Evaluating RUP Software Development Processes Through Visualization of Effort Distribution

Werner Heijstek (heijstek@liacs.nl) Michel R. V. Chaudron (chaudron@liacs.nl)

Leiden Institute of Advanced Computer Science
Foundations of Software Technology, Software Engineering Section
Leiden University, Niels Bohrweg 1, 2333 CA Leiden, The Netherlands

Abstract

In this exploratory case study, effort distribution visualizations of industrial software development projects are made in order to assess to what extent patterns can be found that describe the nature of the distribution of effort. The visualization of effort distributions of two Rational Unified Process (RUP) projects are presented and discussed. Data was collected from hour registration systems, visualized in the image of the RUP 'hump' chart and analyzed for striking features or abnormalities. Senior project team members were confronted with the analysis in order to verify the findings. Although the visualizations show interesting patterns, they cannot be interpreted without context information. The visualizations were evaluated to be a useful addition to project post-mortem analysis.

1. Introduction

Analysis of valid [2] software engineering data from industrial practice is necessary to support hypotheses and theories in the academic discipline of software engineering. In addition, conclusions based on analysis of observations lead to insights that are also valuable to industry.

Research regarding distribution of effort in software processes is commonly found in literature on software estimation and planning: [18], [12], [1], [17]. However, a large portion of the research in that area deals with estimating the total amount of effort needed for a project for specific conditions or development methods such as reuse of code [16] or use-case based requirement specifications [8]. What an effective distribution of effort over disciplines is, is currently unaddressed in literature. Important reasons are a lack of data on software process in general and problems with regards to comparability of the data specifically. This study focuses on effort distribution over the lifespan of industrial, custom software development processes. It does so for three reasons:

First, effort distribution is studied to improve our understanding of project dynamics from a resource perspective. Visualization of software engineering process effort distribution aides in reasoning about process dynamics such as the effects of a chosen iteration strategy. Furthermore, such visualizations provide insights into the interaction between the resources spent on disciplines such as implementation and testing or requirements and analysis and design. These insights could for example lead to improved project planning practices in terms of a more optimal resource allocation – which implies cost reduction.

Second, analysis of effort distribution is necessary to develop a method for project management to gain insight in resource allocation. Observations from visualization of effort data during a project elicit trends and provide an alternate view of the progress of a project. Our comparison study provides evidence for a relationship between effort distribution and the number of defects found [11]. That research suggests that the ratio of the effort spent on requirements, analysis and design, and implementation disciplines, can serve as an indicator of the number of defects that will be found during a software development project.

Third, effort distribution visualizations are presented to follow up on earlier work on effort visualization. A figure that is commonly referred to in the context of project planning is the 'hump' figure used in the documentation for the Rational Unified Process (RUP) that depicts the effort that would be spent during a project on each of the nine disciplines RUP prescribes. Port, Chen and Kruchten [19] attempted to validate the RUP hump diagram earlier by means of student experiments. Their conclusions were that the visualization of their data obtained by a student experiment was similar to the RUP hump image. Contrastingly, this study presents data that was obtained from industrial practice to empirically validate the hump image. The work of

978-0-7695-3276-9/08 $25.00 © 2008 IEEE
DOI 10.1109/SEAA.2008.43

Port et al has been followed up before [10] albeit on an aggregate level. This study examines individual projects. The structure of this paper is as follows: The following section will elaborate on the Rational Unified Process and RUP 'humps'. Section 3 will explain the research method and section 4 will outline the results. Section 5 explains the threats to validity, section 6 contains a discussion of the findings. Finally, section 7 contains a conclusion.

2 Concept Definition

2.1 The Rational Unified Process

The RUP is a commonly used project framework in software engineering practice. It provides a disciplined, iterative approach to the assignment of tasks and responsibilities in software development projects [15].

