PROCESS ACTIVITIES IN A PROJECT BASED COURSE IN SOFTWARE ENGINEERING

Éric Germain[1], Pierre N. Robillard[2], Mihaela Dulipovici[3]

Abstract *¾ "Studio in Software Engineering" is a curriculum component for the undergraduate-level software engineering program at École Polytechnique de Montréal. The main teaching objective is to develop in students a professional attitude towards producing high quality software. The course is based on a project approach in a collaborative learning environment. The software development process used is based on the Unified Process for EDUcation, which is customized from the Rational Unified Process. An insight into the dynamics of three teams involved in the development of the same project allows us to present and interpret data concerning the effort spent by students during particular process activities. The contribution of this paper is to illustrate an approach involving qualitative analysis of the effort spent by the students on each software process activity. Such an approach may allow the development of a model that would lead to effort prediction within a software process in order to designate the actions for improving academic projects.*

Index Terms ¾ collaborative learning environment, effort monitoring, project based course, software engineering education, software process discipline.

INTRODUCTION

Delivering quality software is no longer an advantage but a necessity for companies to be successful. The Internet changed the main software development priority from *what* to *when*. The new business environment demands that software products be delivered more quickly, but also that they offer greater functionality and stand higher quality levels. Increased software quality in a reduced time-to-market has become now one of software engineering's most important missions. As the context of software development is changing, software engineering education has to face these rapid changes [1].

The software process is becoming a major concern in most software development organizations. There are different viewpoints on the meaning of software development. Essentially, a software process is a set of activities and artifacts that must be performed and completed by individuals having different roles in the software's life cycle, as presented in the conceptual model using the UML notation for class representation [2]. Identifying these distinct roles, providing specific abilities for each and developing in students a professional attitude towards producing high quality software are challenges faced by software development process teaching.

Efficiently controlling and improving a software engineering process implies monitoring the effort involved in the development of a software product in order to better understand how the effort is distributed [2]. The measure of effort (the amount of staff-hours spent on any given activity) allows identification of effort spent by students (as software developers) on various tasks such as programming or documenting. This, in turn, allows introspection into the software process and could help identify the benefits and weaknesses of the process in order to help improve it.

The process used is based on the UPEDU (pronounced Yoopeedoo), which is an acronym for Unified Process for EDUcation, which is customized from the Rational Unified Process (RUP) [3]-[4]. UPEDU introduces students to the software process activities and their corresponding artifacts and enables them to understand the roles played in software development.

This paper presents the results of the effort measurements carried out within the framework of the undergraduate-level, one-semester course "Studio in Software Engineering" at École Polytechnique de Montréal. The objectives are twofold: to characterize the patterns of effort over the project duration, and to evaluate the similarities of a given discipline's effort pattern across various projects developed in a collaborative learning environment. The qualitative analysis of the effort spent by the students on each software process activity may allow the development of a model that would lead to effort prediction within a software process, in order to designate the actions for improving academic projects.

PROJECT DESCRIPTION

The course "Studio in Software Engineering" at École Polytechnique de Montréal is different from a conventional software engineering course in its goal and teaching method. The general objective of the studio is to teach software development through a project-oriented course. The teaching method is based mainly on teamwork. The collaborative

This work was supported in part by National Sciences and Engineering Research Council of Canada under grant A0141.
[1] Éric Germain, École Polytechnique de Montréal, Laboratoire de recherche en génie logiciel, eric.germain@polymtl.ca
[2] Pierre N. Robillard, École Polytechnique de Montréal, Département de génie informatique, pierre-n.robillard@polymtl.ca
[3] Mihaela Dulipovici, École Polytechnique de Montréal, Laboratoire de recherche en génie logiciel, mihaela.dulipovici@polymtl.ca

aspect of the teamwork is often a key issue in software development. All the teams are in competition on the same project and have to follow the same software development process. This course does not contain a final exam but a class presentation and a formal acceptance test session.

The 13-week project is a Client-Server application programmed in Java software language, representing a Time Monitoring Tool (TMT) for software development teams. The TMT system to be developed is a stand-alone tool that is integrated within an organization's Intranet. The tool consists of four major components: a Developer Client Module, a Manager Client Module, a Server Module, and a Database. All components must be executed on a Windows NT environment.

