An Approach for Project Task Approximation in a Large-Scale Software Project Course

Birgit Demuth
Technische Universität Dresden
Email: birgit.demuth@tu-dresden.de

Marc Kandler
Technische Universität Dresden
Email: marc.kandler@tu-dresden.de

***Abstract*—Teaching software development to undergraduate students is a challenging task. One of the challenges is to secure a high education quality for large numbers of students. Different projects should be easily comparable while allowing for different tasks for different teams to reduce the risk of plagiarism. Our solution is to use an application framework. The student project teams' final applications attained varying degrees of complexity, which led to complaints from students. An approach to approximate the size of project tasks is to model project tasks by a feature model. The features in this model are weighted with function points. Then, each project task can be refined as a product configuration derived from the feature model with a predefined total number of function points. We present the approach at the example of the SalesPoint framework and report on the results of its application in our software project courses.**

1. Introduction

Teaching object-oriented software development to undergraduate students is a challenging task for several reasons. Students need to develop a lot of complex skills. These include technical skills in object-oriented software development, but also social skills. To acquire these skills, students need development experiences, mostly through team-oriented project courses. Designing such project courses is a challenge in itself. In particular, we set ourselves high standards of education quality for large numbers of students supervised by small numbers of staff to secure. Therefore, an important requirement is scalability: Different projects should be easily comparable while allowing for different tasks for different teams to reduce the risk of plagiarism. The solution that in our experience satisfies these requirements is to use an application framework for an everyday application domain. In our special case, we have been used SalesPoint for web-based point-in-sale applications [1] in several versions since 1997[1]. Each project team is supported in its software development process by a student tutor. The team gets a one-page project task for a point-of-sale application and negotiates details with the customer (represented by the tutor) during the analysis phase. At the end, it depends on the tutor how extensive the detailed project requirements will be. The experience showed that the final applications of the student project teams were of varying degrees of complexity, which led to complaints from students.

The idea to approximate the size of project tasks was to consider the family of SalesPoint applications as a Software Product Line (SPL) [2]. We modeled the SalesPoint SPL in terms of features to create a feature model [3]. Then, we consider a derived product configuration of this feature model as requirement specification for a specific project task. The intention is to approximate the needed effort (or cost) in the software project course for the implementation of the applications by the students relating to (1) the average effort measured in hours and (2) comparable effort for each project task. We started with the analysis of existing SalesPoint applications developed in former software project courses and derived the feature model by feature extraction. Feature extraction starts from an initial set of data, identifies commonalities and differences of existing products based on domain knowledge, and builds derived features [2]. The next step was to weight the features in the feature model with Function Points (FPs) [4] as the basis for the cost estimation of a single project task. We developed a tool which guides the tutor to derive a product configuration with features he or she wants to have in his role as a customer in the SalesPoint application to be implemented. This product configuration should be of a nearly predefined number of Function Points to achieve project requirements of comparable size for all student projects. The tutor uses his product configuration of the feature model to fulfill his customer role in discussions with the students during the analysis phase of the project. Note that it is not the intention that the students work with the product configuration. They should by themselves elucidate the requirements for their special point-of-sale application.

In the past two years, we applied this approach and tool to approximate the project tasks in our annual soft-

1. www.salespoint-framework.org

2377-570X/17 $31.00 © 2017 IEEE
DOI 10.1109/CSEET.2017.21

ware project course. In this paper, we briefly present our approach, and report on our experiences. Our research questions are the following:

- RQ1 Is the approach of a Weighted Feature Model-based derivation of related project task appropriate to approximate the project tasks in a large-scale software project course relating to (1) a mean effort measured in hours and (2) a comparable effort for each project task.
- RQ2 Do the students still complain about different requirements?
- RQ3 How do the tutors evaluate the approach?
  - RQ3.1 Do the tutors evaluate the approach as appropriate to achieve the goal of RQ1?
  - RQ3.2 Do the tutors evaluate the tool as useful for their role as a customer?

One might argue that students should not be shielded from real-world challenges such as real customers and real projects, or the confrontation with team problems, unforeseen technical and project management issues. While true to a certain extent, our students have to face theses issues, especially with regard to teamwork. Regarding the question of real customers, we have, for many years, implemented a two-track strategy. Besides our SalesPoint based project (academic) approach, we offer real projects in the industry to interested students who are then usually more motivated for their software development activities than in our academic projects [8]. In [9], we reported that our academic approach led, in most cases, to a higher program quality compared to real projects with about the same time resources used. The challenge is to balance the trade-off between students' request for real projects and a strong guidance in academic software development. In a recent empirical study on the evaluation of the effects of experience on code quality and programmer productivity, it was reported that programming experience in academia has a positive influence on programmer performance compared to programming experience gained in the industry which has neither has an effect on quality nor productivity [10]. This confirms our long-term observations in teaching, which is why we prefer academic projects for undergraduate students.

