Wu, Y.-C., Mar, L. W., & Jiau, H. C. (2013). A Service-Based Program Evaluation Platform for Enhancing Student Engagement in Assignments. *Educational Technology & Society*, *16* (4), 122–132.

A Service-Based Program Evaluation Platform for Enhancing Student Engagement in Assignments

Ye-Chi Wu*, Lee Wei Mar and Hewijin Christine Jiau
Institute of Computer and Communication Engineering, Department of Electrical Engineering, National Cheng Kung University, Tainan, Taiwan // yechiwu@nature.ee.ncku.edu.tw // lwmar@nature.ee.ncku.edu.tw // jiauhjc@mail.ncku.edu.tw
*Corresponding author

(Submitted June 15, 2012; Revised September 21, 2012; Accepted November 06, 2012)

ABSTRACT

Programming assignments are commonly used in computer science education to encourage students to practice target concepts and evaluate their learning status. Ensuring students are engaged in such assignments is critical in attracting and retaining students. To this end, WebHat, a service-based program evaluation platform, is introduced in this work to support programming assignment practice. WebHat provides students with immediate assistance via online evaluation services and helps teaching staff monitor student status and progress. Based on the results of this system, teaching staff can dynamically adjust, if necessary, online evaluation services to provide further help to enhance student engagement in the ongoing assignment. To demonstrate the effectiveness of WebHat, an empirical study is conducted with freshman students as the subjects during the *Introduction to Computer Science* course. The results show that students are encouraged to start working on their assignments earlier and develop programs in the desired direction with the aid of the dynamically adjusted online evaluation services. WebHat thus enables students to submit better final products with regard to both functionality and quality.

Keywords

Student engagement, Programming teaching, Programming evaluation service, Service-based learning platform

Introduction

Student engagement is an important factor that contributes to the level of student achievement (Fredricks et al., 2004; Munns & Woodward, 2006; Cano, 2011). In computer science education, programming assignments are commonly used to get students to focus on target concepts and to evaluate their learning status. Through appropriate assignment design with regard to the content, context and expected product, students should be able to progress in their learning and teachers should be able to identify any problems that are occurring on the course. Ensuring student engagement in such assignments is thus critical in attracting and retaining students (Hansen & Eddy, 2007).

Traditional programming assignments proceed with a *release and then evaluate until deadline style*, where the responsibility of teaching staff lies in releasing the initial assignment requirements and then evaluating final assignment products. Students in this learning style complete the programming assignment in a black box manner, and thus most students are not conscious of any problems until they receive unsatisfied grades in the final evaluation. In this context, poor grades not only represent a failure in learning, but also have an adverse effect on student motivation.

Additionally, the success or failure of assignment practice often depends on the individual backgrounds and behaviors of students (Law et al., 2010; Blikstein, 2011). For example, teaching staff usually assume that students have the ability to maintain an appropriate schedule to deal with their assignments, but in reality many students start work too late and subsequently achieve poorer than anticipated grades. Similarly, some students develop their assignment in a wrong direction because of misunderstanding about the requirements. When a significant number of students have such problems, then the entire assignment may be considered a failure. To avoid this, it is thus necessary to monitor student engagement in the assignment and to provide help when it is needed.

This paper thus introduces WebHat, a service-based programming evaluation platform, to support programming assignment practice. A well-designed assignment attempts to ensure student engagement via releasing related

assistance, such as test cases, when the project is announced. Based on the design of the assignment, teaching staff can set online evaluation services on WebHat. By accessing the results of the evaluation of their assignment products while they are being produced, students can acquire immediate assistance before the end of the project. In addition, data about student interactions with WebHat will be collected, and teaching staff can thus monitor the students' status with regard to the assignment, such as whether or not they have started work, and whether they are making efforts to improve based on the results of the online evaluation. Furthermore, WebHat utilizes a service-oriented architecture (Papazoglou, 2003; Papazoglou et al., 2008) to provide teaching staff with the flexibility needed to adjust online evaluation services dynamically. Based on the observations of student status during assignment development, teaching staff can provide students with prompt, active and direct help. The ability to keep accessing the online evaluation services will guide students toward successful outcomes in their assignments.