Most students have little industrial software development experience, and many of the process activities have little meaning for them. The UPEDU enables them to select activities, which are more specific to a given task. The difficulty is to maintain a good balance in the process activities in terms of the various conceptual viewpoints they represent. The presence of too many activities can reduce the learning process to a boring experience. On the other hand, there should be enough activities to build a good software process, which is of academic interest.

The UPEDU process has been adapted to the student projects. The main objective for the students in this study was to realize their project by using that process. The students were asked to use UPEDU in order to help define their software's life cycle and to create the appropriate artifacts for each activity. They were also encouraged to adapt the different roles proposed by UPEDU.

The total duration of the project was spread into four iterations, which proceeded according to an established plan and ended in an internal release. Table I illustrates time allocation for each iteration:

TABLE I
DISTRIBUTION OF ITERATIONS DURING PROJECT DURATION

Week	1	2	3	4	5	6	7	8	9	10	11	12	13
Iteration	1			2			3				4		

The students attending this course were supposed to have completed the required software engineering courses and already had some level of computing skills. However, some of the students had little experience in Java programming. Each student was expected to spend at least 135 hours on this project for a total of 675 person-hours for a 5-person team. The class included 3 teams, two of them (team A and B) being composed of five students and the other one (team C) being composed of four. It was later found that two of three teams had exceeded the allotted time, as shown in Figure 1.

Effort was measured by requiring the students to record the time spent on each of UPEDU's software development activities. Analysis of the data shows the variability and the similitude in the process disciplines for the various teams. It also shows were and how the effort is involved in developing software. The main fields of effort recording information are presented in Table II.

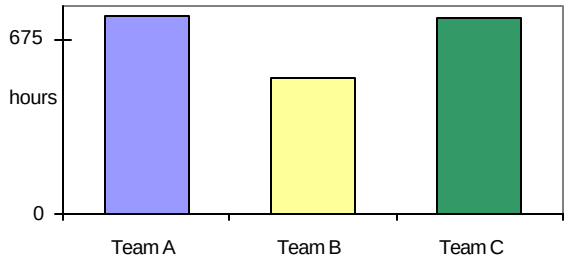

FIGURE 1
TOTAL EFFORT FOR EACH TEAM (IN PERSON-HOURS)

TABLE II
EFFORT RECORDING INFORMATION

Worker Name	Worker	Activity	Artifact	Date	Start Time	End Time	Duration

For each record, the students inserted a short description of the entered data. The duration of the activity was automatically calculated. All the data presented in this article were obtained through analysis of these team databases. Many diagrams are presented showing the distribution of the effort over the time according to the various disciplines, for the three teams. Discussions of the results highlighted salient features of the software process.

RESULTS

Effort Distribution within a Discipline for Each Team

Table III shows the maximum deviation to the mean cumulative effort made by any team for each discipline. It must be noted that the maximum deviation is found to be around 0.20 for 4 of the 6 disciplines, whereas the other two disciplines have figures that are significantly higher. Table IV shows the standard deviation of weekly effort for each team.

TABLE III
MAXIMUM DEVIATION TO THE MEAN CUMULATIVE EFFORT BY DISCIPLINE

Discipline	Week	Deviation
Requirements	2	0.20
Analysis & Design	6	0.20
Implementation	11	0.19
Test	12	0.28
Configuration & Change Management	2	0.33
Project Management	4	0.20

Figures 2 to 7 illustrate the effort distribution within each discipline for each team. Comparison of effort

distribution between teams is required to bring some measure of pattern convergence and to discuss the relevance and validity of using an aggregate model for further analysis. Totals have been normalized with the total effort for a team within a discipline so that the effects of overall productivity variations between teams are removed.