The rest of the paper is organized as follows: In the next section, we summarize related work. Our research approach is presented in Section 3. The design of our feature model and an extract of it is given in Section 4. Then we report on experience with the application of our approach (Section 5). Section 6 presents the results of our case study and evaluates them against our research questions. Threats to validity are listed in Section 7. Finally, we summarize our research and discuss future work.

2. Related Work

Running student software projects are often thematized in education conferences because a software project course comes with many issues regarding organization and supervision. However, large-scale student software project courses are not so often subject of research.

In [5], a framework is proposed how to set up a large-scale student collaboration project, and how to be aware of problematic issues. This framework considers four features as a basis for developing and focusing studies of large-scale student collaboration projects as well as ideas for how to analyze the results of such studies. One of the features is the mechanism for work allocation among the members of a project. The *work* refers to the tasks necessary to fulfill the main project task, and not how the workload is allocated fair among multiple project teams.

General reflections on large-scale student assessment in academic education are written in [6]: what do teachers need to know about valid assessment practices. In our software project course, the student gets the certificate *passed* (successfully finished project) or *failed*. The challenge for the teachers in our case is to fairly allocate the workload.

Ceddia and Dick [7] investigate the estimation of the size of student projects using modified function points because of the huge variability of projects and clients in their Bachelor of Computing program. The mean of the number of FPs in 63 projects is 261 with a standard deviation of 130 points. Such a high standard deviation is exactly what we want to avoid.

To the best of our knowledge, there is no research how to approximate project tasks in a large-scale student software project course.

3. Research Methodology

The main goal of our research is to find a way to define project tasks in large-scale software project courses that have a fair and balanced effort for all participating teams. Figure 1 shows important concepts of our software project course and their relationships by a UML class diagram.

Each student team (Team) consisting of five to six members gets at the beginning a one-page project task (Task) which is the basis for the analysis phase of the project. Each team consists of an assigned tutor who supervises the team and plays the role of the customer in the project. In the customer role, the tutor has to express his detailed requirements to refine the specified one-page task to a Software Requirements Specification (SRS). In order for these requirements to be expressed in such a way that they correspond to the scheduled time and are balanced against other tasks or teams, the tutor prepares the discussion with the students by deriving a product configuration (Product Configuration) from the feature model (Weighted Feature Model). It is important

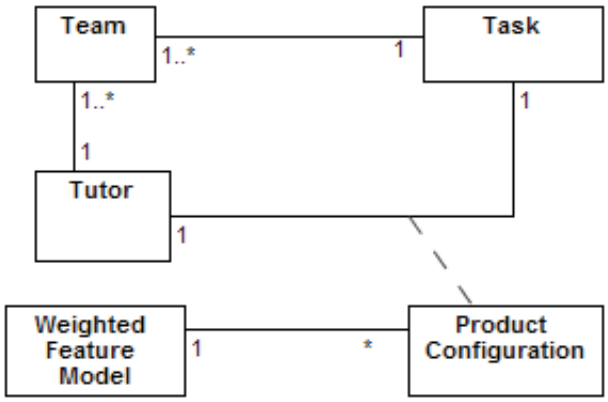


Figure 1. Concepts and their Relationships of our Software Project Course in the Context of the Presented Approach

that this product configuration must nearly meet the predefined total number of Function Points.

After we have performed the project course many years without the feature model, we started to analyze SalesPoint applications to extract features (cf. Subsection 4.1) and designed the SalesPoint feature model (cf. Subsection 4.2). In the next, step we assigned weights to the features (cf. Section 4.3) and implemented a tool which supports the work with the weighted feature model (cf. Section 4.4).

During the past two years, we applied this approach and tool to approximate the project tasks in our annual software project course. In order to evaluate the effects and to respond to our research questions (cf. Section 1), we collected data both by student and tutor questionnaires, and we required feedback of each student to several concerns (cf. Section 5). As we have in previous years regularly collected data after the end of the project, we were able to compare the data to evaluate the new approach (cf. Section 6).

4. Design of a Weighted Feature Model

This section presents our approach of creating a feature model and assigning weights to it. The process we used to get relevant features for the creation of a feature model is described in Section 4.1. Our approach of creating a feature model out of these features is discussed in Section 4.2, where we will also present a small snippet of it. The way in which we assigned a weight to each feature will be shown in Section 4.3. Finally, we will describe our tooling which supports the application of the feature model.

4.1. Extraction of Features

We used the extractive approach [2] to design a feature model for our application framework SalesPoint. We were, in principle, able to analyze hundreds of student projects from a period of 20 years. However, we decided to focus on projects from the winter semester (WS) 2014/2015 which are based on the latest major version of the framework. Of a total of 34 projects, we chose those which had the highest degree of reuse of SalesPoint packages. During the extraction process, we identified such features that are either directly supported by the SalesPoint framework or can at least be implemented by reuse of them. Here we used the following artifacts: (1) the finished application of each team; (2) the related SRS; and (3) their use case diagrams (part of the SRS). We extracted features from each team's project and structured them in order to allow us to assign a weight to each feature.

