Software Metrics for Collaborative Software Engineering Projects

Cyril S. Ku
Department of Computer Science
William Paterson University
Wayne, NJ 07470, USA

and

Thomas J. Marlowe
Department of Mathematics and Computer Science
Seton Hall University
South Orange, NJ 07079, USA

ABSTRACT

Many software metrics have been established in the past to measure the various aspects of the software development process. The scopes of the metrics span across the artifacts, the end product, the process to produce these artifacts, as well as the project management for the process. In recent years, driven by advances in telecommunication, the internet, and wireless technology, and also by economic factors, collaboration in software engineering project has become increasingly popular. As collaboration becomes more widespread, software engineering metrics for collaboration, and new or adapted metrics for collaborative projects, will become more important. Nonetheless, no comprehensive study has been done on the impact of collaboration on development productivity, process structure, or software quality. In this paper, we study some commonly used software metrics to investigate whether collaboration can easily be incorporated, and where possible, to suggest strategies for that incorporation.

Keywords: Collaborative Software Engineering, Process Evaluation, Software Measurement, Software Metrics, Software Quality.

1. INTRODUCTION

Since the late 1960’s, software engineering has been the primary tool employed to address the software crisis [18]—problems in the development and deployment of software including project delay, cost overruns, or products delivered with bugs or with missing specified functionality. Although software engineering has not solved these problems, many successful projects and outstanding software products have been produced. Software metrics which include tools and models have been established to evaluate the quality of software, the process of producing the software, and the management of the software project. In this paper, we investigate the elements and issues of software measurement, and indicate whether these elements can incorporate collaborative efforts into the development process and quality measurements of the collaborative software engineering process. We look at a number of existing metrics and see how they can be applied in the collaborative development environment, which in recent years has become increasingly popular, due to the advances in computing environments and interfaces, telecommunication, the internet, and wireless technology, in combination with economic factors. The motivation is to incorporate collaborative effort and the process of collaboration into the existing metrics, and to propose new collaborative software metrics in the future.

This paper is organized as follows. Section 2 briefly discusses the background of collaboration and our motivation. Section 3 takes a look at how collaboration affects software development life cycle in terms of the software product, process, and project. Section 4 discusses collaborative software engineering metrics. Concluding remarks and future research are discussed in Section 5. Acknowledgements and references follow Section 5.

2. BACKGROUND, RELATED WORK, AND MOTIVATION

First of all, this paper treats collaborative software development using a software engineering process model. Such a model embraces all of the standard phases or disciplines—business modeling, requirements analysis, design, implementation, deployment, maintenance, and evolution— involving collaborating and dispersed teams from, and managed by, one or more institutions. We are not considering Computer Supported Cooperative Work (CSCW) [7]—the use of computer systems and applications such as groupware, workgroup support, or live meeting support for collaborative activities and their coordination (except insofar as the software is developed collaboratively). Nor are we directly addressing the
development of Open Source Software [6], although some of collaborative metrics may be applicable to such projects.

Collaborative effort means two or more people or organizations working together to achieve the common goals of the project. While the concept is not new, and distributed software development within one organization is becoming commonplace, we focus on a newer or less common model—diverse collaboration spanning multiple organizations or branches in different geographic locations, and without monolithic central control. Nowadays, because of advances in communication technologies, collaboration in software engineering can occur around the globe in multiple time zones, to take advantage of expertise, human resources, or less expensive labor markets.

Several important questions arise: What effect does collaboration have on productivity and on time-to-market? What effect does it have on software quality or maintainability? Does collaborative development interfere with measuring process compliance? Are any of these measures affected by the quality of collaboration? And how can that quality be reliably measured? Finally, does the existence, form and quality of collaboration affect the precision and validity of the metrics? If so, what changes will restore their usability?

