Received: 14 October 2018 | Revised: 12 February 2019 | Accepted: 17 February 2019
DOI: 10.1002/smr.2165

WILEY Journal of Software: Evolution and Process

SPECIAL ISSUE PAPER

An evaluation of effort estimation supported by change impact analysis in agile software development

Binish Tanveer[1] | Anna Maria Vollmer[2] | Stefan Braun[3] | Nauman bin Ali[4]

[1] Process Engineering Department, Fraunhofer Institute for Experimental Software Engineering IESE, Kaiserslautern, Germany
[2]Data Engineering Department, Fraunhofer Institute for Experimental Software Engineering IESE, Kaiserslautern, Germany
[3]Product Development, Insiders Technologies GmbH, Kaiserslautern, Germany
[4]Department of Software Engineering, Blekinge Institute of Technology, Karlskrona, Sweden

Correspondence
Binish Tanveer, Fraunhofer Institute for Experimental Software Engineering IESE, Fraunhofer-Platz 1 67663 Kaiserslautern, Germany.
Email: binish.tanveer@iese.fraunhofer.de

Funding information
Bundesministerium für Bildung und Forschung, Grant/Award Number: 01IS15050G

Handling Editor: ICSSP2018 SI

Abstract

In agile software development, functionality is added to the system in an incremental and iterative manner. Practitioners often rely on expert judgment to estimate the effort in this context. However, the impact of a change on the existing system can provide objective information to practitioners to arrive at an informed estimate. In this regard, we have developed a hybrid method, that utilizes change impact analysis information for improving effort estimation. We also developed an estimation model based on gradient boosted trees (GBT). In this study, we evaluate the performance and usefulness of our hybrid method with tool support and the GBT model in a live iteration at Insiders Technologies GmbH, a German software company. Additionally, the solution was also assessed for perceived usefulness and understandability in a study with graduate and post-graduate students. The results from the industrial evaluation show that the proposed method produces more accurate estimates than only expert-based or only model-based estimates. Furthermore, both students and practitioners perceived the usefulness and understandability of the method positively.

KEYWORDS

agile, case study, change impact analysis, effort estimation, expert-based

1 | INTRODUCTION

Software effort estimation methods are broadly classified as data driven (or model based), expert based, and hybrid.[1] Data-driven (or model-based) methods require extensive data for calibration (eg, COCOMO family of models) and are ineffective in the context of agile software development.[1,2] Furthermore, in industry, practitioners mostly rely on expert-based methods for effort estimation.[3-5] However, expert-based estimation methods like Wideband Delphi, due to the reliance on practitioners' judgment are limited by human biases (eg, wishful thinking, group effects) that reduces the prediction capability of this method.[3]

To address the limitations of these methods, some hybrid estimation methods have been proposed like Bayesian belief networks (BBN) and CoBRA.[6] Unfortunately, these methods are either inherently too complicated (eg, BBN) or are too effort intensive (eg, CoBRA) to deploy and use in an industrial setting especially in an agile development context, where change is encouraged and embraced.[1] Researchers have identified the need for considering the impact of a change on the underlying system to improve the effectiveness of expert-based estimation methods.[7,8] Change impact analysis (IA) techniques identify software life cycle objects that are likely to be affected by a given change request.[9]

In our previous work,[8,10,11] to support expert-based estimation methods with change impact analysis, we have synthesized research from two areas: (1) change impact analysis[9] and (2) effort estimation (both expert and model based). We used case study and survey research to understand the needs of industry practitioners and their challenges and development context. The results from these studies and of systematic literature reviews on effort estimation and change impact analysis were used to develop Hybrid Effort Estimation in Agile Software development (HyEEASe) method. HyEEASe was developed incrementally. Initially, a conceptual framework[10] was developed in close collaboration with SAPcompany. The framework was instantiated, and the potential tool support was materialized using mock-ups with SAP. Encouraged by the results,[11] using prototyping the method, HyEEASe, was further refined in close collaboration with InsidersTechnologies.

In this paper, we report the findings from the application and evaluation of HyEEASe at Insiders Technologies and an assessment of the understandability of HyEEASe with graduate students. In the industrial case, the estimates acquired using HyEEASe were compared with (1) the estimates made by the experts using only Planning Poker, (2) the estimates obtained by an estimation model, in this case, gradient boosted trees (GBT), and (3) the actual effort spent during the sprint. We also collected the practitioners' opinion about the method. Using students as participants, we evaluated the perceived usefulness and understandability of HyEEASe and Planning Poker.

This work entailed the following for the industrial study: (1) selection of an appropriate impact analysis technique given the industrial case, (2) integration of the selected impact analysis technique and Planning Poker in HyEEASe, (3) development of an impact model for use in HyEEASe, (4) development of an estimation model using GBT and historical data from the industrial case (5) use of HyEEASe in the industrial case and comparison of results with estimates acquired from GBT, Planning Poker, and actual effort spent.

The evaluation with students, which aimed at assessing understandability of HyEEASe with graduate students at Technical University of Kaiserslautern (TUKL), entailed the following: (1) development of method descriptions for both Planning Poker and HyEEASe, (2) selection of an open source system that is in a domain familiar to students, (3) presentation of the system architecture, (4) creation of user-stories that need to be estimated, (5) development of an experimental design for the study including pretest to block the impact of prior knowledge and experience of students, design of estimation tasks, and the design of a data collection instrument.

The paper is an extension of a conference paper[12] where only the industrial evaluation part of the research was reported. In this paper, the selection and integration of an impact analysis technique for HyEEASe, the detailed work flow as well as the development of both impact and GBT model have been revised and discussed in more detail. Moreover, the overview of the design has been added. The remainder of the paper has been revised, and several sections like the section presenting threats to validity have been extended. Additionally, the design and results of the evaluation of the proposed hybrid method in academic settings are reported in this extension.

The paper is structured as follows: section 2 presents related work. Section 3 describes the selection of impact analysis and effort estimation techniques for their integration in HyEEASe. Section 4 illustrates high level design, whereas section 5 details the work flow. Section 6 explains the development of the impact analysis model that was used in HyEEASe. In section 7, we explain the development of GBT model. In section 8, we describe the case study and the formative evaluation. Section 9 presents the evaluation with students. Section 10 discusses the validity threats. We conclude the paper in section 11 with a summary and indications of future research directions.

2 | RELATED WORK

Various attempts, including estimation models, have been made to improve effort estimation in the agile development context. However, it continues to be a challenge in the software industry.[13,14]

Several studies have suggested the use of function points. However, function points are not commonly used in the industry. Kang et al[15] proposed a dynamic software cost estimation model using function points and a project tracker (using a Kalman Filter) on velocity. Hussain et al[16] proposed a methodology that finds COSMIC function size from informally written textual requirements for estimating development effort in the agile development context.

Another set of techniques leverages the story points instead of function points, which are more relevant for the agile context. Satapathy and Rath[17] used a story-point-based approach and considered different machine learning techniques such as decision trees, stochastic gradient boosting, and random forests to assess estimation prediction. The results showed that the stochastic gradient boosting technique outperformed the other techniques used. Ziauddin and Zia[18] developed a story-point-based regression model. It uses a ranking mechanism to specify story size and complexity, considers and ranks certain other factors that impede the velocity of creating a model and used 21 projects to calibrate it. Popli and Chauhan[19] also considered story-point-based regression analysis for estimating effort.

Bhalerao and Ingle[20] proposed a Constructive Agile Estimation Algorithm, which incorporates factors believed to be vital for accurately investigating the cost, size, and duration of a project. Raslan et al[21] proposed a fuzzy-logic-based effort estimation framework considering user stories. Moser et al[2] proposed a model using neural networks and code metrics to predict the development effort of a user story. Their approach was extended by Abrahamson et al[22] who extracted and used keywords from user stories instead of code metrics to build models (regression, neural networks, support vector machines) for predicting the development effort for a user story. The results show that effort estimation works well, only if the developers write well-structured user stories.

Despite this research on improving effort estimation in an agile context, a lack of empirical evidence on the accuracy of these methods (or models) as well as a lack of information on the data set used to build models, size metrics, and cost drivers is reported by the review.[23] Moreover, most of the improvement approaches either improve expert estimation by providing additional factors to consider during estimation or by proposing a data-driven static model that keeps expert out of the estimation process just like the estimation models proposed for traditional development context thereby violating the Agile Manifesto.[24]

Our method, on the other hand, is hybrid in nature, ie, it consists of a tool-based decision support system that concurrently interacts with the expert while they make estimations and provides them with the required data (in the form of the potential impact of implementing a new user story) thereby assisting them during the estimation process. Additionally, we have developed an estimation model (based on GBT). This study aimed to compare the accuracy of estimates generated by a purely expert-based (Planning Poker), a hybrid (HyEEASe), or a purely model-based (GBT model) estimation and find which of them is more effective.