We decided on a manual feature model design to extract features from artifacts because the diversity of artifacts and student's programs do not support an automatic or semi-automatic approach (which is called feature mining [11]). This decision has multiple reasons, ranging from vastly differing knowledge of the teams and the resulting project quality to the fact that the projects based on an SPL in mind were never considered in former courses. So we had to master several issues in the manual extraction process.

(1) *Inconsistency and Incompleteness of students' projects.* Inconsistency refers to different names for semantically equivalent features. Incompleteness means existing differences between the SRS and the finally implemented application.

(2) *Different but essentially the same implementations of features must be generalized.* For example, one project has a feature/use-case `manage sales personnel`, while another has one called `manage waitresses`. These features have a slightly different meaning in their respective domain, however, essentially they evolve from the management of personnel. Therefore it is irrelevant how exactly this personnel is named. These features are represented in our feature model by `personnel_management`.

(3) *Feature names should be short and applicable to the whole domain, but generally self-explanatory.* In conjunction with the generalization requirement, this was not trivial and only partially possible. For the two extracted features (from different projects) `edit shift schedule` and `edit calendar of events`, the introduced abstraction focuses on time-based events, which are performed in a plan or calendar. However, a feature name could be misunderstood as the tutor might not associate a *plan* with a *schedule* or *event calendar*, but with something else. For this reason, each feature is provided not only by a name but also by a description and examples of different sub-domains.

4.2. Creation of the Feature Model

The feature model was created basically according to the structure of the analyzed projects and their com-

ponents. Components were decomposed into more fine-grained features to allow them to assign an accurate weight. The resulting feature model is a tree, in which each child B of a node A requires A to be selected (or later on implemented), in order for B to become selectable.

A node in the feature model is either *Mandatory*: The node must be selected if the parent is selected, or *Optional*: It is possible to select the node. Furthermore, a node is either *Abstract*: The node has no direct representation through artifacts (i.e. is no concrete feature), but is a means to group other features; or *Concrete*: The node represents a feature, which can be selected and implemented. Features in a group with the same parent are by default logically joint by an AND constraint. But it is also possible to include them into an OR-Group (at least one child must be selected if the parent is selected), or into an XOR-Group (exactly one child must be selected if the parent is selected). In addition, a feature model allows specifying Cross-Tree Constraints (CTC) which define constraints on features in different branches of the tree.

Figure 2 shows a small snippet of the SalesPoint feature model. In this snippet, the abstract root node has only one child, namely the concrete and mandatory feature *A_2_Personnel_Management*. The feature *A_2_Personnel_Mangement* consists of the features (*B_1_Personnel_Overview*, *B_6_Create_Personnel*, and *B_1_Remove_Personnel*). The mandatory feature *B_1_Personnel_Overview* provides the functionality to list personnel data through a simple user interface. It is the prerequisite for the use of the optional sub features *C_5_Filter_Personnel*, *C_3_Sort_Personnel*, and *C_3_Search_Personnel*. The optional children of *A_2_Personnel_Mangement* are the feature *B_6_Create_Personnel*, which provides the functionality as well as a user interface to create a new employee, and the feature *B_1_Remove_Personnel*, which represents the removal of an existing staff member. Note that each feature name (*<level >_<weight >_<mnemonic >*) is composed of the level of the feature tree, the assigned weight to the feature (cf. Subsection 4.3) and a mnemonic name.

4.3. Assignment of Weights to each Feature

We used the Function Point Analysis (FPA) as defined by the IFPUG[2] as the foundation for the assignment of weights to the features in the feature model. The number of Function Points (FP) is a metric which describes the functional size of a feature measured by the necessary cost to implement the feature whereby we measure the cost based on hours.

The FPA distinguishes five components of three complexity levels to categorize functionality, and assigns fixed values, the so-called Function Points, depending on the complexity level the functionality is put in. The components are *External Input (EI)*, *External Output (EO)*, *External Inquiry (EQ)*, *Internal Logical File (ILF)*, *External Interface File (EIF)*. Each of them is ranked as *Low*, *Average*, or *High*. According to the Component and the complexity ranking, an FP value is assigned to each function.

In the FPA of the SalesPoint SPL, the components of the FPA were adjusted to match the needs of our project course. We removed *EQ* features from further considerations because the requirements for *EQ* were nearly never met. We shifted the according functions to the *EO* category. This has been done as our projects and the contained functions always consist of backend and frontend implementation. Additionally, we removed the component *EIF* from the scope, as the teams develop an application without interfaces to other systems. Therefore, no external data is handled, and the *EIF* component is obsolete.

We introduced two new components. This is the *Miscellaneous* component to be able to measure relatively small features or features that do not require real effort in terms of another component. Furthermore, we decided to exclude any non-trivial algorithmic functionality from the other components and classify them in the *Processing Complexity* component. It allows for a more accurate estimation, as we can assign FPs to this component that reflect the difficulty from the student teams' point of view.

