SETAP: Software Engineering Teamwork Assessment and Prediction Using Machine Learning

Dragutin Petkovic[1], Marc Sosnick-Pérez[1], Shihong Huang[2], Rainer Todtenhoefer[3], Kazunori Okada[1], Swati Arora[1], Ramasubramanian Sreenivasen[1], Lorenzo Flores[1], Sonal Dubey[1]

[1]Department of Computer Science
San Francisco State University
San Francisco, U.S.A.
{petkovic, msosnick, kazokada, ltflores}@sfsu.edu, {sarora, sreeni, sdubey}@mail.sfsu.edu

[2]Department of Computer Science and Engineering
Florida Atlantic University
Boca Raton, U.S.A.
shihong@fau.edu

[3]Department of Applied Computer Science
University of Applied Science, Fulda
Fulda, Germany
rainer.todtenhoefer@informatik.hs-fulda.de

***Abstract*— Effective teaching of teamwork skills in local and globally distributed Software Engineering (SE) teams is recognized as an important part of the education of current and future software engineers. Effective methods for *assessment* and early *prediction* of learning effectiveness in SE teamwork are not only a critical part of teaching but also of value in industrial training and project management. This paper presents a novel analytical approach to the assessment and, most importantly, the *prediction* of learning outcomes in SE teamwork based on data from our joint software engineering class concurrently taught at San Francisco State University (SFSU), Florida Atlantic University (FAU) and Fulda University, Germany (Fulda). Our approach focuses on assessment and prediction of SE teamwork in terms of ability of student teams to apply best SE processes and develop SE products. It differs from existing work in the following aspects: a) it develops and uses only objective and quantitative measures of team activity from multiple sources, such as statistics of student time use, software engineering tool use, and instructor observations; b) it leverages powerful machine learning (ML) techniques applied to team activity measurements to identify quantitative and objective factors which can assess and predict learning of software engineering teamwork skills at the team level. In this paper we provide the following contributions: a) we present in detail for the first time the full team activity measurement data set we developed, consisting of over 40 objective and quantitative measures extracted from student teams working on class projects; b) we present a ML framework which applies the Random Forest (RF) algorithm to the team activity measurements and team outcomes, focusing on predicting teams that are likely to fail; c) we describe in detail our now fully implemented and operational data processing pipeline, consisting of data collection methods from multiple sources, ML training database creation, and ML analysis subsystems; and finally d) we present very preliminary results of ML analysis results based on the data from our joint software engineering classes in Fall 2012, and Spring 2013, with the data from 17 student teams. While our ML training database is currently small, it continuously grows. Our preliminary results, verified with two independent accuracy measures, show that RF is able to predict SE Process and SE Product team performance in intuitively explainable manner.**

Keywords—Assessment, Software Engineering Teamwork, Machine Learning, Education

I. INTRODUCTION

Modern software development involves intensive, often globally distributed teamwork, with teams being required to develop easy-to-use, maintainable software on schedule and on budget, satisfying detailed specifications. The need for improved teaching, industrial management, and training in software development is evidenced by the unacceptably high incidence of failure of industrial software projects: about 9% are abandoned, about a third fail, and over half experience cost and schedule overruns [1-5]. The research also indicates that these failures stem primarily from failures in non-technical aspects of software engineering (SE) such as communication, organization and teamwork [1][4-8]. Therefore, there is a critical need to develop methods to effectively teach and assess teamwork skills. Given today's distributed, global SE development environment, it is also important that these methods take into account locally and globally distributed SE teams. Effective methods for *assessment* and early *prediction* of learning effectiveness in SE teamwork are not only a critical part of learning and teaching, but also of value in industrial training and project management.

Most existing approaches to assessing achievement of SE teamwork skills are based on *qualitative* and *subjective* data captured via surveys taken at the end of the SE class, with only rudimentary data analysis techniques applied to the collected data (for example [9-10]). These qualitative surveys may include questions such as "rate the impression of your SE teamwork experience in the class", etc. While these approaches are very worthwhile, they are in general lacking in the following ways: a) qualitative and subjective data used in these methods are not precise nor amenable for sophisticated data analysis; b) detailed user behavior and "meta-data" readily available from today's SE tools, such as software development tool usage and communication patterns among the team members are rarely used; c) the more complex decision methods such as machine learning (ML) [11] that are widely used today in many applications ranging from medicine, marketing, analysis of customer and user behavior (e.g. on-line shopping [12]) to SE problems such as software reuse and evolution management [13-14] are rarely used; d) methods for

This research is sponsored by National Science Foundation Transforming Undergraduate Education in Science Grant #1140172.

early prediction of SE team failure are rarely developed and difficult to implement when data collection through class surveys is performed only at the end of the class.

Our work focuses on assessment and prediction of SE teamwork learning of teams, not of the individual students that make up those teams. We define the learning of SE teamwork as an ability of a team: (i) to learn and effectively apply *SE processes* in a teamwork setting, and (ii) to develop software that satisfies defined requirements. We therefore assess not only the quality of the team's output (i.e. the software product), but also on the team's ability to follow best practices in SE teamwork. Since we study the behavior and communication of the team as a whole during the semester, and not prior personal and communication profile of each student, special attention has been devoted to team selection. Our focus on the team instead of the individual has also been motivated by the observation that team cohesion and communication are critical in the success of SE team projects.