3 | IMPACT ANALYSIS AND EFFORT ESTIMATION TECHNIQUE SELECTION

The selection and integration of impact analysis and effort estimation techniques were done by using our proposed framework[10] and guidelines[25] to instantiate it. The results of the state-of-the-art analysis of effort estimation in agile development, indicate the expert-based estimation methods are most often used, with "Planning Poker" being the most frequently employed method.[13] This motivated the choice of Planning Poker as the estimation method that will be supported by impact analysis techniques. By doing so, the scope of this study was reduced in terms of the selection of estimation techniques.

From the review of change impact analysis techniques, it was found that there exist two main types of analysis approaches, which are static and dynamic impact analysis.[9] Dynamic impact analysis involves program execution and the collection of the execution trace information. It is, therefore, more expensive than static impact analysis due to the overhead of expensive dependency analysis during program execution. Moreover, the impact set identified through dynamic impact analysis often includes false negatives. Due to these reasons, dynamic impact analysis was not considered the optimal option for performing impact analysis in this context.

Static impact analysis has three types, ie, static structural analysis, textual analysis, and historical analysis. Textual analysis extracts conceptual dependence (coupling) based on the analysis of the comments and/or identifiers in the source code. It relies heavily on the developer's writing skills and choice of words to describe implicit relationships in identifiers and comments of source code. Therefore, there is a risk that different developers used different vocabulary to express the same functionality or vice versa. This requires natural language processing for analyzing code comments and identifiers, which in itself is a research topic and is beyond the scope of this study.

In general, to start an IA, an impact set is required as input, whereas the impact volume is expected as the output. The impact set comprises component(s) identified by the experts after analyzing the change request (or user story). The volume of impact means, "what" the impact was (ie, which component/s were changed in the past concerning a particular user story), and "how much" it was (ie, quantified information about the impact).

In the context of this study, it was realized that historical analysis could be used to find out what (which classes) were changed, and static structural analysis could be used to find out how much the identified classes were changed. Since a combination of impact analysis techniques can provide better results (impact) than the consideration of single techniques,[9] therefore, structural static and historical impact analysis were adapted and applied. With this selection, the scope of the study was also reduced in terms of impact analysis techniques selection. The quantification of the impact is specified by measuring the changes through structural static impact analysis using code metrics as seen in Table 1.

The code metrics (LOC, CBO, WMC) (as described in Table 1) for the method level were chosen as these were found to be the most relevant ones in impact analysis research as well as in the state-of-the-practice analysis for the task of effort estimation. Li et al[26] took five metrics from Chidamber and Kemerer,[27] added three of their own, and found a strong correlation between these metrics and maintenance effort. Briand et al[28] also found relationships between most of the coupling/cohesion metrics and fault proneness of classes. They further examined if coupling measures could be used for estimating an impact set.[29] Antonio et al[30] used the size of the modification, which is a key factor for maintenance effort and cost estimation, for estimating object-oriented programs. Similarly, Zimmermann et al[31] applied data mining to version histories to extract the cochange coupling between files for performing IA.

For each identified class in the impact set, the quantified added and deleted values of code metrics (LOC, WMC, and CBO), as well as the changed code metrics of cochanged classes, is returned through static impact analysis. Cochanged classes refer to those classes that were changed together in the version control system while implementing a user story for the same sprint. Such classes seem to have cochange coupling.[31] The information required for historical analysis comprised of the following:

- Class name (class in the identified impact set).
- User story ID (previously implemented user story associated with the new user story via a similar class). "Similar" in the current context refers to the common classes impacted/used between new and already implemented user stories.
- User story type (defect or new feature).
- User story description, actual effort, estimated effort, estimation trend (overestimated or underestimated or just right).
- Difference between actual and estimated effort.
- Total number of affected classes (for respective user story) as well as the details of cochanged classes.

TABLE 1 The code-related impact factors and metrics

Metric name	Explanation
Size in LOC/SLOC	Refers to the number of lines that contain source code.
Cyclomatic complexity or Weighted Methods per Class (WMC)	Refers to the complexity of the potentially impacted methods/classes measured as McCabe's cyclomatic complexity. It is the sum of the cyclomatic complexity of all nested functions or methods.
Coupling between objects (CBO)	Refers to the coupling of the potentially impacted methods/classes. CBO is a count of the number of other classes to which it is coupled. Class A is coupled to class B if class A uses a type, data, or member from class B.

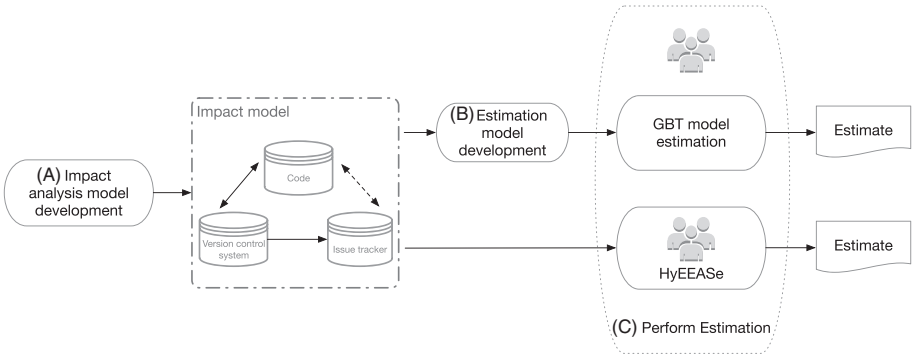

FIGURE 1 High-level design overview

Even though impact analysis is a well-researched field, commercial tools for applying impact analysis are sparse. What is most commonly found are research prototypes, which are often not actively maintained. Due to these constraints, it was not possible to use any existing tool for utilizing impact analysis in the current context. Thus, a prototype tool was developed in MS Access 2010 to support the method.

HyEEASe integrates structural static and historical impact analysis techniques with Planning Poker estimation technique. It is intended to support the experts in identifying the potential impact of a new user story based on the quantified impact information of a similar user story implemented in the past, ie, based on what and how much was changed in the past.

4 | OVERVIEW OF THE DESIGN

The aim was to compare the accuracy of an expert-based estimation method (Planning Poker), an expert-based estimation method with objective information regarding change impact (HyEEASe), and a pure model-based estimation method (GBT model). To perform estimation with HyEEASe, an impact analysis model is required. HyEEASe is composed of two main processes (A. Impact analysis model development, and the use of impact model information in C. Perform estimation) as shown in Figure 1 and explained below.

- Impact analysis model development. In this process, the impact analysis model is developed using static and historical IA. To build this model, three major components, ie, a code repository, a version control system, and an issue tracker system were required. Generally, in software development companies, the user stories are stored in the issue tracking system with a unique ID. The code is directly (straight line) connected to its version control via commit messages. These commit messages keep this ID of the user story when the code is checked into the repositories and are thus traceable. Therefore, the code is indirectly (dashed line) connected to the issue tracker via these user story ID tags. To extract impact information effectively, certain prerequisites, while committing changes to the version control, are as follows.
  - The user story ID should be placed in the commit message while committing.
  - Changes related to only one user story should be committed through one commit at a time (although there may be multiple commits for any user story during a sprint).
- Estimation model development. In this process, the GBT model is developed using a subset of data that is extracted through an impact model.
- Perform estimation. In this process, experts perform estimation for user stories either using HyEEASe or the GBT model. Therefore, this process acts as a decision point for experts. In Figure 2, the processes A and B are shown in dashed rectangles as they are not directly involved in the workflow whereas the process C is shown as the decision point as mentioned here.

5 | DETAILED WORKFLOW

HyEEASe workflow begins when a new user story is to be estimated. HyEEASe workflow is shown in Figure 2 and is explained below.

- Step 1. Impact set identification.
  Input: user story, experts.
  Process: in this step, the experts, identify the initial ‘‘impact set" (the ‘‘seed" for triggering change IA) based on their opinion after analyzing the user story description and doing discussion. This impact set is composed of class/es that the experts identify as candidates that will potentially be impacted due to the implementation of this user story.
  Output: identified potentially impacted code parts (class/es).
- Step 2. Apply impact analysis.
  Input: impact set (seed), impact model (generated from A. Impact analysis model development process).

