A Formal Model for Evaluation of the Software Development Process based on Quality Indexes

Denis Kozlov[1]

[1] Department of Computer Science and Information Systems, University of Jyväskylä, 40014 Jyväskylä, Finland
dekozlov@cc.jyu.fi

Abstract: The focus of the paper is evaluation of the software development process regardless of the underlying software development approach. The process of evaluation is scrutinized from two perspectives: (1) "measurement for management", when results of evaluation are used for managing and improvement of the software development, and (2) "measurement for analysis", when results of evaluation are used only for determination of the current state of the process. A formal model for evaluation based on quality indexes in two variations for each of these cases is proposed. According to the proposed model the software development process is viewed as a chain of sub-processes, each of which is characterized by an integral quality index or a set of integral indexes, depending on the case (1) or (2), and the aggregated quality index of the whole process is calculated from these. Some principles of accuracy assurance for the proposed model are discussed.

Key words: software development process, process evaluation, quality metrics, formal model

1 Introduction

Nowadays there are a number of different approaches how to develop high-quality software that would satisfy needs and expectations of its users. Some of these approaches differ from each other significantly; however they all have a common feature: a quality assurance mechanism that enables to reach the quality of the developing software. Usually this mechanism is called a quality management system [1].

A problem of the quality management system is that in general there are no standards or internationally recognized methods for measurement and evaluation of different processes and activities in the framework of the quality management system. The reasons for this are clear. On the one hand, the family of the standards ISO 9000 et al is oriented to the quality system in general, without taking into account specific peculiarities of the enterprise. On the other hand, standards and papers that describe the issue in quite general form [2-4] emphasizing only the groups of attributes\metrics

without discussing concrete approaches for measuring them. For instance, the attribute "reliability" can be evaluated based on the metric "mean time to failure", but this metric can be measured in different ways by taking into account such influencing factors as operational profile of the user that operates the software to be evaluated and different definitions for time and failure [5].

One more disadvantage of the ad hoc papers is that they often present the approach for measuring attributes of an entity (software or software development process) based on errors [6-7]. For instance, the reliability is measured based on the errors that already happened. Although this approach might be useful from a practical point of view, however, the learning of an organization by its own mistakes is always connected with costs. So the errors of the software at the output are not the only indicators that something in the software development process is wrong.

Finally, the ad hoc papers and standards represent often evaluation (or measurement) of the software development process in a very complex way so that persons who are going to evaluate the process should have a special knowledge or background in order to be able to evaluate. Moreover the evaluation process itself seems to be quite cumbersome.

The paper represents an approach that is based only on experts' judgment. One of the advantages of this approach is that all the sub-processes are measured in relative units. Thus it enables to eliminate dealing with heterogeneous (non-equidimensional) quality attributes that confront with one another from the viewpoint of dimensions.

2 Proposed Model for Evaluation of the Software Development Process

The quality assurance mechanism (or quality management system), which is a common feature for different kinds of software development approaches (and therefore can be considered as a certain "common denominator") can be represented in the following way by taking into account software development peculiarities (Fig. 1).