The RUP is an adaptable process framework that is architecture-centric and risk-driven. In RUP, software engineering processes are organized into phases and iterations. A project consists of four phases which correspond with the first four main stages of the waterfall model as described by Ghezzi, Jazayeri and Mandrioli: requirements definition, system and software design, implementation and unit testing, and integration and system testing [9]. During the *inception phase* the business case and the financial forecast are created as well as a use-case model, a risk assessment and project description. The *elaboration phase* is used to perform problem domain analysis and to shape the architecture. During the *construction phase* the development and integration of components are the central activities. Finally, the *transition phase* the software system that is developed will be implemented at the customer's organization. In RUP, the effort that is spent on activities are categorized into 9 disciplines. The *business modeling* discipline is concerned with activities that bridge business and software engineering in order to understand business needs and to translate them to software solutions. The *requirements* discipline is concerned with elicitation and organization of functionality and non-functional demands and aims to create a description for what the system should do. The *analysis and design* discipline is concerned with the mapping of requirements to a formal design. The resulting design model acts as a blueprint for the source code. A modeling language such as the Unified Modeling Language (UML) can be used to design classes and structure them into packages with well-defined interfaces. During the *implementation* discipline, the actual implementation of the components is made, either by reuse or by creation of new components. The *test* discipline serves to verify the completeness and correctness of the implementation of the requirements. This discipline is also responsible for the elicitation of defects and their respective fixes. The *deployment* discipline is concerned with product releases and end-user delivery of these releases. Activities that fall in the *configuration and change management* discipline deal with change requests with regard to project artifacts and models, and version control of these changes. The *project management* discipline focuses on progress monitoring of iterations through collection and analysis of metrics, planning iterations and management of risk. The *environment* discipline aims at activities that facilitate the configuration of a project and project support in general by means of tools and supporting processes.

2.2 RUP Humps

The term RUP 'hump' refers to a plot of effort spent over time during a particular discipline. The RUP 'hump chart' consists of a collection of humps for all RUP disciplines. This diagram was created in 1993 during a workshop on architecture and process [14] and was inspired upon work by Booch [7] and Boehm [4] [6]. It has been part of the Rational Objectory Process after reviews by Dyrhage and Bylund and moved on to play a more important role in the RUP in 1998 when it served as the opening page for the digital version of the process [14]. Its final form was published by Kruchten in 1998 [15]. An older version was later used by Jacobson, Booch and Rumbaugh [13] and an altered version was used by Royce [20]. A recent version of the RUP chart is depicted in figure 1. Over the years this diagram has be-

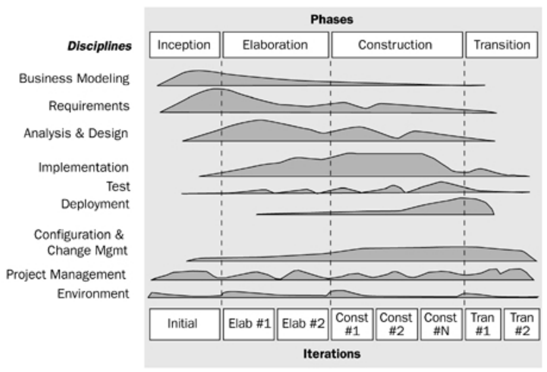

Figure 1. A recent version of the RUP 'hump' chart

come increasingly connected with RUP in such a manner that sometimes it is perceived as if it was a logo for the process. The chart has been spread widely over the Internet. A known misconception about the hump chart is, that it is based on empirical assessment of actual projects rather then on the educated guess of Kruchten.

"... I always insisted that these humps were just illustrative, as well as the number and duration of iterations shown on the horizontal axis, but

many people wanted to read much more meaning in that diagram that I intended. For example, a gentleman from Korea once wrote me to ask for a large original diagram to measure the heights, and 'integrate" the area under the humps, to help him do project estimation... " [14]

In 2005, Port, Chen and Kruchten [19] tried to empirically validate the RUP hump chart. They assessed the effort spent in a group of 26 student projects which served as an introduction to software engineering. The projects had a lead time of 24 weeks. The students participating were all graduate level students at the University of Southern California's Center for Software Engineering. All projects were structured around the CS577 Model-Based Architecting and Software Engineering (MBASE) guidelines [5]. In their research, Port, Chen and Kruchten create a mapping from the CS577 effort reporting categories to the RUP disciplines and they note that, although CS577 projects are representative of RUP projects, they "do not strictly follow all the RUP guidelines". Their finding was that "CS577 projects generally follow the suggested RUP activity level distributions with remarkably few departures".

An important difference between the experiments conducted by Port et al and this study is that their effort was already reported in terms of RUP disciplines. An effort mapping was therefore not necessary.

3 Methods

Detailed hour registration data was collected from the software development department of a large IT service provider. This data was visualized, consistent with the RUP hump chart and these visualizations were analyzed. Finally, senior project members were confronted with the process visualizations. The following paragraphs describe the research environment, the data collection process, visualization process and the validation of the data.

