UNDERSTANDABILITY OF THE SOFTWARE ENGINEERING METHOD AS AN IMPORTANT FACTOR FOR SELECTING A CASE TOOL

INFORMATICA 3/92

Keywords: CASE tool, engineers testing, learning properties, software engineering

I. Rozman, J. Györkös, K. Rizman
University of Maribor
Faculty of Technical Sciences
Smetanova 17, 62000 Maribor, Slovenia

ABSTRACT — **The article highlights the understandability of a software engineering methodology as an important criterion for selecting a CASE tool. This aspect is treated through the comparison of learning properties for two very well known methodology on which the CASE tools are usually based on. The first one is SA-SD and the second one is JSD. In the purpose to compare both methodology a group of young engineers has been tested. Each of them wrote a seminar theme, answered a questionnaire and explained his observations. At the end of the paper, a general conclusion is presented.**

1 Introduction

Computer - Aided Software Engineering (CASE) tools are coming into widespread use in the software engineering. These tools automate methods that can improve software development practice and help to improve the quality of a software product. Choosing a CASE tool can be a difficult and confusing task because the great number and variety of tools are on the market. Many potential user of the CASE tool base their selection upon literature from the vendors, perhaps with a trial period using a demo system.

In article [6] a method for selection a CASE tool is described as a six step process:

- define the CASE requirements,
- contact vendors for information,
- select systems and vendors for in-dept evaluation,
- require vendors to respond in writing to the CASE requirement and to loan demonstration copies of the system trial use and evaluation,
- if possible, visit vendors at their home sites,
- select a CASE system for purchase.

In the first step, analysis and/or design methodologies which support the tool as an overall system requirement are expressed. It should be noted, that the tool

- must have a full semantic understanding of the graphical and textual object comprising each analyses and design component,
- must handle a minimum number of levels (e.g. 10 or 15) in hierarchical analyses or design methodologies, and a minimum number of components such as data dictionary entries, data-flow diagrams, or structure charts,
- must support completeness and consistency checking across the analyses and design and through all levels of the analyses hierarchy.

But, the influence of methodology to be learned itself, is not observed in this article. Methodology is prerequisite and essential for using the CASE tool. The importance of this criterion, as an important criterion for characterizing the CASE tool, was factually not expressed in last workshop CASE 90 [2]. But from our experience, we know that a full satisfaction with a CASE tool can be achieved only if users like to use them. A tool must be based on methodologies which are close to the way of the user's thinking and close to his/her experience in the past, or that it aids his/her existing knowledge.

In order to highlight the problem of friendliness of a CASE tool to the user, we have compared two formal methodologies regarding to the user's capability to understand it and to learn how a particular method can be applied to the software development. The both chosen methodologies are taken from the two main groups and are quite different. These groups are:

- Data Flow Oriented Methodologies and
- Data Structure Oriented Methodologies.

From the first group the well known SA-SD methodology (Structured Analysis according to DeMarco [1] and Structured Design according to Yourdon and Constantine [5]) was chosen. From the second group the JSD methodology [4] (Jackson System Development, author is M. Jackson) was chosen, because this methodology is the most typical and in certain expert circles most widely used. And what has not the least importance, some authors classify this methodology near the object oriented ones (e.g. [3]).

Our comparison is based on an experiment in which we tested some engineers to find out which methods are easier to learn.

2 Experiment description

Experiment has been applied to twelve person tested who learned the both mentioned methodologies during the post - graduate study. Eight of them are electrical engineers, two are mechanical engineers and two are civil engineers. With the aid of lectures attended (fifteen hours) and literature studied each of them had to solve a problem, approx. 3000 LOC, in the form of a seminar work and according to methodologies cited above. The problems to be solved were mainly procedural oriented, but some of them had important data component, too. If difficulties in individual design appeared consultations were given. When their work was finished they presented their results and answered the questionnaire given. At the same time they passed the examination.