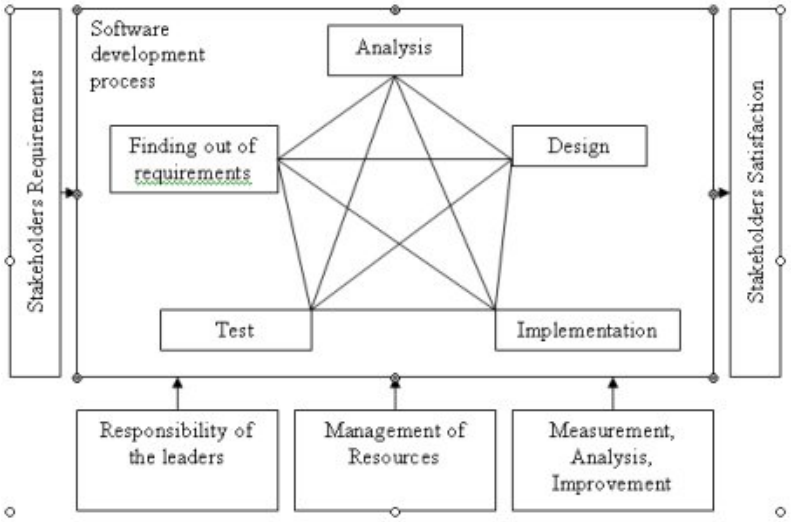

Fig. 1 Software development process from the viewpoint of quality management

In other words the software development process regardless of the software development approach underlying it includes the following components:
Software development process in its pure form
Management of resources (hardware, software, materials, time, people etc.)
Measurement and analysis of current data aimed at improvement of the software development process
Responsibility of the leaders (managers). Here we adopt the concept from the quality management area, however, in case of the software development process this component means responsibility and understanding of all the members that they are parts of the software development process and therefore influence on the final results rather then only "leaders" of the organization.
According to the quality management methodology the whole product development process can be represented as a set of sub-processes, which can be carried out in the same department or in different departments of the organization. The consumer of one process serves as a supplier of other process, so that the software development process can be viewed as a chain of sub-processes. The idea of the process chain is not new; it was developed as a part of the Supply Chain Management. However, we chose this model for our measurement approach due to the fact that regardless of the chosen software development approach the software development process in every organization can be represented using this model.
However, in order to implement this methodology to the case of software development process some changes are required. The point is that in the modern software development process there are no processes and sub-processes in the strict sense: analysis of requirements, design, implementation, test and maintenance are often carried out at the same time (so called Unified Process), so that the boundaries between processes are blurred. The pentagon on the Fig. 1 reflects the idea that the above phases of the software development process are activities rather then processes in the strict sense.

However, they can be assumed also as processes, if all the influences and connections with other activities are taken into account. For instance, the activity "Test" can be viewed as a process if all the connections with other activities/phases and external influencing factors (costs, resources etc.) are considered (Fig. 2). Here we assume that activity is a phase of the software development process that take place at the same time as another phase, whereas process is a phase of the software development process that take place only when the previous phase is completed.

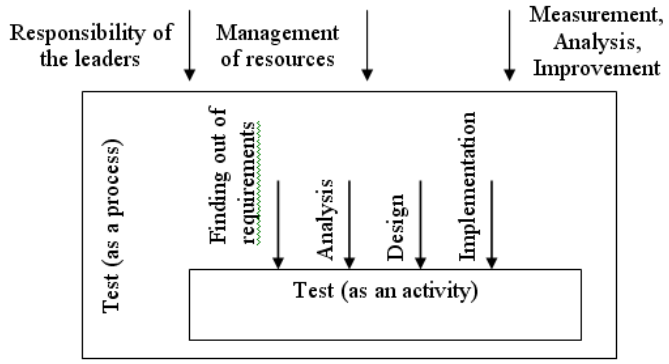

Fig. 2 Phase "Test" as a process

By representing all the phases of the software development process in this form, we will get a linear chain of the processes instead of the pentagon (Fig. 1). Thus our model bears a resemblance to the Waterfall model or the V-model with that difference that all the processes of the "pure" Waterfall model are subjected by the influence of external factors typical for the quality management system – responsibility of the leaders, management of resources and measurement.

For the purposes of formalization of the above assumptions and further discussion let us represent the proposed model in the following form (Fig. 3).

Here P_i – sub-process i; P_{i-1} – a process that is carried out before the process i; P_{i+1} – process that follows the process i; A_i – activity i that is corresponding to the process i; A_{i-1},A_{i+1},…, A_{n-1} – activities that are corresponding to the processes P_{i-1}, P_{i+1},…, P_{n-1}; m – number of external factors and activities influencing on the current sub-process; RL – Responsibility of the leaders; MR – management of resources; MAI – Measurement, Analysis and Improvement.

