Chapter 3

Global Architecture and Design Process Evaluation Through Effort Visualization

The objective of this chapter is to evaluate how resources (person-hours) are allocated in global software development projects and co-located projects. To this end, patterns in process resource allocation in general and in architecture and design processes in particular, are analyzed by means of effort distribution visualization. We collected data from four large-scale industrial software development projects. Data is obtained from various sources within these projects.

This chapter is based on the following publications:

- Werner Heijstek and Michel R. V. Chaudron (2007) **Effort distribution in model-based development.** *In Proceedings of the 2nd Workshop on Model Size Metrics (MSM 2007) pages 26–38, Nashville, Tennessee, USA*
- Werner Heijstek and Michel R. V. Chaudron (2008) **Evaluating RUP Software Development Processes Through Visualization of Effort Distribution.** *In Proceedings of the 34th Euromicro Conference on Software Engineering and Advanced Applications (SEAA 2008) pages 266–273, Parma, Italy*
- Werner Heijstek and Michel R. V. Chaudron (2008) **Exploring Effort Distribution in RUP Projects.** *In Proceedings of the 2nd International Symposium on Software Engineering and Measurement (ESEM 2008) page 359, Kaiserslautern, Germany*

3.1 Introduction and Objectives

Software architecture and design are both processes and artifacts. As processes, software architecture and design influence one another. If perceived from a chronological stance, a first version of a software architecture is often designed before a more detailed design specification is made. However, such a design specification might lead to insights that in turn influence a newer instance of the software architecture.

While these process interactions are unique for each software project, one may expect to find patterns in process interactions in software projects. These patterns may be expected to be similar for custom software development projects.

The developers of the RUP (Kruchten, 2003b) defined a set of processes and created visualizations of the effort distribution between these processes in a diagram that would later be referred to as the "RUP Hump Chart" (Figure 1.3). In this chapter, we will use a similar visualization to investigate how resource allocation for distributed software development differs from co-located software development. Specifically, we will address **RQ1** (Section 1.3). This exploratory research question aims (in part) to uncover how software architecture is coordinated in the context of global software development. We will therefore specifically analyze the role of the "analysis and design" discipline in these visualizations.

Research regarding distribution of effort in software processes is commonly found in literature on software estimation and planning: (Milicic and Wohlin, 2004, Iwata et al., 2006, Baldassarre et al., 2006, Menzies et al., 2006). 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 (Lopez-Martin et al., 2006) or use-case based requirement specifications (Braz and Vergilio, 2006). What an effective distribution of effort over disciplines is, remains unaddressed in literature. Important reasons are a lack of data on software process in general and problems with regards to comparability of data in particular. 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 analyzing 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 better resource allocation — which implies cost reduction.

Second, analysis of effort distribution is necessary in order to develop a method for project management to gain insight in resource allocation. The effort perspective provides an objective overview of what is happening (or has happened) in a software

development project. Team members might work on several projects at the same time. By focusing on the hours spent on specific tasks in a particular project, the dynamics of the tasks that are executed can be better understood in isolation than by using observation. In addition, observations from visualization of effort data during a project elicit trends and provide a view of the progress of a 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 RUP that depicts the effort that would be spent during a project on each of the nine disciplines RUP prescribes. Port et al. (2005) attempted to validate the RUP hump diagram earlier by means of student experiments. They conclude that the visualization of their data 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 Port et al. has been followed up before (Heijstek and Chaudron, 2007) albeit on an aggregate level. This study examines individual projects.

Another study in which RUP humps are redrawn based on a project data has been executed by Hindle et al. (2010). In their study, a variety of existing artifacts is used to draw RUP Humps of two major open source projects. Hindle et al. concluded that for both projects, the humps *"allowed [them] to find interesting requirements- and analysis- related behaviors"*. In particular, they found that they could uncover important events — such as major component re-designs — that would not have shown up in common project metric overviews. The visualizations made by Hindle et al. focused on product software (e.g. they used 14 and 9-year timescales) whereas in this work, we focus on greenfield[1] project-based software development. Timescales in this domain are measured in months.