II. Our Approach

Our approach to assessment and, most importantly, *prediction* of learning of SE teamwork consists of three basic steps [22]:

Step 1: Collect the objective and quantitative data on student team activity during joint SE classes at SFSU, FAU and Fulda

Step 2: Create a ML training database comprising student team activity measurements, instructor observations, and grades for student SE teamwork achievement and product quality

Step 3: Apply random forest (RF) ML method to the data to discover models and factors that determine and predict SE teamwork achievement and product quality of student teams.

Our approach is novel in that it: a) develops and uses only objective and quantitative measures from multiple sources such as statistics of student time use and SE tool use, counts of emails exchanged and of issues needing attention etc.; and b) applies powerful ML techniques which uses these quantitative and objective measures of teamwork activity to assess *and predict* team's achievement in learning and applying of SE teamwork skills. Our analysis is focused on teams and their communication and behavior dynamics during the SE project development.

We chose the RF ML approach [15-16] for its accuracy, ease of implementation, availability as open source software, and for its ability to rank variables in terms of their predictive power, which can illuminate the most important factors for assessment and prediction.

III. Organization of the Joint Software Engineering Class

Our research is integrated and critically dependent on a teamwork intensive, globally distributed SE class taught at San Francisco State University (SFSU), Florida Atlantic University (FAU) and Fulda University in Germany, which has been ongoing since 2006 [17-21]. This class provides an environment where student teams are "embedded and observed" in as realistic a project and teamwork development environment as possible, thus providing realistic data for the research.

TABLE I. Process and Product Grading Rubrics Used in Determining Outcomes and ML Classes A and F.

	SE Process Grading Rubrics
1.	Team participation at the meetings
2.	Quality and timing of follow-up on outstanding issues
3.	Ability to deal with feedback constructively
4.	On time delivery of software on non-software items
5.	Quality and completeness of non-software deliverables such as website design, documentation, and milestone docs.
6.	Number and severity of teamwork issues in which the instructor had to intervene.
7.	Ability to apply best SE and teamwork practices as taught in class and as advised
8.	Ability to effectively use collaborative software development and communication tools
	SE Product Grading Rubrics
1.	Correctness and reliability of operation
2.	Functionality actually delivered vs. team commitment
3.	Ease of use, user interface
4.	Performance
5.	Architecture
6.	Database design
7.	Code quality and comments
8.	Presentation and effectiveness of final demo

Our SE class now involves about 140 students each year, working in 25 to 30 teams of 5 to 6 students each, which is jointly and concurrently taught at SFSU, FAU, and Fulda in Fall semesters and additionally at SFSU in Spring semesters. The exact number of teams varies with class enrolment and is growing.

A. Term Project and Grading

During the class, all student teams develop the same web application, with mandatory use of a suite of modern SE development and communications tools. Starting with only a single page, high-level description of the product, student teams develop their application in five well-defined milestones:

1. M1: high level requirements
2. M2: detailed requirements and specification
3. M3: prototype development and review
4. M4: beta release
5. M5: final delivery and demo

Student team composition may be *local* (comprising students from the same school) or *global* (comprising students from each of SFSU and FAU or SFSU and Fulda schools). Teams meet weekly in class for a mandatory meeting with instructors, and are expected to meet independently outside of class. Teams (especially global teams) may use Skype or Google Chat for meetings outside of class. Observations of student teams are made and recorded by instructors during the in-class meeting, for the components of *SE product* and *SE process* as described in more details in TABLE I.

To ensure teamwork culture and student commitment, all members of a student team share the same grade for the SE process and SE product components, which each contribute 25% to the student's overall class grade. To ensure that students quickly learn the class SE tools, an individual milestone, M0, is instituted early in the class, requiring students to install and learn the tools through the development of a small example application. M0 is worth 5% of a student's grade. A comprehensive final exam testing students' knowledge of class material contributes the remaining 45% to a student's grade.

Great effort has been made to smoothly integrate the research and its related data collection with the SE class teaching and grading. Class teaching is "just in time" i.e. teaching topics are offered at the time when students need them for project milestones. For about one hour at the end of each class, instructors meet with student teams where the instructors observe, advise and record their observations in instructor observation logs (IO). Students complete a weekly time card survey (WTS), which is used to collect "time spent" information (e.g. time spent on meetings, coding, documentation). It is made very clear to all students that no information collected or derived from this research influences student grade. Students are given the choice to participate in the research study or not, and those who choose to participate in the study sign informed consent documents. Students who choose not to participate are grouped together into a team, and the entire team's data is discarded for that semester. To assure strict adherence to student privacy, analysis is done and published only at the aggregated team level from *team activity measurements* (TAM).