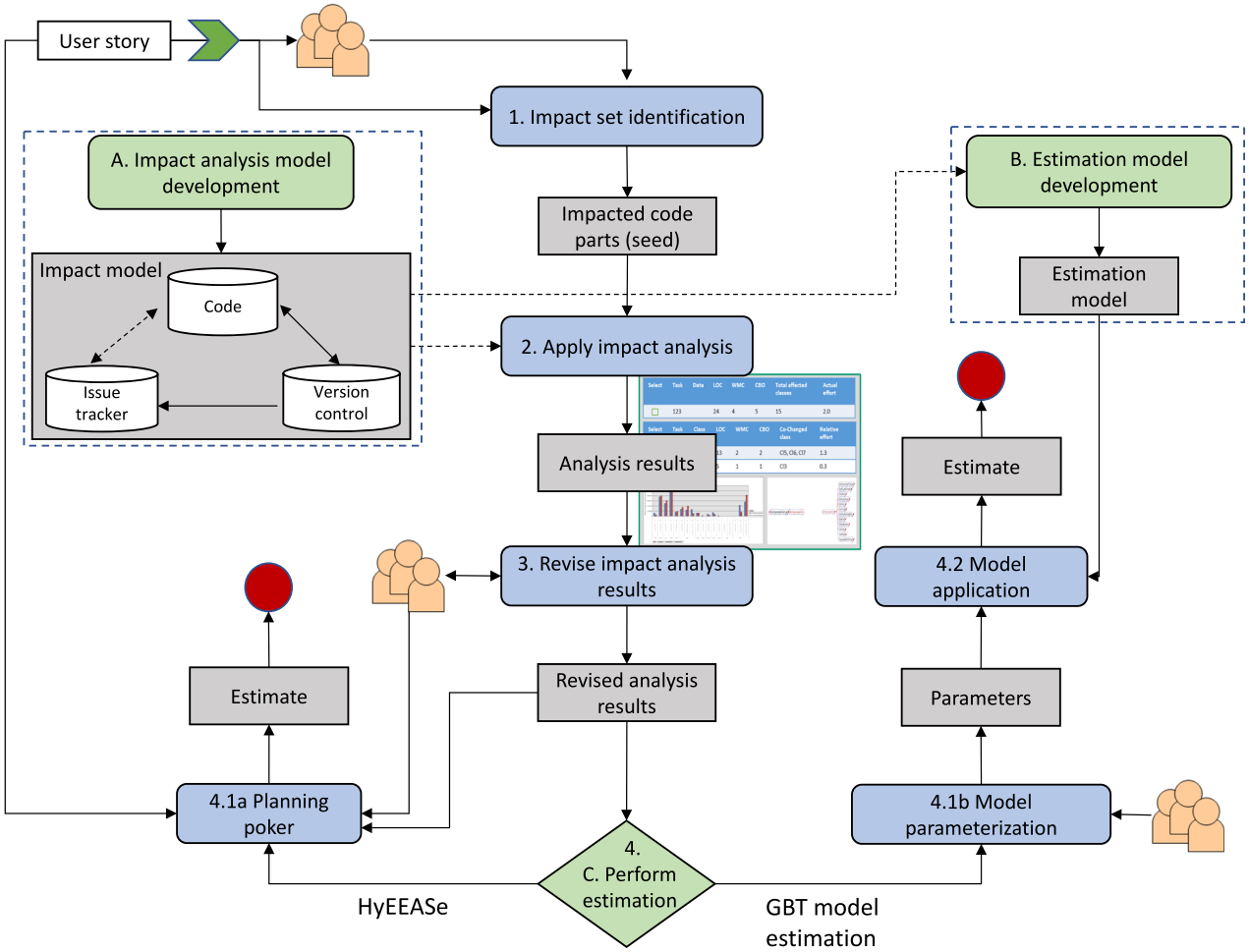

FIGURE 2 Detailed workflow

Process: through the selected static and historical impact analysis techniques, impact analysis is performed. Static impact analysis returns the quantified impact through code metrics, and historical impact analysis returns the effort data, other affected classes, cochanged classes as mentioned in section 3. The analysis results are visualized in the form of tables, charts, and dependency graphs providing structural dependence information.

Output: analysis results.

- Step 3. Revise impact analysis results.

  Input: analysis results, experts.

  Process: the experts can interact with the visualized impact analysis results and make selections regarding the potential impact of the selected user story.

  Output: revised analysis results.
- Step 4. Perform estimation.

  This is a decision point in the workflow. It has two paths, where the experts choose the path to perform estimation. One path leads to estimation with HyEEASe, and the second leads to estimation with the GBT model.

  - HyEEASe: if experts choose this path, they would consult, discuss the revised impact analysis results provided to them in an expert based estimation method like Planning Poker, and then provide an estimate for the user story. As the experts are involved in the process, they are shown inside the rounded rectangle.
  - GBT model estimation: if experts choose this path, they would like to use GBT model estimates by supplying the model parameters. The experts are shown outside the round rectangle since they are not involved in the process other than providing the input. The model provides them with an estimate for the user story.

- Step 4.1a Planning Poker.

  Input: revised analysis results, experts, user story

  Process: the experts will take into account the revised analysis results and play Planning Poker to estimate effort for the user story.

  Output: an estimated effort of the user story.

- Step 4.1b Model parameterization.
  Input: experts.
  Process: the experts will provide input to GBT model parameters.
  Output: parameters of the model are provided.
- Step 4.2 Model application.
  Input: estimation model (generated from B. Estimation model development process), model parameters.
  Process: the model is then applied to obtain an estimate for the user story.
  Output: an estimated effort of the user story.
  The workflow ends once the user story is estimated. The whole information comprising the impact model, expert estimation, and the estimation model is stored in an experience base, which can be updated with current sprint data to support estimation for the next sprint data. With this experience base, organizational learning is enabled at the project level.

To summarize, the impact information is visualized for the experts such that they can interact with it to find the revised potential impact along with the historical effort data. With the help of this information, the expert is asked to perform estimations for the given new user story. The experts can also use the GBT model to obtain an estimate for a given user story. The estimate is saved and also serves as test data for applying the estimation model in the next sprint.

6 | IMPACT ANALYSIS MODEL DEVELOPMENT

Impact analysis model comprised of static and historical impact analysis results. It is shown as process A in Figure 2. For performing static and historical impact analysis, the following data were required.

- User story metadata like ID, type, description, sprint date, commit ID, and effort data.
- A mapping displaying all user stories that affect a certain class (*class → userstory*).
- The total number of classes affected when a single user story is implemented.
- The quantified code metrics for each affected class, ie, LOC, WMC, CBO.

The development of impact analysis model followed the following subprocesses.

1. Data collection.
   Input: issue tracking system, source code, version control system.
   Process: the data for applying impact analysis were gathered from 72 sprints at Insiders Technologies. The data were collected from three components, ie, the issue tracking system, the version control system and the source code as shown in Figure 1.
   The user story information was stored coherently in a database, which was extracted using SQL queries. There existed an *m x n* relationship between a user story and a class. A user story can affect multiple classes. However, capturing these affected classes was complicated because the information on which user story affected which class/es is only stored implicitly in version control. The version control system stored the affected ‘‘files" instead of storing ‘‘classes" against each commit.
   To gather the affected classes, first, every commit had to be captured with its respective user story and analyzed the affected files. A script was created that mapped each user story to its corresponding commits. For each commit, the changed files were analyzed using a static code analyzer namely the Understand tool™.[32] The output of this analyzer consisted of class and file metrics (including the affected class names), which was used in combination with the version control information to create the *class → userstory* mapping for each class.
   Output: mapping of user stories and affected classes to corresponding commits. Code metrics for each affected class at the commits.
2. Perform static impact analysis.
   Input: mapping of user stories and affected classes to corresponding commits. Code metrics for each affected class.
   Process: the data were preprocessed for creating an impact analysis model with the information from static impact analysis. Changes in LOC, WMC, CBO metrics in both dimensions (added/deleted) were collected at the commit level. For any deleted and any newly added files at a certain commit, data from all the commits in a sprint for a certain user story were synchronized. Classes with no changes were discarded. The information about the changed metrics for each class regarding each user story was then aggregated to determine how much was changed in an affected class for a specific user story in a given sprint.
   Output: impact analysis model with information obtained from static impact analysis.
3. Perform historical impact analysis.
   Input: mapping of user stories and affected classes to corresponding commits.
   Process: historical effort data for these user stories were extracted. Differences between actual and estimated were calculated.
   Since effort is generally estimated for the user story, a means for finding the relative effort of each affected class in that user story was needed. Therefore, a metric-based algorithm was devised. Weights were assigned to the code metrics, which were obtained from experts using the cumulative voting[33] method. Two experts were involved in this activity, both had several years of experience in software

development. They were working in the capacity of a product owner and scrum master and were both actively engaging in the estimation task in the company. Before finding the relative effort, the relative change (RC_i) of each class in a user story was found. The relative change means how much a certain class has contributed towards the total change for a certain user story regarding each metric (Y_i) across the total of all affected classes (TC), calculated as follows:

$$RC_i = \frac{Y_i}{\sum_{i=1}^{TC} Y_i}.$$

Then the weighted sum of the relative changes of all affected classes in a user story was calculated. The weighted sum was divided it by the sum of weights, and the result was multiplied by the associated user story effort (AET_j). This gave the relative effort of each class (REC_i).

$$REC_i = \frac{\sum_{i=1}^{TC} W_i * RC_i}{\sum_{i=1}^{TC} W_i} * AET_j,$$

where W_i is the weight of the corresponding code metric (Y_i), RC_i is the relative change of the ith class, AET_j is the associated effort of the jth user story, and REC_i is the relative effort of each ith class affected by the jth user story.