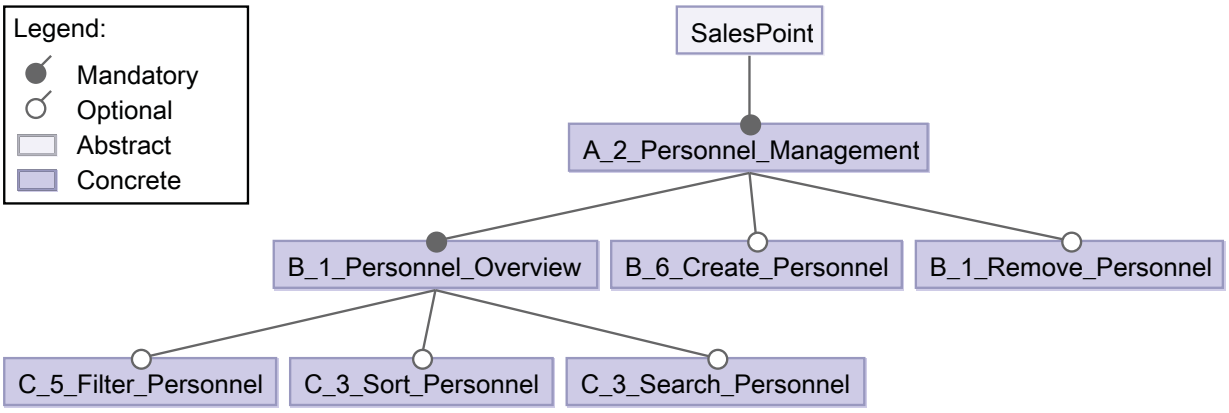
Finally, we also adjusted the Function Points assigned to the components in order to represent the effort within our project course and according to the skills of the students appropriately. A complete overview of our components mixed with the complexity levels and the respective FPs assigned can be seen in Table 1.

TABLE 1. Comparison between the FPs of the original FPA and the SalesPoint adjusted version of the FPA

Nr.	Component with Complexity	FPA	Adjusted FPA
1	Low EI	3	3
2	Average EI	4	5
3	High EI	6	8
4	Low EO	4	3
5	Average EO	5	6
6	High EO	7	9
7	Low EQ	3	-
8	Average EQ	4	-
9	High EQ	6	-
10	Low ILF	7	7
11	Average ILF	10	10
12	High ILF	15	15
13	Low EIF	5	-
14	Average EIF	7	-
15	High EIF	10	-
16	Low Processing Complexity	-	5
17	Average Processing Complexity	-	10
18	High Processing Complexity	-	15
19	Miscellaneous	-	1

2. International Function Point User Group, www.ifpug.org

Figure 2. A Snippet of the SalesPoint Feature Model.



4.4. Tooling

The weighted feature model is designed to support tutors to create a software product configuration that is the basis for their discussions with the students in the analysis phase. We used the widespread tool FeatureIDE[3] in order to visualize our feature model as a feature diagram (cf. Figure 2). It allows for an easy selection of features and verifies whether the selected configuration is valid. It even offers support for the detection of the causes of invalid configurations, which makes it overall an appropriate tool for our cause.

We extended the FeatureIDE tool to dynamically calculate the total number of FPs from the weights encoded in the feature names during the configuration process. This has proven to be a great help, as tutors get direct feedback about whether or not the size of their feature list conforms to a predefined weight.

5. Case Study

We have been working with the feature model in two project courses, in the winter semester 2015 (*WS15*) and 2016 (*WS16*). The intention was to overcome the following general problems which led to the complaints of the students:

1) The given one-page description of the tasks was very general and vague, but also vastly differing in the effort.
2) The tutors had no clear guideline as to how much functionality they should require from the teams in order to keep the needed effort equal for all teams.

In order to evaluate the effects and to respond to our research questions (cf. Section 1), we collected data both from student and tutor questionnaires, and we required feedback of each student to several concerns. As we have in previous years, we regularly collected data after the end of the project, and were able to compare student's data to evaluate the new approach (cf. Section 6). In each course, all projects were archived, and each team completed a questionnaire about the team's experience within the project course and their general evaluation. Furthermore, each student had to write an individual experience report, which is also included in the analyzed data set. We used project data from the last winter semester course before the application of the feature model (*WS14*) as the foundation for the comparison in our case study. Additionally, we designed a questionnaire for the tutors to evaluate their experience and to ask them for feedback.

From each student, the project course requires six credits points which correspond to an approximate total effort of 180 hours. The course lasts 12 weeks which means that every student should plan about 15 hours per week for the project work. A team consists of 5 to 6 students, which means that the project is small compared to industrial projects, but it is a first large and work-related project for inexperienced students. The intended time investment is not a strict demand, as experienced students might need considerably less time to meet our requirements.

In Subsection 5.1 we present data we collected from the student teams, their responses to the questionnaire and their experience reports. Subsection 5.2 summarizes the results from the questionnaires completed by the tutors.