Background

In the recent years, e-learning systems have been widely utilized as a platform for programming practice (Verdú et al., 2011). One of major benefits of teaching this way is that students can acquire some pre-defined immediate assistance from e-learning systems. To enhance student involvement in programming practice, different levels of assistance are provided by various e-learning systems. For example, ClockIt (Norris et al., 2008) and Retina (Murphy et al., 2009) focus on assisting students in solving syntax, semantics and runtime errors to make their program executable. In contrast, Submit! (Pisan et al., 2003), RoboProf (Daly & Horgan, 2004), Marmoset (Spacco et al., 2006), Web-CAT (Edwards & Pérez-Quiñones, 2008), and NOBASU (Funabiki et al., 2010; 2012), focus on providing online testing to assist students in developing an assignment product that will meet specific requirements. Besides assisting students in ensuring that their assignment product is executable and correct, some systems focus on assisting students in getting more involved in the assignment in order to encourage them to keep improving the quality of their work. For example, Lawrence (Lawrence, 2004) proposed a Critical Mass project in which students develop game strategies via programming for a final Critical Mass tournament. Figure 1 is an example of two major support for a current e-learning system applied in the programming assignment process. As shown in Figure 1(a), the e-learning system provides a fixed set of online evaluations for a specific kind of assistance that can help the students develop their work.

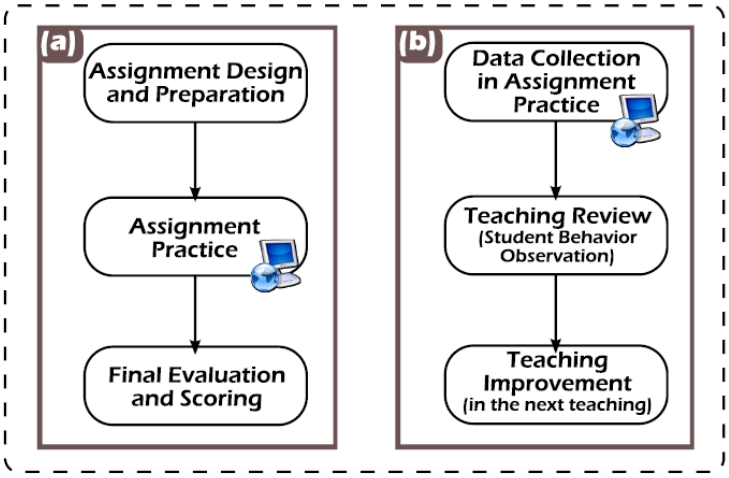

Figure 1. Two major support for a current e-learning system in (a) The programming assignment process, and (b) The teaching review process

The data collected in e-learning systems which reflect the different levels of frustration in learning are valuable. To help teaching staff review their teaching, many existing e-learning systems collect the data about student behavior in programming assignment practice (Spacco et al. 2006; Edwards & Pérez-Quiñones, 2008; Norris et al., 2008; Murphy et al., 2009; Blikstein, 2011). As shown in Figure 1(b), teaching staff can observe student behavior via the data collected by the e-learning system in order to extract certain factors that can be associated with success or failure in programming. For example, according to studies conducted on Web-CAT (Allevato et al., 2008) and ClockIt (Norris et al., 2008; Fenwick et al., 2009), student success in programming (i.e., getting higher grades in the

assignment) is highly related with factors such as an early start in assignment product development, a high frequency of accessing online evaluations, and a low complexity of the finished assignment product. These factors reveal some possible directions for teaching staff consider with regard to improving their teaching.
Many e-learning systems provide student immediate assistance and the related data collected by e-learning systems is effective for teaching review. However, as shown in Figure 1, the support offered by such systems is applied in the distinct processes. These systems only provide fixed assistance. Teaching staff cannot provide further help through the existing e-learning systems even if any student frustration is observed during assignment practice. The data collected during on-going teaching can only be used to improve the next stage of teaching.