The structure of this chapter is as follows: The following section (3.2) will elaborate on related work regarding RUP "humps". Section 3.3 explains the research method and Section 3.4 outlines the results. Section 3.5 explains the threats to validity, Section 3.6 contains a discussion of the findings. Finally, Section 3.7 contains our conclusions and future work.

3.2 Related Work

In this section "RUP humps" and related studies are discussed.

3.2.1 RUP Humps

The term RUP "hump" refers to a graph of effort spent over time during a particular discipline. The RUP hump chart consists of a collection of humps for all RUP dis-

[1] also referred to as bespoke software development: the development of software "from scratch" as opposed to e.g. product software development, based on prior releases

ciplines. This diagram was created in 1993 during a workshop on architecture and process (Kruchten, 2003a) and was inspired upon work by Booch (1995b) and Boehm (1986, 1988). 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 (Kruchten, 2003a). Its final form was published by Kruchten in 1998 (Kruchten, 2003b). An older version was later used by Jacobson et al. (1999) and an altered version was used by Royce (1998). A recent version of the RUP chart is depicted in Figure 1.3. Over the years this diagram has become increasingly connected with RUP in such a manner that it is sometimes perceived as a logo for the process. IBM refers to the RUP Humps as the *"widely recognized RUP Lifecycle Diagram"* (O'Neill, 2007). 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 than 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 than 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..."* (Kruchten, 2003a)

3.2.2 Other Related Work

Port et al. (2005) 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 (Boehm et al., 1999). In their research, Port et al. 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 the study in this chapter is that their effort was already reported in terms of RUP disciplines. An effort mapping was therefore not necessary.

Hindle et al. (2010) report on a study in which they employ visualizations that are consistent with the RUP hump chart. Their objective is software process recovery. Hindle et al. used data from mailing-list archives, version control systems and bug-tracker systems to draw what they refer to as Recovered Unified Process Views (RUPVs). In their study, they present two cases. First, they draw RUP Humps of the development

process of the open source operating system "FreeBSD"[2] over a period of 16 years. Second, they reconstruct the development process of SQLite[3] over a period of 10 years. Lacking precise effort data, they use techniques such as word-bags and topic analysis to reconstruct discipline activity. Hindle et al. concluded that the humps *"allowed [them] to find interesting requirements- and analysis- related behaviors* and that they *"were able to find important events that would not have shown up in a commits per month signal."* Hindle et al. note that their humps are not only useful for project managers, who can use the visualizations in their dashboards, but also for (new) developers, unfamiliar with the project culture or consultants or investors want to gain an overview of a project's processes. For this study, we specifically use large industrial software development projects as opposed to open source projects concerned with product software development.

3.3 Methods

In this section, the methods used for our study, are outlined. Detailed hour registration data was collected from the software development department of a large IT service provider (organization ABC from Chapter 2). 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.

3.3.1 Project Context

Data is collected from four industrial projects developed by a single software organization. Within this organization, specific departments offer services to software projects. These services include estimation and measurement, "assembly line" support (e.g. development environment configuration), process coaching, tool support and infrastructure support. The estimation and measurement department is responsible for quantitative analysis of projects before, during and after execution. The assembly line department offers "continuous integration" services with regard to software development. Process coaches are responsible for providing help and training to project department 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 department, 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 department is responsible for the tools that are used for supporting the services. Tools for version control, change and defect tracking and modeling of requirements and

[2] http://www.freebsd.org/
[3] http://www.sqlite.org

design are supported by this part of the facility. The infrastructure department is responsible for offering the technical capabilities to make use of all services. Besides supporting computer hardware and being responsible for project hardware and back-ups, the infrastructure department configures and maintains virtual environments for project members to work in. The business modeling discipline is not used within the software development organization as this part of projects is done by a different organization.

