Towards Data-driven Software Engineering Skills Assessment

Jun Lin*†‡, Han Yu†, Zhiqi Shen*† and Chunyan Miao*†
*School of Computer Engineering, Nanyang Technological University (NTU), Singapore
†Joint NTU-UBC Research Centre of Excellence in Active Living for the Elderly (LILY), NTU, Singapore
‡School of Software, Beihang University, Beijing, China
{jlin7, han.yu, zqshen, ascymiao}@ntu.edu.sg, linjun@buaa.edu.cn

***Abstract*—Today's software engineers often work in teams to develop complex software systems. Therefore, successful software engineering in practice not only require team members to possess sound programming skills such as analysis, design, coding, and testing, but also soft skills such as communication, collaboration, and self-management, etc. However, existing examination based assessments are often inadequate for quantifying students' soft skill development. In this paper, we explore alternative ways for assessing software engineering students' skills through a data-driven approach. Leveraging our proposed *HASE* online agile project management (APM) tool, we conducted a study involving 21 Scrum teams consisting of over 100 undergraduate software engineering students in multi-week coursework projects in 2014. During this study, students performed close to 170,000 software engineering activities logged by HASE. By analyzing the collected activity trajectory dataset, we demonstrate the potential for this new research direction to enable software engineering educators to have a quantifiable way of understanding their students' skill development, and take a proactive approach in helping them improve their programming and soft skills.**

I. Introduction

Most hiring managers in software engineering companies understand that a successful member of software engineer needs to be strong in both *programming skills* (e.g., software design, coding, and testing skills) and *soft skills* (e.g., communication, collaboration, and self-management skills) [1]. Programming skills can be gauged, at least in part, from students' performance in examinations and programming contests. Soft skills are much harder to assess, especially during the limited time given in job interviews. Although the concept of these skills can be taught, the ability to apply them consistently in practice can only be acquired through one's own experience.

In tertiary education institutions, software engineering students are often assessed by a combination of examinations and coursework projects. Many educators have realized the limitations of examinations in assessing students' practical skills. Thus, coursework projects often serve as an opportunity for students to both practice and demonstrate their skills. However, as an instructor has to face tens or even hundreds of students in a semester, it is not practical for him/her to know the weaknesses and strength in each student's skills in detail through observation. Technologies that can subjectively quantify students' skill development are needed to enable instructors to proactively and effectively help each student.

With the emergence of systems capable of collecting personal behavior trajectory big data [2], data-driven analysis of people's characteristics over time is changing how students' performance can be measured. Some funding agencies are starting to support research in data-driven student assessment technologies to complement traditional examination scores. For example, the Ministry of Education in Singapore has started an initiative to build technological solutions capable of holistically assessing students' 21st Century Competencies (e.g., critical thinking, self-directed learning skills)[1].

Following a similar line of thinking, in this paper, we explore how software development behaviour data can be used to assess students' programming and soft skills. As agile software development (ASD) involves many human factors reflecting developers personal characteristics compared to other plan-driven methodologies [3], we focus on tracking students' activities in the ASD process. For this purpose, we conducted a 12 week study involving 125 undergraduate software engineering students from the Beihang University, Beijing, China. The students self-organized into 21 ASD teams of 5 to 7 persons. Each team developed one software system of significant complexity following the Scrum ASD method as part of their course requirements. Some examples of the coursework projects include "A Personal Healthy Living App", "A Social Network App for Senior Citizens", and "An Activity Tracking App for the Elderly", etc.

Students in this study carried out software engineering activities at various stages of the Scrum methodology in our online agile project management (APM) tool - the Human-centred Agile Software Engineering (HASE) platform (http://www.linjun.net.cn/hase/) [4]. HASE mainly supports activities during the *sprint planning* and *sprint review/retrospective* phases. Such activities include proposing tasks, estimating the priority, difficulty and time required for each task, deciding how to allocate tasks, collaboration information, reviewing the timeliness and quality of completed tasks, and providing feedback on individual team member's mood at different points in time during a sprint. During the study, students logged 169,137 ASD activities in the HASE platform. By analyzing the collected dataset to reflect students' programming skills,

[1]The plan for the initiative can be accessed from: http://www.moe.gov.sg/media/press/2010/03/moe-to-enhance-learning-of-21s.php

collaboration, and mood stability, we demonstrate the potential of this research direction and discuss its implications for software engineering education.