5.1. Data from the Student Projects

We collected and analyzed student project data from the three named courses *WS14*, *WS15*, and *WS16*. The *WS14* data is mainly used as the reference to assess

3. wwwiti.cs.uni-magdeburg.de/iti_db/research/featureide/

TABLE 2. Data from student projects

#	Evaluated Data	WS14	WS15	WS16
S1	Number of teams	34	31	27
S2	Mean time per student per week invested	15,6 hours	14,1 hours	13,8 hours
S3	Standard deviation of #S2	3,7	3,0	3,2
S4	Number of teams that deviated >= 5 hours from the target effort	9 (26,5%)	4 (12,9%)	6 (22,2%)
S5	Number of teams that named differences in effort between teams, or criticized that the effort needed to complete the course exceeds the scheduled time	11 (32,3%)	4 (12,9%)	2 (7,4%)
	Mean team's evaluation of...			
S6	the completion of the task	2,6	2,8	3,1
S7	Standard deviation of #S6	0,8	0,6	0,6
S8	the quality of the implementation	2,4	2,6	2,9
S9	Standard deviation of #S8	0,8	0,7	0,8
S10	Number of teams that thought their task was of comparable effort to most other teams	No Data	No Data	22 (81,5%)

the applied feature model based approach in *WS15* and *WS16*. Table 2 lists consolidated data from these three courses. Each row represents one evaluated statistic while the respective column represents the project course year the data is from.

Our questionnaire asked each team (total of cf. #S1) to note the mean time per week per student invested in the project (#S2). The time was tracked by the students during the whole course on a day-by-day basis. We see a continuous decline in the mean time, even below the scheduled time of 15 hours per week per student. The standard deviation of the mean time (#S3) decreases as does the mean time.

Looking at each team's individual answer to the question regarding the mean time invested, we also took into account how many teams deviated at least 5 hours from our target commitment of 15 hours (meaning 10 and below, as well as 20 and above) which would account for a discrepancy of at least 33% (#S4).

Each team member had to write an experience report, in which they could freely write about how they liked the course or what problems they had (#S5). Even if this is only related to our approach indirectly, we decided to include these results in our evaluation. In particular, we looked for hints that showed that the students felt either (a) an imbalance between teams and their tasks (especially regarding the required effort), or (b) complained that the effort required to finish the course exceeded the scheduled project time. With (a) we targeted the communication between the students and teams, as it was very likely that friends of different teams would communicate about their tasks and how their project was going. Therefore, we hoped to get any - even if highly subjective - feedback regarding the fairness of the course. With (b) we assumed that only students who are confident in their ability would write such a statement. We excluded any general complaints about the course being too demanding, as students tend to complain too often. However, we counted any justified critique and checked it against the time invested by the respective student or team. Additionally, we attributed the critique in a single report to the whole team, as students mostly tended to avoid reporting the same problems another team member already submitted.

Within the questionnaire, each team had to state, how they assess their completion of the task (#S6) and how they rate the quality of their implementation (#S8). For both questions, we evaluated the standard deviation (#S7 and #S9). The scale for both was as follows:

- with major deficiencies (1 point)
- with minor deficiencies (2 points)
- fulfilled (3 points)
- overfulfilled (4 points)

The target value is 3 points (fulfilled), as a mean close to this denotes that the tasks we handed out were appropriate for the course.

In the questionnaire handed out in *WS16*, we introduced a question which asked for an evaluation of the students regarding the comparability of the project tasks (#S10). In that course, the tasks were already balanced, so hopefully of comparable effort, which is why we wanted to see if this held true in the team members' eyes.

5.2. Data from the Tutors

With the goal to evaluate and improve our approach, we also designed a questionnaire for the tutors. It was first handed out at then end of *WS15* and with minor adjustments also in *WS16*. As the tutors are mostly former project course participants, they tend to know both sides, which is why they are an important asset in evaluating their answers.

In *WS15*, we had 17 (resp. 18) tutors who supervised at least one team, and we reached a response rate of 100%. In *WS16*, however, we had 14 (resp. 15) tutors but only managed to get a response from 10 due to organizational problems. In both courses, one author of this paper was excluded from the evaluation to prevent potential biases.

Table 3 summarizes the data we collected.

Question #T1 targets the state of the course before our approach was used and was therefore only asked in *WS15* to compare the before and the after directly. As each tutor either participated as a student in the course

TABLE 3. Data from tutors

#	Question	*WS15*	*WS16*
T1	Do you think that the basic demands on the teams varied strongly in the course before?	14x Yes, (...) 3x No	Not asked
T2	Do you think that the basic demands on the teams varied less strongly in this course?	15x Yes 2x No	Not asked
T3	Do you think that the basic demands on the teams were equal at the beginning of this course?	Not asked	6x Yes 4x No
T4	Do you think that the software product configurations and the added weights are appropriate to compare the tasks and the corresponding required effort?	4x Yes 11x Yes, (but...) 2x No, (...)	4x Yes 5x Yes, (but...) 1x No, (...)
T5	Did you have the impression that the complexity of the application(s), which your team(s) had to implement (based on the product configuration), was comparable to those of other teams?	15x Yes, (...) 2x No, (...)	10x Yes, (...) 0x No, (...)
T6	Do you consider the creation of a product configuration as being useful?	14x Yes 3x No	10x Yes 0x No
T7	Is the predefined total task weight (currently 250 FPs) for mandatory requirements appropriate for a team of five students?	11x Yes 6x No, (...)	8x Yes 2x No, (...)
T8	Did the feature model contain all features you wanted to select?	13x Yes 4x No	Not asked
T9	Was the provided manual sufficient and helpful?	17x Yes 0x No	9x Yes 1x Cannot judge
T10	Are the included feature descriptions sufficient?	16x Yes 1x Cannot judge	Not asked

or was already a tutor in *WS14*, we see every tutor as fully qualified to answer this question.