3.3.2 Data Collection

Data was primarily gathered by means of extracting data from the applications CA Clarity[4] (an hour registration system), Open Workbench[5] (a front-end to Clarity), IBM ClearQuest[6] (a defect tracking system) and the log files of source lines of code (SLOC) counters. These data were triangulated by examining various other sources of electronic data: As a first check, project documentation stored in the software configuration and change management system (SCCMS, IBM ClearCase[7]) systems such as management summaries and memos were consulted. Incomplete or inconsistent data was later compared to the measurement reports created by the Estimation and Measurement department 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.

3.3.3 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 and task effort for each week of the project 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 3.1. These log files were analyzed by means of a set of GNU Bash[8] and Python[9] scripts that counted the amount of time and effort that were spent on the

[4] http://www.ca.com/us/project-portfolio-management.aspx
[5] http://sourceforge.net/projects/openworkbench/
[6] https://www-01.ibm.com/software/awdtools/clearquest/
[7] https://www-01.ibm.com/software/awdtools/clearcase/
[8] https://www.gnu.org/s/bash/
[9] http://python.org/

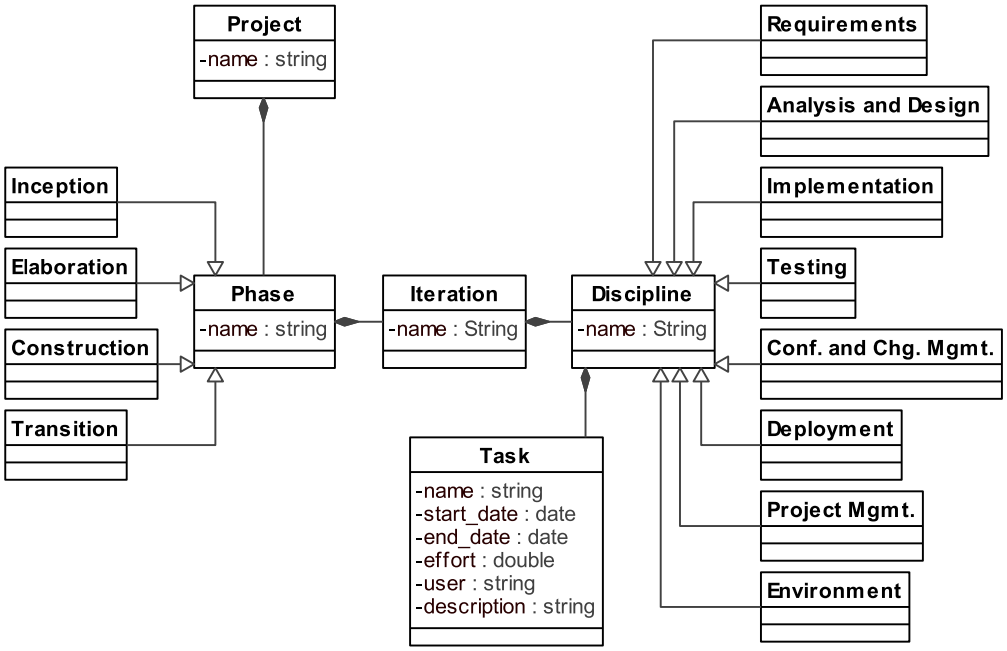

Figure 3.1: *Class diagram depicting the structure of the examined effort log files*

task-level. Then, both time and effort data were normalized and data points in the form of

$$x = \left(\frac{\text{task effort}}{\text{discipline effort}}\right) \quad \text{and} \quad y = \left(\frac{\text{task time}}{\text{project time}}\right) \tag{3.1}$$

The data points for each discipline were then visualized by means of R (R Development Core Team, 2011) package `ggplot2` (Wickham, 2009).

3.3.4 Validation

After the projects were finalized, the process visualizations were validated with senior project members such as the project leader and the configuration manager. Also, the estimation and measurement department members were asked to elaborate on the humps. 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?