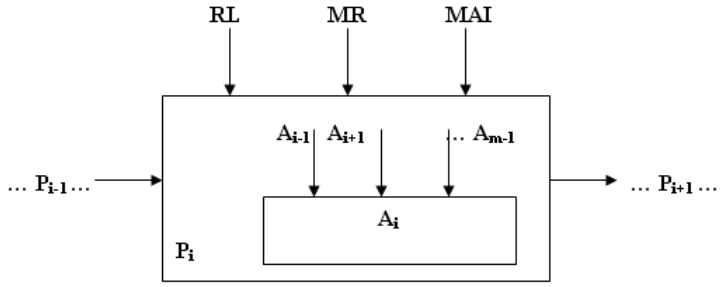

Fig. 3 Model for evaluation of the software development process

According to the proposed model the process of evaluation of the software development process consists of two steps:
Evaluation of every sub-process in the chain based on its integral quality index;
Evaluation of the whole software development process as a chain of sub-processes based on the aggregated integral quality index.
It should be noted that there are two perspectives on evaluation of the software development process:
Evaluation of the sub-processes and activities of the software development process for the purpose of analysis of its current state (whether the process satisfies certain requirements and to which extend, "measurement for analysis").
Evaluation of the sub-processes and activities of the software development process for the purpose of its improvement ("measurement for management").
The first case can take place, when e.g. external experts/auditors evaluate the software development process of a company in order to find out the level of compliance to the defined requirements in the framework of an international certification.
The second case is more important for the company itself, when it is oriented on a long-term software development and wants to use the results of evaluation for improvement.
By developing a framework for software development process evaluation both perspectives should be taken into consideration.

3 Integral Quality Index of Sub-processes

In the proposed model a unified formal sub-process is used. It means that each sub-process in the chain is supposed to be evaluated as an indivisible unit without taking into account its possible small ad-hoc sub-processes. Due to this restriction the evaluation process of different sub-processes seems to be formally unified and pared-down. However, this unification is achieved at the cost of accuracy of evaluation.
By taking into account the above model, the integral quality index of the unified sub-process i can be expressed in the following way:

$$Z_{pi} = F\ \{ \delta_{ij} \times X_{ij};\ \gamma_{ik} \times Y_{ik} \} \quad (1)$$

where Z_{pi} – integral quality index of the unified sub-process i;
X_{jj} - private quality index j of the sub-process i;
δ_{jj} – weight of the private quality index j;
Y_{ik} - influencing factor k on the sub-process i;
γ_{ik} – weight of the influencing factor k on the process i;
$j = \{1 \ldots n_i\}$, n_i – number of quality indexes for the sub-process i;
$k = \{1 \ldots m_i\}$, m_i – number of influencing factors on the sub-process i.

Therefore the integral quality index of the unified sub-process i is viewed as a function of its private quality indexes and factors that influence on this sub-process.

3.1 Private Quality Indexes of the Sub-processes

Some basic quality attributes (named as goals) for the software development process are discussed in [8] and [9]. We assume that the same set of attributes can be implemented not only for the whole software development process, but also for its sub-processes. The quality attributes (or quality indexes as it is used in the current paper) for sub-processes of the software development process can be listed and defined in the following way:

Effectiveness – ability to determine what an internal customer (a customer of the process/activity depending on results of the current process/activity) needs, produce what an internal customer needs and verify is that what has been produced is what an internal customer needs;

Efficiency – ability to carry out the process i by the optimum speed and by the optimum costs;

Maintainability – ability to find out and remedy faults in the sub-process i or work out where to make changes due to exposing the designers' and programmers' thought processes in such a way that their intentions are clear;

Predictability – ability to predict accurately how long it will take to carry out the sub-process i and which resources are necessary for it (time, stuff, hardware etc.)

Repeatability - ability to replicate all the activities and sub-phases of the sub-process i in future projects (e.g. in future releases of the same software);

Quality – ability to ensure a high quality product/results of the sub-process i; ability to provide a clear link between internal customers' desires/requirements and a product/results of the activity;

Improvement – ability to identify and prototype possibilities for improvement in the sub-process.

Tracking – ability to keep track of how good predictions concerning necessary resources are, and hence how to improve them.

Human factor – ability to satisfy professional and psychological needs and expectations of the sub-process's members.

One of the possible ways for measuring of the above attributes is the evaluation by experts based on different classifications for software development metrics, for instance [10]. In this case each of the attributes is measured according to the matrix of correlations given in [9]. For example, in order to evaluate the attribute "Tracking" the following groups of metrics can be used: Project Management Metrics and Quality Management metrics. In case of effectiveness all the groups of metrics will be used (Table 1).