TABLE IV
STANDARD DEVIATION OF WEEKLY EFFORT BY DISCIPLINE

Discipline	Team A	Team B	Team C	Mean of teams
Requirements	0.15	0.14	0.10	0.13
Analysis & Design	0.06	0.13	0.09	0.09
Implementation	0.08	0.11	0.08	0.09
Test	0.16	0.16	0.07	0.13
Configuration & Change Management	0.10	0.23	0.09	0.14
Project Management	0.10	0.13	0.09	0.11

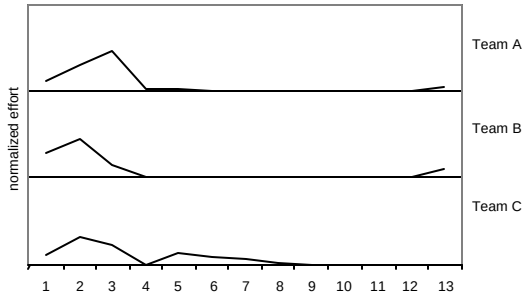

FIGURE 2
EFFORT COMPARISON – REQUIREMENTS

Comparative analysis of effort by team for the Requirements discipline shows a similar pattern for all three teams. The pattern consists of intensive effort at the beginning of the project with 60% to 80% of total work completed at the end of the first iteration and 70% to 90% of total work completed after four weeks. Maximum work done by a team during a single week is approximately 45% for weeks #2 and #3.

The Analysis and Design discipline is more uniformly spread along the time span, with maximum work done by a team within a single week peaking at 25% of the total work on that discipline, with the exception of week 6 for team B with a burst of 40%. Nonetheless, the average standard deviation for weekly team effort is 0.09, which is the lowest of the six disciplines. Effort emphasis is put on weeks 3 to 6.

Work in the Implementation discipline has a quite similar to the Analysis & Design discipline effort distribution among the three teams, with an average standard deviation also equal to 0.09. Work is performed most intensively throughout the second half of the project.

Patterns for the Test discipline are less uniform between the three teams than for the preceding disciplines. This discipline shows a maximum deviation to cumulative average of 0.28, which represents the second greatest deviation, second to the one for the Configuration & Change Management discipline. Figure 5 illustrates that teams A and B follow a similar pattern of intermittent work in the first 5 to 6 weeks of the project, absence of work for the following 5 to 6 weeks, and intensive work in the last two weeks. Work in team C has been much more uniformly spread, with a weekly effort standard deviation of 0.07.

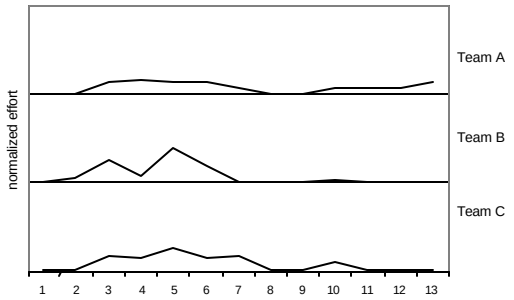

FIGURE 3
EFFORT COMPARISON – ANALYSIS & DESIGN

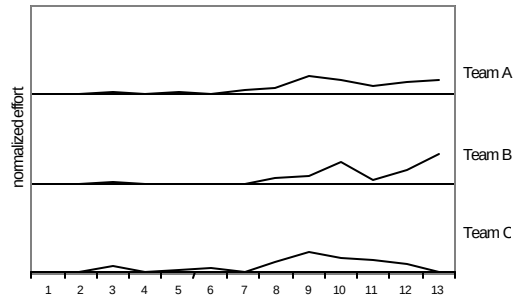

FIGURE 4
EFFORT COMPARISON – IMPLEMENTATION

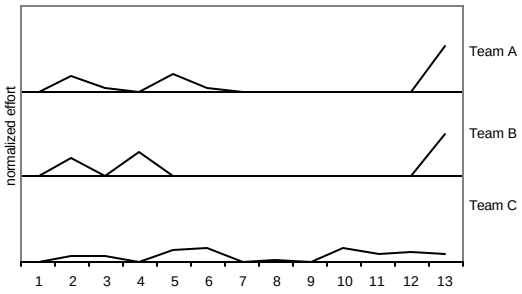

FIGURE 5
EFFORT COMPARISON – TEST

The Configuration & Change Management discipline shows the most important variations between the three teams, with the only common point being medium-to-high

work intensity during week 2. This is especially true for team B, which performed 80% of their work in that sole week. In comparison, team C had performed only 40% of their total work in that discipline after week 4.

As for the Requirements discipline, the Project Management discipline offers much similarity between teams and shows uneven effort distribution over time. Much effort is spent at the beginning of the project, amounting to 60% or more after the first four weeks. For two of the teams, significant work is performed in the last two weeks of the project.