This relative effort is a means to support experts in roughly assessing the associated effort a certain potentially impacted class may have in a specific user story. All the user story and class level impact information along with the respective actual and calculated relative effort were stored in the created impact analysis model for performing impact analysis.

Output: IA model with added historical impact analysis information. This information was combined and visualized through the developed prototype to support experts in making estimates (see step 2 and step 3 in Figure 2).

7 | ESTIMATION MODEL DEVELOPMENT

Boosting is a nonlinear regression procedure in which weak classification algorithms are sequentially applied to the incrementally changed data, to create a series of decision trees that produce an ensemble of weak prediction models. While boosting trees increases their accuracy, it also decreases speed and human interpretability. On the other hand, gradient boosting procedures generalize tree boosting to minimize these issues.[34] Boosting such as stochastic gradient boosting (SGB) has already been applied for predicting development effort[35,36] and have outperformed other models based on neural networks and random forests by achieving higher prediction accuracy.[17]

GBT is the ensemble of either regression or classification tree models that use forward-learning ensemble methods to obtain predictive results through gradually improved estimations. Furthermore, ensemble methods provide more accurate prediction accuracy compared with individual models and hence are more reliable.[37] This motivated the choice of using GBT for building the estimation model.

The estimation model was developed using the following subprocesses. It is shown as process B in Figure 2.

1. Dataset.

   Input: the model utilizes a subset of the same dataset that was used to create the impact analysis model in section 6.

   Process: the dataset comprised a total of 345 user stories including both types of user stories, ie, defects and new features. However, only user stories of the type "new feature" (210 user stories) were considered for building the model because the experts had already defined a standard estimated effort for defects that rarely changes. In the dataset, seven parameters were identified and collected for implementation purposes. The parameters demonstrate the total number of changes (additions and deletions) in LOC, CBO, and WMC of class/es and the number of classes affected when a single user story is implemented. The last parameter is the actual effort spent on finishing the development of the user story. This dataset was used to predict the total effort required to develop a user story in a sprint.

   Output: parameters.

2. Modeling method.

   Input: dataset classified into parameters.

   Process: several modeling methods were considered before opting for GBT. Multivariate linear regression was considered, but there was multicollinearity among the independent parameters, and homoscedasticity was not met either. Moreover, the data were highly right skewed; therefore, the data processing was needed comprehensively to apply multivariate linear regression. As an alternative, GBT regression models were considered, as they are capable of handling conditions such as multicollinearity by themselves. GBT is also an example of ensemble methods that provide more accurate prediction accuracy compared with individual models and hence are more reliable.[37]

   Output: GBT as preferred modeling method.

3. Data normalization.

   Input: dataset classified into parameters.

   Process: it was observed that the dataset was not normally distributed (based on the skewness and kurtosis values). Hence, to make the data normally distributed, logarithmic transformation was applied over the dataset.

   Output: normalized dataset.

4. Model creation.

   Input: normalized dataset.

   Process: the model was implemented using RapidMiner.[38] First, the dataset was divided into a training set and a test set using a 10-fold cross-validation technique. Then, the GBT model was created and applied to the test data. The performance of the model was calculated by using evaluation metrics described in section 8.3. Afterwards, the model was recreated on the whole dataset to be applied on the estimation data gathered during the evaluation workshop.

   Output: GBT model.

8 | EVALUATION WITH INSIDERS TECHNOLOGIES

We conducted a case study with Insiders Technologies to evaluate HyEEASe. The experts used the prototype for estimating tasks for their ongoing project. We also collected their feedback at the end of the session regarding their experience of using HyEEASe.

In this section, we will describe the evaluation approach in terms of evaluation objects, context regarding the case company, and its estimation process, goals, target population, design, execution, results, and analysis.

8.1 | Evaluation objects

The objects of the evaluation were the HyEEASe hybrid estimation method and the GBT model. Firstly, HyEEASe was evaluated by using the prototype in a live iteration at the case company for estimating the tasks. HyEEASe was evaluated for estimation effectiveness and perceived usefulness about the impact information of the potentially impacted code.

Secondly, the GBT model was also evaluated offline for estimation effectiveness. The model parameters were gathered by taking input from the experts at the case company.

8.2 | Evaluation goal

The goal of the evaluation was to understand and evaluate the HyEEASe concerning its usefulness and effectiveness from the perspective of agile development teams in the context of effort estimation in agile software development. Additionally, we wanted to evaluate the effectiveness of the GBT model. The idea, as mentioned earlier, was to analyze whether using a purely expert-based (Planning Poker) or a hybrid (HyEEASe) or a purely model-based (GBT model) estimation improves estimation accuracy. We also wanted to identify needs for improvement to increase the quality of the method and prototype.

8.3 | Evaluation criteria

To evaluate the quality of the prototype, we explored several quality aspects such as usefulness, reliability, relevance, completeness, visualization, and acceptance as shown in Table 2. To quantify these aspects, a five-point Likert scale was used (scale: five-point scale, with 5 being "Strongly agree" and 1 being "Strongly disagree"). We asked the experts to rate the quality aspects after they have used the prototype.

To evaluate the accuracy of the produced estimates, the following accuracy metrics[17] were considered.

- Mean Absolute Error (MAE), which is the average of the absolute errors between the actual and the predicted effort.

$$MAE = \sum_{i=1}^{TT} |AE_i - PE_i|,$$

  where AE_i = the actual effort collected from the dataset for the ith test data, PE_i = the output (predicted effort) obtained using the developed model for the ith test data and TT = total number of tasks in the test set.

TABLE 2 Results of the evaluated quality aspects of the prototype (median values)

Quality aspect	Reference	Result
Usefulness	McKinney[42]	4
Reliability	McKinney[42]	3.75
Relevance	Lee and Strong[43]	3.25
Visualization	Nelson et al[44]	3
Acceptance	TAM[45]	3
Completeness	Goodhue et al[46]	2

- Mean of Magnitude of Error Relative to the estimate (MMER), which is one of the criteria used for evaluating effort estimation models. It is shown that MMER can provide higher accuracy than Mean Magnitude of Relative Error (MMRE).[39,40] MMER is the mean of MER.

$$MMER = \frac{1}{TT}\sum_{i=1}^{TT}\frac{|AE_i - PE_i|}{PE_i}.$$

- Prediction accuracy (PRED) (x), which is the average of MAE's off less than or equal to x as shown in MAE. The accuracy of the estimates directly corresponds to PRED(x) and is conversely relative to MMER.

$$PRED(x) = \frac{1}{TT}\sum_{i=1}^{TT}\left\{\begin{array}{l} 1 \text{ if MAEi} \le x \\ 0 \text{ Otherwise} \end{array}\right\}.$$

These metrics were calculated for the estimates estimated by both the HyEEASe method and the GBT model as soon as we obtained the actual efforts spent on each task at the end of the sprint.

8.4 | Evaluation design—sample and population

The project for which we gathered data and applied the tool for performing estimations has been running since 2000. The current Scrum team includes nine developers working in 2-week sprints. The current software release has approximately one million lines of code and fifty components developed in C++.

We contacted the corresponding agile development team with whom we had collected the data for our method because they were the right stakeholders and also the end users of the method and tool.

Therefore, their feedback is very relevant. Four participants from the team with various roles were able to participate. One of these participants was both the Scrum master and a developer; the rest were developers. The aggregated experience of all the participants was 3.5 years (median value) in agile development and 2.5 years (median value) in estimation.

Since the planned total number of tasks (ie, new features and not defects) for the running sprint were only two, this motivated us to include other already implemented tasks for our evaluation. These tasks were chosen randomly from the database together with our contact person other than the participants. Associated threats are mentioned in section 10. While choosing the tasks, we made sure that they were estimated and/or implemented way back in time and with developers other than the ones participating in the evaluation. It was done to reduce any bias and/or participant's familiarity with the estimated or actual effort. Keeping the duration of the evaluation in mind, two such tasks were selected making a total of four. One was overestimated, and the other was just right. We wanted to analyze how our HyEEASe method and GBT model would perform compared with the initial purely expert-based estimation.

The protocol used in execution includes (1) the procedures for inviting the participants and making arrangements for the fieldwork, (2) the informed consent form allowing us to collect data for the intended analysis and informing the participants about the evaluation as well as their rights, (3) the data collection procedures, and (4) the data analysis procedures.