Process metrics	**Quality attributes of SDP**	**Effectiveness**	**Tracking**
Management Metrics	Project Management Metrics	x	x
	Quality Management Metrics	x	x
	Configuration Management Metrics	x	
Life Cycle Metrics	Problem definition metrics	x	
	Requirement analysis and specification metrics	x	
	Design metrics	x	
	Implementation metrics	x	
	Maintenance metrics	x	
Resource Metrics	Personnel Metrics	x	
	Software Metrics	x	
	Hardware Metrics	x	

Table 1: Correlations between metrics related to SDP and its quality characteristics, adopted from [9] (x – the most significant correlations)

A process quality attribute (or a private quality index) X_{ij} can be calculated as:

$$X_{ij} = \frac{\sum_{l=1}^{L_j} v_{jl} \times A_{jl}}{\sum_{l=1}^{L_j} v_{jl}} \qquad (2)$$

where v_{jl} – weight of the metrics group l for the process j; L_j – number of groups of metrics for the process j;

A_{jl} – integral estimate of the metrics group l for the process j:

$$A_{jl} = \frac{\sum_{t=1}^{T_{jl}} w_{jlt} \times a_{jlt}}{\sum_{t=1}^{T_{jl}} w_{jlt}} \quad (3)$$

where w_{jlt} – weight of the metric t in the group of metrics l for the process j; a_{jlt} – estimate of all the relevant metric t in the group of metrics l for the process j; T_{jl} – number of groups of metrics.

It should be noted that concrete metrics chosen from the groups of metrics for evaluation of a certain attribute of the sub-process i can be different for different sub-processes.

Let us support the equation (2) by an example. By taking into account the classification of metrics related to the software development process [10] the attribute "Tracking" can be estimated in the following way:

$$X_{tracking} = \frac{v(PMM) \times A(PMM) + v(QMM) \times A(QMM)}{v(PMM) + v(QMM)} \quad (4)$$

where A(PMM) and v(PMM) – integral estimate of all the metrics constituting the group Project Management Metrics and its weight; A(QMM) and v (QMM) – integral estimate of all the metrics constituting the group Quality Management Metrics and its weight;

The integral estimates of the metrics groups PMM and QMM can be calculated according to (3) by taking into account their sub-classifications [9].

If the results of evaluation are supposed to be used as a basis for further improvement of the software development process, so it is reasonable to present each sub-process as a set of quality indexes X_{ij}:

$$I_i = \{X_{ij}\} \quad (5)$$

However, in case of the measurement for analysis, when the results of evaluation are supposed to be used only for determination the current state of the process or a degree of conformity to certain requirements (e.g. in the framework of a certification) it is worth also calculating the integral quality index of the sub-process i:

$$I_i = \frac{\sum_{j=1}^{9} \theta(j) \times X(j)}{\sum_{j=1}^{9} \theta(j)} \quad (6)$$

where X(j) – quality index j of the sub-process; θ(j) – its weight;

3.2 Taking into Account Influencing Factors

The previous assumptions were referred to the case when there are no factors influencing on the sub-process i. However, according to our model (Fig 2) every sub-process is viewed as "an activity + influencing factors". Each sub-process is subjected by the influence of three factors of the quality management system: responsibility of the leadership, management of resources and measurement, analysis and improvement (Fig. 1). Other influencing factors will depend upon the sub-process.
According to the automatic control theory the influence of the above factors can be taken into account in the following way (Fig.4).

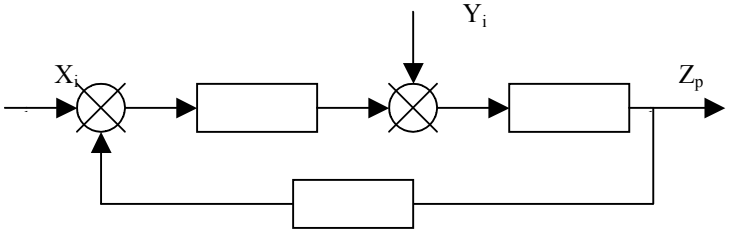

Fig. 4 Sub-process from the view point of the automatic control theory

The sub-process i here is represented as a black box, the income of which is the sub-process quality index X_i without taking into account influencing factors and the outcome – its quality index Z_{pi} by taking into consideration the influencing factors. At a certain stage of the sub-process i the influencing factor Y_{ij} takes place. The impact of the factor Yij can be either positive or negative. In order to minimize a possible negative impact of a factor on the sub-process a feedback is required. The new value Zpi of the quality index i will be associated with the initial value Xi according to the formula:

