ISSN: 0976-3376

Available Online at http://www.journalajst.com

ASIAN JOURNAL OF SCIENCE AND TECHNOLOGY

Asian Journal of Science and Technology
Vol. 07, Issue, 06, pp.3031-3034, June, 2016

RESEARCH ARTICLE

SUPPORTING PROJECT MANAGEMENT WITH THE USE OF SOFTWARE ENGINEERING DATA

***Raed Abduljabbar Aljiznawi and Naseer Hwaidi Alkhazaali**

School of electronic information and communication engineering, Huazhong University of Science and Technology, #1037 Luoyu Road, Wuhan 430074, P.R. China

ARTICLE INFO

Article History:
Received 18th March, 2016
Received in revised form
24th April, 2016
Accepted 14th May, 2016
Published online 30th June, 2016

Key words:

Development,
Process, Software metrics,
Production.

ABSTRACT

Based on software data from a software development project, a number of analyses have been carried out. The results discovered the reasons for cost overruns, showed how effort planning could be refined during development, and identified a relationship between the quality of the design documents and the effort consumed for their production. This provided an insight into the development process and establishes norms for interpreting metrics values obtained in future projects. In this way, it is a first step for a company towards increasing its software development process maturity level.

INTRODUCTION

Several software development processes can be scamper in an interrelated way to universal project management processes. These processes exist primarily for supporting the management of software development, and are generally twisted toward addressing business concerns. A software development process is irritable primarily with the production features of software development, as divergent to the technical aspect, such as software tools. The central activities of software project management include planning, estimating, tracking and decision making. It attributes of a mature engineering discipline that the end-product quality level is planned, reliable project plans are made, progress is tracked on a detailed level, and estimates and decisions can be made, based on well demented experience with previous successful projects. This situation is not usually found in the relatively new discipline of software engineering. One ground is that, over the years, software engineering has adapted to a swift technological improvement. This, together with the fact that the end product is imperceptible, has led to a condition where many software development projects came out in an ad hoc fashion and very often fail to meet their success criteria. For software development, the important management parameters are project cast and duration, and product quality.

****Corresponding author: Raed Abduljabbar Aljiznawi,***
School of electronic information and communication engineering, Huazhong University of Science and Technology, #1037 Luoyu Road, Wuhan 430074, P.R. China.

Measurements of cost and calendar time are fairly easy. Control of these parameters is often attempted and is possible within limits, although mast managers have difficulties forecasting project cost and duration. Discussion of the quality of the end product starts with defining the appropriate quality factors and their measurement units. However, the next steps of predicting, at the start of the project, the expected final product quality factor levels, and controlling progress towards specified quality factor levels during the project, are not usually attempted. To improve the present state of affairs, it is necessary, during a software development project, to collect data that describe the different production processes, as well as the resulting partial and intermediate products. Thereby, the technical achievements, as well as the overall project status and progress, can be assessed. By analyzing the data, a quantitative basis for decisions can be established, with results improving as norms for interpreting measurement values become existing from finished projects. When selecting data and analyses for use in a project, or across projects in an organization, the emphasis should be on data that are simple and easy to collect and analyses giving results that can easily be understood. The benefits of introducing a software metrics programme will then contain improved tracking and control of a development project; early identification of atypical measurement values, which may indicate a previously undiscovered problem ;an accumulated set of data describing completed projects, for use in planning and estimation of future projects; the possibility of optimizing the software development process, since the problem areas can be identified

and the results of changes to the existing process can be The points require that norms for measurement values are available and, in many cases, such norms must be derived from data collected during previous projects. This paper describes a number of data analyses, together with the conclusions that were drawn from their results. The analyses were performed on a very comprehensive data set collected during a 6 period of a development project. The analyses confirmed how the experience could help software companies to advance project planning and monitoring, through the establishment of company norms and by introducing specific changes to the development process.