3.1 Research Environment

The software development department offers services to projects in a range of categories that are called competence centers. Most of the facility's customers use a subset of these services although some adopt the entire software delivery process. The competence centers are estimation and measurement, assembly line, process coaching, tool support and infrastructure. The estimation and measurement team is responsible for quantitative analysis of projects before, during and after execution. The assembly line team offers 'continuous integration' services with regard to software development. Process coaches are responsible for providing help and training to project team members to help them to work more efficiently, more effectively and according to RUP specifications. The process coach uses the output of the estimation and measurement team, the assembly line and interviews with project members to assess the status of the project and to seek for areas of improvement. The tool support team is responsible for the tools that are used for supporting the services. Tools for version control, change and defect tracking and visual modeling of requirements are supported by this part of the facility. The infrastructure team is responsible for offering the technical capabilities to make use of all services. Besides supporting computer hardware and being responsible for project specific domain control and back-ups, the infrastructure team configures and maintains virtual environments for project members to work in.

3.2 Data Collection

Data was primarily gathered by means of extracting data from the applications CA Clarity, Open Workbench, IBM ClearQuest and the log files of source lines of code (SLOC) counters. This data was triangulated by examining various other sources of electronic data. As a first check, project documentation stored in IBM ClearCase systems such as management summaries and memos were consulted. Incomplete or inconsistent data was later compared to the measurement rapports created by the Estimation and Measurement team of which backups are kept on the development department's own servers. These servers contain information on both current and past projects in which the development department's services were used. If ambiguities regarding project data still exist after consulting the prescribed administration systems, the informal project documentation and the measurement assessments, the project's process coach and project manager were consulted.

3.3 Visualizing Effort Data

Visual representations were made by automated interpretation of effort information that was entered by project members into Clarity. We created a custom view for the effort data so that the columns task type, task description, effort in person-hours, starting-date and ending-date were listed in that particular order. The ordering of the items was hierarchical so that a project consists of phases, phases consist of iterations, iterations consist of disciplines and disciplines consist of tasks. The data structure of the log files is depicted in a class diagram in figure 2.

These log files were analyzed by means of a script that counted the amount of time and effort that were spent on task-level. Then, both time and effort data were normalized and data points in the form of $x = (\frac{task\ effort}{discipline\ effort})$ and $y = (\frac{task\ time}{project\ time})$ were created for every task that was

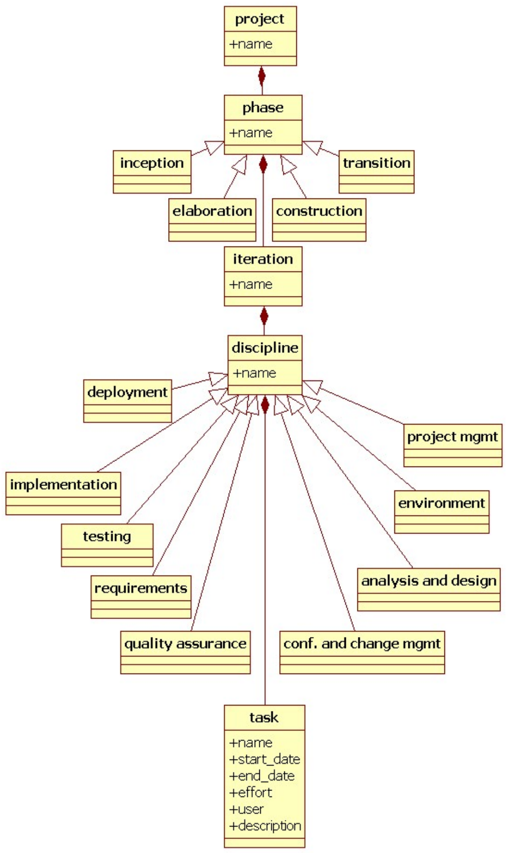

Figure 2. class diagram depicting the structure of the examined effort log files

executed within a discipline. The data points for each discipline were written to a separate file. A visualization tool was used to interpret the data files and to plot a diagram for each discipline.

3.4 Validation

After both projects were finalized, the process visualizations were shown to senior project members such as the project leader and the configuration manager. Also, the estimation and measurement team members were asked to elaborate on the images. Questions asked during these unstructured interviews include:

- To what extent can you recognize the development strategy in the image?
- What other factors influence the visualization?
- Can you explain the reason for the amount and length of the phases and iterations?
- Would you find it useful to see these images during a project?
- Why is there a certain anomaly or unusual effort peak depicted in a certain phase or iteration?