II. Related Work

To the best of our knowledge, there has yet to be published previous studies using software engineering activity data to assess software engineers' skills. Nevertheless, as the skills assessment has always been an important problem, other methods have been applied in an attempt to address it.

Author in [5] advocated evidence-based software engineering (EBSE) similar to what is happening with evidence-based medicine. A technological platform for tracking and analyzing important factors in software engineering such as skills factors and lifecycle factors were called for, and the benefits of which were analyzed. Nevertheless, the work intends to produce methods to support the development of high quality of software through objective analysis of performance related indicators. Although similar in principle to our work, [5] does not specify how EBSE can assess the skills demonstrated by the software engineers or how such insight can be used to improve software engineering education.

In [6], the authors presented the results of a systematic literature review concerning agile pair programming effectiveness. The paper analyzed compatibility factors, such as the feel good, personality, and skill level factors, and their effect on pair programming effectiveness. Four metrics were used in the analysis: 1) academic performance, 2) technical productivity, 3) program/design quality and 4) learning satisfaction. As the study was not focused on assessment, the general findings are not useful for skills assessment. Nevertheless, it did point towards the importance of soft skills in software engineering.

The study reported in [4] started to track personal performance data with agile project management tools to study task allocation related decision-making under Scrum. It employed the same research techniques as reported in this paper. However, the study in [4] focused on analyzing students' programming skills and did not consider their soft skills such as collaboration and mood.

III. Study Design

In this section, we present our research approach, and the metrics adopted in our analysis.

A. Research Approach

We use the HASE APM platform to unobtrusively track the student participants' activities in the Scrum ASD process, including their decision-making, collaboration, task assignment and mood, etc. The platform provides five main features to support agile project management which cover the sprint planning and sprint review/retrospective phases:

1) *Registration*: In order to build user profiles, HASE requires registrants to specify their self-assessed competence levels in different areas of expertise such as familiarity with specific programming languages, system design methodologies, and user interface (UI) design tools, etc. This information will only be used to compute an initial assessment for a user in the absence of peer ratings or performance data. Once data from these relatively more objective sources become available, the user's self-assessment will be excluded from the assessment result.
2) *Team and Role Management*: HASE supports the creation of teams, the selection of product owners and stakeholders into the teams, and the assignment of different roles within a team (e.g., programmers and UI designers).
3) *Task Management*: Task information including task description, skills required for the task, and the person who proposed each task is displayed for all team members to view. The difficulty value of each task τ, is recorded using an 11-point Likert scale [7] (with 0 denoting "extremely easy" and 10 denoting "extremely hard"). Each team member can input his/her estimated difficulty value for each task into the HASE platform. The HASE platform then uses the average difficulty value for the task (D_τ). The students were asked to take into account the technical challenge as well as the amount of effort required when judging the difficulty of a task. The priority value of each task τ, is also recorded using an 11-point Likert scale (with 0 denoting "extremely low priority" and 10 denoting "extremely high priority"). Each team member can input his/her estimated priority value for each task into the HASE platform. The HASE platform then uses the average priority value for the task.
4) *Sprint Planning*: HASE records the teams' decisions on which tasks are assigned to which team member during each sprint. Once assigned, the status of the task becomes "Assigned". The assignee i inputs his/her confidence value ($Conf_\tau^i$) for each task τ on an 11-point Likert scale (with 0 denoting "not confident at all" and 10 denoting "extremely confident"). Each team member also inputs the estimated required time to complete each task (in number of days). The HASE platform uses the average estimated time required to generate the deadline for the task (T_τ^{est}). Apart from a primary assignee, multiple students can collaboratively work on a task. The collaborator information for each task is also recorded by HASE.
5) *Sprint Review/Retrospective*: Once a task is completed, the assignee changes its status in the HASE platform to "Completed". This action will trigger HASE to record the actual number of days (T_τ^{act}) used to complete this task. HASE also provides functions for team members to peer review the quality ($Qual_\tau$) of each completed task τ. The quality of a completed task is recorded in the platform using a 11-point Likert scale with 0 representing ("extremely low quality") and 10 representing ("extremely high quality"). The average quality rating for each task is used by HASE as the final quality rating for that task.
6) *Team Morale Monitoring*: During the sprint planning