Monitoring of software development

The work reported here is one result of a concerted effort to improve the prediction, monitoring and assessment of software product quality. This was undertaken as part of the ESPRIT project REQUEST (Reliability and Quality of European Software Technology). In the area of software quality, work has concentrated on developing COQUAMO (Constructive Quality Model) (Petersen, 1987). The original idea of COQUAMO was to transfer the approach that had been successfully applied in COCOMO (Cost Model) (Boehm, 1981) to the field of software quality. COQUAMO has now been developed into three models, the first of which is a predictive model for software quality (using an approach similar to that of COCOMO) to be applied in the early phases of development. The second part consists of a monitoring model for use during the project, and the third part is a quality assessment model for the later stages in the development. Introduction of the monitoring model (Kitchenham and Walker, 1989) as part of COQUAMO enables it to maintain the natural activities of a project manager during a software development project. In this way, software metrics are utilised in software project control activities, based on the general project control prm cedure of setting quantitative targets, measuring against these targets and responding to deviations. The approach aims to work at a more detailed level and include more metrics than other software metrics programmes (Grady and Caswell, 1987; Duncan, 1988). In support of the development of COQUAMO, large-scale data collection and storage have been carried out for soft- ware metrics data as part of Request (Dale, 1987). For the monitoring model, this allows significant metrics and relationships to be identified. In-depth analyses have been carried out for a number of project data sets, with the aims of developing the model and investigating the assumptions behind it, as well as investigating the possibilities for automating the analyses of project data. One set of analyses has been based on the ideas of using anomalies (Eoerflinger and Basili, 1985), i.e. atypical metric values, to detect the deviations that, at an early stage, may indicate potential quality problems [SI. This paper shows how detailed analyses of data from a large software development project may be used to support general project management, by establishing quantitative norms, and how they may also be used to improve the software development process in a company. Collection and analysis of software engineering data from just one project allow initial norms to be established. These can be used to support planning, as well as interpretation of data, in subsequent projects. The data collected from these projects may then be used to check, refine and expand the set of norms.

To begin with, the empirical foundation for the norms is actually weak, since they are based on the information extracted from one project. It is, however, an advantage to utilize this quantified experience as support in the next project, rather than continuing a purely qualitative approach. As data become available from more projects, the initial assumptions concerning relationships between parameters can be verified and the uncertainties of numerical values can be reduced. In addition, the coverage of the norms can be expanded, and gradually this will lead to a set of company norms, in which the project manager can have confidence, and which will help managers of future projects to learn from the experience gained in previous projects. This process does not follow a strict scientific approach, of first establishing a number of hypotheses, then collecting a statistically adequate number of datasets and finally verifying or rejecting the hypotheses based on data analyses. In a commercial environment, the attitude will be to analyze the first obtainable data set, draw conclusions from (or base decisions on) the results of the analyses and then introduce changes to optimize the development process. The description below is concerned with the first step a company must take in order to pioneer a quantitative basis for the management of software development projects. This includes the collection of a first data set, which describes one of the company's own projects, together with a number of analyses of the data, leading to the establishment of a first set of norms for software engineering data.

Processes

The software development project from which data were obtained can be characterized in the following way. The product: the software product is a real-time information system for use in a highly integrated, but geographically widespread, environment. Requirements for total system reliability are very high, since the consequences of severe faults are critical. It was developed for one customer, consists of four subsystems and has a total (all inclusive) size of 73 OOO lines of code. The process: the development process followed an in- house software development handbook prescribing a life- cycle, documentation level and V & V activities, corresponding to a standard third-generation development approach. In practice, this was followed fairly strictly. This was the case, despite the fact that it was the first time the project group had followed the already existing development handbook, which was well tested in use by other development groups. The development was based on detailed plans, with effort allocated to tasks in the range of 2W3Xl man-hours and follow-up supported by an extensive data collection. The code was written in a high-level dedicated application language and supported by dedicated tools, as well as host and target environments. The personnel: all developers were involved in most of the project, and the majority had several job functions. The average experience among the developers in the application area and in the software development was fairly high. The project: the project was carried out by %25 developers from one department at one site, delivering a total of 19 working years within a calendar time of two years. The period covered the activities from analysis until the product was released to the customer. The delivery took place on time, and the total cost (-overrun) was kept within acceptable limits.

The data and data collection

As an integrated part of the development process and the project management a large amount of data was collected. This was supported by existing administrative procedures and viewed by all involved as a necessity for successful project management. The following data items were among those made available to REQUEST, subject to the condition that the identity of the provider remains undisclosed for commercial reasons:

planned and actual effort for each development and inspection task broken down, so that the planned task effort is in the range of 600 man-hours; [7 size of documents (pages); Information was made available so that it was possible to link the effort consumed to produce a specific document or module to its size, the effort consumed by V & V activities (inspections), as well as the number of errors detected.