2.1 Questionnaire

The questionnaire is divided into two logical unit. The first two questions are general. The answers are expected to reflect the influence of previous skill of students for modelling a real world. The last four questions are expected to give an individual note of both methodologies and to show which methodology the young engineers are more familiar with. A descriptive manner of answering is foreseen for the last four questions. The narrowness of possible answers and their misguiding influence are avoided, but a descriptive treatment of the results of the questionnaire is needed. For the presentation in the last question the well known Osgood's Semantic Differential with four-level scale is used .

Q U E S T I O N N A I R E

1. Which programming language is most often used in your programming experience?

a) FORTRAN

b) Pascal

c) Other

d) None

2. Your task is to model a part of the real world into a software system. When such task is given to you, do you think that you understand the required reality:

a) Completely

b) Only main aims are known; the details are unknown.

c) The details are known; their uniting into a system is unknown.

3. If you don't understand the problem completely, the reason is as follows:

a) Ignorance of the skilled perspective of reality.

b) It is not known how to express the function of the problem (It can only be "felt")

c) Others

4. What do you think of a problem that should be solved?

a) The model of reality is seen.

b) The model is seen as a transformation of the input picture into the output picture.

5. Which software formal developing method are you going to use with your future work?

a) JSD

b) SA-SD

c) None

6. Why are you going to use the method marked under point 5?

a) ---------------------------------------

b) ---------------------------------------

c) ---------------------------------------

(give at least two answers)

7. Why are you not going to use the other method mentioned or even none of them?

a) ---------------------------------------

b) ---------------------------------------

c) ---------------------------------------

(give at least two answers)

8. With the marks 0 (very bad), 1 (bad), 2 (good), 4 (very good) try to express the degree of satisfaction of particular activities in the life cycle for both methods

	JSD	SA-SD
A - assistance in formulation of requirements		
B - quick detection of faults		
C - surveyability		
D - simple maintenance of end-product		
E - simple completion of end-product		

Figure 1. Questionnaire

2.2 Interpretation of Answers

The answers to the first five questions are shown in the tabular form on the figure 2.

answer	1	2	3	4	5
a	10	2	7	3	1
b	2	8	2	10	11
c	0	2	3	x	0
d	0	x	x	x	x

Figure 2. Tabular presentation of answer to the first five questions

Most young engineers answered that they are familiar with the FORTRAN language. This is understood because the young engineers mostly learned this language during their regular education. To have only a skill about FORTRAN means that these people are "half illiterate" as regards programming. It is sure that ignorance of data part of the program makes the understanding of any software development methodology difficult. In the SA-SD methodology this is seen from the difficult understanding of the data dictionary and data handling in the files as a data base. In the JSD methodology, where entity structure is dominant, ignorance of the data part of the program was undoubtedly one of the reasons for more difficult understandability.

The answers to the second question show that the tested persons tried to get the most abstract picture of all important functions. Two of them decided that they had known details but they had not had a clear perspective about uniting details into the system. The conversation in the experiment showed that in these two cases tested people were working on the problem long time and the seminar work done during the experiment was prolonging their continuous work.

The answers to the questions three and four show that in the most cases the functional decomposition was used as appropriate way of thinking for decomposing a problem into small pieces. The fifth question demanded a concrete decision regarding the methodology used in the future by the individual. The majority (eleven persons) decided to use the SA-SD methodology. Let us consider some most important arguments for preferring SA-SD methodology which were answered under question six:

- methodology was easy to learn,
- the philosophy of this methodology is close to my way of thinking,
- a clear model is quickly reached and
- the methodology is simple and efficient.

The JSD methodology was negatively appreciated (answers to question seven). The arguments are as follows:

- it is difficult to learn,
- analysis from the entity/action aspect is difficult to understand and
- implementation is not simple.

The presented seminar works also showed worse understanding of the JSD methodology. Where usually had the tested people problems? In the first "entity action step" they were in doubt about boundary of the modelled system. They enumerated actions easier as they decided about entities. The second problem, which appeared, was the understanding what the entity is? They usually forgot that actions must appear on the entity what is not the same as the entity in the E-R (Entity-Relationship) diagram.