Assignment practice with WebHat

To enhance student engagement during assignment practice, this work introduces WebHat with dynamic evaluation services adjustment. Figure 2 illustrates the detailed process during assignment practice with WebHat. To ensure the achievement of a specific assignment, the teaching staff specifies particular evaluation services in WebHat, which will then provide appropriate online assistance to students during assignment practice. As illustrated in the *Student Session* of Figure 2, students keep developing their assignment products during the practice, and when they finish stable versions of their products, they can access the online evaluation in WebHat and check the results. These results provide a basis to guide the subsequent product development, such as fixing bugs and further evolution of the program. Students remain involved in the cycle of assignment product development and evaluation until they have finished work or the deadline is reached, whichever is sooner.

Simultaneously, in the *Teaching Staff Session*, the data about student interactions with WebHat is also collected to help teaching staff monitor their status during the assignment. WebHat provides statistical reports about this status, and teachers can utilize this information to observe student behavior. When student frustration is observed, teachers can offer further help and adjust the evaluation services available in WebHat. The assistance can be immediately provided by the system to enhance student engagement in assignment practice.

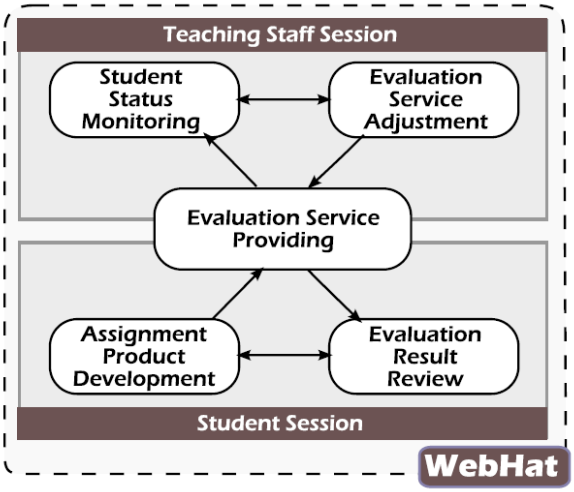

Figure 2. Detailed process during assignment practice with WebHat

Figure 3 presents the architecture of WebHat, which applies a service-based mechanism to support dynamic evaluation service adjustment (Wu & Jiau, 2012). A general flow of assignment evaluation is maintained by an assignment evaluation backbone. As shown in Figure 3(a), students keep submitting stable modifications of their assignment products via *Assignment Product Uploader*, and WebHat checks each iteration to perform the evaluation. All of the evaluation results are record in *Evaluation Result Records*, and both teaching staff and students can access these results via a web browser. The *Evaluation Result Displayer* is responsible for generating a feedback page to display the evaluation results based on who is making the request, a teacher or a student.

Service Manager, as shown in Figure 3(b), maintains a set of service specifications, which is used to describe concrete evaluation services with specifying the necessary functionality at specific points of the assignment evaluation backbone. Teaching staff can specify new service specifications or adjust existing ones through *Service Manager*, even during runtime. Based on the specified service specifications, WebHat can provide the concrete evaluation services with adapting corresponding service components into the evaluation flow (Figure 3(c)). Such adaptation can be performed dynamically to enable evaluation service adjustment during assignment practice.

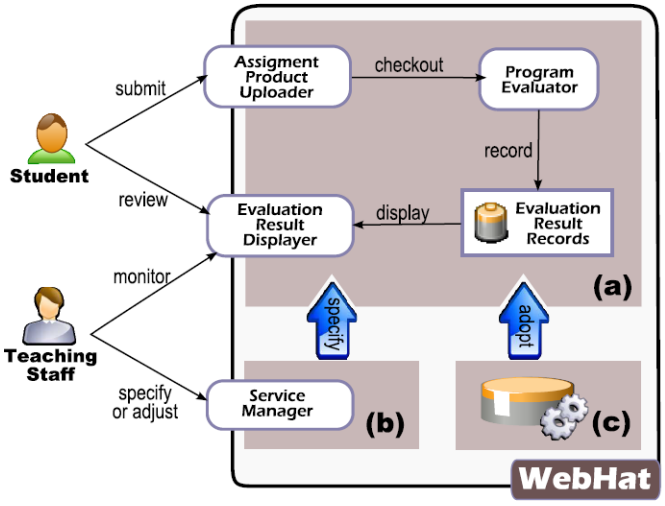

Figure 3. Architecture of WebHat system: (a) Assignment evaluation backbone. (b) Service management subsystem. (c) Service component pool.