The *SE process* component of the outcomes of student teamwork learning is graded by instructors reviewing observation logs and student project documentation, using the rubric in TABLE I. to evaluate proper adherence to SE processes by the student team and its members. The *SE product* component of the learning outcomes is graded both by instructors and by independent observers, who use the rubric in TABLE I. to evaluate the quality of the team's final product. Each team receives an absolute score in points, and is also ranked relative to the other teams in the same class for that semester. Following grading and ranking, for the purpose of this research, the instructors classify each team's SE process and SE product achievements into two ML classes: *at or above expectations* receives class label A, and *below expectations or needing attention* receives a class label F.

B. Team Organization

In order to focus our analysis only on factors influencing team success exhibited during the class and minimize the influence of an individual student's experience and skills developed prior to the class, it is critical to form student teams with approximately the same overall distribution (mix) of skills and experience. While we note the importance of prior student personality profile, we focused only on the student SW and team experience in composing the teams. To minimize influence of personality and communication issues of each student we used mentoring and coaching during early stages in the class. The decision not to use student personality profiles was also motivated by concerns of cost and efficiency of obtaining the profiles given a tight class management schedule and by privacy implications. We do however pay considerable attention to communication and personality profile in choosing team leads.

We have developed, and recently improved and formalized, the following process for student team selection:

- A Team Placement Survey (TPS) is administered to all students at the start of the class. This survey comprises 17 graded (Lickert) scale, Y/N questions, and a small programming proficiency test. In the TPS, the student is asked about their prior product development and teamwork experience, GPA, gender, etc. The student is asked to self-rate on a scale their proficiency in various programming languages used in the class. Finally the TPS includes 3 simple programming proficiency tests on the languages that will be used during the semester.
- Each TPS is rated by the weighted sum of responses to questions and instructor grading of the programming tests to determine *student skill scores* for each student. Teams are formed such that *team skill scores,* obtained by averaging student skill scores are approximately equal.
- Global teams are formed primarily from students who have volunteered to be on a global team.
- Each team is asked to recommend a team lead, who is evaluated and must be approved by instructors. Global teams have a team lead in each participating school.

IV. Collection of Data On Team Activity

Team activity measurements (TAM) are computed for each team by aggregating individual *student activity measurements* (SAM) of participating team members.. The design of TAM and SAM was motivated by the real-world experience and intuition gained from teaching joint SE class for several years, trying to formalize and understand how to better assess and predict student teamwork learning.

The challenge was to choose only objective and quantifiable measurements (e.g. time spent, counts of events/issues, tool usage like e-mails, postings, etc.) suitable to more advanced data analysis techniques like ML and motivated by our teaching and teamwork evaluation experience. For example, it was observed that teams who struggle to establish communication early on tend to fail more often, so we measure time spent in meetings and collect statistics about e-mail usage. We observed that teams writing poor software repository commit messages, such as messages that are empty or repeated, tend to produce a lower quality software product, so we measure the percent of unique commit messages to the code repository. We have also noticed that teams completing assignments late or having a high number of teamwork-related issues tend not to do well, so we measure both percent of late delivery and the count of issues requiring instructor intervention. To measure dynamics in time and within the team, we compute standard deviation of certain measures such as e-mail usage and repository commits over time in weeks,

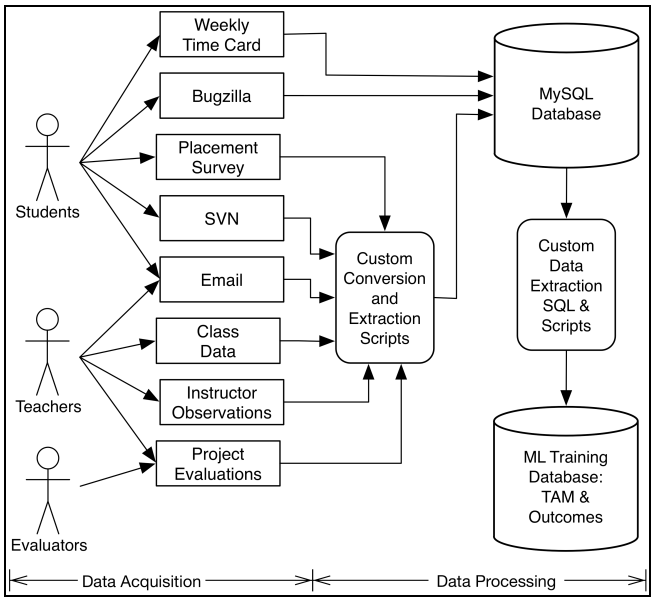

Fig. 1. Dataflow for the data acquisition and data processing phases of the SETAP project, which results in the creation of the ML training database.

and over team members. These measures are intended to help reveal cases such as a student doing most of the work in a team instead of the work being evenly distributed among the team members. We have also conjectured that TAM predictive importance and student team dynamics may change during the course of project development (i.e. in each milestone), hence we collect TAM data separately for time intervals corresponding to five project milestones as well as for the whole class period.