3.4 Results

For this study, four projects were analyzed. All projects were executed by the same IT organization but for different clients, in different domains, under different circumstances and with different team members. The projects took place over a period of 7 years. RUP was adhered to as strictly as possible as this is stimulated by the IT organization in general and by the process coaches, discussed earlier, in particular. Also, the project leader as well as the other team members already had experience with using RUP. The IT organization emphasizes cooperation with the client and therefore primarily defines success in terms of client satisfaction. In the standard post-mortem analysis client interview process, on average, all projects scored over 4 on a scale of 1 to 5. Table 3.1 contains an overview of relevant project characteristics. The following subsections will describe the effort distribution visualizations, elicit the striking phenomena that can be observed from these images and try to explain these occurrences for each of the projects. Lastly, the explanations for each phenomenon given by involved project seniors will be described.

Table 3.1: *Project Characteristics*

	project A	**project B**	**project C**	**project D**
application type	webshop	administration	administration	search
domain	education	insurance	financial	government
dev. language	.Net	.Net	Java	.Net
func. points	866	912	2,000	600
person-hours	8,941	11,492	53,837	14,536
peak staff (FTE)[1]	7	12	28	13
sched. pressure	yes	no	no	yes
cost structure	time–material	fixed price	fixed price	fixed price
offshore	no	no	yes	yes

[1] Full-Time Equivalent

3.4.1 Project A

Project A consisted of building a web-enabled content management system and business-to-business web shop. The client was from the educational domain. The project employed 13 to 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 a 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 schedule pressure due to time limitations. Project A used some agile practices such as daily stand–up meetings 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 Figure 3.2. The horizontal axis was scaled to fit the entire

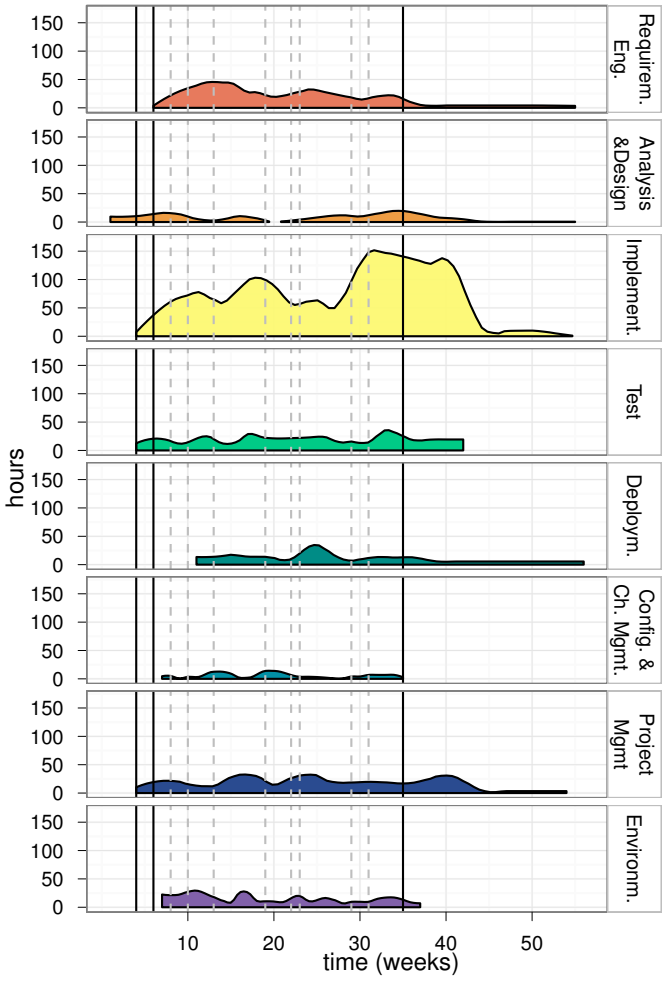

Figure 3.2: *Effort distribution visualization for project A*

project span. The black vertical lines on the vertical axis represent the phase delimiters and the gray, dotted lines represent the iteration delimiters.