As a direct follow-up to the prior discussed question, we asked (#T2) whether or not the tutors perceived that the basic demands varied less strongly in *WS15*. Our target was to evaluate, whether the feature model approach was appropriate in order to (at least subjectively) match the requirements each team has to fulfill.

In *WS16*, we adjusted the question (#T3), as some tutors did not experience *WS14*, and therefore no direct comparison was possible. However, the question is now more general and does not only demand an assessment about a comparison to a past state. On an abstract level, we wanted to figure out whether or not all teams started from the same level of complexity and extent, or if even the starting conditions were unequal (regarding the requirements of the course).

Question #T4 checks whether or not the tutors see our approach as an appropriate solution to the referred problem.

While we already asked if the tutors see that each team had equal starting conditions, we wanted to evaluate whether this was also the case during the course, especially regarding the complexity of the application to develop, which stems from the initially created product configuration (#T5).

We also wanted to know if the tutors see the creation of the product configuration as useful (#T6). This allows us to determine whether the effort it takes to (a) complete the setup (includes reading the feature model manual), (b) learn how to handle the tool, (c) get familiar with the feature model and read the descriptions of the features, and (d) finally create the product configuration is worthwhile.

In order to determine whether or not we need to adjust the predefined total weight for the tasks, we asked the tutors if they see the given value as sufficient (#T7). Besides a target weight for a task, we also allowed a variance (currently 30 FPs) to give the tutors more freedom in their feature choices.

As we used the extractive approach to design a feature model from a subset of existing applications from one course, we wanted to evaluate if we were able to represent the whole application domain appropriately (#T8). This question was only asked in *WS15*, as in *WS16* we wanted to gather more data about missing features that will not be discussed in this paper.

In order to support the tutors during the first steps as described above (#T6), we wrote an extensive manual to guide the installation, usage, and even provide necessary information about SPLs in one comprehensive document (#T9).

Finally, we asked the tutors to evaluate if the provided descriptions for each feature were sufficient (#T10). Almost all tutors in *WS15* found this to be true, so we removed the question in *WS16*.

6. Evaluation

In this section, we evaluate the collected data presented in Section 5 and answer our research questions (RQ) (cf. Section 1). In Subsection 6.1, we address RQ1 and RQ2 by discussing the implications of the data collected from the teams and the respective students. Subsection 6.2 addresses RQ3 by evaluating the responses of the tutors.

6.1. Evaluation of the Data from the Student Projects

We asked if the approach of a Weighted Feature Model-based derivation of related project task is ap-

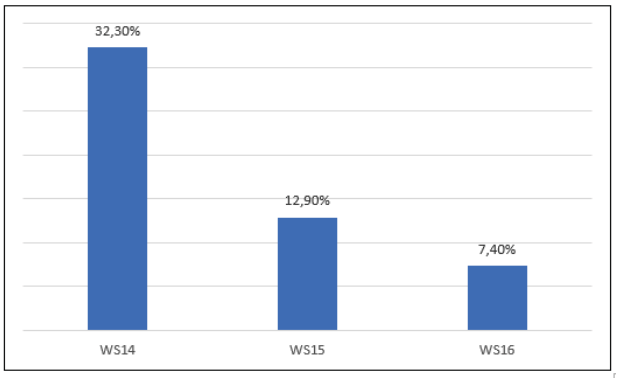


Figure 3. Percentage of teams that named differences in effort between teams, or critized that the effort needed to complete the course exceeds the scheduled time (#S5)

propriate to approximate the project tasks in a large-scale software project course relating to (1) the average effort measured in hours and (2) comparable effort for each project task (RQ1). The collected data #S2, #S3, and #S4 in Table 2 show that the project time invested has decreased in the courses *WS15* and *WS16* (hereinafter referred to Weighted Feature Model approach based courses (*WFMs*)). The standard deviation (#S3) and the deviation in hours from the target effort (#S4) have also decreased which indicates that the effort for the different tasks are becoming equal overall. Teams evaluated in *WFMs* the completion of tasks (#S6) and the quality of their implementation (#S8) better than in *WS14*. We take these data and the corresponding mostly reduced standard deviations (#S7 and #S9) as an indication that the quality of the student projects could be increased with a balanced task complexity. Overall, we, therefore, answer RQ1 positively.