A. Data Collection Methods

Fig. 1 depicts the data collection process, starting from collecting SAM measures, and using several methods such as:

Mandatory student Weekly Timecard Surveys (WTS) collect information from each student about the time the student spent during the week on coding, meeting, teamwork, etc. It is difficult to collect this data by other means without the use of significantly more invasive collection methods. We are aware that students might enter somewhat erroneous numerical estimates, but we found that, when averaged over all the team members, these estimates are reasonably reliable when used in ML analysis. We emphasize to students that they are not graded on the *content* of their WTS, only on the *submission* of the WTS.

- *Tool Logs (TL)* collect statistics/counts of individual student usage of SE communication and development tools, such as the number of e-mails between the team members, number of postings to source code repository, quality of commit messages.
- *Instructor observation logs* (IO) of team activity such as team participation, number of issues requiring instructor intervention, number and percent of issues closed on time, etc., are also recorded weekly and included in the calculation of the TAM. Other basic data such as semester, team lead gender, etc., are collected from Class Data (CD).

Time stamps are kept for all data, since further analysis is performed in selected time intervals, T*i*, which correspond to the five predefined milestones: T1 - start to end of Milestone 1; T2 – start to end of Milestone 2 and so on; T6 is composed of the data from T1 through T5, the dataset for the entire semester.

Once collected, SAM data for team members are aggregated with other data into TAM for each team and are computed for each time interval T1 to T5 and T6. It is only TAM data that are used for ML analysis, and no analysis is performed on any data that may individually identify a student.

TABLE II. provides a description of rows 1-47 of the TAM, the method of extraction of the data points, and the name and brief description of the each TAM data item.

B. Data Collection Infrastructure

Teacher, student, and team accounts, software development and communication tools, and data extraction software are hosted on a virtual Ubuntu server hosted in the Amazon cloud, which we call the SETAP server. Each student and each student team is provided a Unix shell account on this server in which to develop their project; the student team's final project must be served from their group server account. The student team projects are developed for a LAMP (Linux, Apache, MySQL, PHP) stack, which is provided on the system. The server provides SE tools, which the students are required to use during project development, including Subversion (http://subversion.apache.org), and Bugzilla (http://www.bugzilla.org). Students are provided email service through the class server, and are required to communicate with their team and instructors with it. A custom web interface has been developed for the creation and management of these student and group services.

For processing of student email and Subversion activity, custom PHP and shell scripts capture and convert data from these tools' system logs and store the data to the central MySQL (http://www.mysql.com) database. WTS are sent out every week of the semester by an automated script, which also notifies instructors of survey non-responders. The surveys are administered to students using LimeSurvey [25] via a web interface. Both LimeSurvey and Bugzilla write directly to the database, so no additional data conversion is necessary. At the beginning of the semester, class data (CD) are entered into an excel spreadsheet. The spreadsheet is uploaded to the system through a custom web interface. The CD are then converted and stored to the database by custom scripts. Placement survey data, instructor observation data, and project evaluations are done on paper, and manually entered into a spreadsheet; these spreadsheets are then uploaded to the system via a custom web interface, where custom scripts convert the spreadsheet data and store the information to the database. Strict quality control (code reviews, testing, manual data checking etc.) of data

TABLE II. ML TRAINING DATABASE. TAMS (ROWS 1-47) AND SE PROCESS AND SE PRODUCT OUTCOMES (ROWS 48-53). MEANS OF COLLECTION: CD – CLASS DATA; WTS – WEEKLY TIMECARD SURVEYS; IO – INSTRUCTORS' OBSERVATIONS; TL – TOOLS LOGS

Row	Means	*Measure:* Description
1	CD	***Year:*** Year the measures collected
2	CD	***Semester:*** Semester the measures collected
3	CD	***Time Interval****:* Time interval for which measures are collected (i.e. milestones 1-5 or complete semester)
4	CD	***Team number:*** ID of the team
5	CD	***Team member Count****:* How many students in the team
6	CD	***Percent of female team members****:* Percent of female student members in the team
7	CD	***Team Lead Gender:*** Gender of the team lead
8	CD	***Team Distribution:*** Local (from the same school); Global (from different schools)
9-11	WTS	***Time spent on meetings:*** Average per student/week, standard deviation (SD) over weeks, SD over team members
12-14	WTS	***Time spent on non-coding tasks:*** Average per student/week, SD over weeks & team members
15-17	WTS	***Time spent on coding tasks:*** Average per student/week, SD over weeks & team members
18-19	WTS	***Lead admin time:*** Average time and SD over weeks the team lead (local or global) spent management tasks
20-21	WTS	***Global Team Lead Admin Time:*** Average time and SD over weeks spent on admin for global portion of teamwork
22-24	WTS	***Time students spent helping other team members:*** Average per student/week, SD over weeks & team members
25-26	IO	***Meeting participation:*** Average per team, SD over weeks of percent of team members being present at in-class scrums
27	IO	***Percent of late deliverables:*** Percent of deliverables (e.g. documentation, programs) not delivered on time.
28	IO	***Instructor teamwork intervention count:*** Count of instructor process intervention for teamwork issues
29-31	TL	***Number of e-mails within a team:*** Average per student/week, SD over weeks, SD over team members
32-33	TL	***Number of e-mails by team lead to team members:*** Average per week, SD over weeks
34-36	TL	***Number of commits to code repository:*** Average per student/week, SD over weeks, SD over team members
37-39	TL	***Length of commit message to code repository:*** Average per student/week, SD over weeks; SD over team members
40-42	TL	***Uniqueness of commit messages:*** Average percent per student/week, SD over weeks, SD over team members
43-45	TL	***Number of commit files changed:*** Average number per student/week, SD over weeks, SD over team members
46	IO	***Total number of instructor initiated issues:*** Count of instructor requested formal response to an issue/checkpoint
47	IO	***Percent of late issue responses:*** Percent of responses to issues that were late
48-50	IO	***SE Process Outcome Grades:*** Letter (A or F), percentile, and class rank of team's SE process outcome grades
51-53	IO	***SE Product Outcome Grades:*** Letter (A or F), percentile, and class rank of team's product outcome grades.