8.5 | Organizational context

The case company also employs the "Clean Code" policy.[41] According to this policy, no comments or identifiers are added in the code, which meant textual analysis could not be applied for performing impact analysis. Furthermore, based on the argumentation in section 3, a combination of historical and structural static impact analysis technique was selected, adapted, and applied to perform IA.

At the case company, a user story is the highest level of granularity, which is initially estimated roughly. Later, these stories are decomposed into smaller "Tasks," which are then estimated in a Planning Poker meeting and finally implemented. It is for these tasks that impact analysis has been performed. The intention was that the experts use the prototype during Planning Poker and, by consensus, estimate the effort for a given set of tasks. The case company uses a multistage process to estimate a user story. The different stages are as follows.

- Package grooming: in this stage, the team gives an initial rough estimate for a user story so that the product owner (*PO*) can prioritize the user story in the backlog. After that, the user story is included in a future sprint.
- Grooming: in this stage, the requirements for the user story are refined by at least two members of the development team together with the PO.
- Research: in this stage, the user story is researched by at least two members of the development team and is broken down into low granularity substories, called tasks, which can be implemented independently from each other.
- Q&A: in this stage, the developers present the user story to the rest of the team to reestimate the whole story more precisely using Planning Poker. The team then divides the estimate of the story into its corresponding tasks based on the associated complexity and risk.

HyEEASe, a hybrid method, was applied in the Q&A stage where the tasks got their final estimates during Planning Poker.

TABLE 3 Improvement suggestions given by practitioners and corresponding remarks from authors

Suggestions by practitioners	Remarks by authors
General suggestions: in addition to the impact data, link to the changed code of old user stories may also be useful.	Because of clean code policy at the case company, we did not find any comments/identifiers in the code. Therefore, no direct way exists for tracking and linking the changed code with the user story. Nonetheless, some work around could be found in future work, such as tracking the changed lines of code in the version repository against particular commits and user stories against every associated task and every affected class.
Missing features: additional metrics, eg, modified LOC, testing metrics are missing.	The modified LOC is rather a constraint instead as the underlying code analyzer does not support it. However, a work around may be possible in the future to compensate for these missing metrics.
Constraints: expert familiarity and knowledge (concerning prototype and data usage) necessary for increasing productivity.	We agree that to use the prototype, training is required. However, in the prototype, additional help via menus/files can be supported additionally to increase understandability. An upfront investment is required to understand the prototype to increase productivity (a cost-benefit trade-off).

8.6 | Execution and data collection

We conducted an interactive workshop where we first introduced the prototype, and later, the participants used it themselves for estimating efforts for the tasks. To reduce bias and increase the credibility of the empirical data and findings, multiple data sources were used. These sources include observation of the effort estimation process in the evaluation workshop, the questionnaire where the participants rated the quality aspects of the prototype and the moderation cards where the participants provided their open feedback. The execution steps are as follows.

- Introduction: we explained the goals and procedure of evaluation and collected the signed informed consent forms.
- Training: we introduced the idea of using change impact analysis for effort estimation and demonstrated the prototype by explaining its functionality and resolved any open questions.
- Task estimation: in this part of the evaluation, we asked the participants to use the prototype to estimate the effort for four tasks. After considering information provided by the prototype, they estimated the effort for each task. The estimations were stored using the prototype interface. If they had different opinions, they discussed their estimated efforts and reestimated them until they arrived at an agreement. During this activity, we observed the participants, took notes, and answered questions. After considering the information provided by the prototype, they estimated the effort for each task.

  Using HyEEASe, the experts estimated the effort for each task. If they had different opinions, they discussed their estimated efforts and reestimated them until they agreed. During this activity, the participants were observed, notes were taken, and questions were addressed.

  After using HyEEASe, experts provided input to the parameters of the GBT model. The estimates produced by the GBT model were not shown to the experts to keep their estimates (made through using HyEEASe) unbiased.
- Feedback: after the estimation, the experts were provided with a questionnaire to rate several quality aspects as seen in Table 2. To collect their feedback, we then performed another interactive session. We asked the participants to tell us about the strengths of the prototype by writing them on moderation cards. After the cards had been explained and clustered, we collected needs for improvements and documented.

This evaluation workshop lasted 4 hours. During the whole evaluation, two researchers were present; one was the moderator, and the other one was acting as an observer and taking notes. The estimates made through using HyEEASe and the input to GBT model parameters were collected through the prototype, quality aspects rating was collected using a questionnaire and feedback through moderation cards.

8.7 | Data analysis

The input provided by the experts to the parameters of the GBT model was then analyzed. In case of any missing parameters, the impact analysis database was consulted to compensate for the missing values. The model was then applied to this new task dataset offline, and estimates were obtained. The quantitative and the qualitative data were analyzed, compared, and results were integrated. The quantitative analysis was done by using the evaluation criteria described in 8.3. For the qualitative data, themes were derived from the feedback session.

8.8 | Results and discussion

This section presents the results of the evaluation and discusses them simultaneously.

Quality of the prototype: the results of the evaluated quality aspects are shown in Table 2. The practitioners were asked about their level of agreement regarding the quality aspects on a five-point Likert scale.

The values are median values where N = 4. Overall, the participants found the prototype useful, reliable, and relevant for estimation. However, regarding its acceptance, the participants were neutral (median = 3). This might be because during the feedback session, they stated that if the developer is unfamiliar with the underlying system and the prototype, it might be difficult to apply it (for details, see improvement suggestions in Table 3).

Moreover, in their view, the concept does not take into account certain code metrics such as modified LOC or test metrics, which made them disagree on its completeness (median = 2). Here, we must emphasize the constraint that the underlying static code analyzer does not provide these metrics; therefore, we do not have them in this prototype. Nonetheless, we intend to compensate for this lack in our next development iteration.

The summary of the positive feedback with categories is shown in Table 4. In general, the participants appreciated the availability and accessibility of the impact and relevant historical information. They found the prototype useful to support estimation when experts do not agree on a particular estimation. The prototype also visualizes the impact information and shows past estimation trends (over/underestimation). Impact information visualization helps them to learn which other parts of the system they need to consider while estimating.

The summary of improvement suggestions, their categories, and explanations by the practitioners along with remarks from the author addressing the corresponding suggestion are presented in Table 3. The participants suggested that in addition to impact information, it could be useful for estimation if a link to code changes was provided. We appreciate the suggestion; however, as the case company has a clean code policy, ie, no commits/identifiers are added in the source code, it would be difficult to establish such links without this information. However, a workaround is possible. Furthermore, the practitioners had concerns regarding missing metrics in the prototype. For example, they emphasized the importance of having the code metric modified LOC. We agree with their concerns, however, not using these metrics was a limitation of the underlying static code analyzer.

Quality of the estimates: to evaluate the quality (effectiveness/accuracy) of the estimates, we analyzed the effort estimated by the experts using HyEEASe as well as the effort estimated by the GBT model (by applying it offline) and compared it with the actual effort data. The results of this comparison are reported in Table 5. It is clear from the table that MMER and PRED of expert-based estimation using HyEEASe is better than both purely expert-based (Planning Poker) and purely model-based estimation (GBT model). Also for the already implemented overestimated task, the estimates were improved (getting closer to actual effort) when reestimated using HyEEASe. Furthermore, upon sharing the results with the participants afterwards, we were informed that for one of the new planned tasks (T1 in Figure 3) for the running sprint, the team revised their initial purely expert-based estimates (from 1.0 to 1.5 PD) in their Q&A session, based on the impact information they got from our evaluation. During the evaluation, they had estimated T1 as 2.0 PD using HyEEASe where the actual turned out to be 2.3 PD.

This indicates that the HyEEASe method enabled the experts to do better estimations by supplying all relevant and necessary impact information and led to improve accuracy of estimates than the purely expert-based estimation (Planning Poker).

On the other hand, the performance of the GBT model for estimating tasks was not good. One possible reason could be the missing data. The experts could only supply values for the LOC parameter. The missing parameters were provided by taking mean values from our database. This mechanism of providing missing information was perhaps not optimal, hence resulting in the low performance of the model. The comparison of estimated effort with/without HyEEASe and with the GBT model with the actual effort is shown in Figure 3. From the figure and the above-mentioned results, we can conclude that neither the purely expert-based (Planning Poker) nor the pure model-based estimation (GBT

TABLE 4 Positive feedback given by practitioners

Category	Feedback
Visualization	Several kinds of visualization are available regarding the impact set (impact classes). Code structural dependencies are visible through dependency graphs. Visibility of impact. Dependencies of impact classes on other classes are visualized (through charts/tables).
Estimation support	Assists in implementing a change during estimation. Shows errors regarding the estimated effort of the related tasks implemented in the past.
Learning	The prototype enables learning and is useful especially if the experts do not agree on the estimated effort for a user story.
Impact granularity and scope	Shows the potential impact at the class level Enables one to think about which classes may be impacted before implementing the user story.
Access to (historical) data	Availability of huge amount of historical data regarding related tasks and classes. Shows impact factor data on the related tasks and classes. Easy access to past data. Data accessibility on different granularity levels, ie, tasks, class.