In the second "entity structure step" tested person had problems to understand that each structure in Jackson's diagram has only two levels. The first level is the structure's name and the second level form the parts of this structure, while in the Structure Chart each box represents a program module. In the "initial model step" - step three seems difficult to understand what really a process marked by zero and process marked by one means. In the beginning, it was not very clear to the students why the same object is treated as structure of entity in step two and as process in step three. This was the most repeated question which tested persons had during the working their seminar work.

When the first steps were understood the remaining three did not cause greater problems. We are surprised that the function of scheduler (implementation step) is quickly understandable, what means that testes had enough knowledge about concurrent programming. The essence of Jackson's pseudo cod is well accepted, too.

The answers to the eighth question are shown in a graphical form as a average values of collected points of semantical differential. The results are surprising. Despite of the negative attitude towards the usage of JSD methodology, both methodologies are relatively equally assessed; figure 3. Cumulation of points for question "E" is greater for JSD as for SA-SD. This means that young engineers decided correctly, accepted the essence of both methodologies, and assessed that JSD is more appropriate for completion of end product than SA-SD.

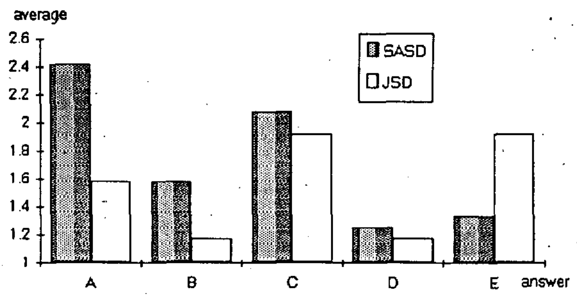

Figure 3. Presentation of the answers to the eight question

3 Conclusion

It should be stressed that it is not our intention to compare all aspects of both methodologies used in the experiment but only to present that the learning aspect of a methodology is an important criterion which has to be taken into account in the process of selecting a CASE tool. To assess this criterion we suggest the usage of a questionnaire like this one, presented in our paper. We are aware of the fact that such exhaustive investigation the problem of the understandability the methodology, as it is presented in the paper, is too time consuming. But however, some problems such as: types of problems which are usually solved by the group, previous skill of group about modelling real world and programming, familiarity with software engineering methodologies, satisfaction with proposed methodology, should be answered by the questionnaire. The most time needed for questionnaire can be saved with reducing problems with which person are tested. In our experiment we intentionally have not used any CASE tool because we were observing only methodologies themselves. The usage of any tool would make our picture about methodology learning unclear. We have not evaluate the influence of the tool's teaching component. This is a very difficult task which demands additional investigation and it is valid for a particular tool only.

The results obtained speak in favour of the SA-SD methodology because it is more understandable to the group of engineers who represent, in our case, the potential user of selected CASE tools. Perhaps, we are wondering, why SA-SD is better assessed than JSD? Our opinion that we obtained such result is, that in engineering curriculums people have gotten much more skills about programming with procedural languages than programming with object-oriented languages. So, we can legitimately expect that those CASE tools which support data flow oriented methodology (like SA-SD) need a shorter training period for people with engineering education than those CASE tools which support other methodologies.

If we want to select a CASE tool for another group of users which have got more knowledge and skills in object-oriented programing it is possible that we achieve another fully opposite conclusion about the methodology supported.

References

[1] T. DeMarco *Structured Analysis and System specification*, Prentice-Hall, 1979

[2] R. J. Norman, R. V. Ghent. (Editor), *CASE'90, Four Internal Workshop on Computer-aided Software Engineering*, Irvine, CA (December, 1990).

[3] B. Henderson-Sellers, J. M. Edwards, *The Object - Oriented Systems Life Cycle*, Communication of the ACM, vol. 33, no. 9, Sept. 1990, pp. 143-159

[4] M.A. Jackson, *System Development*, Prentice-Hall, 1983

[5] E. Yourdon, L.L. Constantine, *Structured Design: Fundamentals of a Discipline of Computer Program and System Design*, Prentice-Hall, 1979

[6] L.Zucconi, *Selecting a CASE Tool*, ACM SIGSOFT, vol. 14, no. 2 Apr 1989, pp. 42-44