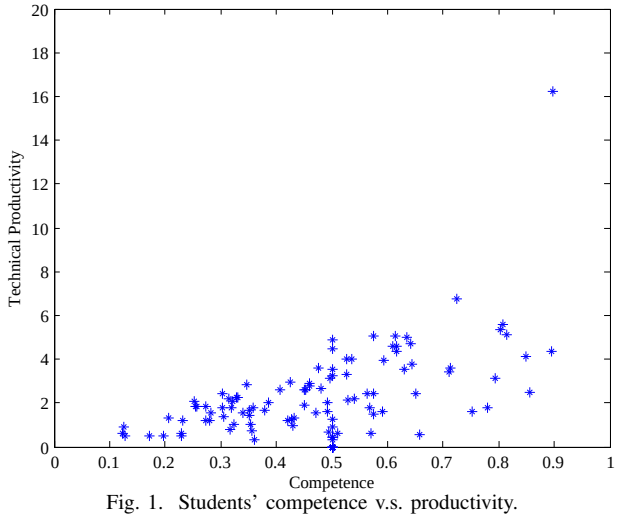

Fig. 1. Students' competence v.s. productivity.

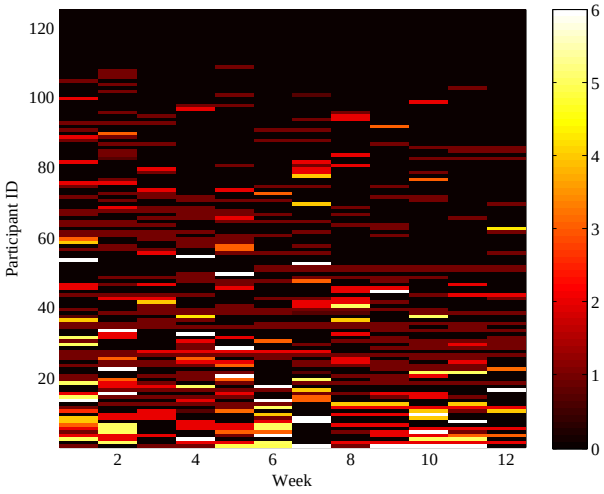

Fig. 2. Average number of collaborators per task.

meeting, team members can report their current mood values into the HASE platform. A person i's mood at the beginning of a sprint t ($m_i^{begin}(t)$) is represented on a 5-point Likert scale with 1 representing "very low" and 5 representing "very high". During the sprint review/retrospective meeting, each task assignee i can report his/her mood after completing a task at the end of sprint t ($m_i^{end}(t)$) using the same 5-point Likert scale.

The input data to the HASE platform required from ASD teams are as a result of students' activities following the Scrum methodology. In this way, users of HASE can behave as if they are using any APM tool without expending additional effort to help with data collection. Thus, the data collection process remains unobtrusive to the participants. Over the 12 week period of the coursework project, the HASE platform collected 169,137 behaviour trajectory records related to software engineering activities from the 125 students who participated in this study. The anonymized dataset is made available to the research community to support further studies and analysis[2].

[2]Link to the dataset.