TABLE 5 Comparison of estimation accuracy

	Purely Expert-Based effort estimation (Planning Poker)	Expert-based effort estimation using HyEEASe	Purely Model-based effort estimation (GBT model)
Mean Absolute Error (MAE)	0.72	0.22	0.88
Mean of Magnitude of Error Relative (MMER)	1.55	0.23	1.03
Prediction accuracy (PRED) (25)	0.5	0.75	0.5

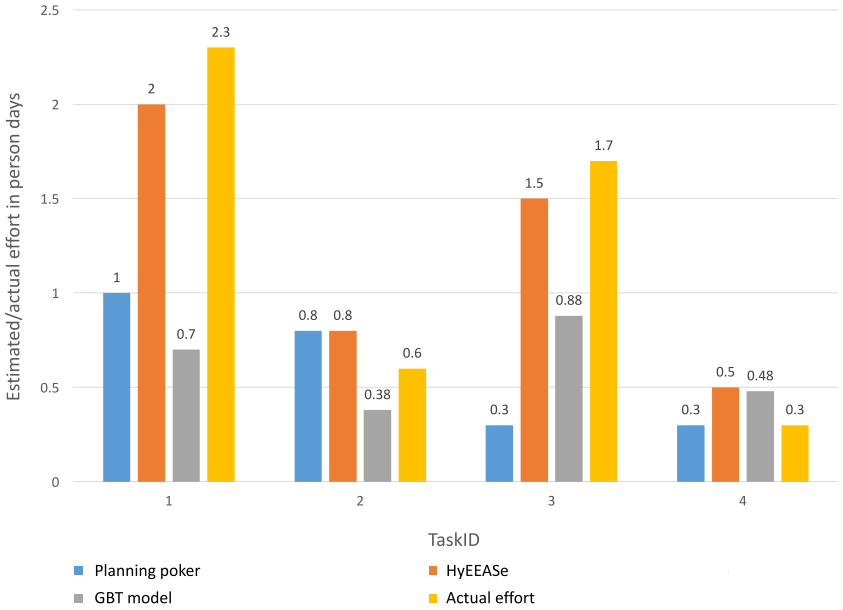

FIGURE 3 Comparison of estimation accuracy—pure expert-based (Planning Poker) vs expert-based using Hybrid Effort Estimation in Agile Software development (HyEEASe) vs pure model-based (gradient boosted trees [GBT] model)

model) improved estimation accuracy. However, expert-based estimation with a support system (HyEEASe) managed to improve estimation accuracy.

These results are in line with results found in the literature concerning model-based estimation methods and their performance in terms of improving estimation. It is established[47] that it is not possible to conclude whether model-based or expert-based models perform better. However, expert estimates seem to be more accurate if domain knowledge is not included in the model or when uncertainty is high. The results also align with our previous case study[8] in which we found that to improve expert-based estimation, decision support is required that provides useful historical data and impact information.

9 | EVALUATION WITH STUDENTS

The previous preliminary assessment at SAP SE[10] and this evaluation at Insiders Technologies GmbH indicated positive results for HyEEASe to improve the accuracy of effort estimation. Therefore, we wanted to introduce the method to software engineering students along with Planning Poker (the most frequently used method in the industry[13]). This also gave us the opportunity to objectively assess the understandability of the HyEEASe method. Students are appropriate participants for this study as the focus is to evaluate the understandability. The premise is that if the description of the method is understandable for students, the same results can be expected for practitioners. The participants in the study were BSc and MSc students taking Software Process and Project Management (SPPM) course at the Department of Computer Science, Technical University Kaiserslautern, Germany. In this study, the understandability of Planning Poker and HyEEASe was compared.

9.1 | Evaluation goal

The goal was to analyze Planning Poker and HyEEASe estimation methods for the purpose of comparison with respect to understandability from the point of view of BS and MS students (novice developers) taking the SPPM course.

9.2 | Evaluation criteria

To evaluate understandability, we explored several quality aspects such as perceived ease of use, usefulness, and learnability of the estimation methods as shown in Figure 4. Using a questionnaire based on TAM,[45] we elicited the subjective opinion of students about these quality aspects. To quantify these aspects, a five-point Likert scale was used (scale: five-point ratio scale, with 5 being strongly agree and 1 strongly disagree). We asked the students to rate the quality aspects on this scale after they have used the estimation methods.

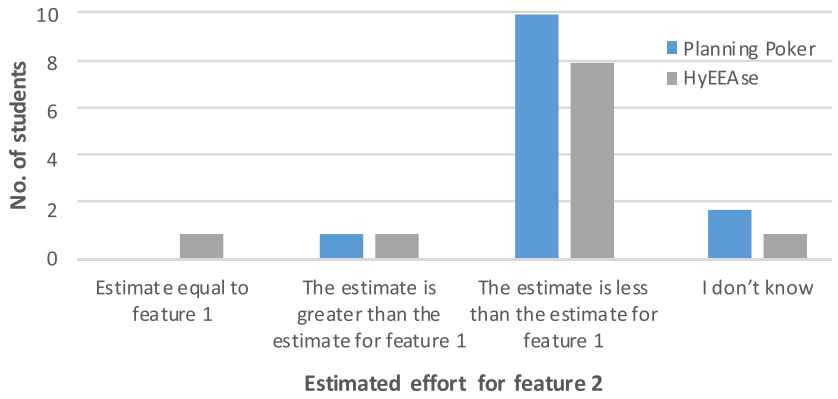

FIGURE 4 Student feedback regarding the understandability of estimation methods

9.3 | Evaluation design and sampling strategy

Students were assigned an estimation method using stratified random sampling. The strata were defined based on the students' experience in software development and awareness of estimation methods. Using a questionnaire, students level of experience and awareness was judged, and experienced/inexperienced students were assigned randomly to two groups. Altogether, 24 students participated, 13 of whom were introduced to Planning Poker and 11 to HyEEASe thus an imbalanced design. Each group had a mix of experienced and inexperienced students making them comparable.

The selection of subjects was based on convenience, ie, students participating in the SPPM course at the university. The students had the freedom to refuse participation, without any penalty for the individual. However, to encourage participation, students were promised 5% bonus on their score in the final exam. To ensure equal treatment (right to service), in a follow-up seminar, the first author presented both methods to students in the course. All students were welcome to attend the lecture regardless of their participation in the experiment.

9.4 | Execution and data collection

Each group of students was given a booklet that included the same introduction to the system, ie, open source system Moodle* and its high-level architecture. Also, the same description of two feature requests was provided to both groups. We choose Moodle as a system since it is highly likely that most of the students are familiar with a learning management systems. Similarly, the two features picked for estimation were related to plagiarism detection and reporting, which is also a familiar concept for the students.

Each group was given a descriptive narrative of how a fictitious team that would use the estimation method to arrive at an estimate for one of the features. A conscious attempt was made to provide an objective, accurate and unbiased narrative of using the two methods. Both descriptions were reviewed internally by the coauthors of this paper and by two senior researchers with expertise in software effort estimation and empirical research. Furthermore, both narratives arrived at the same estimate of (eight story points).

Then, the students were asked to estimate the related feature (feature 2) independently. The students were asked to provide a coarse grain estimate that whether feature 2 will require the same, a less, or greater effort than feature 1 along with a rationale. They were given additional sheets for feature estimation along with rationales, which they handed over afterwards. They were also given the questionnaire to provide their opinion about the understandability of the estimation methods as soon as they finished feature estimation.

We analyzed the quantitative and qualitative data and then compared and integrated the results. For the quantitative analysis, we report descriptive statistics including the median value and number of students giving estimates and rationales in both the groups. For the qualitative data, we derived patterns.

9.5 | Results, analysis, and discussion

Figure 5 shows that a clear majority of the students from both groups picked the right answer, ie, the feature 2 will require less effort to implement than feature 1. Since the students were also asked to justify their choice, we also analyzed their rationales provided in the open text.