$$Z_{pi} = X_i \times \frac{F_{wf}}{F_{\prod u}} - Y_i \times \frac{F_{Yi}}{1 + F_{wf}} \qquad (7)$$

where X_i – the value of the quality index i without taking into account influencing factors; F_{wf} – transfer function without feedback; F_{Yi} – transfer function from the point of impact to the output; Y_i - a weighted arithmetic mean of all the factors influencing on the sub-process i:

$$Y_i = \frac{\sum_{k=1}^{m_i} \gamma_k \times y_k}{\sum_{k=1}^{m_i} \gamma_k} \quad (8)$$

$F_{\sum u}$ – transfer function of the sub-process viewed as a set of typical dynamic units. The main idea of $F_{\sum u}$ is that every sub-process can be represented as a set of dynamic units – sub-activities and small processes in the framework of the sub-process. For instance, such dynamic units can be: relationships among the stuff, costs, time etc.

$$F_{\prod u} = \prod_{i=1}^{f} F_{ui} \quad (9)$$

where F_{ui} – a dynamic unit i of the sub-process

4 Aggregated Integral Quality Index of the Software Development Process

The final result of the evaluation of the whole software development process can be seen as follows.
In case of "measurement for analysis" the aggregated integral quality index of the software development process is represented as a set of integral quality indexes of its sub-processes:

$$Z_p = \{Z_{pi}\} \quad (10)$$

In case of "measurement for management" the aggregated quality index of the software development process is viewed as the weighted geometric mean of the integral quality indexes of its sub-processes:

$$Z_p = \left(\prod_{i=1}^{ni} Z_{pi}^{\delta_j}\right)^{\frac{1}{\sum_{i=1}^{ni}\delta_i}} = \exp\left(\frac{1}{\sum_{i=1}^{ni}\delta_i} \sum_{i=1}^{ni} \delta_i \times \ln Z_{pi}\right) \quad (11)$$

Despite the fact that "geometric mean is a an average which is useful for sets of numbers which are interpreted according to their product and not to their sum" [11], the use of this mean is righteous because all the sub-process in the software development process are independent from the viewpoint of their influence on the

quality of the final product. The real connections between processes are considered in the model by taking into account the difference between the sub-process and activity discussed in the part 2 of the paper.
The problems by evaluation of the equation (11) are:

- The constituents of the formula are heterogeneous; they are evaluated in different scales.

In order to make them all equidimensional, the following assumptions should be taken into account. Due to the fact that some of the quality indexes of the sub-process are evaluated only based on the expert judgment, expressed in ordinal scale (e.g. in points) and therefore can not be applied to another scale, so other quality indexes and influencing factors expressed in the ratio scale (such as costs, time etc) should be transformed and evaluated in other scales. For instance, a quality attribute of the sub-process is calculated during a certain period. Based on this statistical data it is possible to make a graph and define the upper and lower boundaries – the maximum and minimum of its value. The letter should be determined when the software development process is mature and quasi-stable. The ratio of the current evaluated costs to the upper boundary determines the same costs but expressed in the ordinal scale.

- Estimation of the weight is supposed to be carried out by experts

The problem of expert judgments is a low accuracy of their results. In order to minimize it, some widely accepted methods for defining weights should be used. Among those are: overall integral index [12], consensus relation [13], analytic hierarch process [14] or fuzzy analytic hierarchy process [15]. Some other possible ways to reduce the inaccuracy are discussed in the following chapter.

5 Principles of Accuracy Assurance for the Above Approach

The weak points of the above approach from the viewpoint of accuracy of its final results are:
The evaluation of the partial integral quality indexes for sub-processes and the integral quality index for the whole software development process is conducted by experts. However, the judgment of human beings even possessing knowledge and attainments in software development field is always connected with errors. These errors can be divided into two major groups: random errors and systematic errors. Random errors are hardly to predict and exclude, so we will not take them into account.
Systematic errors associated with the experts can be minimized due to several ways.
First of all, the requirements to the experts should be articulated – who can be considered as experts by evaluation of the software development process. The requirements should include such parameters as seniority of the experts, independence from the evaluating organization, sex, age (in order to gather a group of homogeneous experts).
Secondly, the process of the evaluation should be described as thoroughly as possible. One of the "stumbling blocks" by organizing the evaluation process by experts is the way how to assign weights to different processes.