collection SW and the data itself has been applied during development to ensure accuracy of collected data.

V. CREATION OF THE MACHINE LEARNING TRAINING DATABASE

In the final part of the data collection and processing phase, data is extracted from the database by custom scripts and SQL queries, which organize and extract the TAM data and instructor evaluations into a ML training database. The ML training database is composed of TAM data for each team paired with ML class labels A or F for each of *SE process* and *SE product (*A for teams *at or above expectations* and F for teams *below expectations or needing attention*) to constitute feature vectors in ML training database. This ML training database is constructed at the team level with no individual student data and in addition, to protect student privacy, this database contains no individually identifiable student information.

The complete ML training database shown in TABLE II. consists, for each team, of: a) the *Team Activity Measurements* or TAM (rows 1-47), and b) SE process and SE product "ground truth outcomes" (rows 48-53) determined as explained above. Currently, our ML training database contains carefully vetted information from 17 teams: 11 teams participated in the Fall 2012 semester, and 6 teams participated in Spring 2013 (SFSU only, no global teams). In total, for SE Product there were 12 teams classified as A and 5 as F, and for SE Process 8 teams were classified as A and 9 teams classified as F. We note a slight imbalance between A and F classifications for SE Product. Our ML analysis procedure has been designed to address this imbalance.

VI. APPLYING RANDOM FOREST MACHINE LEARNING

Our goals are to develop optimal predictive RF models which, based on TAM data, best predict occurrence of teams graded F for SE product and SE process learning outcomes separately. In the future, using built-in RF variable importance functionality, we will work on discovering which TAM data has the highest predictive power for class F, thus indicating factors which most influence the learning of SE teamwork. The key challenge in this early phase is the small size of the training database where each item corresponds to one student team completing our class, which is limited by the number and enrollment size of classes we can effectively teach. With time this database will grow at an expected rate of 25 to 30 teams per year. Another challenge from our focus on predicting teams labeled F inspired the exploration of alternative accuracy measures for RF predictor than the commonly used Out of Bag Error (OOB) [15], which averages misclassification error for all class labels (e.g. A and F). We also note our somewhat unbalanced training data, where the class of interest (F) may constitute a minority; we believe this problem might persist with more data collected.

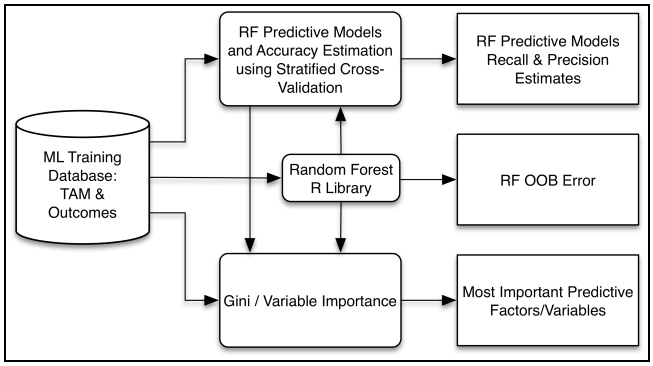

Fig. 2. Data flow of the ML portion of SETAP project.

Given our objectives, and in light of the above challenges, we have developed ML analysis methods using RF as depicted in Fig. 2. For accuracy measures, given our focus on class F, we use *recall and precision* in detecting class F as our primary accuracy measure. *Recall* is the number of samples correctly classified as class F vs. all existing samples of class F. *Precision* is the number of samples correctly classified as class F vs. all samples classified as F. To estimate the recall and precision in the presence of unbalanced training data we use stratified sampling in cross validation (CV) so that each class is fairly represented in every fold of a CV process. To make our predictions practical (i.e. having good prediction with minimal false predictions), we are specifically interested in finding maximal recall for class F at a fixed high precision (e.g. 90%). To account for different dynamics in each SE class milestone, we perform prediction in different time intervals T1-T5 corresponding to each milestone M1-M5, and T6 corresponding to the whole class period. We also wish to investigate if predictors from earlier time intervals (T1-T3) can be used for early *prediction* of teams which later fail, enabling earlier instructor intervention.

As a secondary accuracy measure, and to check for consistency, we use above-mentioned Out of Bag Error (OOB) common with all RF implementations. This measure is computed in same early intervals T1-T3.