Example assignment

WebHat was utilized in an assignment during the *Introduction to Computer Science* course. The goal of the assignment was to help students understand basic programming concepts. Since practicing continuously is the key to better understanding, it is important to motivate students to keep working on their assignment products. To attract the students' interest, the teaching staff selected a simulation game for programming practice, *Resource Craft* (Jiau et al., 2009), as the material for this assignment. In the game, each player controls game entities to find and collect resources. These resources are then transformed to the corresponding player's score. In *Resource Craft*, players aim at developing a strategy that can maximize the final score in a game with a limited duration. To develop the strategy, the player writes Java program code to query and control the game entities using defined interfaces. Through the process of strategy development, the player learns the basic programming concepts in Java. *Resource Craft* supports both single player and two-player competition modes.

Table 1. The evaluation services provided in WebHat

Group	Service Name	Description	Duration (days)
Pre-planned	1P Mode	Evaluate strategy using single player mode.	2-16
	2P Mode	Evaluate strategy using two-player competitive mode.	2-16
	Top Player	List top 5 students with high scores in 1P mode and 2P mode.	11-16
Feedback	Error MSG	Present error messages to help students identify the reason for the failed execution.	3-5

Besides applying a game-based approach, the example assignment also adopted a competitive-based evaluation method (Lawrence, 2004; Guerreiro & Georgouli, 2008) to encourage students to improve their assignment products through positive competition. In this assignment, each student was asked to develop a game strategy in *Resource Craft*, and a *Resource Craft* tournament was held by teaching staff as the final evaluation. In the tournament, every student competed with each other in two-player competition mode, and the student whose strategy attained the higher

score won that particular game. After the tournament, students were ranked according to the number of games that they had won, with better grades being given to the higher ranked students. The duration of this assignment was from November 20 2010 to December 5 2010, a total of 16 days.

For the success of this assignment practice, the teaching staff developed and released an appropriate set of evaluation services to students using WebHat, which helped them assess and further enhance their work. To enable the service implementation, *Resource Craft* runtime software was included as a service component of WebHat. Based on the service component, four online evaluation services were developed to assist students in evaluating their strategies. Once students developed stable modifications and submitted new strategy revisions to the system, they could check the results of the evaluation and then work towards further enhancement of their products.

Table 1 lists the four evaluation services, which are categorized into two groups. In the *Pre-planned* group, the three evaluation services (1P Mode, 2P Mode and Top Player) were planned in advance and prepared by the teaching staff before the assignment practice. In the *Feedback* group, the Error MSG evaluation service was developed during the assignment practice to help students resolve the difficulties faced while developing strategies. These evaluation services were not released simultaneously, but instead were released sequentially during the assignment to appropriately guide students in developing their strategies.

- **1P Mode and 2P Mode**. Providing interactive online assistance enables students to assess the effects of their assignment development (Hulls et al., 2005; Fernández Alemán, 2011). The 1P Mode and 2P Mode services are therefore provided in WebHat throughout the duration of the assignment. In 1P Mode, the service directly used the strategy submitted by the student to execute *Resource Craft*. By evaluating the strategy in a simplified and non-competitive scenario, 1P Mode helped the student check whether it behaved as expected in the early development stage. In 2P Mode, the strategy submitted by the student competed with a strategy that was developed by the teaching staff. The game score from 2P Mode helped the student further assess the competitiveness of his strategy, as the higher their score, the more competitive the strategy. By investigating the evaluation results, students gained direct feedback about the development status of their strategies.