Sub-processes as parts of the software development process are often heterogeneous. Therefore the evaluation of the software development process as a set of sub-

processes by calculating the weighted geometric mean is always associated with measuring error.
In order to minimize this error we propose the following solution: the evaluation of the integral quality index for the software development process should be calculated twice by two different groups of persons. On the one hand it should be calculated by real experts by taking into account the above proposals concerning minimization of the systematic error of the experts. On the other hand the integral quality index should be calculated by the members of the organization as a part of self-assessment process. In that case the total integral quality index of the software development process is expressed by the formula:

$$X_{(total)} = 0.8 * X_{(exp)} + 0.2 * X_{(self)} \quad (\mathbf{12})$$

where $X_{(total)}$ – total integral quality index of the software development process;
$X_{(exp)}$ – integral quality index of the software development process calculated by the experts;
$X_{(self)}$ – integral quality index of the software development process calculated by the organization as a self-assessment process.
It is readily seen that the above formula reflects Pareto's principle: 80% of errors are caused by the evaluation by 20% less experienced evaluators. Therefore we assume that the confidence level to the evaluation results of the experts – 80%, whereas one to the self-assessment results – 20%.
It should be noted that there is no strict theoretical evidence of Pareto's principle; however, this principle has proven oneself in the practice by numerous examples, so that it might be reasonable to use it.

6 Limitations and Further Research

We see at least two areas for a further research following from the model presented in the paper.
The focus of the paper is on a theoretical representation of the new model for evaluation. We do not give any evaluation of the model, since it is being planned now. The evaluation of the model could be based on a case study conducted in a software development company in Finland. The results of the case study are supposed to be published.
In the chapter 5 only few principles of accuracy assurance are discussed, since the issue of inaccuracy in qualitative evaluation of quality attributes is a separate topic that exceeds the boundaries of the paper. In the future we plan to investigate such issues as: 1) how to pick up experts for conducting of the evaluation (measuring of the competence); 2) what is the number of experts sufficient for carrying out of a qualitative evaluation; 3) how to organize the experts' work in order to eliminate possible errors (especially, if a group of experts is distributed geographically, rather then gathered in one place); 4) how to validate results of a qualitative evaluation; 5) how to deal with incomplete and missing data used during a qualitative evaluation etc.

References

1. ISO 9001:2000 Quality management systems - Requirements
2. IEEE 1074 Standard for developing software life cycle processes
3. ISO 12207 Software Life Cycle Processes
4. ISO/IEC 15504: Information Technology - Software Process Assessment
5. C. Kaner, P.B. Walter: Software Engineering Metrics: What Do They Measure and How Do We Know? 10th International Software Symposium, METRICS 2004
6. J.C. Munson: Software Engineering Measurement. Auerbach Publications, 2003
7. M. Lorenz, J. Kidd: Object-oriented software metrics, PTR Prentice Hall, Englewood Cliffs, New Jersey, 1994
8. S. Tyrrel: The Many Dimensions of the Software Process, available at http://www.acm.org/crossroads/xrds6-4/software.html
9. D. Kozlov: Classification of metrics related to the software development process as a prerequisite for its improvement. In QAOOSE Workshop Proceedings of the 18th European Conference on Object-Oriented Programming "ECOOP-2004", Oslo, 14–18 June 2004
10. Software Metrics Classification, available at http://irb.cs.uni-magdeburg.de/sw-eng/us/metclas/index.shtml
11. Mean, available at http://en.wikipedia.org/wiki/Mean/
12. G.G. Azgaldov: The foundations of quality. Moscow, Economics, 1982
13. H.P. Young, A. Levenglick: A consistent extension of Condorset's election principle. SIAM Journal of applied mathematics, 35(2), 1978, 285-300
14. T. L. Saaty: Models, methods, concepts and applications of the analytic hierarch process. Boston: Kluwer Academic, 2001
15. L. Mikhailov, P. Tsvetinov: Evaluation of services using a fuzzy analytic hierarchy process. Applied Soft Computing, 5(2004) 23-33