In Figure 3.2, many iterations can be identified of which the varying length and the long third (construction) phase, are the most striking features. In the interviews, the amount of iterations showed to be correct whereas the length of the iteration was not always correct. The amount of iterations was said to be relatively high because of the volatility of the requirements. The different iteration lengths are the result of the fact that the effort logging database was used for hour registration and that this registration served directly as the basis for invoices for the client. 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 interviews with project leaders, it emerged that explanations for the resource allocation as represented by the RUP humps were sometimes not directly related to software-related events. For example: in all disciplines, sudden drops of effort can be seen. The drop in all disciplines around 30 weeks can be attributed to a national holiday. Other drops were associated with team training or illness of team members.

The fact that, for example, the analysis and design and implementation disciplines are still in a peak around 40 weeks, 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 10 weeks of the project is not logged as a result of the schedule pressure.

When compared to the original RUP hump chart in Figure 1.3, 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 Figure 3.2 we see that more effort is spent on analysis and design around week 35 (the beginning of the transition phase) 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 later. 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 week 40 is a direct effect of this architecture redesign.

An important remark during the interview with the project leader of project A was that 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. 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 always strictly 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 rates associated with them.

3.4.2 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 client–induced limitations on the architecture, a complex application front-end which could not be expressed in function points and a range of client change requests during the project which caused the functionality of the system to expand. Project B was a fixed price project. Figure 3.3 displays a visualization of how effort was distributed over the RUP disciplines for project B. As was the case for project A, all disciplines but the business analysis discipline were used.

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 client 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 week 15 as requirement, analysis and design and implementation effort dips.

A striking feature of Figure 3.3 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 50 percent FTE. The long transition period was attributed to client change requests (RFCs), infrastructure problems at the client 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 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 3.3 displays a large amount of phases and 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, six iterations were executed in which six sets of use-cases were implemented.

