The Road in Software Engineering Education from Structured Programming to Object-Oriented Modelling

Dr. József Tick

Budapest Polytechnic, Hungary, tick@bmf.hu

Abstract: Higher level software engineering education has always followed the drastic paradigm changes that happened in the profession in the last twenty five years. As a result, software engineers have got to object oriented modelling, the use of CASE tools, the usage of client-server software architecture and component-based software development from the old structured programming. This paper examines how well software engineering education went after this progress. Higher level software engineering education has converged to the curve of the progress in the form of a broken line. This paper examines first and foremost the educational aspects, especially highlighting those educational methods that enforced a mentality change in the attitude of the students as young software engineers.

Keywords:Software Engineering, Education, Structured Software Development, Object-oriented Modelling, CASE tools

1 Introduction

The last twenty five years have resulted in a quite large and increasingly speeding technical development, in which informatics has done its own share. Furthermore, in the field of informatics the improvement in software development has been significant without any question.

This paper, primarily, summarises the statements based on the features, relations and experiences of software engineering education in the John von Neumann Informatics Faculty of the Budapest Polytechnic, and earlier in the form of programming technologies in the ancestor institution, the informatics course of Kalman Kando Polytechnic.

Modern higher education, within which, of course, the education of software engineering, has to follow the progress in the profession. This tracking can only happen in the form of a broken line, since:

- Higher education has a large time constant, in which the training term itself is of a 3-5- year period.
- A change in the education requires a big change in fixed assets especially in the labs, which means the demand of great financial resources (the purchase of software licences in large numbers, occasionally a change in the computer park)
- The knowledge and methodology background of a change must be set up (new learning materials, the elaboration of case studies).

As a result of the above, great changes have been discrete and separate periods could be defined. Of course, some features of some periods can overlap. The beginnings and ends of the different periods cannot be easily determined, however, the events that can be linked to the introduction of a certain new paradigm can also be well distinguished. Describing it on a time axle, the life cycle of each new paradigm can be examined. These events are as follows:

1. The introduction of structured programming
2. The introduction of structured software development methodologies
3. The introduction of the usage of structured CASE tools
4. The introduction of object-oriented programming
5. The introduction of object-oriented software development methodologies
6. The introduction of the usage of object-oriented CASE tools
7. The introduction of integrated software development tools

Taking the always existing overlaps into consideration, 5-7 year long time periods can be determined in the development of software engineering. These periods on the time axle with their most significant features are:

1. The era of structured development
2. The era of object-oriented development
3. The era of integrated software development systems

The knowledge components make up a hierarchy in which the components are built on each other. A change in each component, thus a change in the profession, can be followed in education only with a certain time shift. These components (theories, methodologies and tools) have their own characteristics like their own place in the education, the optimal methodology of their teaching, their life-cycle in the profession and the rate of immediate usable knowledge in the software production.

Taking the elements of software development and the above mentioned components into account, the field of development can be basically split into four categories:

1. The category of development methodologies
2. The category of development tools
3. The category of programming languages
4. The category of project management

When describing the eras these four factors must be considered. Figure 1. presents the component lifecycles in software engineering education at the Budapest Polytechnic.

2 The eras in software engineering higher education at the Budapest Polytechnic

According to the above list, the eras distinguished by their features follow below:

2.1 The era of structured development

The “boom” of structured programming, a so called structured euphoria, can be placed at the beginning of the eighties. Thanks to the work of Wirth [1] half the world used PASCAL and later thanks to the Borland firm software products were developed in Turbo Pascal. Naturally, education followed the trend and with the introduction of Turbo Pascal 3.0 students developed software in Pascal in the labs. By now Pascal is, of course, not used in software engineering, however, some new versions of Turbo Pascal are still in use in some other fields. A real world success, C language replaced programming in Pascal in the education, which lasted quite a long.

First the methodology of Jackson, called JSP [2], later the methodologies of DeMarco [3], Stevens, Myers, Constantine [4] and Yourdon [5] have gained ground in the software engineering specialisation. In order to complete the list the methodologies of Gane, Sarson [6], Warnier and Orr [7] as well as of Jackson’s JSD [8] are also taught, although of minor importance.

Considering CASE tools, firstly the product of Siemens, the Easy-Case and later the product of Cadre Technologies, the Teamwork, have been introduced.

By the end of the era, in project management, teamwork has been accepted the usual way of solving different tasks, which has not been supported by CASE tools at all yet.