We will attempt to answer these questions using a three-fold approach: First, using existing metrics where appropriate to measure and categorize the differences between collaborative and non-collaborative projects. Second, modifying existing metrics and creating new metrics if needed to account for collaboration. Finally, structure existing, modified, or new metrics to properly account for changes in software development process, and additional structures and artifacts, introduced by collaboration (compare, for example, [11, 12, 13]). Such metrics can then be used to understand the differences between collaborative and non-collaborative development, the tradeoffs better understanding of collaborative environments and approaches will allow modification of metrics to measure collaborative overhead in various dimensions. In this paper, we consider the first two questions. We do not address the third issue, except in passing, but it will be important for future work.

Environments have been created to support collaborative software engineering [4, 19] and the evaluation of such environments has been proposed [5]. These recent research efforts have concentrated on creating and evaluating these environments—which is however not the same as measuring the quality of the software or of the collaboration itself. Our eventual goal is to provide such metrics, if possible by modifying existing, well-accepted metrics or their underlying models to incorporate the effects of collaboration, and the structural and process differences required to support it.

The motivation of our approach is as follows: (1) existing metrics modified to incorporate collaborative development will require only small changes in data gathering, evaluation, and interpretation; (2) the results of the modified metric can be compared with results for the existing metric, both to validate the changes and to see the effects of collaboration; and (3) use of slightly modified but well-established metrics is far more likely to obtain management buy-in and technical acceptance.

In the following section, we look at a number of standard software engineering metrics and how collaborative efforts may affect measurement or interpretation. Then in Section 4, we list the factors that need to be considered when collaborative software engineering metrics are being designed.

3. CONSIDERATION OF COLLABORATION IN SOFTWARE METRICS

Traditionally software metrics can be classified into three categories [9]: product metrics, process metrics, and project metrics. Product metrics describe the characteristics of the software such as size, complexity, design features, performance, quality level, etc. Process metrics are the measurement that indicates the effectiveness of, or compliance with, the process or activities of the software development life cycle. Project metrics deal with the project management aspects of the software development project such as cost estimation, number of developers, staffing, scheduling, productivity, and so on. Some metrics belong to multiple categories. The three categories of metrics are not mutually exclusive—they interrelate to provide indicators about various aspects such as effectiveness of the software development life cycle, the productivity of the process, and the quality of the software product—and in particular the viability of software development as practiced by the development team.

In order to see the impact of collaboration on software development life cycle, an ideal case will be to use some software metrics on the project without collaboration and use the same metrics on the same set of projects with collaboration, and then to compare the results—although of course one couldn’t use the same sets of developers. A more realistic and still reasonable approach will be to apply the metrics to sets of similar projects (projects with similar characteristics in terms of size, complexity, and duration), and to compare those results. We would also want to compare standard metrics to versions modified to account for collaboration as they apply to collaborative and if possible non-collaborative projects, to determine the incremental value of the modifications, and to see whether the modified versions properly measure good practices and results for collaborative projects.

In the following subsections, we look at examples in each of the three categories of software metrics, and discuss the effects of collaboration on these metrics and their interpretation. Similar analyses can be applied to other metrics.
3.1 Product Metrics

One of the simplest but still useful metrics for a software product is the size of the software, normalized for the development language, platform, and software methodology. Size, once so normalized, is an indicator of design complexity, coding and testing effort, and of maintainability and evolvability. The most common size metrics are LOC (lines of code) or KLOC (thousand lines of code) and FP (function point) counts. If these measures are used in quality metrics such as the number of faults per 1000 lines of code, then the effects of collaboration may be important. That is, comparing the number of defects per KLOC with collaboration and without collaboration will tell us the impact of collaboration. Further, the number of KLOC produced with collaboration or without collaboration is an indicator of productivity. (This is one example for which isolating the code overhead of program structures to support collaboration will be useful.)

Most of the product metrics do not include factors, attributes, or elements in their formulae that we can include collaborative characteristic. In other words, KLOC is KLOC, although KLOC produced with collaboration and KLOC produced without collaboration may be different. However, in considering defects per KLOC, it may prove useful to separate defects within a single collaborator's module from defects at the interfaces, or resulting from module interactions—and to compare these to analogous categories in code developed without collaboration.