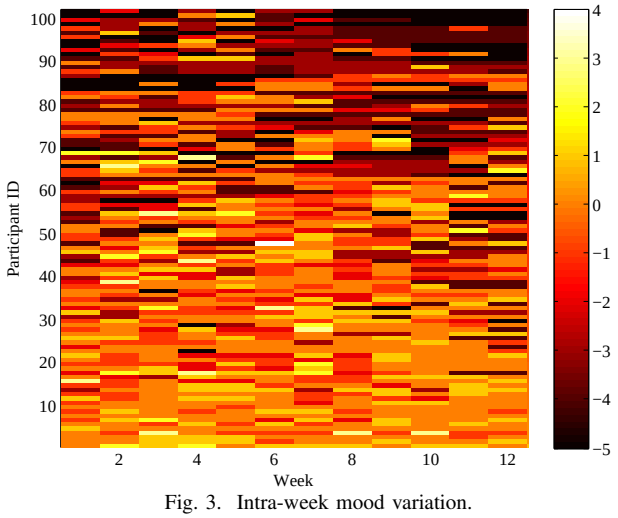

Fig. 3. Intra-week mood variation.

B. Metrics

In this paper, we adopt the exploratory data analysis (EDA) approach [8] to analyze the data collected. EDA is an approach for analyzing data sets to summarize their main characteristics, often with visual methods. It is primarily for understanding what can be learnt from the data beyond the formal modeling or hypothesis testing task. We use the following metrics to facilitate our analysis:

1) *Technical Productivity* (μ_i): it refers to the average amount of workload a student i can complete during a sprint. In this study, we use the task difficulty value as an indicator of the workload of a task as the task difficulty values reported by students denote both the technical challenge and the amount of effort required to complete the task.
2) *Competence* ($Comp_i$): it refers to the probability a student i can complete a task assigned to him/her with satisfactory quality before the stipulated deadline. In this paper, the outcome of a task needs to achieve an average quality rating higher than 5 out of 10 in order to be considered as having satisfactory quality. This metric is similar to a student's reputation. Thus, we adopt a reputation computation model - the Beta Reputation model [9] - which is widely used in the fields of online services, artificial intelligence and network communications [10], [11], [12], [13], [14]. It is calculated as follows:

$$Comp_i = \frac{\alpha_i + 1}{(\alpha_i + 1) + (\beta_i + 1)} \in (0, 1) \qquad (1)$$

where α_i and β_i are calculated as:

$$\alpha_i = \sum_{\tau \in \phi(i)} 1_{[T_\tau^{act} - T_\tau^{est} \leq 0 \text{ and } Qual_\tau > 5]} D_\tau \qquad (2)$$

$$\beta_i = \sum_{\tau \in \phi(i)} 1_{[T_\tau^{act} - T_\tau^{est} > 0 \text{ or } Qual_\tau \leq 5]} D_\tau. \qquad (3)$$

The function $1_{[\text{condition}]}$ in Eq. (2) and Eq. (3) equals to 1 if "condition" is true. Otherwise, $1_{[\text{condition}]}$ equals to 0. $\phi(i)$ denotes the set of tasks i has previously worked

on until the current point in time. The "+1" terms in the numerator and denominator of Eq. (1) are *Laplace smoothing* terms [15] which ensure that if i has no previous track record, $Comp_i$ evaluates to 0.5 indicating maximum uncertainty about i's performance.

3) *Team Morale (Begin)* ($M_j^{begin}(t)$): it refers to the average of the mood values reported by members of team j during the sprint planning meeting of sprint t.
4) *Team Morale (End)* ($M_j^{end}(t)$): it refers to the average of the mood values reported by members of team j during the sprint review/retrospective meeting of sprint t.

IV. Results and Analysis

An exploratory data analysis has identified certain personal characteristics which may become useful markers for assessing students' skills in the future. Figure 1 shows the participants' competence scores versus their productivity scores at the end of the study. It can be observed that the participants' performances in terms of these metrics are quite distinguishable. In general, participants who demonstrated high competence tend to also be able to handle high workloads allocated to them ($r = 0.7443$, $p < 0.01$). One participant achieved significantly higher competence and productivity scores than the rest of the participants.

Collaboration is generally regarded as a useful way to improve the effectiveness and efficiency of a software team. Figure 2 shows a heat map of the number of collaborators per task each participant had for each of the 12 weeks. The lighter the color of a point on the figure, the more collaborators per task that participant had for that particular week. The color scale mapping different color gradients to the actual number of collaborators per task is shown on the right-hand side of the figure. Participants are ranked according to their average number of collaborators per task per week. Those who are shown at the bottom of the figure ranked the highest among their peers. It can be observed that this metric can distinguish the behaviors among different participants clearly.