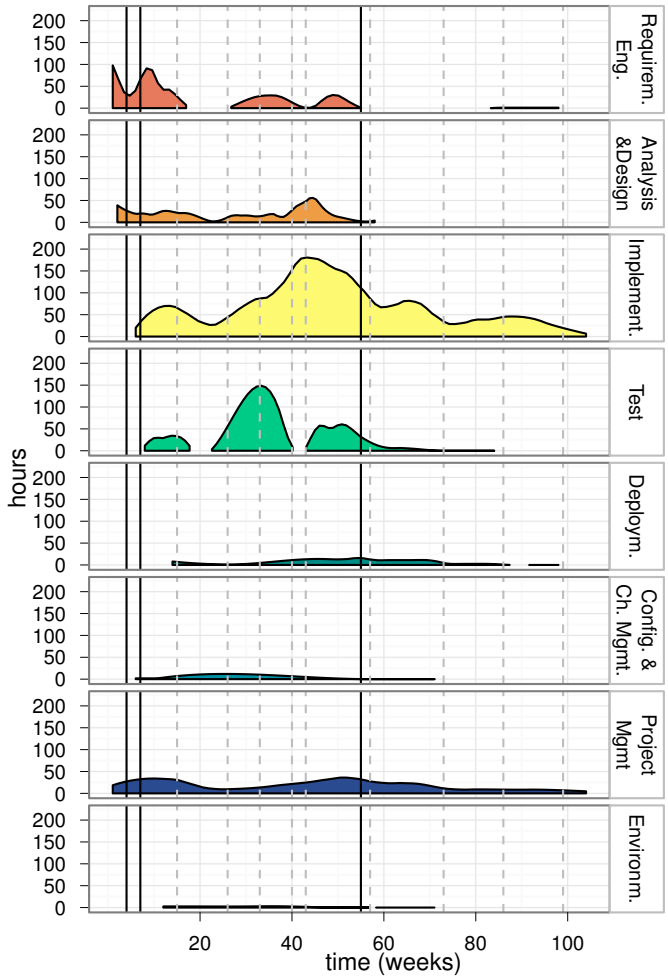

Figure 3.3: *Effort distribution visualization for project B*

3.4.3 Project C

The objective of project C was to develop a web-enabled registration system for various types of financial products. The client was a large, international financial organization. Project C employed model-driven development techniques and was partly executed offshore. During the two years the project ran, it had various difficulties both due to the complexity and novelty of the model-driven development tools and techniques and due to problems associated with offshore development. The requirements for this project were not particularly volatile. However, the software architecture was not very stable and difficulties existed with communicating the architecture to the offshore development location. The effort distribution visualization for project C is depicted in

Figure 3.4. In this visualization, only six disciplines used by the project team are shown.

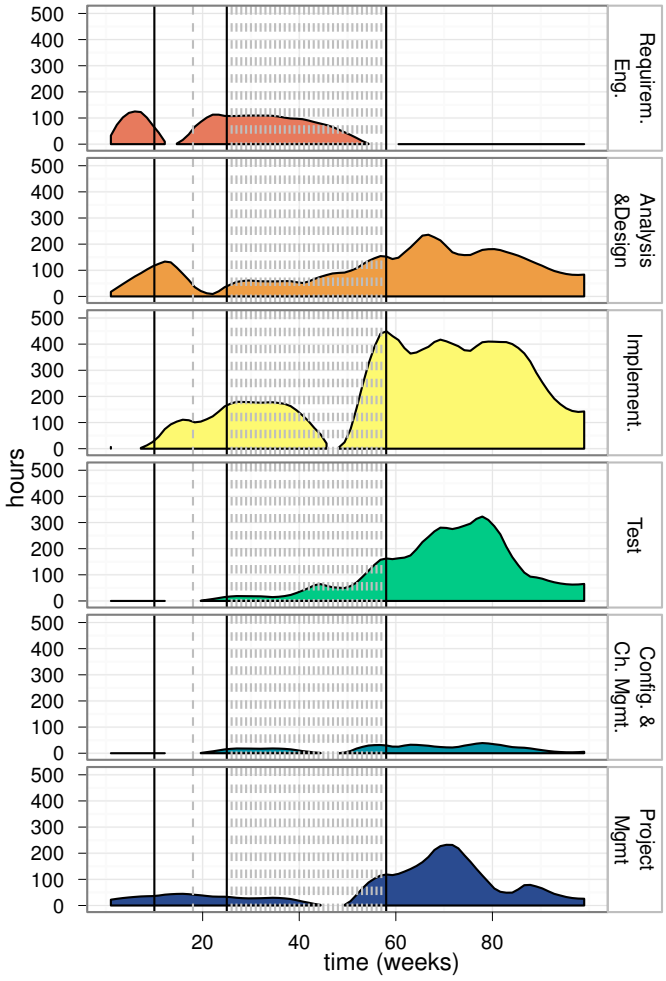

Figure 3.4: *Effort distribution visualization for project C*

The business modeling, deployment and environment disciplines are not used. The developed system was never deployed at the client site and the environment discipline was not seen as necessary.

The most striking feature in the RUP humps for project C are the 33 iterations that comprise the third (construction) phase. While RUP was used as a development method, during the construction phase, a Kanban approach (Ohno, 1988, Poppendieck and Poppendieck, 2003) was used. As a result, iterations were weekly. In the second (elaboration) phase, the only other iteration line can be seen. The Kanban approach also explains why requirements engineering effort is almost as high as it is in the first

two phases, during most of the construction phase.