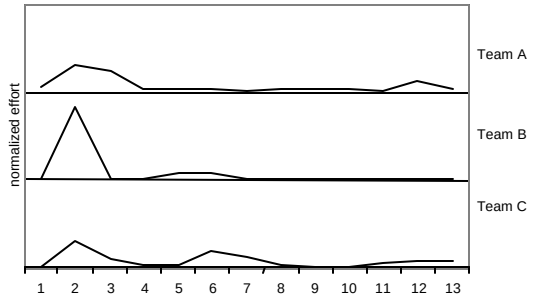

FIGURE 6
EFFORT COMPARISON – CONFIGURATION & CHANGE MANAGEMENT

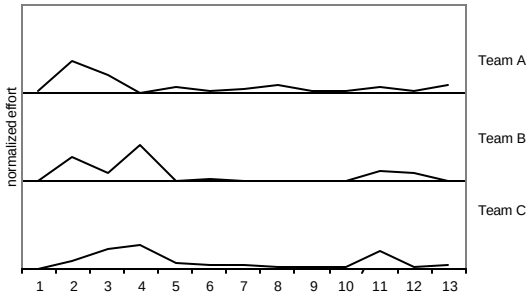

FIGURE 7
EFFORT COMPARISON – PROJECT MANAGEMENT

Convergence of data for all three teams

For data analysis purposes an aggregate model combining the results for all three teams can be used to illustrate the typical patterns of effort during a project. Figure 8 illustrates the non-normalized, aggregate effort distribution for the six disciplines.

Limitations of such a model can be analyzed in the light of the relative effort made on each discipline. For instance, even though the maximal deviation from average cumulative effort amounts to 0.33 for the Configuration & Change Management discipline, that effect is reduced because of the small total amount of work that has effectively been performed on that discipline relatively to the other disciplines.

The total relative effort for each discipline is illustrated in Figure 8. Effort spent in the Implementation discipline constitutes by itself 45% of the total time, and the Implementation and Analysis & Design disciplines combined make up almost two-thirds of that total time.

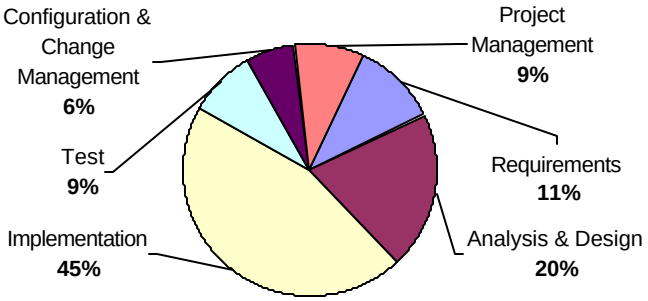

FIGURE 8
Distribution of Effort through Disciplines for All Teams

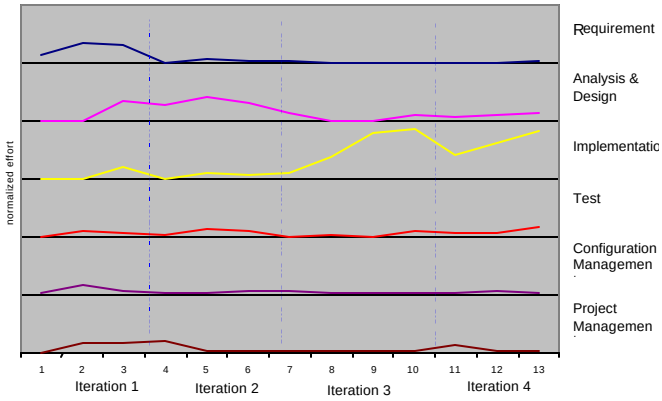

FIGURE 9
COMPARISON OF EFFORT FOR EACH DISCIPLINE FOR ALL TEAMS
(SCALE = MAXIMUM 200 PERS*HR FOR EVERY CURVE)

Table V shows the total effort for each discipline. It can be observed that the four most effort intensive disciplines (Requirements, Analysis & Design, Implementation, Project Management) correspond to the most stable ones according to Table III. We can therefore conclude that the aggregate model can be used with proper precaution to evaluate patterns that will not diverge by more than 20% for the most effort intensive disciplines.