- **Top Player**. Although 1P Mode and 2P Mode evaluation services help students evaluate their developing strategies, they are still unaware of development status of the other students. To maintain a competitive atmosphere during the assignment, the Top Player service ranked the top students with the highest final game scores in 1P Mode and 2P Mode execution. Since the Top Player service aims to encourage students to compete with each other, its effects are highly dependent on the level of the students' engagement. Hence, the Top Player service is not introduced at the beginning of assignment practice, but instead is released only when more than half the students have started the process of strategy development and some of them have developed competitive strategies. In the example assignment, the teaching staff introduced the Top Player service on day 11.

Figure 4. Screenshots of the provided services. (a) Evaluation results of the 1P Mode and 2P Mode services. (b) Error messages generated by the Error MSG service. (c) The Top Player service in the assignment.

- **Error MSG**. The activeness of students' engagement is the key factor in deciding when to introduce the Top Player service. When the 1P Mode and 2P Mode services were first released, the teaching staff, through WebHat, noticed that only few students submitted their strategies for evaluation. Furthermore, most of the strategies that were submitted failed to be executed. Motivating students to develop and submit their strategies more actively, the assistance in compilation error removal is required to lower the barriers in starting assignment development (Kelleher & Pausch, 2005). The teaching staff therefore developed and released the Error MSG evaluation service. The Error MSG service could check compilation errors of submitted strategies via Eclipse Java Development Tool (Eclipse JDT) and interpret the error messages to help students identify possible causes for failed executions. The teaching staff took one day to develop Error MSG and released it in WebHat from day 3 to day 5. By limiting the Error MSG release to the early stage of the assignment, students were expected to be more active in developing and submitting their strategies so that they could find out what was wrong with them.

Based on specifications of the evaluation services, WebHat evaluates students' submitted strategies and presents the results on a web-based interface. Figure 4 shows the screenshots of the result presentation in terms of all provided evaluation services. The upper box of Figure 4(a) lists all available evaluation result presentation choices. A student can select one of them for the result inspection. For example, the line chart of Figure 4(a) presents the results generated by 1P Mode and 2P Mode evaluation for all submitted strategies. As shown in the line chart, some zero scores are appeared in both 1P Mode and 2P Mode evaluation in the early stage of assignment practice. This indicates the difficulties in developing a workable strategy. By checking the results of Error MSG evaluation, the information with regard to compilation errors can be investigated. In Figure 4(b), an error message, "File Not Found", in revision 10 is presented and possible treatments, re-specifying an available file or re-submitting another specific file, are suggested for handling the error. The service helps students remove errors and promotes them to start assignment early.

Besides, students can assess the improvements of their strategies by checking the trends of both the 1P Mode and 2P Mode scores by the line chart of Figure 4(a). For example, the line chart in Figure 4(a) reveals that, between revisions 11 to 20, the student's 1P Mode score increased steadily while the 2P Mode score remained unchanged at approximately 50 points. This alerts the student to the fact that although he is getting familiar with the manipulation of game entities in this stage, the competitiveness of his strategy remains relatively low. The student subsequently began to find out the factors regarding strategy competitiveness and gets the upward trend for the 2P Mode score from revision 21. In the following revisions, the student can keep tracking the 2P Mode score evaluation and adjust his development plan accordingly. For example, the unexpected score decrement in revision 24 warns the student that the newly made changes make the strategy become less competitive. The student needs to perform a special treatment on these changes, such as bug fixing or strategy setting reconfiguration, to retain its competitiveness. Once the treatment is done, the student can confirm the effectiveness of the treatment through the line chart, as the strategy competitiveness continues to be improved after revision 25 in Figure 4(a).

Figure 4(c) shows a screenshot of the Top Player service, in which a bar chart represents a rank of the student in all students by their highest game scores in 2P Mode execution. As shown in Figure 4(c), the student's best score in 2P Mode evaluation came to 208 points and was ranked in top 41%. Besides, the scores of top five students are also highlighted in the ranking bar to motivate students to keep improving their strategies for better competitiveness.

Evaluation

To evaluate the effectiveness of utilizing WebHat, the empirical data from the aforementioned assignment is investigated. A total of 109 students took the *Introduction to Computer Science course*. Among these, 86 successfully submitted their final assignment products and participated in the final tournament. Each version of the strategy submitted by every student, the submission time, and the evaluation results are collected as empirical data. Two questions are examined in this investigation.