Moreover, considering function points, it may be possible to consider the level of collaboration in FP computations, even though it is a process activity rather than a product characteristic—as one might consider whether there is continuity between the developers and maintainers of a system.

Other product metrics that inherently do not account for collaboration are MTTF (Mean Time To Failure) and MTBF (Mean Time Between Failure): cyclomatic complexity [15]; module cohesion and module coupling, including object-oriented metrics of class size, cohesion, and coupling; and SMI (Software Maturity Index) [8] where the number of modules is counted and the ratio is calculated and tracked between releases. It will still be interesting to compare collaborative and non-collaborative projects on these metrics, and to account for any effects of collaborative process structures and artifacts.

3.2 Process and Project Metrics

Process metrics are usually collected across all projects and tracked over long periods of time. They are used to make strategic decisions. Project metrics, on the other hand, are used to track and control on-going projects. They are used to make tactical decisions as a project proceeds. Finally, process compliance metrics have both project and process aspects, and are used in software quality assurance.

In general, these categories of metrics concern about the productivity of the software team and the quality of the software product produced. Boehm [1] has shown that the skill and motivation of people to be the single most influential factor in quality and performance. Since people are the major participant in collaboration, collaboration has direct impact on productivity and quality. Unlike product metrics, which usually do not directly measure collaboration, process and project metrics can include direct attributes that take collaboration into account.

As mentioned in Section 3.1, a product metric such as KLOC can become a project metric to measure productivity if you use KLOC, and likewise FP, in measuring cost per KLOC, KLOC per person-month, FP per person-month, and so on. A simple example of quality metric that does not need to measure collaborative effort but whose result will be quite different if the activity involves collaboration is MTTC (Mean Time To Change). MTTC is a measure of the time it takes to go through an iteration (or iterations) of a change request. The activities of the iteration include the time to analyze the change request, design appropriate modification, implement the design, test the modification and perform regression testing, and finally deliver the modification to the users. Each of the activities mentioned above is obviously affected by collaborative effort, and by the quality of dependence tracking [11]. If most faults can be quickly isolated into one module, or to interface incompatibilities, then MTTC may actually be reduced; on the other hand, frequent problems crossing module boundaries are likely to result in increased MTTC.

We should also note that a negative change in one metric may correlate with desirable results in another. For example, MTTC may increase for a collaborative project, but the specialized expertise in and additional validation needed for a collaborative project may result in larger MTTF, indicating increased system reliability. We need to take a global perspective in evaluating the impact of collaboration.

We are even more interested in project and process metrics that need to take collaboration into account. One standard class of examples is the estimation models for software projects where historical data are being used to determine parameters. For example, COCOMO (the Constructive Cost Model) [1, 2], developed by Boehm in the 1980’s, used historical project data at TRW. Depending on the nature of the project (different types of software and environment), the model predicts the effort, schedule, personnel, maintenance, etc., for the various phases of software development. COCOMO incorporates mathematical formulae, statistical modeling, and expert judgment to estimate the various costs and efforts for software development. Obviously, if we want to use COCOMO for estimation for collaborative software projects, historical collaborative data need to be used to
derive a different set of multipliers, and perhaps even additional variables, for the formulae of the model. COCOMO itself has evolved from using the waterfall model of software development to accommodate a wide variety of modern software engineering methodologies, including object-orientation in COCOMO II [3]. Some of the multipliers in COCOMO II even take whether a project is being developed in multiple sites into account. These modifications are a good starting point for a collaborative model, but experimental data and research effort are needed to incorporate collaborative effort into estimation models such as COCOMO.

Finally, one can consider metrics applied to other process disciplines. Testing metrics [9, 20], for instance, can be thought of as partially product quality metrics and partially process compliance metrics. A white-box metric such as “all branches covered” clearly makes as much sense for code developed collaboratively as it does for single-site development. Metrics for integration can also be used without change, but collaborative design will specify, design and use interfaces between partner components very differently, and the results may need to be interpreted differently.