4 Results

For this study, two projects were analyzed. These two projects are similar in size, duration and have in common that they used the Microsoft .Net development environment and that they were led by the same project manager. Each project was executed for a different customer, in a different domain, under different circumstances and with different team members. Project A took place after project B but not immediately. Project management did not specifically address effort distribution between project A and project B. RUP was adhered to as strictly as possible as this is stimulated by the IT organization. Also, the project leader as well as the other team members already had experience with using RUP. The IT organization emphasizes cooperation with the customer and therefore primarily defines success in terms of customer satisfaction. In the standard post-mortem analysis customer interview process, on average, both projects scored over 4 on a scale of 1–5. Table 1 contains an overview of important project characteristics. The following subsection will describe the effort distribution visualization, elicit the striking phenomena that can be observed from these images and try to explain these occurrences for each of the two projects. Lastly, the explanations for each phenomenon given by involved project seniors will be described.

Table 1. Project descriptives

	project A	**project B**
APPLICATION TYPE	Webshop	Administration
DOMAIN	Education	Insurance
SIZE (SLOC)	80,000	62,000
SIZE (FUNCTION POINTS)	866	912
LEAD TIME (DAYS)	323	725
EFFORT (PERSON-HOURS)	8,941	11,492
PEAK STAFF (FTE)	7	12
SCHEDULE PRESSURE	yes	no
COST STRUCTURE	time–material	fixed price

4.1 Project A

Project A consisted of building a web-enabled content management system and business to business web shop. The customer was from the educational domain. The project employed 13–15 people with a peak of seven full–time equivalents during the construction phase. 866 Function points were realized in 8,941 person–hours and resulted in 80,000 source lines of code. During the execution of the project, the requirements changed and were expanded to great extent. At the start of the project, the project manager had 4.5 years of experience in managing IT projects. Project A was executed under a schedule pressure due to lime limitations. Project A used some agile practices such as daily stand–up meetings [3] and writing code with the responsible testers, the end users and the designers in the same room. The reason for applying these practices was the volatility of the requirements. The effort distribution visualization for project A is depicted in Fig. 3.

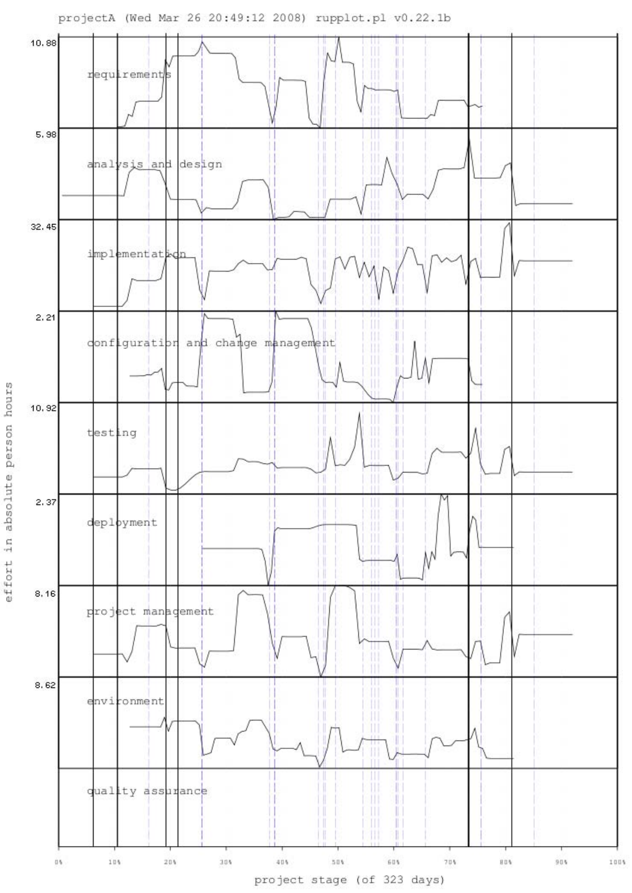

Figure 3. Effort distribution visualization for project A

In 3, nine disciplines used by the RUP are shown. The business modeling discipline is not used within the software development organization. The quality assurance discipline is a discipline that is used by the software development department to log hours spent on process quality. Neither of the projects described in this study made use of these hours. On the y–axis, the absolute effort in person hours for each discipline is depicted. The scaling is not uniform so that each peak fits well in the image and to prevent effort representations of disciplines on which traditionally less time is spent, to be depicted too small. The x–axis was scaled to fit the entire project span. The thicker continuous lines on the x–axis represent the phase delimiters and the thinner dotted lines represent the iteration delimiters.