1. Do the dynamically adjusted evaluation services in WebHat enhance student engagement in assignment practice?
2. Do students with higher engagement levels in WebHat submit better final assignment products?

The question 1 is first examined by describing the effects of dynamic evaluation service adjustment on motivating students to continuously engage in assignment practice and enhance their assignment products; and then the question 2 is investigated through discussing the correlation between students' engagement level and the performance of their

final assignment products. In the following subsections, the term "strategy" is used to represent the assignment product, because students were asked to submit a *Resource Craft* strategy to complete the assignment.

Effects of dynamic evaluation service adjustment

Table 2. Summary of student submissions in the early phase of assignment

day	# submit	# success	#success/#submit
1	0	0	0
2	4	0	0
3	14	5	0.36
4	25	17	0.68
5	15	9	0.60

Error MSG and Top Player were released at different stages of the assignment practice, with both aimed at motivating students to engage in the assignment practice. Error MSG was developed to encourage students to start the assignment practice and submit executable strategies early in the process, while Top Player was aimed at motivating the students to focus on enhancing the competitiveness of their strategies before the final tournament. This section examines whether these two services had the expected effects in enhancing student engagement during assignment practice.

Effects of Error MSG. Table 2 summarizes the daily submission count (**#submit**) in the early stages of assignment practice. The **#success** column shows the number of successful submissions (i.e., those with successfully executed strategies). The daily successful execution ratio is shown in the **#success/#commit** column. On day 2, the 1P Mode and 2P Mode evaluation services were first released and students began to submit their strategies. Four strategies were submitted and all of them failed in execution. With the aim of encouraging the students to start work, the Error MSG service was provided from days 3 to 5. Right after the release of the Error MSG service on day 3, the number of submissions greatly increased (up to 14) and the students started to submit successfully executed strategies (a total of five on this day). On days 4 and 5, the daily successful execution rate was significantly higher, with 25 and 15 submissions, respectively, about 65% of which were executed successfully. This indicates that the Error MSG service did indeed motivate the students to develop strategies earlier, as well as assisted them developing successfully executed strategies.

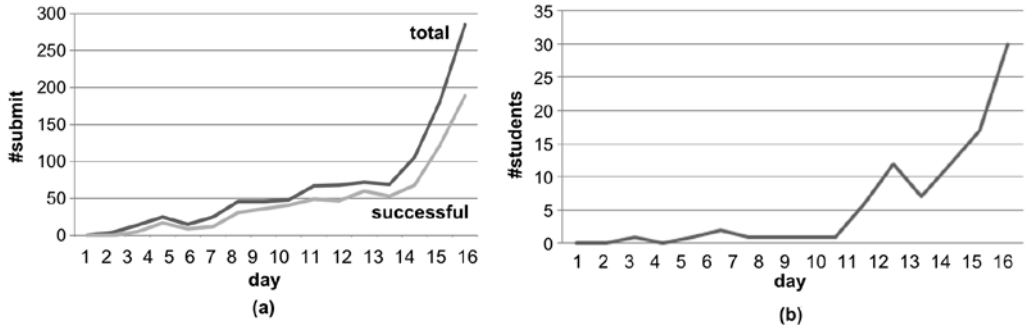

Figure 5. The submission statistics over the duration of the assignment. (a) Total and successful submission. (b) Competitive strategy submission

Error MSG also caused a chain reaction in motivating students to start their strategy development earlier, as it motivated more active students to start work, which then caused other students to begin the assignment. As a result, the positive effect of Error MSG was sustained throughout the rest of the assignment. Figure 5(a) presents the number of daily submissions, and even after terminating the Error MSG on day 5, the number of total and successful submissions still increased stably over time.

Impacts of Top Player. Figure 5(b) presents the daily number of students who submitted competitive strategies. A strategy is classed as competitive if it achieves a score of more than 200 points in the 2P Mode evaluation service. Before day 10, there were already some students (one or two per day) who submitted competitive strategies. The Top

Player service was released on day 10 to bring a more competitive atmosphere to the assignment. By announcing the best performing students, the Top Player service successfully encouraged others to develop more competitive strategies. As shown in Figure 5(b), after releasing Top Player service on day 11, the number of students who submitted competitive strategies rapidly increased throughout the final stage of the assignment practice.