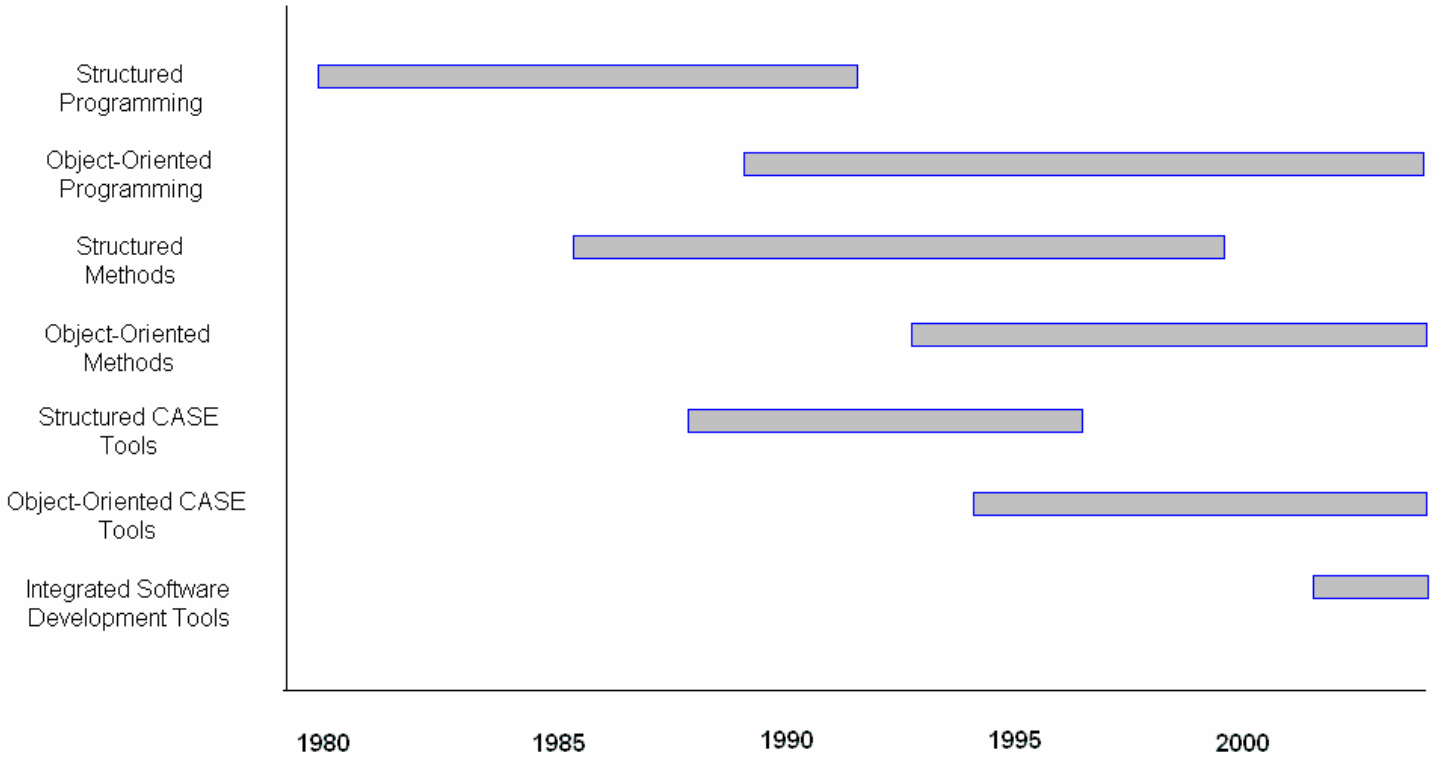

Fig. 1. The component lifecycles in SE education at the Budapest Polytechnic

2.2 The era of object-oriented development

Similarly to the structured development the upside down triangle syndrome was valid in this phase as well, which means that this era also started with programming followed by design, and later by analyses.

At the beginning of this era object-oriented programming started with the introduction of Turbo Pascal 5.5, which was followed by advanced versions. The OO concept was completed by the introduction of C++, which was the next step in the education as well. Tracking the development, the use of JAVA and thus the use of network-based client-server architecture were chosen from the applied languages.

In software engineering education, the methodology named after Booch [9] came out first as an object-oriented methodology. This was followed by The Object-Oriented Analysis and Design noted by Coad-Yourdon, [10] [11] and it was completed by Jacobson [12]. Apart from these ones, the responsibility driven methodology by Rebeca Wirfs-Brock [13] was also introduced.

The above methods were followed by two successful and widespread Object oriented modelling technique-based methods, namely, OMT, which is linked to Rumbaugh [14] and later, UML [15], which was elaborated by Rumbaugh, Jacobson and Booch and has slowly become the exclusive leader in the profession

From CASE tools, first Software through Pictures, which supports OMT, later Rational Rose, which also supports OMT, and then a version of Rational Rose, which supports UML were introduced in the education of software engineering.