By analyzing the open text answers, we identified nine reasons they have used to justify their effort estimate (see Figure 6). One pattern is that students using HyEEASe have stated a broader variety of rationales and also have relied more on objective data to motivate their answer. This suggests that the students using HyEEASe were considering relevant data, as intended by the method when estimating the effort required to implement a feature. The strength of argumentation is seen as an indicator for a better understanding of a concept.[48]

*Moodle: an open-source learning platform https://moodle.org/

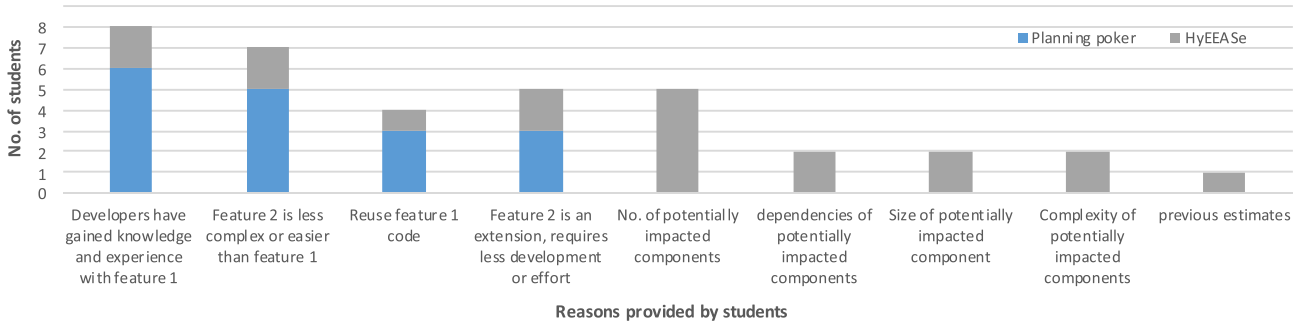

FIGURE 5 Students' estimates of the relative effort required to implement feature 2

		HyEEASe						Planning Poker					
		Strongly disagree	Disagree	Neutral	Agree	Strongly agree	I do not know	Strongly disagree	Disagree	Neutral	Agree	Strongly agree	I do not know
Perceived ease of use	**I think the estimation method…**												
	1… is clear	0	0	4	6	1	0	0	0	2	6	5	0
	2... is understandable	0	0	2	7	2	0	0	0	1	4	8	0
	3... requires a lot of mental effort to apply it	0	5	3	1	2	0	0	2	3	4	4	0
	4… is easy to use	0	3	1	6	1	0	0	2	1	5	5	0
Usefulness	**I think the information provided by change impact analysis is…**												
	… informative	0	0	2	5	4	0	NA					
	… valuable	0	0	2	4	5	0						
	in general, useful for estimating effort for enhanced features	0	0	1	5	5	0						
Learnability	**After using the estimation method …**												
	I learnt about the factors influencing estimation of enhanced features.	0	0	0	9	2	0	NA					
	I am convinced I can *learn over time* , which factors influence estimation of enhanced features.	0	0	0	7	4	0	0	0	1	9	3	0
	I am convinced it will *help me learn fast about* the factors influencing estimation of enhanced features.	0	0	3	5	3	0	0	1	1	11	0	0

FIGURE 6 Reasons provided by students to justify their effort estimate

The results of the subjective assessment of HyEEASe and Planning Poker by the students on "perceived ease of use", "usefulness", and "learnability" indicate positive results (see Figure 4). Please note that the questions about the usefulness of change impact analysis and code related factors did not apply to the group using pure Planning Poker.

Overall, Planning Poker got superior ratings than HyEEASe on both perceived ease of use and learnability. However, students did consider that HyEEASe requires less mental effort. This is partly because of the tool support, that was developed considering the input from two companies, which presents the information in a manner to facilitate effort estimation.

The positive rating of HyEEASe (see Figure 4) and the improved quality of argumentation for justifying the estimates (see Figure 6) also indicate positive results for HyEEASe.

10 | THREATS TO VALIDITY

In this section, we have discussed some threats to the validity of the study and how we tried to mitigate them.

Construct validity: to avoid inappropriate measurement of the evaluation quality aspects, we considered and adapted existing reliable test instruments such as TAM.[45] On the basis of peer reviews and experiences gathered from previous research,[11] we checked the appropriateness of the questions for our evaluation goals. The resulting questionnaire was used to elicit the opinion of practitioners. Only a relevant subset of this questionnaire was posed to students to elicit their feedback about the estimation methods. To avoid mono-method bias, we combined and compared quantitative and qualitative analysis.

Internal validity: to ensure internal validity, only participants with sufficient experience in agile methodologies were approached in the case study. To avoid evaluation apprehension, the practitioners were assured of the anonymity of their personal and organizational data by the informed consent provided. Later, we shared the quantitative and qualitative results with the respective team to get their reflections. To reduce any bias or participant's familiarity with the estimated or actual effort for the two already implemented tasks in the dataset, we made sure that they were randomly selected, were estimated and/or implemented way back in time. Additionally, developers other than the ones participating in the evaluation implemented these tasks.

Regarding the evaluation with students, there could be selection bias as the sample was convenient. We however assigned the students different estimation methods using stratified random sampling, where strata were defined based on their self-assessment of knowledge and experience in agile software development and software effort estimation. To reduce the impact of unfamiliarity with the software system, ie, regarding the instrumentation, we presented the students with an open source system and features and enhancement requests related to it.

External validity: convenience sampling and small sample size for the evaluation with student reduces the statistical generalizability of our results. However, given the consistent positive evaluation from both students and practitioners participating in the study gives confidence that the method is understandable. In case of the evaluation at the case company, given that most commonly used effort estimation method in the industry is Planning Poker and that most companies have version management and issue tracking systems, the results will likely be applicable in most cases. The use of existing data from various information silos also reduces the cost of the intervention for any adopting company.[49] However, to increase the representativeness of our results, we will need to use the method for multiple sprints in the company. More replications in different companies will also improve the generalizability of the results. The general instance of HyEEASe, however, will need to be operationalized in each case company. However, the main limitation of the applicability of the method is that it will not be applicable to completely new development projects. For the evaluation with students, to avoid selection-treatment interactions, replications are required. To mitigate setting treatment interaction, the features for estimating efforts were related to a real open source system.

Conclusion validity: by involving more than one researcher in each step of the evaluation, we increased the reliability of our study. We aggregated the results of the participants and only used median values and the number of practitioners to identify common practices/views and to point out potential future research areas. However, because of the small sample, we could not perform a more rigorous statistical analysis.

To motivate students to participate in the experimental study, all participating students were promised extra points in the form of 5% of their score in the final exam. They were further encouraged to participate since they can learn about the state-of-the-art effort estimation methods. The extrinsic incentive was fairly small, so the students' participation was voluntary, as can be seen that a few students did not sign up for the study. Furthermore, by concealing the goal of the study, by promising confidentiality of responses and by disassociating incentive of participating from the performance in the experiment, we increased the validity of the study.

11 | SUMMARY AND FUTURE WORK

In this study, we reported the results of evaluating a hybrid effort estimation method for estimation accuracy and several quality aspects. We evaluated the use of change impact analysis to support expert-judgement based estimation in industrial settings. Furthermore, we collected feedback from practitioners and students regarding the understandability and ease of use of the proposed method.

We conclude that both students and the practitioners, participating in this study, perceived the overall method and the prototype as very useful for supporting effort estimation. The impact information and the visualizations not only support them during estimation but also enable learning about the system. Furthermore, in the practitioners' view, considering modified LOC and testing metrics among the impact factors could improve the effectiveness of impact analysis for effort estimation.

Regarding estimation accuracy, a purely model-based estimation (GBT model) did not perform better than an expert-based estimation supported by HyEEASe providing additional useful impact information.

In the future, we intend to refine our prototype, taking into account the improvement suggestions we received from the practitioners. On the basis of our evaluation results, we expect practitioners to be able to plan software releases better by making more informed decisions with more transparency during the estimation process.

ACKNOWLEDGEMENTS

This research is funded by the German Ministry of Education and Research (BMBF) project Abakus, grant number 01IS5050G. We express our gratitude to practitioners at Insiders Technologies for their participation in the study. The work of Nauman bin Ali is supported by ELLIIT, a Strategic Area within IT and Mobile Communications, funded by the Swedish Government.