Research question RQ2 addresses the non-balanced effort in the project course for all teams as a reason for student complaints. In comparison to *WS14*, fewer teams named differences in effort between teams or criticized that the effort needed to complete the course exceeds the scheduled time in *WMFs* (#S5 graphically depicted in Figure 3).

Furthermore, the majority of teams (81,48%) in *WS16* explicitly confirmed (#S10) that they saw their own task as compared to the tasks of other groups. So we see question RQ2 negated which means that we have basically no complaints from the students anymore.

6.2. Evaluation of the Data from the Tutors

The responses to the tutor questionnaire allow an evaluation of the research question RQ3, and they are mostly positive (cf. Table 3).

Research question R3.1 asks if the approach is appropriate to approximate the project tasks in a large-scale software project course. Data regarding #T1 and #T2 compare the situation before and after our approach has been applied. While the tutors see that the basic demands varied less strongly in *WS15*, we can only guess that this is due to the effectiveness of our approach. However, as our work was the only change in the course that targeted this variation, we assume that the assumption is justified. On the other hand, #T3 data shows that we only reached a slight improvement in comparison to the state of the course in *WS14*. Even though 60% of the tutors think that the demands are equal, we still see room for improvement. #T4 and #T5 data show that an overwhelming majority of tutors feel that the approach is appropriate to achieve the desired goal, though they still would like to improve either the feature model, the assigned weights, or even both. All in all, we come to the same conclusion as in the evaluation of the student data that the approach helps to approximate the project tasks (cf. Subsection 6.1).

The data set consisting of #T6, #T7, #T8. #T9 and #T10 essentially address research question RQ3.2. Tutors see the initial creation of a product configuration at the beginning of the course as useful (#T6). This indicates that the costs (i.e. their invested time) are outweighed by the benefits, and therefore the approach must be an asset to their role as a customer. Tutors generally think that our predefined weight for a task, or the variance we allowed for it, is adequate, but a notable number also disagrees. This can most likely be seen as related to data #T4 and should be looked into in the future. However, we can perhaps agree that the predefined weight is too low, as the mean time spent for the project (#S2) has dropped compared to *WS14*. #T8, #T9, and #T10 data give a positive answer to the question whether the tutors are sufficiently supported by the feature model in their work. Thus, the question RQ3.2 can also be regarded as essentially positive.

7. Threats to Validity

According to Wohlin et.al. [12], we consider the following validity threats to empirical studies in software engineering that basically refer in our research to (1) the design of artifacts collecting data in the case study (construct validity), (2) the comparative survey of collected data (internal validity), and (3) the generalization of the results outside the scope of the SalesPoint framework and our didactic course concept (external validity).

In terms of *construct validity*, we need to take into account threats arising from the design of the questionnaires. As students did not know about our approach, they would neither be able to infer benefits, nor drawbacks, which is why we assume that the answers collected by *student questionnaire* to be real. Moreover, the teams were required to fill out the student questionnaire in a closing and feedback session. Unfortunately, we found hints of false data occasionally being submitted, which might influence our evaluation of this data.

A further threat to validity is the time spent of the students that they had to record. We made sure that students make an accurate record of the times. But this is difficult because the students often discuss project issues between the lectures and therefore do not always work continuously on the project. In addition, the different motivation, the teamwork and the technical skills of the students influence the time spent. This would explain the increased variance of the number of teams that deviated >= 5 hours from the target project effort (#S4) in WS16.

All tutors were handed an extensive document (*tutor questionnaire*) with all aspects regarding the usage of our approach. Additionally, we gave to every question asked a detailed description and selected examples, wherever misunderstandings could happen or different tutor experience might play a role. Unfortunately, the tutors might have had the desire to rate their own work positively But as the tutors knew that their answers would influence whether or not our approach will be used in future courses again, we assume that their answers are unbiased. Otherwise, they would harm themselves if they will decide to supervise teams again in the future.

Besides the questionnaires, we evaluated student experience reports. These were not anonymous, which has been remarked by several students as a hindrance for a truly objective feedback. However, as the aspects we researched in this work are not harmful to the students if they gave us their personal feedback, we conclude that the answers regarding the goals of our approach represent unbiased results. One possible threat to construct validity is the format of the experience report. While it was freely written in *WS14*, we changed the format afterward to include general questions, but nothing related to the evaluation of this work. The students could still freely write their opinion.

Our project course changed considerably between *WS14* and *WS15*, not just for the feature model approach. We introduced several technological changes like version management on GitHub, Continuous Integration of student applications, and also putting more focus on code quality (potential threats to *internal validity*). But these changes do not influence the analysis phase in which the features to be implemented are defined, and thus not the answers to the corresponding questions. One author of this research was also tutor in all three evaluated courses. We excluded his answers on all tutor questionnaires to avoid biased data. The results of his supervised teams are still part of the analyzed data, as we found no abnormalities compared to other teams.