Specific questions we want to answer are:

- Can RF predict teams labeled F in SE Process and SE Product with sufficient accuracy (indicated by high recall and low OOB)?
- In which of early intervals T1-T3 does RF achieve the best and sufficiently accurate prediction?
- Do recall and OOB accuracy measures indicate best predictions in the same time intervals?

As depicted in Fig. 2, the ML portion of the system consists of the combination of custom Unix shell, Perl and Python scripts, which use the randomForest package run by the R statistical computing package [26-27]. Again, to ensure accuracy the code and the data have undergone multiple reviews by experienced programmers, and tested on a synthetic test data set with known results. We also leveraged our experience and cross validation code from a project where we used RF technology on bioinformatics data [28] and tested and compared results on two independent ML systems.

A. Derivation of Random Forest Predictive Models

The derivation of predictive RF models and estimation of accuracy in predicting teams classified as F is accomplished by training RF on a ML training database to find the best RF models, or predictors, for class F. The process of optimizing RF consists of finding the optimal values of RF accuracy (i.e. recall/precision or OOB) for the number of decision tress (*ntree*) to be included in the RF ensemble (forest), and the number of variables (*mtry*) to be evaluated at each tree node during training, which maximizes the recall measure at the desired 90% precision. The range and interval of grid search

```
Loop over ntree (1000, 50000)
    Loop over mtry (2, 4)
        Loop over 3-fold stratified cross-validation (K=1..3)
            Train RF on data from two folds, test on third fold
            Vary RF cutoff to achieve 90% Precision for class F,
                Record Recall for class F
        Average Recall measures for class F over 3 folds
```

Fig. 3. Pseudocode for RF optimization using CV

for *ntree* and *mtry* follows recommendations from literature, given the nature and size of our training database. Due to the very small training database and a relatively large number of TAM measures, to avoid over-specification we currently use {2, 4} for *mtry* and to provide for enough random tress given large number of TAM data we use {1000, 50000} for *ntree*. To handle our unbalanced data set, we adapt a 3-fold stratified cross validation as recommended in [23-24]. 3-fold stratified cross validation first partitions the training database into larger training and smaller test sets, consisting of 2/3 and 1/3 of the data, respectively. These partitions are constructed so that samples labeled A and less frequent samples labeled F participate proportionally in each partition. Training and testing RF for each partition is followed by computing recall at precision of 90% for class F by varying the RF decision threshold to achieve the desired (90%) precision. Finally these recall values are averaged over 3 different repetitions of cross validation. The pseudo code for this process is in Fig. 3.

For OOB error estimates we used standard open source RF implementation from randomForest library [26-27] using the same range for *ntree* and *mtry* values as above.

The template is used to format your paper and style the text. All margins, column widths, line spaces, and text fonts are prescribed; please do not alter them. You may note peculiarities. For example, the head margin in this template measures proportionately more than is customary. This measurement and others are deliberate, using specifications that anticipate your paper as one part of the entire proceedings, and not as an independent document. Please do not revise any of the current designations.

Derived predictive models for class F consist of an ensemble of decision trees provided by the RF implementation (in our case the randomForest R library [26-27]) for optimally determined *ntree* and *mtry* values as above, and are derived for each interval T1-T3, and separately for SE process and SE product prediction.

VII. Results and Discussion

The system (methods and software implementation) for data collection and creation of the ML training database is fully operational and deployed for continuous data collection for three concurrent SE classes at SFSU, FAU and Fulda with over 140 students and 25 to 30 student teams each year. This system is fairly complex and took about 3 years to develop and test. ML analysis system (algorithms, software) based on RF is fully operational as well.

In this first step, we have created a ML database with over 40 TAM measures for each of 17 student teams from the period of Fall 2012 and Spring 2013 including graded

outcomes of their achievement of SE Process and SE Product learning. The data has been carefully checked for accuracy due to complexity of collection process and large number of parameters. Currently, we are organizing, checking and adding the data from additional 25 teams from our joint SE class in Fall 2013.

We note the current small size of our current training database. Our approach was to first establish the "proof of concept", test and prepare the software infrastructure for more data and then embark on more data processing and analysis.

We performed two kinds of ML experiments by computing: recall and precision for detecting class F, and standard RF OOB error of misclassification in intervals T1-T3, for SE Process and SE Product separately.

To estimate RF prediction for class F with recall and precision, we used 3 fold stratified cross validation, looking for best RF predictor for class F both for SE process and SE Product. We focused on early millstones M1, M2, M3 (the time intervals T1, T2, T3) looking for early predictions so that class intervention can be effectively implemented. We

TABLE III. BEST RF PREDICTORS FOR SE PROCESS

Time period (milestone)	Recall at 90% precision	Ntree	mtry
T2 (Milestone 2)	0.667	1000	2
T2 (Milestone 2)	0.667	1000	4
T2 (Milestone 2)	0.667	50000	4

TABLE IV. BEST RF PREDICTORS FOR SE PRODUCT

Time period (milestone)	Recall at 90% precision	Ntree	mtry
T3 (Milestone 3)	0.6	50000	4

measured recall for class F (teams that need attention or are below expectations) at 90% precision. Results are in TABLE III. and TABLE IV.