4. FACTORS IN COLLABORATIVE SOFTWARE ENGINEERING METRICS

We have examined the effects of collaboration on the various software metrics and what factors need to be modified in these metrics. Can we design a set of metrics for collaborative software engineering? It is beyond the scope of this paper to design such metrics. In attempting to design collaborative software engineering metrics, many factors or variables must be considered. Mattessich in his book [14] lists twenty factors that can make or break any group effort. His book reviews hundreds of scientific studies about successful and not-so-successful collaborations. The factors fall into six general categories: environment, membership, structure and process, communication, purpose, and resources. Mattessich talks about any undertaking that involves people and organizations working together. They are, of course, applicable to collaboration in software development endeavor. We list some of the things that need to be considered under Mattessich’s six categories when designing a metric for collaborative software engineering:

- Environment – This comprises both the working environment and the software environment (are there collaborative software tools available?). If your software development organization does not have a history of collaboration, then there is a learning curve in educating, shaping expectations, and developing buy-in among all the people participating in the software development life cycle, both technical and management. Trust is important for collective reliability, competence, and intentions.
- Membership – Who (system analysts, domain experts, designers, developers, testers, etc.) participate in the collaborative process? Variables include size and breadth of the group. Again, trust and understanding and mutual respect are important factors in collaboration.
- Structure and Process – Software project management needs to organize the different collaborative partners. What is the structure of the collaborative efforts? What are the processes involved within this collaborative structure? Which stages or phases of the software development life cycle are involved in the collaboration?
- Communication – Successful collaborative groups communicate often and well. How do the various groups communicate and how often they need to do it? What level of communication is needed (e.g., project manager level or programmer level)? Communication technologies and collaborative software tools play a role in here, so does the structure of the collaborative teams.
- Purpose – Each participant in the collaboration need to have specific and realistic goal and objectives. Collaborative efforts in every phase of the software development life cycle need to have common vision and shared tasks, both short-term and long-term.
- Resources – Collaboration takes substantial efforts, consistent funding and staffing. It includes hardware and software resources.

One also has to consider whether the collaboration occurs within different divisions of a single organization, or involves multiple organizations, cultures, or languages. In the latter case, each of the above bullets carries with it intensified risk, and a need for management as well as technical coordination [16, 17]. In designing metrics for collaborative software engineering, the challenge is to come up with a small set of measures that takes the most significant elements in each of the above categories into account. Perhaps many metrics can be constructed to characterize all the attributes of collaborative software engineering efforts. These metrics will probably include qualitative or subjective evaluation and quantitative or objectives data. A good reference for validating proposed metrics for software engineering is presented in [10].

5. CONCLUDING REMARKS AND FUTURE RESEARCH

In this paper, we examine the attributes in some of the existing software metrics that need to be modified or incorporated. We also discuss the factors need to be considered when new collaborative software engineering metrics are constructed. Instead of creating new collaborative software engineering environments and establishing new metrics for such environments, we feel that initial efforts should look at existing metrics and
models and see how they can be used to fit into collaboration. Such a survey will also reveal aspects of collaboration not easily covered by modifying existing metrics; in such cases, we can explore additional models and metrics. This paper merely defines and structures the problem; much future work will be needed to evaluate the effectiveness of collaboration in terms of development productivity and software quality.

Further, no matter how much theoretical analysis is applied to a metric, its effectiveness and usefulness have to be demonstrated and validated using real-world experimental data. The complementary next step in our research will be gathering experimental data as available. We also need to take a close look at the factors of collaboration to understand what types of data we need to gather to see the effects of collaboration.

6. ACKNOWLEDGEMENTS

This research was supported in part by the ART (Assigned Released Time for Research) program, Office of the Provost, William Paterson University of New Jersey.

7. REFERENCES