TABLE V
TOTAL EFFORT PER DISCIPLINE

Discipline	Total effort (pers*hr)
Requirements	210
Analysis & Design	400
Implementation	900
Test	170
Configuration & Change Management	110
Project Management	180

Aggregate disciplines by week

Figure 8 shows that the teams have successively concentrated their works on the three most effort intensive disciplines. More specifically, weeks 1 to 3 have been focused on the Requirements discipline, weeks 3 to 6 on the Analysis & Design discipline, and weeks 8 to 13 on the Implementation discipline. It must be noted that both the magnitude and time span of those focus areas rise chronologically from one discipline to another. In contrast, effort on the three other disciplines (Test, Configuration & Change Management, Project Management) is both smaller and more evenly distributed. Due to the relatively small amount of work, spikes detected in the normalized figures for the latter have a much lesser amplitude than for the three most effort intensive disciplines.

Aggregate disciplines by iteration

Figure 10 illustrates the cumulative work by iteration for each discipline. It can be observed that work on the Requirements discipline is clearly concentrated in the first iteration and that the same is true for the Analysis & Design Discipline with the second iteration. The Implementation discipline dominates by far the third and fourth iterations in a very similar fashion.

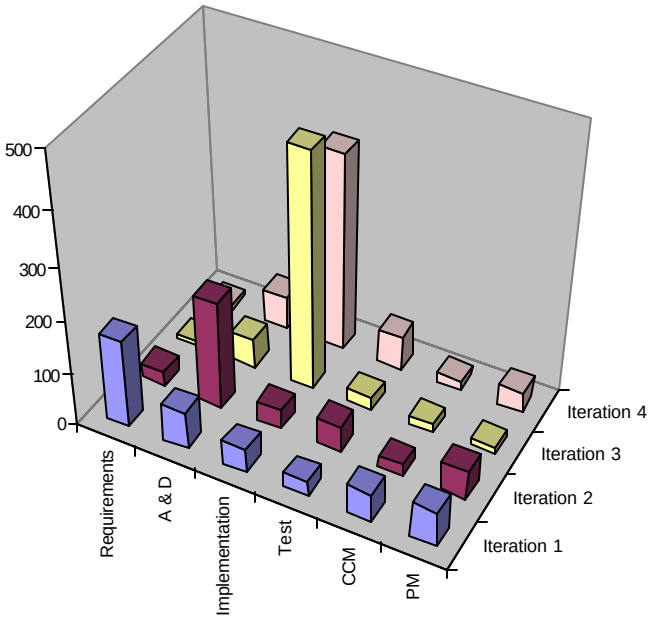

FIGURE 10
EFFORT BY ITERATION

Work on the other disciplines, while not completely even among the iterations, does not show as many spikes due to the relatively low absolute amount of work involved. More effort has been put on tests at iteration #4 and on configuration management at iteration #1, and project management tasks seem not to have been much performed at iteration #3.

DISCUSSION

Even though the process used is based on the concept of iterations, one can find behaviors that are related to the classic waterfall software process model, particularly within the engineering disciplines. This is shown for instance in figure 10, where the Requirements, Analysis & Design and Implementation disciplines dominate clearly one or two iterations. However, the Test discipline offers a behavior that is very different from the waterfall model. Much effort is spent on tests at the beginning of the project, but very few activities are performed during weeks 8 to 10. It must be noted that, in UPEDU, the unit test activity is not part of the Test discipline but rather of the Implementation discipline. In the first two iterations, the process requires a lot of test planning effort. In particular, use-case construction requires the availability of proper test plans. Meanwhile, validation and acceptance tests are performed only at the very end of the project (week 13).

The Configuration Management discipline shows a very unusual profile. One would expect the effort to be made in a continuous fashion. It has rather been found that much of the effort was concentrated in iteration #1. A lot of effort was required for the construction of the Configuration Management Plan, as students are not familiar with that concept. In fact, except for the effort spent on configuration management planning, the overall effort for this discipline is very low.

The Project Management discipline shows a quite similar situation, whereas a uniform distribution was expected but not found. High effort during the first iteration was due to overall planning activity as well as to the mutual familiarization of the team members, which were not used to work together. Also, a lot of effort has been made in project management activities during iteration #4 in order to plan for the last steps of the projects, which faced very strong deadlines. Even though every team invested an important amount of effort throughout the project, it has been found that the effort spent during iterations #2 and #3 was significantly lower than for the rest of the project, possibly because of the absence of any perceived need to do more management activity.