To assess RF performance using OOB as an accuracy measure, we performed the same set of experiments as above (same *ntree*, *mtry* and time intervals) using the radomForest software library from R statistical computing [26-27]. Significantly, OOB results are consistent with recall and precision, namely they provide best predictions in the same intervals T*i* as recall measure both for SE Process and SE Product. Specifically: a) for SE Product the best OOB predictor was again in interval T3 with an OOB of 18% for ntree=1000 and mtry=4, detecting 3 out of 5 teams F; b) for SE Process the minimal OOB of 24% for ntree, mtry combinations (1000, 2; 1000,4; 50000,2; 50000, 4) was again in interval T2, detecting 6 out of 9 teams F.

These results have an intuitive explanation. It makes sense for T2 (comprising Milestone 2 – the detailed specification phase) to be the best predicting interval for teams' SE Process performance since in that milestone teams are at the peak of establishing communication and teamwork while developing many non-coding deliverables. Similarly, it makes sense that T3 (comprising Milestone 3 – the prototyping phase) is the best interval in predicting SE Product teams' performance because this is when the teams start implementing their project. In terms of recommendations for educators and managers these results indicate intervals when most attention has to be paid to identifying teams that could fail.

We note again a very preliminary nature of results but we conjecture that "proof of concept" has been validated with both recall and OOB measures agreeing, and with the intuitively justifiable interpretation of the results as above

Some bias in determining classes A and F is still present since these grades are partially determined by class instructors, an issue hard to eliminate completely. To mitigate this we created as objective grading rubrics as we could and we also involved external graders for SE Product grade.

We also recognize that we are simultaneously collecting activity measures from student teams as the students are learning and being coached, resulting in a "non-stationary observed process". We mitigate this by careful recoding of team behavior with time stamps and whereby our more aggressive coaching and help is generally withheld until after M3 so that team behavior with all its issues can be exposed in early stages (Milestones M1-M3).

VIII. CONCLUSIONS AND FUTURE WORK

In this paper we presented, for the first time, the details of the SETAP Project's full data definition, data collection and first preliminary ML analysis methods and results, and the software pipeline used to implement them. The whole system (methods, algorithms, software) for data collection, creation of ML training database and ML analysis is fully operational and deployed for continuous data collection and analysis from three concurrent SE classes at SFSU, FAU and Fulda with over 140 students in 25 to 30 teams each year. Out preliminary results show consistency when tested with two independent accuracy measures and prediction intervals indicated by our ML approach offer intuitive explanation.

Future work includes first and foremost gathering and processing more data, which is under way with about 25-30 teams each year. In parallel we plan to investigate variable importance measures associated with RF in order to derive factors (e.g. TAM measures) that have most predictive power. Knowing such measures will help educators and SW managers better focus on predicting team performance.

We also recommend and plan ourselves the investigation of other ML and analysis methods, specifically those which deal well with small data sets and offer some explanation and ranking of the variables.

We plan to significantly streamline and improve our SW system for data gathering and collection into a training DB including creating on-line forms for instructor observations. In the long run we plan to leverage the fact that more and more tools for SE development and communication offer statistics of their usage as a byproduct, thus potentially enriching our TAM observation data.

Our future work will also include special attention to differences between local and global teams deploying, among others, clustering techniques on ML training database.

In order to enable others to try their own analysis techniques, and noting the very significant cost and time it takes to collect the data for the ML training database, we plan to make it publicly available once more data is collected.

ACKNOWLEDGEMENTS

We gratefully acknowledge the advice of Dr. Byron Dom regarding the ML component of our project.

REFERENCES

[1] "Standish Group Report: CHAOS Summary 2009." http://www1.standishgroup.com/newsroom/chaos_2009.php Accessed May 18, 2011.

[2] Sauer, Chris and Christine Cuthbertson. April 2003. "The State of IT Project Management in the UK 2002-2003." *Computer Weekly*. London.

[3] Sauer, Chris, Andrew Gemino, and Blaize H. Reich. November, 2007. "The impact of size and volatility on IT project performance," *Communications of the ACM*. Vol 50(11), pp. 79-84.

[4] Jones, Capers. 2010. *Software Engineering best Practices: Lessons from Successful Projects in the Top Companies*. McGraw Hill. ISBN 978-0-07-162161-8.

[5] Reel, John S. 1999. "Critical success factors in software projects." *IEEE Software*. Vol 16(3), pp. 18-23.

[6] Charette, Robert N. September, 2005. "Why software fails." *IEEE Spectrum*. Vol. 42(9), p. 42.

[7] Pressman, Roger. 2005. *Software Engineering: A Practitioner's Approach, Sixth Edition*. McGraw Hill.

[8] Curtis, Bill, Herb Krasner, and Neil Iscoe. 1988. "A field study of the software design process for large systems." *Communications of the ACM*. Vol. 31(11), pp 1268–1287.

[9] "CATME - Comprehensive Assessment for Team-Member Effectiveness." June, 2012. http://www.catme.org. Accessed June 6, 2012.