Threats to *external validity*, which refer to the generalization of our results, arise by the application of the feature model approach applied to a concrete framework and by the concrete organization and rules in our project course. We performed the presented approach by example. We are aware that this only applies initially to our special scenario. One crucial problem lies in the weight of each feature. First evaluations have shown that our approach with fixed weights for each feature might only be applicable in small domains with relatively simple functionality, like ours. With rising complexity, we assume that features start influencing each other more considerably, therefore increasing the required effort and actually also increasing the Function Point Count of the whole application. However, we see the potential of generalizing the approach and adjusting it to the different needs of comparable project courses. As long as a general domain for the SPL is defined and a fine-grained feature model can be created in accordance with the components of the Function Point Analysis, our approach can be adopted.

8. Conclusions

We presented an approach to approximate project tasks for a family of similar applications in an everyday application domain. The aim of the approximation of project tasks is the result of the requirement to balance the effort for many teams in a large software project course. We use the SalesPoint framework in our course to enable students to implement point-of-sale applications. The solution for achieving our goal is to consider the family of SalesPoint applications as a software product line, and we create a feature model for this SPL. Then, a rough description of a project task can be refined by derivation of a product configuration of the SalesPoint feature model. The idea behind this method is that each feature has an assigned weight in terms of function points. By summing up the FP of the product configuration, the effort to implement the individual applications can be compared and balanced. The challenge to the application of this approach is a well-designed feature model and the calculation of a realistic number of function points for all features. We master this challenge by a comprehensive analysis of existing SalesPoint applications and additional project data to extract features and to calculate corresponding function points.

We applied our approach in the recent two courses and collected extensive data. In this case study, we compared these data with data of a former course to evaluate the effects of our approach. Our research question was if the use of a Weighted Feature Model-based derivation of related project task is appropriate to approximate the project tasks in a large-scale software project course. Based on the evaluation of the collected student project data, we are sure that the feature model will help to approximate the student project task, which will eliminate student complaints about unequal requirements on their projects. At the same time, we were able to validate this observation by the survey of the tutors, who evaluated the concept, including the tooling provided, as helpful.

Our conclusion is that we see the potential of generalizing the presented approach for other large-scale software project courses. As long as a family of software applications is the subject of a project course with many

student teams our approach can be adopted. In our next courses, we will refine the feature model including the weights for the features based on the analysis of the recent course and the experience of the tutors.

Acknowledgments

The authors would like to thank the tutors in our software project courses for their dedicated support of the student teams and their help in the evaluation of the approach presented in this paper.

References

[1] Steffen Zschaler and Birgit Demuth and Lothar Schmitz, *Salespoint: A Java framework for teaching object-oriented software development*, Sci. Comput. Program 79: 189-203 (2014).

[2] Sven Apel and Don Batory and Christian Kästner and Gunter Saake, *Feature-Oriented Software Product Lines. Concepts and Implementation*, Springer, 2013.

[3] K.C. Kang and S.G. Cohen and J.A. Hess and W.E. Novak and A.S. Peterson, *Feature-oriented domain analysis (FODA) feasibility study*, Technical Report CMU/SEI-90-TR-021, SEI, Carnegie Mellon University, November 1990.

[4] Allan J. Albrecht, *Measuring Application Development Productivity*, Proc. of IBM Application Development Symp. In IBM Application Development Symp.: 83-92 (1979).

[5] M. Wiggberg and M. Daniels, *Reflecting on running large scale student collaboration projects*, 38th ASEE/IEEE Annual Frontiers in Education Conference, Saratoga Springs, NY: S3C-8-S3C-12 (2008).

[6] Ch. Ungerleider, *Reflections on the Use of Large-Scale Student Assessment for Improving Student Success*, Canadian Journal of Education 29: 873âĂŘ883 (2006).

[7] J. Ceddia and M. Dick, *Automating the Estimation of Project Size from Software Design Tools Using Modified Function Points*, Sixth Australasian Computing Education Conference (ACE2004), Dunedin, New Zealand, 30 (2004)

[8] B. Demuth and L. Schmitz and B. Wittek, *Softwaretechnologie-Praktikum im Grundstudium: universitäres oder reales Projekt?*, Conference on Software Engineering im Unterricht der Hochschulen (SEUH), 2005

[9] B Demuth and S. Götz and H. Sneed and U. Schmidt, *Evaluation of Students' Modeling and Programming Skills*, Proceedings of the Educators' Symposium co-located with ACM/IEEE 16th International Conference on Model Driven Engineering Languages and Systems (MODELS), CEUR Workshop Proceedings, Vol-1134, 2013

[10] O. Dieste and A. M. Aranda and F. Uyaguari and B. Turhan and A. Tosun and D. Fucci and M. Oivo and N. Juristo, *Empirical evaluation of the effects of experience on code quality and programmer productivity: an exploratory study*, Empirical Software Engineering, Springer, 22: 2457–2542 (2017)

[11] P. Dolla and Z. Tu and H. Tao and S. Belongie, *Feature Mining for Image Classification*, IEEE Conference on Computer Vision and Pattern Recognition, Minneapolis, MN: 1-8 (2007).

[12] C. Wohlin and et.al., *Experimentation in Software Engineering*, Springer, 2012