach to their work and utilizing a methodology in a proper manner. This is similar to a formal definition of quality assurance: Establish organizational procedures and standards for quality (Sommerville, 1995).

PROPOSAL

The proposed approach to address this situation is to establish a much more structured approach to project supervision to ensure a student performs quality planning and another person rather than the project supervisor perform the activity of quality control. (Any quality management activities should be separate from project management to ensure independence. Quality management in this context is the overall management of the quality of the software process and product.)

Formal definitions of quality planning and quality control as provided by Sommerville (1995) are given below:

- Quality planning

Select applicable procedures and standards for a particular project and modify these as required;

- Quality control

Ensure that procedures and standards are followed by the software development team.

QUALITY CONTROL

Teaching staff in the timetabled project laboratory classes each week can perform the activity of quality control. This is a clearly defined task of ensuring students follow a methodology using the appropriate techniques available. However the definition of quality planning given above needs more attention to actually determine what tasks need to be performed.

QUALITY PLANNING

At first glance one might consider the only task is to select an appropriate methodology suitable for the process you wish to follow in developing the software. An additional task that should be performed is that of selecting the desired software product quality attributes for that **particular** project. To define these quality attributes alone is insufficient. To help determine if the software product has these quality attributes requires some sort of measure to be able to be taken.

MEASURING SOFTWARE PRODUCT QUALITY ATTRIBUES

This task has been considered by Hughes & Cotterell (1999) who state 'Trying to find measures for a particular quality helps to clarify ideas about what that quality really is……In some cases we can measure the quality directly while in other cases the thing being measured is not the quality itself but an indicator of the degree to which the quality is present'. A student project proposal should include identified product quality attributes that must be present for their software to be considered acceptable.

EXAMPLES OF MEASURES

One example of a product quality attribute could be 'usability' and the first task is defining what this attribute means in the context of this project. This definition should be the result of negotiation between the student and the project supervisor. This definition could be …users should be able use the software after two hours training…or …the average response time is 30 seconds. This type of attribute can be quantified and measures taken to determine if the desired quality attribute has been achieved. Another possible product quality attribute could be 'maintainability', which includes incorporating new requirements. This is more difficult to quantify and measure. As mentioned by Hughes & Cotterell, above, sometimes it is not the quality measure but an indicator that it is present. How a project supervisor deals with 'maintability' is again subject to negotiation with the student. One could say you do not know how easy or difficult it is maintain software until a sufficient number of changes have been made over a period of time. Obviously this would be of no use in the context of a student project. However the project supervisor could take a number of innovative approaches such as: applying a design quality metric such as the *design structure quality index* ***(DSQI)***, Pressman (1997) provides a example of this metric; giving a new requirement after the software development has been completed. The ***DSQI*** measure can help determine whether further design work should be undertaken. A good measure would be indicative of a design with good characteristics that would enable relatively straightforward and efficient changes to be made to the software. Likewise the ease or otherwise of incorporating a new requirement after completion of the software will indicate the degree of maintability present in the software.

CONCLUSION

Many student projects are more implementation-oriented dealing with more general applications which are relatively easily understood when compared with more realistic projects. The major focus for students therefore is the implementation issues. A project student can create a tangible software product without any real compliance to any software development process model. However to have a chance of achieving the required product quality attributes, which more realistic projects would demand, necessitates sufficient attention being given to the software development process model. The required software product quality attributes identified at the commencement of the project must be achieved to be indicative that an acceptable software product has been developed.

Such an approach would ensure that sufficient attention is given to the software development process model to ensure that the desired quality attributes are achieved.

REFERENCES

Bennett S, McRobb S, Farmer R, (1999): Object-Oriented Systems Analysis and Design using UML, McGraw-Hill

Booch B, Rumbaugh J, Jacobson I, (1999): Unified Software Development Process, Addison Wesley.

Goodland M, Slater C, (1995): SSADM A Practical Approach, McGraw-Hill.

Hughes B, Cotterell M, (1999): Software Project Management 2nd Edition, McGraw-Hill.

Pressman R, (1997): Software Engineering A Practitioner's Approach 4th Edition, McGraw-Hill.

Sommerville I, (1995): Software Engineering 5th Edition.

Vliet J, (1993): Software Engineering Principles and Practice, John Wiley & Sons.