The effort distribution of project C, is further removed from the 'reference humps' than Projects A and B were. The reasons for this seem to be both the model-driven nature of this project and the fact that an offshore development team (GSD) was employed. Three reasons lead to this observation:

First, the code was generated from models. The effort logged as analysis and design can therefore be regarded as implementation effort. The effort spent on implementation should be split between effort spent on the code for the project and the code of the code generator. The effort data was not detailed enough to allow to deduce this distinction in the data.

Second, most effort analysis and design and implementation effort seems to be spent during the last (transition) phase. Part of these phenomena can be explained by the fact that the model-driven development paradigm was new for the development team. However, project management and team members mostly explain these late effort spikes as having to do with issues related to the difficulties of collaborating the with offshore development team. The project ran over time and over budget to a serious extent and had to ramp up development effort during the transition phase. Consequently, technically, the transition phase was another construction phase. The project management effort spikes in the last phase are explained to be a direct result of solving communication problems with the offshore team. The complexities of the meta-model used for code generation, were the reason that a only very few knowledgeable experts in the (onshore) development team were available. As they were all senior developers, they constituted costly resources that the contractor preferred not to be consulted too extensively. This inhibited offshore developers from attaining an understanding at a similar expert level. As a consequence, elaborate onshore guidance was needed throughout the project. Onshore support became even more vital when development was ramped up in the later phases of the project as more questions arose. The relatively high test effort, overall, was explained to be a result of examining consistency of implementation with the meta-model, or architecture compliance checking.

3.4.4 Project D

The objective of project D was to build a document retrieval system that was to be used by various governmental organizations. The employed development methodology was RUP. The project employed transfer-by-development stage offshoring (Mockus and Weiss, 2001) where different development stages are executed at different locations, sequentially. As a result, the inception and elaboration phases were executed by a small onshore team and implementation was executed by an offshore team of developers. The effort distribution visualization for project D is depicted in Figure 3.5. Disciplines missing from the effort registration of Project D were business analysis, environment and configuration and change management. Environment and configuration and

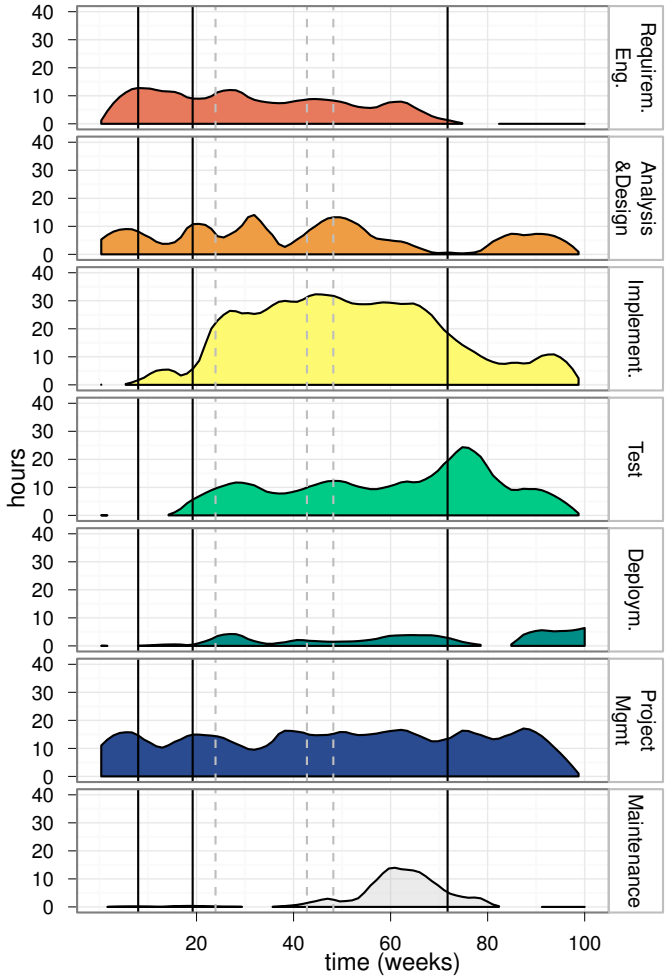

Figure 3.5: *Effort distribution visualization for project D*

change management effort were said to be merged with project management effort. The maintenance discipline is not an official RUP discipline. It was used for this project as the contractor was also set to maintain the system they were building. Therefore, it was seen as important to implement the system ensuring maintainability. Activities to that end were separately logged as if they were part of a maintenance discipline. Most effort spent on this discipline is implementation and analysis and design effort.