In Fig. 3, many iterations can be identified of which the varying length and density in between 30% and 70% of project time are the most striking features. In the interviews the amount of iterations showed to be correct whereas the length of the iterations were not confirmed to be consistent with reality. The amount of iterations was relatively high because of the volatility of the requirements. The different iteration lengths are the result of the fact that CA Clarity was used for hour registration and that this registration served directly as the basis for the invoices for the customer. Because there were so many iterations and because there was a certain amount of schedule pressure during the construction phases, setting the exact dates for each iteration was not a top priority. The phases were confirmed to be correct.

In the requirements discipline, as in many other disciplines, sudden drops of effort can be seen, in this case just before 40% and 50% of project time. These drops could mostly be attributed to vacations, team training and illness of team members.

The fact that, for example, the analysis and design and implementation disciplines are still in a peak around 80%, makes the apparent ending of the project around that time seem abrupt. The project leader confirmed that, during development, the system was tested in the production environment as a result of problems with simulating the production environment. The effort spent in the last 15% of the project is not logged as a result of the schedule pressure.

When compared to the original RUP hump chart in Fig. 1, the visualization of effort distribution of project A shows distinct differences with regard to how analysis and design effort is spent. In the original RUP hump the analysis and design effort peaks early and peaks several times later, albeit somewhat lower, as the design is reworked in hypothetical consecutive iterations. In Fig. 3 we see that more effort is spent on analysis and design around 70% than in the initial stages of the project. Besides changes in requirements that have fundamental impact on the design of the application, this could indicate that the first construction iterations were based on a poor design which was reworked. The latter was the case: Rework in the design was caused by architectural decisions which were not confirmed. This rework is a pattern that can be clearly deduced from the visualization. The peak of implementation effort that can be seen around 80% is an effect of this architecture redesign.

An important remark during the interview with the project leader of project A was that according to him, the effort for the requirements, analysis and design and implementation disciplines could essentially be combined in one hump to more accurately represent the spending of person-hours. From this remark it could be deduced that a team member with the role of programmer who participated in a requirement specification workshop, recorded his or her working hours as effort spent on implementation. This finding implies that the RUP defined roles and disciplines were not used as separate entities from an effort registration perspective. The reason for this was the cost structure for project A which required every hour to be recorded according to price. In this cost structure it is more important to know the amount of hours that a certain team member worked because different team members have different prices.

4.2 Project B

During the execution of project B, a car insurance application was built. The final application had to interface with various, already existing databases. At the peak of the project, during the construction phase, 12 fte were working in the project simultaneously. During the transition phase, this amount was reduced to 0.5 fte. Before project execution, 912 function points were counted. At the time of application deployment, a total of 62,000 SLOCS were delivered. The project produced more lines of code than predicted due to customer induced limitations on the architecture, a complex application front-end which could not be expressed in function points and a range of customer change requests during the project which caused the functionality of the system to expand. Project B was a fixed price project.

Figure 4 displays a visualization of how effort was distributed over the RUP disciplines for project B. The overall project trend was recognizable for the project leader who was interviewed about the visualization results. The project started as a traditional RUP project but at an early stage, the customer thought the tempo of the project was too high. Consequently, staff was reduced. The effects of this decision is clearly visible in the visualization at around 17% as requirement, analysis and design and implementation effort dips.

A striking feature of Fig. 4 is the low amount of effort that is spent in the second half of the project. The effort spent in this last – transition – phase is spent by one person who works on the project half of his time. The long transition period was attributed to customer change requests (RFCs), infrastructure problems at the customer deployment site and dependence on other projects which were executed by different organizations at other locations. In the transition phase most effort is attributed to the implementation stage. This, however, is not correct as the 0.5