Stability of mood is a sign showing one's maturity and self-management skills. Figure 3 shows a heat map of the intra-week mood change (which is computed as $\Delta m_i(t) = m_i^{end}(t) - m_i^{begin}(t) \in (-5, 5)$ for each week) over the 12 weeks. 102 out of the 125 participants provided valid reports on their $m_i^{begin}(t)$ and $m_i^{end}(t)$ values. The color scale mapping different color gradients to the intra-week mood change is shown on the right-hand side of the figure. Participants are ranked according to their average intra-week mood change values per week over 12 weeks. Those who are shown at the bottom of the figure ranked the highest among their peers. It can be observed that this metric can distinguish the behaviors among different participants quite clearly. The mood of those who ranked high on this metric tends to increase at the end of a week after a sprint of development. And as their mood at the beginning of the week also tends to be high, the increments are generally small. Thus, their mood remain relatively stable throughout a sprint. Those who ranked low on this metric (top part of the figure) tend to have big negative mood swings, especially towards the end of the study.

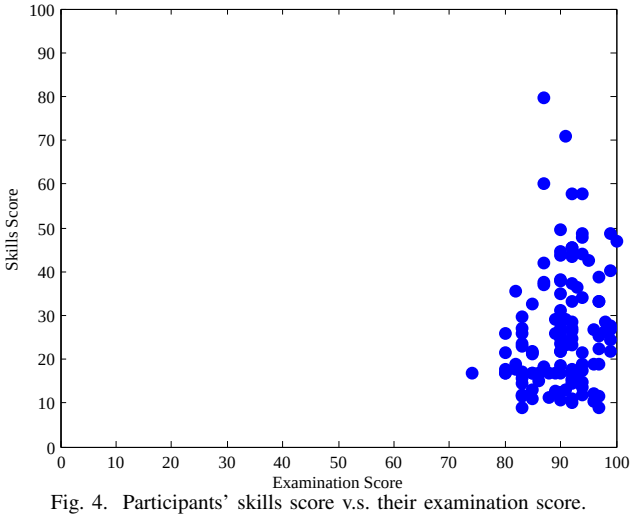

Fig. 4. Participants' skills score v.s. their examination score.

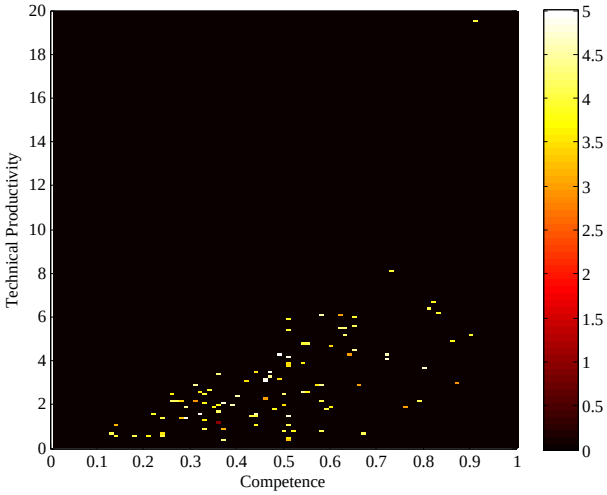

Fig. 5. Students' average morale before a Sprint.

In order to explore if the assessment of participants' skills may help us identify students who are good at hands-on software engineering but did not stand out in examinations, we construct a *skills score* to aggregate the effect of competence, productivity, collaboration, and mood stability into one scalar measurement. In this study, the skills score, $S_{skills}(i)$, for a participant i is computed as:

$$S_{skills} = \frac{S_{\mu_i} + S_{Comp_i} + S_{col_i}}{1 - S_{\Delta m_i}} \times \frac{100}{3}, \tag{4}$$

where $S_{\mu_i} \in [0, 1]$, $S_{Comp_i} \in [0, 1]$, $S_{col_i} \in [0, 1]$ and $S_{\Delta m_i} \in (-1, 1)$ are the normalized scores for i in terms of productivity, competence, collaboration, and mood stability, respectively ($S_{skills}(i) \in [0, 100]$).