A striking feature is that the iterations in the third phase have an unequal length. This is the result of a feature-oriented iteration strategy. Not all features were of equal size and as a result, iteration length differed. In this project, a substantial amount of effort was required to ensure that the offshore development team complied with

the architecture made onshore. This impact of the use of an offshore development team is visible in various parts of the hump image. First, the analysis and design discipline peaks in the fourth (transition) phase. It also runs until the end of the project. These features are caused by the effort that was spent communicating and altering the architecture design. Second, the test discipline hump is relatively large. The explanation for this phenomenon is that a significant amount of time was spent checking architecture design compliance. Third, even when accounting for the environment and configuration and change management effort being merged with project management effort, this discipline has an unusually large hump associated with it. Various changes in the offshore team composition required project management effort to increase. On the one hand, new team members needed to be made familiar with the project and the system's architecture. On the other hand, due to these team composition changes, the project became delayed and had to be re-planned.

3.5 Threats to Validity

Correctness of hour registration data directly influences data visualization. For example, the iteration mismatches and the sudden end of the effort distribution data that can be seen in project A is an artifact of the hour registration rather than a process change. These effects were uncovered by validating the visualizations with project leaders. Remarks of project team members include that the distinction between requirements and analysis and design disciplines is not always clear when logging effort data. We manually reclassified task descriptions and moved tasks between these two disciplines when necessary. In the case of project A, the testing discipline is also confused with requirements and analysis and design at some occasions. We again reclassified tasks based on their description when needed and validated this process with at least one project team member. Project B registered extra phases for change management (CM) 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 the result of 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 they are and should be reclassified and validated before they can be used for process analysis.

3.6 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 under-spending resources during the design phase which force a 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. Another pattern that could be observed was the impact of distributed teams in projects C and D. Observed influences of employing offshore development teams on effort distribution were increased analysis and design for communication of architecture and increased test effort for increased testing of architecture compliance.

RUP's distinction between engineering and supporting disciplines is not a good predictor of which disciplines are thought to be most important to log. The disciplines that are most often missing from an effort registration system are business modeling, deployment and environment — of which the latter two are defined as engineering disciplines. The business analysis discipline was missing from all four of the cases under study because this particular task is not part of the expertise of this department. When projects run late, project management effort increased because of the required rescheduling of the work that remains to be done. Also, the client needs to be managed more intensively. This change is either logged in detail or it leads to a situation in which little to no effort is logged because of schedule pressure. Non-standard development approaches have significant impact on the shapes of the humps. An example is the model-driven development approach taken in project C. In this project, modeling is synonymous with implementing. The definitions of the analysis and design and implementation disciplines are therefore less clear.

According to the project leaders, all four effort distribution visualizations give an accurate indication of how effort was actually spent globally. 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 client. 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.

3.7 Conclusions and Future Work

In this chapter, we followed up on a method for visualization of software development process effort and adapted it to provide a view on how resources are allocated in large-scale, custom software development projects. Both distributed and co-located projects were used. Evidence was found for aberrant distribution of analysis and design effort in projects in which offshore development teams are employed. These aberrations are related to unclarities related to communication and coordination of software architecture.

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 what 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 in which cases A, B, C and D were executed, 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 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.

The RUP hump plots can be extended to include a cumulative effort plot per discipline and plots of the cumulative number of source line of code, defects found and functionality realized over time measured in, for example, function points or use case points. The plots can then be used during project execution to analyze the project status and they can also become a part of a standard project post-mortem.