Collecting Software Engineering Data

The challenge of collecting software engineering data is to make sure that the collected data can supply useful information for project, process, and eminence management and, at the same time, that the data collection process will not be a burden on development teams. Therefore, it is important to regard carefully what data to collect. The data must be based on well-defined metrics and models, which are used to drive improvements. Therefore, the goals of the data gathering should be established and the questions of interest should be defined before any data is collected. Data classification schemes to be used and the level of precision must be carefully specified. The collection form or template and data fields should be pretested. The amount of data to be collected and the number of metrics to be used need not be overwhelming. It is more important that the information extracted from the data be focused, accurate, and useful than that it be plentiful. Without being metrics driven, over-collection of data could be wasteful. Overcollection of data is quite common when people start to measure software without an a priori specification of intention, objectives, profound versus trivial issues, and metrics and models. Gathering software engineering data can be exclusive, especially if it is done as part of a research program, For example, the NASA Software Engineering Laboratory spent about 15% of their development costs on gathering and handing out data on hundreds of metrics for a number of projects (Shooman, 1983). For large commercial development organizations, the relative cost of data gathering and processing should be much lower because of economy of scale and fewer metrics. However, the cost of data collection will not at all be immaterial. Nonetheless, data collection and analysis, which yields intelligence about the project and the development process, is vital for business success. Indeed, in many organizations, a tracking and data collection system is often an integral part of the software design or the project management system, without which the chance of success of large and complex projects will be reduced.

Data analyses and results

Results are presented below from a number of analyses which have been chosen so as to focus on the establishment of company norms. The norms can be used to interpret measurements made in future projects, and further data from these can then be used to check and improve the norms. Furthermore, some of the analyses give an insight into the development process and show how the process may be optimized. The analyses include detailed comparisons between planned and actual effort for tasks occurring up to the end of coding .The results are used to highlight the sources of significant overruns in expenditure, when compared to the planned cost, and to determine how this can be avoided in future projects. Collection and analysis of software metrics data, even from just one software development project, will provide a company with an improved insight into their development process. This insight will develop further as more data sets become available and they are used to check and improve the initial norms. The analyses of the data may indicate that the underlying part of the process is performed in an acceptable way. In such cases, the data may be used as norms within the company for planning of future projects, or so that data values from future projects can be compared to the norms. When the average effort required to produce one, page in a document is influenced by the complexity of writing the document, the higher error rates recorded for high values of man-hours/page are acceptable, and the trend may be used as a norm as described above. On the other hand, the analyses may reveal problems with an aspect of the development process and possibly may also point to the cause of the problems. In such cases, the increased insight may be used to plan and implement changes to the development process, and to document the impact of the changes by repeating the analyses on data from projects carried out under the changed conditions. The severe overruns seen for these tasks indicate that the effort allocation should be improved to avoid gross underestimations for some tasks (i.e. the planning process must be changed), rather than to continue using the same planning process and then increase the planned values by W%. The data, analyses and results presented here will, in a future project, provide important guidance on how to 0 0 0 By collecting and analyzing a detailed data set, the company has taken a first step towards increasing their software development process maturity level and ultimately, their productivity and end-product quality. The description of the data collection and analysis can therefore serve as an example of the activities that many companies must undertake in the near future, in order to build up quantitative support for project management, in the form of norms for interpreting data from future projects.

Acknowledgement

The authors acknowledge with sincere gratitude to the research supervisor, project team members and those who peer reviewed the article and for constructive suggestions.

REFERENCES

Boehm. B.W. 1981. 'Software engineering economics' (Prentice-Hall Inc.).

Dale, C. 1987. 'The Request database for software reliability and software development data' in Directorate General XI11 (Eds.): 'Esprit '87: achievements and impact (Elsevier Science Publishers B.V., North-Holland).

Duncan, A. S. 1988. 'Software development productivity tools and metria'. Roc. 10th Int. Cod. on Software Engineering, Sine pore, April, pp. 414

Eoerflinger, C.W., and Basili, V.R. 1985. 'Monitoring software development through dynamic variables'. *IEEE* Tram., SE-11, (9), pp. 978-985

Grady, R.B., and Caswell, D.L. 1987. 'Software metrics: establishing a company-wide program' @entice-Hall Inc.

Humphrey, W.S., and Sweet, W.L. 1987. 'A method for assess- ing the software engineering capability of contractors'. Soft- ware Engineering Institute Technical Report CMU/SEI-87-TR- 23.

Kitchenham, B.A., and Walker, J.C. 1989. 'A quantitative approach to monitoring software development', *Soft. Eng. J.,* 4, (l), pp. 2-13

Kitchenham, B.A., Andersen, O., and Klim, S. 1989. 'Interpreting software metria data: a case study'. Request document, R1.10.3.

Petersen, P.G. 1987. 'Software quality: The Constructive Quality Modeling System' Directorate General XI11 (Eds.): 'ESPRIT '86: Results and Achievements' (Elsevier Science Publishers B.V., North-Holland.

****