[10] Davis, Denny, Michael Trevisan, Patricia Daniels, Kenneth Gentili, Cynthia Atman, Robin Adams, David McLean, and Steven Beyerlein. 2003. "A Model for Transferable Integrated Design Engineering Education." *World Federation of Engineering Organizations.*

[11] Baldi, Pierre and Søren Brunak. 2001. *Bioinformatics: The Machine Learning Approach, Second Edition (Adaptive Computation and Machine Learning)*. MIT Press.

[12] Perner, Petra, ed. "Advances in data mining: applications in medicine, web mining, marketing, image and signal mining," 6th Industrial Conference on Data Mining, ICDM 2006. *Lecture notes in Computer Science*. Vol. 4065. Springer. 2006.

[13] Zhang, Du and Jing-Pha Tsai, eds. 2005. *Machine Learning Applications In Software Engineering.* Vol. 16. World Scientific Publishing. ISBN: 981-256-094-7

[14] Rajwinder Singh, Neeraj Mohan and Dr. Parvinder S. Sandhu. "Evaluation of Success of Software Reuse using Random Forest Algorithm." In International Conference on Artificial Intelligence and Embedded Systems (ICAIES'2012) July 15-16, 2012 Singapore.

[15] Breiman, Leo. 2001. *Random Forests*. Machine Learning. Vol. 45(1), pp. 5–32.

[16] Liaw, Andy and Matthew Wiener. 2002. *Classification and Regression by randomForest*. R News. Vol. 2(3), 18--22. http://www.r-project.org/doc/Rnews/Rnews_2002-3.pdf. Retreived October 22, 2013.

[17] Petkovic, Dragutin, Rainer Todtenhöfer and Gary Thompson, "Teaching practical software engineering and global software engineering: case study and recommendations," *Proceedings of the 36th ASEE/IEEE Frontiers in Education Conference*. San Diego, CA, 2006. pp. 19–24.

[18] Petkovic, Dragutin, Gary Thompson, and Rainer Todtenhöfer. "Teaching practical software engineering and global software engineering: evaluation and comparison." In *Proceedings of the Eleventh Annual Conference on Innovation and Technology in Computer Science Education*. Bologna. Italy, 2006. pp. 294–298.

[19] Petkovic, Dragutin, Gary Thompson, and Rainer Todtenhöfer. "Assessment and comparison of local and global SW engineering practices in a classroom setting." *Proceedings of the 13th Annual Conference on Innovation and Technology in Computer Science Education*. Madrid, Spain. June 2008. pp. 78–82.

[20] Thompson, Gary, Dragutin Petkovic, Shihong Huang and Rainer Todtenhöfer. "Teaching distributed collaborative development techniques in a software engineering class setting." *Integrating FOSS into the Undergraduate Computing Curriculum, Free and Open Source Software (FOSS) Symposium.* Chattanooga, Tenesee, USA. 2009.

[21] Petkovic, Dragutin, Gary Thompson, Rainer Todtenhöfer, Shihong Huang, Barry Levine, S. Parab, G. Singh, R. Soni, and S. Shrestha. "Work in progress: e-TAT: online tool for teamwork and "soft skills" assessment in software engineering education." *Frontiers in Education (FIE) 2010, IEEE*. 27-30 Oct. 2010. pp. S1G-1-S1G-3,

[22] Petkovic, Dragutin, Kazunori Okada, Marc Sosnick, Aishwarya Iyer, Shenshaochen. Zhu, Rainer Todtehoefer, Shihong Huang. "A Machine Learning Approach for Assessment and Prediction of Teamwork Effectiveness in Software Engineering Education." *Frontiers of Education FIE 2012*, Seattle, WA, October 2012. pp. 1-3.

[23] Chen, Chao, Andy Liaw, and Leo Breiman. July, 20014. "Using Random Forest to Learn Imbalanced Data." Statistics Technical Report. ID: 666. UC Berkeley. http://statistics.berkeley.edu/sites/default/files/tech-reports/666.pdf Retreived June, 20, 2014.

[24] Khoshgoftaar, Taghi, M., Moiz Golawala, and Jason Van Hulse. "An Empirical Study of Learning from Imbalanced Data Using Random Forest." *19th IEEE International Conference on Tools with Artificial Intelligence, 2007. ICTAI 2007*. Vol. 2, pp. 29-31.

[25] "LimeSurvey – the free and open source survey software tool." http://www.limesurvey.org/en. Accessed June 20, 2014.

[26] Liaw, Andy and Matthew Wiener. 2013. "Breiman and Cutler's random forests for classification and regression". R Package 'randomForest'. http://cran.r-project.org/web/packages/randomForest/randomForest.pdf. Accessed June 20, 2014.

[27] Team, R Core. 2012. "A Language and Environment for Statistical Computing."

[28] Buturovic Ljubomir, Wong Mike, Grace W.Tang, Russ B. Altman, Dragutin Petkovic. 2014. "High Precision Prediction of Functional Sites in Protein Structures." *PLoS ONE* Vol 9 (3).