Figure 4 plots the participants' skills scores against their examination scores for the subject of software engineering in the same semester. The examination paper used was the standard software engineering end-of-semester examination paper from the Beihang University, which has been designed by the professors in charge of the course and reviewed by the university examination board. It can be observed that,

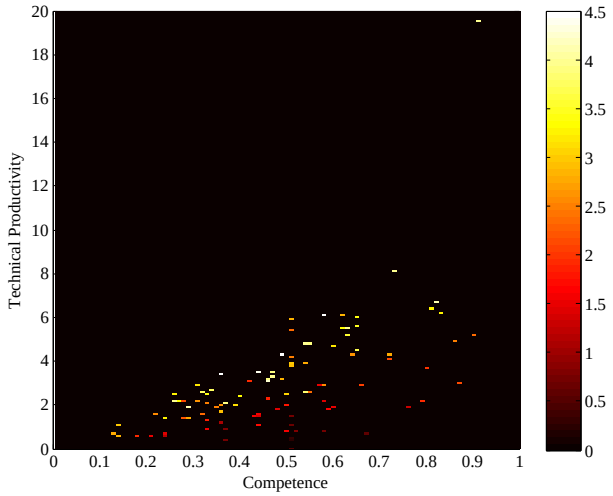

Fig. 6. Students' average morale after a Sprint.

according to their exam scores, their performance clustered in the range of 80 to 100 marks, enabling almost all of them to achieve a grade of A or A+. However, their skills scores spread from as low as 10 marks to as high as 80 marks, making their performance more distinguishable compared to their exam scores. The skills scores have only a weak positive correlation with the exam scores ($r = 0.2129$, $p < 0.05$). Furthermore, the top three best performing participants in terms of skills scores achieved only average exam scores among their peers, and many participants with high exam scores achieved low skills scores.

We acknowledge that there may be other ways to compute the skills score and we refrain from claiming that our current formulation for the skills score is the most effective. Nevertheless, the results show that the data-driven skills score can potentially help us distinguish the performance of software engineering students better than examination-based assessments.

Figure 5 shows the distribution of students' average self-reported mood values during the sprint planning meeting at the start of each sprint. The colour scale represents the average self-reported mood values. The average mood value is 3.86 out of 5. The correlation between students' mood during the sprint planning meetings and their competence values is ($r = -0.0025$, $p = 0.9394$), indicating no statistically significant correlation. The correlation between students' mood during the sprint planning meetings and their technical productivity values is ($r = 0.1505$, $p < 0.01$), indicating a statistically significant albeit weak positive correlation.

Figure 6 shows the distribution of students' average self-reported mood values during the sprint review/retrospective meeting at the end of each sprint. The colour scale represents the average self-reported mood values. The average mood value is 3.80 out of 5 which is slightly lower than at the beginning of the sprint. The correlation between students' mood during the sprint review/retrospective meetings and their competence values is ($r = -0.0148$, $p = 0.5946$), indicating no statistically significant correlation. The correlation between students' mood during the sprint review/retrospective meetings and their technical productivity values is ($r = 0.4207$, $p < 0.01$), indicating a statistically significant positive correlation. Therefore, based on these analysis, team members with high technical productivity tend to have high morale, especially at the end of a sprint after completing the tasks allocated to them.

V. Implications

By providing a technological platform for the longitudinal tracking of software engineers' behaviour trajectory data related to software development, we open up new possibilities for different parties involved in software engineering researchers, educators, and practitioners.

Implications for software engineering researchers: The availability of large software engineering behaviour datasets will present new challenges to researchers to develop new analytics techniques. With detailed information on each user's demographics, skills indicator values over time, detailed interactions with the software engineering tools provided, interactions with other team members, and decisions made, etc., the high dimensionality of the datasets makes it a challenge to identify which feature, or combination of features, can form accurate predictors for certain behaviours of interest. Machine learning [16] can be leveraged to develop useful techniques for this purpose.