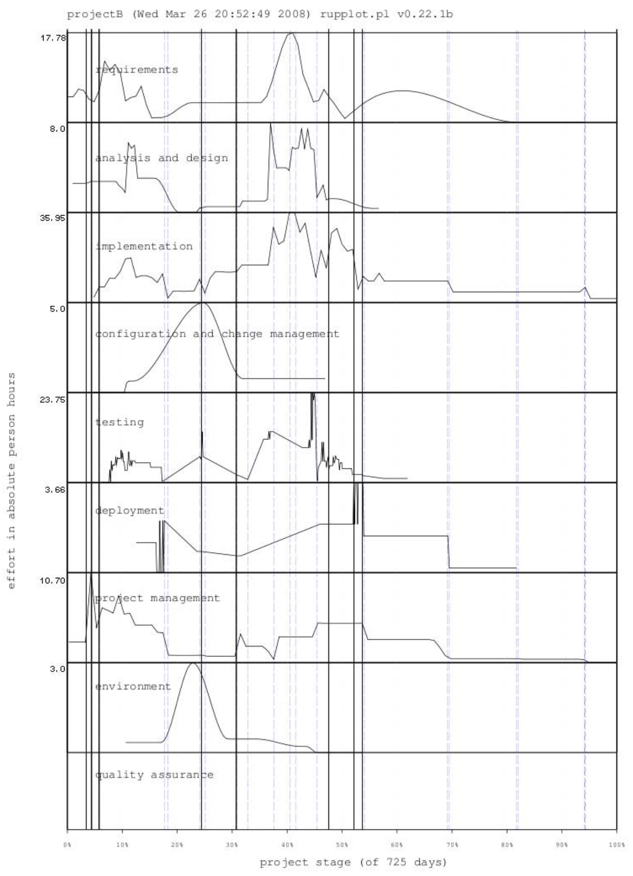

Figure 4. Effort distribution visualization for project B

fte assigned to the project was responsible for multiple disciplines. Although time was spent on requirements, analysis and design, implementation, testing and deployment, this person choose to attribute all effort spent to the implementation discipline, due to time constraints. This portrays the central role that the implementation discipline plays and how this discipline is used as a default for effort logging under time pressure.

Figure 4 displays a large amount of phases an iterations. Not all phases depict real phases. Instead, some phases were used for registering hours for impact analysis and changes. The iterations were all recognized by the project leader. For example, during the construction phase, which is found roughly between 31% and 48%, six iterations were executed in which six sets of use-cases were implemented.

5 Threats to Validity

Correctness of hour registration data is an issue as outlined by the iteration mismatches and the sudden end of the effort distribution data in project A. In both cases schedule pressure caused the project to be less inclined to keep an hour registration. Remarks of project team members include that the distinction between requirements and analysis and design disciplines is not always clear when logging effort data. In the case of project A, the testing discipline is also confused with requirements and analysis and design at some occasions. Project B registered extra phases for change management and changes for administrative purposes. Also in project B, the implementation discipline was used to attribute effort to that in reality was spent on other disciplines. This merge was due to time constraints. This mismatch poses a possible treat to validity. Because of these mismatches, the data in the hour registration system can not always be used as-is and should be validated before it can be used for process analysis.

6 Discussion

The visualizations gave an insightful impression of spending of person-hours. Certain patterns were clearly visible. An example of such a pattern is the pattern seen in project A: The rework that had to be done on a poor design and had clear implications on various disciplines later in the project. This is a clear and understandable example of underspending resources during the design phase which force the project to spend extra resources on the design in a later stage of the project. Contrastingly, if too large an amount of effort would have been spent on the design, the project would have been forced to spend less time on subsequent disciplines to prevent project overrun. Therefore, finding a balance between the amount of resources to be spent on disciplines prevents that the resource allocation is dictated by shortages.

Both effort distribution visualizations give a good indication of how effort was actually spent. However, RUP's flexibility led to differences in how effort was recorded. From the feedback during the interviews it became apparent that formal arrangements regarding expenditure, such as cost–structure, influence effort registration. This problem should be accounted for in future exploration and analysis of effort registration data. Using separate applications for logging effort data for analysis and for billing purposes can help to increase data comparability. However, effort registration is often not a priority in commercial software development. The objectives of scientific analysis of a software engineering project and the objectives of the project itself are conflicting. The prime objectives of a software project are to deliver relevant and functional software in a timely manner. Contrastingly, the benefits of scientific research, such as in this case, quantitative post–mortem project analysis, are not directly relevant to the customer paying for the effort spent. Also, such analysis does not guarantee results and if it does, those results may be difficult to operationalize on the short term and so they constitute a long–term investment. Logging effort distribution poses other problems such as the challenge of defining what type of effort should be logged or the possibility that team members may see detailed logging of their activities as intrusive and a threat to their privacy.