The comparison of all disciplines over the 13 weeks offers the following observations:

- The Requirements and Analysis & Design disciplines constitute the major part of the effort spent during the first half of the project. In particular, the activities of the requirement discipline are executed in parallel with the other activities, since many tasks must be performed at the very beginning of the project in order to conform to the process.
- The Implementation discipline constitutes by far the most important component of the project. It is especially prominent during weeks 8 to 10. When work on implementation issues begins, it monopolizes all the

team's energy and very few other activities occur simultaneously.

- The construction of a prototype required in order to complete the requirements specification shows up as an effort burst during week #3 in the Implementation discipline.
- The results show that the use of this particular process (UPEDU) might have enforced the beginning of the implementation work only at the middle of the project. According to that interpretation, in the case where this process would not have been used, the implementation work would have started much earlier. Even though implementation effort is important, it is well cut-out and starts only when the prerequisites have been fulfilled.

The comparison of all disciplines over the four iterations that have been considered offers the following observations:

- As expected by the process, some effort has been spent within each discipline and iteration.
- Some disciplines dominate one or many iterations. Specifically, the Requirements discipline dominates iteration #1 and the Analysis & Design discipline dominates iteration #2, while the Implementation discipline dominates iterations #3 and #4. It must be noted that the Test discipline does not dominate any iteration.
- As expected by the process, the Configuration & Change Management and Project Management disciplines' effort does not dominate any particular iteration.

CONCLUSION

This work shows that the study of activities and of their corresponding disciplines may allow the development of a model that would lead to the prediction of effort spent within each discipline and iteration. Such a model would be very useful for planning, scheduling and reporting tasks.

Some process activities can be confusing or ambiguous for some students [5]. This can be overcome by adequate training or by redefinition of some activities. In particular, it is important that the activities considered for effort measurement be relevant and well understood by the development teams. Effort recording is a difficult task. Some participants are not well disciplined to record all their activities or to assess correctness of entered values. We have passed over that difficulty by assigning a person that was in charge of ensuring accurate recording. It is also important that the process be defined as to be appropriate and tailored to the specific project.

The way the teaching techniques are used in the course "Studio in Software Engineering" allowed students not only to gather new knowledge but also to develop abilities that can be directly applied in their soon-to-come professional projects. Furthermore, students are well prepared for independent lifelong learning with relevant skills.

UPEDU allows students to "play" different roles in the process, which is a good practice in specialization by level of responsibility. [6]

The goal of the study is to show the feasibility of measuring activity effort within a process with the objective of building a model. As a note of caution, the results of this study are derived from student projects realized in a very well defined context. Therefore, it would be inappropriate to generalize to industrial projects the results presented in this paper on that basis.

ACKNOWLEDGMENT

This project would not have been possible without the participation of all the students who enrolled in the "Studio in Software Engineering" course at the winter 2001 semester. We are grateful to Houcine Skalli for his work as Teaching Assistant during the course, and to Martin Robillard who provided the formal specification for the software to be built. We would also like to thank to John Slavich for his helpful advice.

REFERENCES

[1] Robillard P.N., D'Astous, P. Kruchten, P. Software engineering process with UPEDU, Addison Wesley, August 2002.

[2] Robillard P. N., "Case study analysis of Measuring Effort in a Software Engineering Process ", *Third Maghrebian Conference on Computer Sciences (MCSEAI'2000)*, Fes, Morocco, 2000.

[3] Robillard P. N., Kruchten P., d'Astous P., "YOOPEEDOO (UPEDU): A Process for Teaching Software Process", *14th Conference on Software Engineering Education & Training*, Charlotte, NC, USA February 2001,pp 18-26.

[4] Kruchten P., "The Rational Unified Process: An Introduction", Addison-Wesley, 2000.

[5] Germain É., Dulipovici M., Robillard P.N., "Measuring Software Process Activities in Student Settings", *Proceedings of the 2nd ASERC Workshop on Quantitative and Soft Computing Based Software Engineering (QSSE 2002)*, February 18th-20th, Banff, Alberta, Canada, pp 44-49.

[6] Shaw M., "Software Engineering Education: A Roadmap", *The Future of Software Engineering*, ACM, 2000, pp373-380.