In this era projects carried out by the students during the semesters were done in groups of four, where team roles were well defined, the project was well documented and the project management was well supported by a CASE Tool. A presentation and evaluation of project happened at the end.

2.3 The era of integrated Software Development Systems

The applied OO methodologies have crystallised and Unified Modelling Language (UML) was almost exclusively used in each more important development firm. The methodology linked to UML, the Rational Unified Process (RUP) was rapidly spreading among software developers.

Thanks to this technology, the development of much larger and more complex applications became possible. By now, software development looks like a „construction from Lego set" by clicking onto or moving the (existing, large numbered) components.

During software development, the application of CASE tools, out of which mostly the usage of tools supporting complex application-development, has stepped

forward. In the industrial type software development the Rational XDE CASE system, the product of the Rational firm acquired by IBM, has much the biggest market share. This tool supports the development of OO type UML based stand-alone or rather client-server architectural and platform free software systems that apply software-reuse.

During realisation, Rational XDE can be linked to all the three presently market leading, component based development environments that apply visual techniques, support team work and version management and build on network technology. These environments enable extremely effective, rapid and safe software development. They fulfill the demands set up by the present modern applications (running under windows, integration with database, web based distributed application) to the highest extent.

These systems have been made for similar purposes, but were built on different phylosophies and offer different services. They are competitive products, however, each tool has its own widescale technological background:

- **„Visual Age for Java, by IBM"**, and the replacing **„WebSphere Studio"** is a component based visual development system, which is JAVA based, well cooperates with Oracle and well builds up with Rational XDE. Due to its JAVA background it enables the development of real multiplatformed applications.
- **Microsoft „Visual Studio NET"** is built on the new technology group (NET) of Microsoft and is also a component based visul development tool, which, however, has enabled the development of applications exclusively on MS Windows-based platform so far.
- **„JBuilder" by Borland,** which can be considered a follow-up of the PASCAL -> DELPHI line, is a JAVA based popular visual development system, that can be well integrated with database handling.

The usage of the above mentioned tools and the development work with them are quite similar, however, the technological line they represent and the product line of the companies supporting them are quite different. In order to make the students' participation in the development work at their work place easy , it is practical to give the opportunity to present all three systems to the students during their higher education studies.

Naturally, the number of lessons does not give the opportunity to teach all the three trends to the students. Within the compulsory number of lessons the use of WebSphere Studio would be practical in the labs. This is justified by the fact, that this product is quite popular and widespread in Hungary, it enables platform free development and it applies all the techniques that are required by the different development environments. The other two trends have to be taught in optinal courses.

Conclusions

The above defined three eras are determined arbitrarily, however, they well describe the progress that software engineering education made in the last decades from structured programming to the application of integrated software development systems. This overview tried to present those components that meant the most important building bricks in the education of software engineering at the Budapest Polytechnic. With the help of the selected methodologies, languages and tools the students were trained to skill level in software development in each era. This has been proved by the good job rate among graduates and by the positive feedback from the profession for a long time.

References

[1] Jensen, K., Wirth, N.: PASCAL – User Manual and Report
Springer Verlag, 1974

[2] Jackson, M.: Principles of Program Design
Academic Press, 1975

[3] DeMarco, T.: Structured Analysis and System Specification
Prentice-Hall, 1979

[4] Stevens, W. P., Myers, G. J., Constantine, L. L.: Structured Design
IBM Systems Journal, vol.13, no. 2, 1974

[5] Yourdon, E. N., Constantine, L. L.: Structured Design
Yourdon Press, New York, 1978

[6] Gane, T., Sarson, C.: Stuctured Systems Analysis
McDonnell Dougles, 1982

[7] Orr, K. T.: Structured Systems Development
Yourdon Press, New York, 1977

[8] Jackson, M.: System Development
Prentice-Hall International, 1983

[9] Booch, G.: Object Oriented Design with Applications
The Benjamin/Cummings Publishing Company, Redwood City 1991

[10] Coad, P., Yourdon, E.: Object-Oriented Analysis
Yourdon Press, New York, 1991

[11] Coad, P., Yourdon E.: Object-Oriented Design
Yourdon Press, New York 1991

[12] Jacobson, I., Christerson, M., Jonsson, P., Övergaard, G.: Object-Oriented Software Engineering – A Use Case Driven Approach
Addison Wesley, 1992

[13] Wirfs-Brock, R., Wilkerson, B., Wiener, L.: Designing Object-Oriented Software
Prentice Hall PTR, Englewood Cliffs, New Jersey, 1990

[14] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorensen, W.: Object-Oriented Modelling And Design
Prentice Hall International Edition, 1991

[15] Rumbaugh, J., Jacobson, I., Booch, G.: The Unified Modeling Language Reference Manual
Addison Wesley, 1998