However, before this step can happen, additional efforts are needed to complement the behaviour datasets with labelled meta-data on what the observed behaviour patterns mean. This can potentially be achieved by conducting follow-up interview-based studies with the participants through carefully designed questionnaires once unique behaviour patterns have been identified. This also further opens up the research question on how to present the behaviour pattern data in a human interpretable format to facilitate the interviews.

Implications for software engineering educators: Software engineer educators may be a viable source of knowledge in the effort of building up a repository of meta-data for the behaviour patterns obtained by the tracking platform. As they frequently interact with students who may be using the proposed tracking platform, they can potentially provide insights into the meanings of the behaviour patterns. The challenge here is for software engineering researchers to provide tools to enable educators who are willing to contribute meta-data for the behaviour patterns to do so with ease. Techniques from the field of crowdsourcing [17] may offer a starting point for such an effort.

Once new techniques for automatically assessing the a student's skill development based on his/her behaviour patterns are developed, new forms of real-time personalized inventions may become available to educators. The simplest possibility is for the system to send out alerts on students who may require help in specific areas to course instructors. Through mining the behaviour patterns of many students and cross-checking with their academic performance, or even employment prospects if such data are available, the system may be able to suggest behaviour trajectories that are the most beneficial for students from different backgrounds, thereby making data-driven personalized software engineering training

possible. The envisioned behaviour data tracking platform can potentially convert software engineering education into a test bench for open science and enable a more adaptive and individualized learning experience.

Implications for software engineering practitioners: The behaviour data are tracked in an unobtrusive manner by the APM tool automatically. The peer review related functions represent activities that an ASD team member already having to perform when following the ASD practice. Overall, the proposed APM tool based behaviour data tracking approach does not require software engineers to incur additional overhead. However, the data analytics functions provide ASD teams with insights into detailed team dynamics and performance information which can be useful for decision-making. Furthermore, with the behaviour data as input, automatic context-aware software engineering task allocation decision support mechanisms [18] become a distinct possibility. These mechanisms can be based on similar mechanisms available in the field of crowdsourcing [19], [20], [21].

VI. Discussions and Future Work

In this paper, we explore a novel data-driven approach to assess software engineering students' skills. Different from traditional interview/internship-based methods, our study is based on participants ASD activity trajectory data collected unobtrusively during normal ASD processes through our HASE APM platform. This type of data objectively reflects developers ASD activities and performance at fine granularity.

As the data collection and analytics technologies further develop, software engineering students may eventually perform all coursework activities in a technology platform capable of unobtrusively collecting their behavior data and continuously assessing a wide range of their skills over time. In this way, the students' practical skill development can be monitored by their instructors so that pedagogical methods can be personalized to help individual students in specific areas. Such a tool will enable software engineering educators to have a quantifiable way of understanding their students' skill development and take a proactive approach in helping them develop programming and soft skills. The skills scores may, one day, be part of a student's academic profile and be taken into consideration by industry recruiters to help companies identify well rounded software engineering talents suitable for their teams.

From this study, we see the start of a series of research and applications in data-driven software engineering skills assessment. In future research, we plan to conduct surveys/interviews to understand more in-depth how students collaborate. We will continue using the HASE platform to collect agile programming activity data over subsequent semesters, and expand our data collection effort to include more universities so as to investigate the possible effects of socio-cultural factors. More finely grained data such as the time each student spent on a task and the breakdown of the usage of the time will also be collected in future versions of the HASE platform. The resulting datasets will be published to support the discovery of new insights.

Acknowledgment

This research is supported by the National Research Foundation, Prime Minister's Office, Singapore under its IDM Futures Funding Initiative and administered by the Interactive and Digital Media Programme Office.

References

[1] P. Freeman, A. I. Wasserman, and R. E. Fairley, "Essential elements of software engineering education," in *Proceedings of the 2Nd International Conference on Software Engineering (ICSE'76)*, 1976, pp. 116–122.

[2] P. Heymann and H. Garcia-Molina, "Turkalytics: analytics for human computation," in *Proceedings of the 20th International Conference on World Wide Web (WWW'11)*, 2011, pp. 477–486.