7 Conclusion

The visualizations of how effort was distributed over RUP disciplines were seen as useful in the sense that they can play a role in verifying to which extent resources should have been spent. As one project leader put it: *"The [RUP hump] image should not yield any surprises [at any given time during project execution]."* The visualizations are mainly dependent on a few factors such as the type of project in terms of cost or billing structure and the definitions of RUP disciplines used. In the organization described in the research environment, the visualizations are to be used as a standard extension to the tools used for post mortem project analysis.

More data is needed in order to categorize software development processes. Collecting data that was recorded in a uniform manner can help us to determine patterns of effort distribution and to relate these patterns to various project specific success or failure related factors. Comparing average RUP humps for organizations can give insights in typical decisions taken in terms of project management style or the implicit organizational attitudes with regard to the software engineering process and to what extent these have a structural impact on project results.

References

[1] M. T. Baldassarre, N. Boffoli, D. Caivano, and G. Visaggio. Speed: Software project effort evaluator based on dynamic-calibration. *22nd IEEE International Conference on Software Maintenance*, 0:272–273, 2006.

[2] V. R. Basili and D. M. Weiss. A methodology for collecting valid software engineering data. *IEEE Transactions on Software Engineering*, 10(6):728–738, 1984.

[3] M. Beedle, M. Devos, Y. Sharon, K. Schwaber, and J. Sutherland. SCRUM: A pattern language for hyperproductive software development. In N. Harrison, B. Foote, and H. Rohnert, editors, *Pattern Languages of Program Design 4*, pages 637–652. Addison Wesley, 2000.

[4] B. Boehm. A spiral model of software development and enhancement. *SIGSOFT Softw. Eng. Notes*, 11(4):14–24, 1986.

[5] B. Boehm, M. Abi-Antoun, A. Brown, N. Mehta, and D. Port. Guidelines for the life cycle objectives (lco) and the life cycle architecture (lca) deliverables for model-based architecting and software engineering (mbase)– usc technical report usc-cse-98-519. Technical report, University of Southern California, Los Angeles, CA, 90089, February 1999.

[6] B. W. Boehm. A spiral model of software development and enhancement. *Computer*, 21(5):61–72, 1988.

[7] G. Booch. *Object solutions: managing the object-oriented project*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 1995.

[8] M. R. Braz and S. R. Vergilio. Software effort estimation based on use cases. *30th Annual International Computer Software and Applications Conference*, 1:221–228, 2006.

[9] C. Ghezzi, M. Jazayeri, and D. Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2002.

[10] W. Heijstek and M. R. V. Chaudron. Effort distribution in model-based development. In *Proceedings of the 2nd workshop on Model Size Metrics (Co-located with MODELS 2007) Nashville, TN, USA*, pages 26–38, October 2007.

[11] W. Heijstek and M. R. V. Chaudron. On early investments in software development: A relation between effort distribution and defects in RUP projects. *to appear/ under review*, 2008.

[12] K. Iwata, T. Nakashima, Y. Anan, and N. Ishii. Improving accuracy of multiple regression analysis for effort prediction model. *1st IEEE/ACIS International Workshop on Component-Based Software Engineering,Software Architecture and Reuse*, 1:48–55, 2006.

[13] I. Jacobson, G. Booch, and J. Rumbaugh. *The unified software development process*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[14] P. Kruchten. A brief history of the rup®'s "hump chart". Technical report, University of British Columbia, 2003.

[15] P. Kruchten. *The Rational Unified Process: An Introduction*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003.

[16] C. Lopez-Martin, C. Yanez-Marquez, and A. Gutierrez-Tornes. A fuzzy logic model based upon reused and new & changed code for software development effort estimation at personal level. *15th International Conference on Computing*, 0:298–303, 2006.

[17] T. Menzies, Z. Chen, J. Hihn, and K. Lum. Selecting best practices for effort estimation. *IEEE Transactions on Software Engineering*, 32(11):883–895, 2006.

[18] D. Milicic and C. Wohlin. Distribution patterns of effort estimations. *30th EUROMICRO Conference*, 0:422–429, 2004.

[19] D. Port, Z. Chen, and P. Kruchten. An empirical validation of the rup "hump" diagram. In *ISESE '05: Proceedings of the 4th International Symposium on Empirical Software Engineering*, 2005.

[20] W. Royce. *Software project management: a unified framework*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1998.