ORCID

Binish Tanveer https://orcid.org/0000-0003-2300-068X
Nauman bin Ali https://orcid.org/0000-0001-7266-5632

REFERENCES

1. Trendowicz A, Jeffery R. *Software Project Effort Estimation - Foundations and Best Practice Guidelines for Success*. Heidelberg: Springer; 2014.
2. Moser R, Pedrycz W, Succi G. Incremental effort prediction models in agile development using radial basis functions. In: Proceedings of the 19th International Conference on Software Engineering & Knowledge Engineering (SEKE'2007); 2007; Boston, Massachusetts, USA :519-522.
3. Jørgensen M, Boehm BW, Rifkin S. Software development effort estimation: formal models or expert judgment? *IEEE Softw*. 2009;26(2):14-19.
4. Moløkken K, Jørgensen M. A review of software surveys on software effort estimation. In: Proceedings of the International Symposium on Empirical Software Engineering, ISESE; 2003; Rome, Italy, Italy:223-230.
5. Nguyen-Cong D, Tran-Cao D. A review of effort estimation studies in agile, iterative and incremental software development. In: Proceedings of the International Conference on Computing and Communication Technologies, Research, Innovation, and Vision for the Future, RIVF; 2013; Hanoi, Vietnam:27-30.
6. Briand LC, Emam KE, Bomarius F. COBRA: a hybrid method for software cost estimation, benchmarking, and risk assessment. In: Proceedings of the International Conference on Software Engineering, ICSE; 1998; Kyoto, Japan, Japan:390-399.
7. Bohner SA. Impact analysis in the software change process: a year 2000 perspective. In: Proceedings of the International Conference on Software Maintenance (ICSM '96); 1996; Monterey, CA, USA, USA:42-51.
8. Tanveer B, Guzmán L, Engel UM. Understanding and improving effort estimation in agile software development: an industrial case study. In: Proceedings of the International Conference on Software and Systems Process, ICSSP; 2016; Austin, TX, USA:41-50.
9. Li B, Sun X, Leung H, Zhang S. A survey of code-based change impact analysis techniques. *Softw Test, Verif Reliab*. 2013;23(8):613-646.
10. Tanveer B, Guzmán L, Engel UM. Effort estimation in agile software development-case study and improvement framework. *J Software Evol Process*. 2017;29(11):e1862.
11. Tanveer B, Vollmer AM, Engel UM. Utilizing change impact analysis for effort estimation in agile development. In: Proceedings of the 43rd Euromicro Conference on Software Engineering and Advanced Applications, SEAA; 2017; Vienna, Austria:430-434.
12. Tanveer B, Vollmer AM, Braun S. A hybrid methodology for effort estimation in agile development: an industrial evaluation. In: Proceedings of the International Conference on Software and System Process, ICSSP; 2018; Gothenburg, Sweden:21-30.
13. Usman M, Mendes E, Börstler J. Effort estimation in agile software development: a survey on the state of the practice. In: Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering, EASE; 2015; Turin, Italy:1-10.
14. Ali NB, Petersen K, Schneider K. Flow-assisted value stream mapping in the early phases of large-scale software development. *J Syst Software*. 2016;111:213-227.
15. Kang S, Choi O, Baik J. Model-based dynamic cost estimation and tracking method for agile software development. In: Proceedings of the 9th International Conference on Computer and Information Science (ICIS) IEEE; 2010; Yamagata, Japan:743-748.
16. Hussain I, Kosseim L, Ormandjieva O. Approximation of cosmic functional size to support early effort estimation in agile. *Data Knowledge Eng*. 2013;85:2-14.
17. Satapathy SM, Rath SK. Empirical assessment of machine learning models for agile software development effort estimation using story points. *Innovations Syst Software Eng ISSE*. 2017;13(2-3):191-200.
18. Ziauddin SKT, Zia S. An effort estimation model for agile software development. *Adv Comput Sci Appl (ACSA)*. 2012;314:314-324.
19. Popli R, Chauhan N. Cost and effort estimation in agile software development. In: Proceedings of the International Conference on Optimization, Reliabilty, and Information Technology (ICROIT); 2014; Faridabad, India:57-61.
20. Bhalerao S, Ingle M. Incorporating vital factors in agile estimation through algorithmic method. *IJCSA*. 2009;6(1):85-97.
21. Raslan AT, Darwish NR, Hefny HA. Towards a fuzzy based framework for effort estimation in agile software development. *Int J Comput Sci Inf Secur*. 2015;13(1):37.
22. Abrahamsson P, Fronza I, Moser R, Vlasenko J, Pedrycz W. Predicting development effort from user stories. In: Proceedings of the International Symposium on Empirical Software Engineering and Measurement (ESEM); 2011; Banff, AB, Canada:400-403.
23. Usman M, Mendes E, Weidt F, Britto R. Effort estimation in agile software development: a systematic literature review. In: Proceedings of the 10th International Conference on Predictive Models in Software Engineering, PROMISE'14; 2014; Turin, Italy:82-91.
24. Fowler M, Highsmith J. The agile manifesto. *Software Dev*. 2001;9(8):28-35.
25. Tanveer B. Guidelines for utilizing change impact analysis when estimating effort in agile software development. In: Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering, EASE; 2017; Karlskrona, Sweden:252-257.
26. Li W, Henry S. Object-oriented metrics that predict maintainability. *J Syst Software*. 1993;23(2):111-122.
27. Chidamber SR, Kemerer CF. A metrics suite for object oriented design. *IEEE Trans Software Eng*. 1994;20(6):476-493.
28. Briand LC, Wüst J, Lounis H. Replicated case studies for investigating quality factors in object-oriented designs. *Empirical Software Eng*. 2001;6(1):11-58.
29. Briand LC, Wust J, Lounis H. Using coupling measurement for impact analysis in object-oriented systems. In: Proceedings of the International Conference on Software Maintenance, (ICSM'99); 1999; Los Alamitos, California:475-482.
30. Antoniol G, Canfora G, Lucia AD. Estimating the size of changes for evolving object oriented systems: a case study. In: Proceedings of the 6th International Software Metrics Symposium (METRICS); 1999; Boca Raton, FL, USA:250.
31. Zimmermann T, Zeller A, Weissgerber P, Diehl S. Mining version histories to guide software changes. *IEEE Trans Software Eng*. 2005;31(6):429-445.
32. SciTools Understand. https://scitools.com/features/. Accessed: 2018-01-14.
33. Leffingwell D, Widrig D. *Managing Software Requirements: A Unified Approach*. Boston: Addison-Wesley Professional; 2000.
34. Gradient Boosted Trees. https://docs.rapidminer.com/latest/studio/operators/modeling/predictive/trees/gradient_boosted_trees.html. Accessed: 2018-01-14.
35. Nassif AB, Capretz LF, Ho D, Azzeh M. A treeboost model for software effort estimation based on use case points. In: Machine Learning and Applications (ICMLA), 2012 11th International Conference on, Vol. 2 IEEE; 2012; Washington, DC:314-319.

36. Satapathy SM, Acharya BP, Rath SK. Class point approach for software effort estimation using stochastic gradient boosting technique. *ACM SIGSOFT Software Eng Notes*. 2014;39(3):1-6.
37. Elish MO, Aljamaan H, Ahmad I. Three empirical studies on predicting software maintainability using ensemble methods. *Soft Comput*. 2015;19(9):2511-2524.
38. Gradient Boosted Trees. https://rapidminer.com/. Accessed: 2018-01-14.
39. Foss T, Stensrud E, Kitchenham B, Myrtveit I. A simulation study of the model evaluation criterion MMRE. *IEEE Trans Software Eng*. 2003;29(11):985-995.
40. Kitchenham BA, Pickard LM, MacDonell SG, Shepperd MJ. What accuracy statistics really measure. *IEE Proc-Software*. 2001;148(3):81-85.
41. Martin RC. *Clean Code: A Handbook of Agile Software Craftsmanship*. New York: Pearson Education; 2009.
42. McKinney VR, Yoon K, Zahedi F. The measurement of web-customer satisfaction: an expectation and disconfirmation approach. *Inf Syst Res*. 2002;13(3):296-315.
43. Lee YW, Strong DM. Knowing-why about data processes and data quality. *J Manage Inf Syst*. 2003;20(3):13-39.
44. Nelson RR, Todd PA, Wixom BH. Antecedents of information and system quality: an empirical examination within the context of data warehousing. *J Manage Inf Syst*. 2005;21(4):199-235.
45. Venkatesh V, Bala H. Technology acceptance model 3 and a research agenda on interventions. *Decis Sci*. 2008;39(2):273-315.
46. Goodhue DL, Thompson RL. Task-technology fit and individual performance. *MIS Q*. 1995;19:213-236.
47. Jørgensen M. A review of studies on expert estimation of software development effort. *J Syst Softw*. 2004;70(1):37-60.
48. Ali NB, Unterkalmsteiner M. Use and evaluation of simulation for software process education: A case study. *Proceedings of the European Conference Software Engineering Education (ECSEE)*;2014; Seeon Monastery, Bavaria, Germany: Shaker Verlag;59-73.
49. Ali NB. Is effectiveness sufficient to choose an intervention?: Considering resource use in empirical software engineering. In: Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM 2016, Ciudad Real, Spain, September 8-9, 2016; 2016; Cuidad Real, Spain:54:1-54:6.

How to cite this article: Tanveer B, Vollmer AM, Braun S, Ali Nb. An evaluation of effort estimation supported by change impact analysis in agile software development. *J Softw Evol Proc*. 2019;31:e2165. https://doi.org/10.1002/smr.2165