[3] A. Cockburn and J. Highsmith, "Agile software development, the people factor," *Computer*, vol. 34, no. 11, pp. 131–133, 2001.

[4] J. Lin, H. Yu, Z. Shen, and C. Miao, "Studying task allocation decisions of novice agile teams with data from agile project management tools," in *Proceedings of the 29th IEEE/ACM International Conference on Automated Software Engineering (ASE'14)*, 2014, pp. 689–694.

[5] B. A. Kitchenham, T. Dyba, and M. Jorgensen, "Evidence-based software engineering," in *Proceedings of the 26th International Conference on Software Engineering (ICSE'04)*, 2004, pp. 273–281.

[6] N. Salleh, E. Mendes, and J. Grundy, "Empirical studies of pair programming for cs/se teaching in higher education: A systematic literature review," *IEEE Transactions on Software Engineering (TSE)*, vol. 37, no. 4, pp. 509–525, 2011.

[7] R. Likert, "A technique for the measurement of attitudes," *Archives of Psychology*, vol. 22, no. 140, 1932.

[8] J. W. Tukey, *Exploratory Data Analysis*. Addison-Wesley, 1977.

[9] A. Jøsang, R. Ismail, and C. Boyd, "A survey of trust and reputation systems for online service provision," *Decision Support Systems (DSS)*, vol. 43, no. 2, pp. 618–644, 2007.

[10] L. Pan, X. Meng, Z. Shen, and H. Yu, "A reputation pattern for service oriented computing," in *Proceedings of the 7th International Conference on Information, Communications and Signal Processing (ICICS'09)*, 2009.

[11] H. Yu, Z. Shen, C. Miao, C. Leung, and D. Niyato, "A survey of trust and reputation management systems in wireless communications," *Proceedings of the IEEE*, vol. 98, no. 10, pp. 1755–1772, 2010.

[12] H. Yu, S. Liu, A. C. Kot, C. Miao, and C. Leung, "Dynamic witness selection for trustworthy distributed cooperative sensing in cognitive radio networks," in *Proceedings of the 13th IEEE International Conference on Communication Technology (ICCT'11)*, 2011, pp. 1–6.

[13] S. Liu, H. Yu, C. Miao, and A. C. Kot, "A fuzzy logic based reputation model against unfair ratings," in *Proceedings of the 12th International Conference on Autonomous Agents and Multi-agent Systems (AAMAS'13)*, 2013, pp. 821–828.

[14] H. Yu, Z. Shen, C. Leung, C. Miao, and V. R. Lesser, "A survey of multi-agent trust management systems," *IEEE Access*, vol. 1, no. 1, pp. 35–50, 2013.

[15] Y. Wang and M. P. Singh, "Formal trust model for multiagent systems," in *Proceedings of the 20th International Joint Conference on Artifical Intelligence (IJCAI'07)*, 2007, pp. 1551–1556.

[16] Y. Anzai, *Pattern Recognition and Machine Learning*. Academic Press, Inc., 1992.

[17] A. Doan, R. Ramakrishnan, and A. Y. Halevy, "Crowdsourcing systems on the world-wide web," *Communications of the ACM*, vol. 54, no. 4, pp. 86–96, 2011.

[18] J. Lin, "Context-aware task allocation for distributed agile team," in *Proceedings of the 28th IEEE/ACM International Conference on Automated Software Engineering (ASE'13)*, 2013, pp. 758–761.

[19] H. Yu, Z. Shen, C. Miao, and B. An, "A reputation-aware decision-making approach for improving the efficiency of crowdsourcing systems," in *Proceedings of the 12th International Conference on Autonomous Agents and Multi-agent Systems (AAMAS'13)*, 2013, pp. 1315–1316.

[20] H. Yu, C. Miao, B. An, C. Leung, and V. R. Lesser, "A reputation management approach for resource constrained trustee agents," in *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI'13)*, 2013, pp. 418–424.

[21] H. Yu, C. Miao, Z. Shen, C. Leung, Y. Chen, and Q. Yang, "Efficient task sub-delegation for crowdsourcing," in *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI-15)*, 2015, pp. 1305–1311.