Correlation between engagement level and final performance

The engagement level of a student represents the degree of activeness with which they are engaged in the strategy development. To quantize the student engagement level, the assignment practice duration is split into several periods. The student engagement level is measured in terms of $level_{engaged}$, the number of periods in which a student submitted a strategy. Since the teaching staff expected that it took about three days for a student to revise a strategy and the duration of the assignment (from day 2 to day 16) is split into five periods of three days each. As a result, the $level_{engaged}$ of a student is formulated as followed.

$$level_{engaged}(student) = \sum_{i=1}^{5} engaged_i(student),$$

$$\text{where } engaged_i = \begin{cases} 1 & \text{if student had submitted strategy} \\ & \text{between day 3i - 1 and 3i + 1} \\ 0 & \text{otherwise} \end{cases}$$

The 86 students are divided into three groups according to their $level_{engaged}$. G_{active} contains 23 students whose $level_{engaged}$ is 4 or 5. G_{mid} contains 36 students whose $level_{engaged}$ is 2 or 3. $G_{inactive}$ contains 25 students whose $level_{engaged}$ is 1.

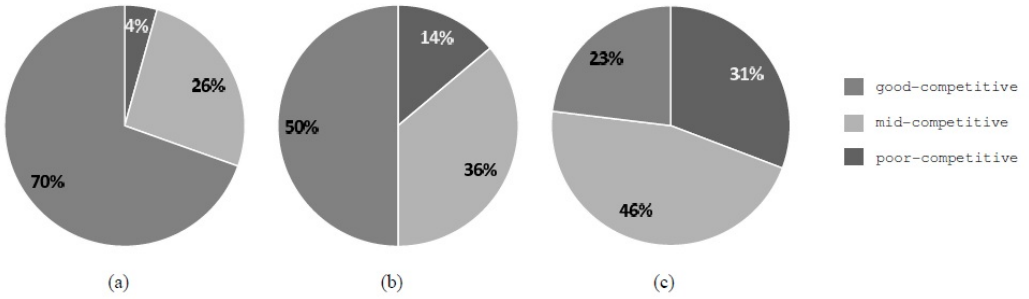

Figure 6. The score distribution in students with different engagement levels over time (a) G_{active}, (b) G_{mid}, and (c) $G_{inactive}$

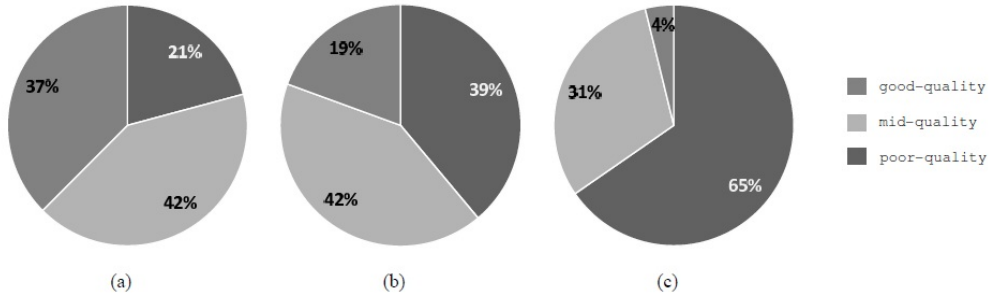

Figure 7. The quality distribution in students with different engagement levels over time (a) G_{active}, (b) G_{mid}, and (c) $G_{inactive}$

Each final submitted strategy was evaluated from two perspectives, *competitiveness* and *quality*. The *competitiveness* of a strategy is represented in terms of the score it attained in the final tournament. The final submitted strategies can be categorized into three different categories: *well-competitive* (with a score higher than 200), *mid-competitive* (between 100 and 200) and *poor-competitive* (less than 100). Figure 6 summarizes the proportions of different competitive categories in three groups. For example, Figure 6(a) shows that there are 70% of students in G_{active}

submitted well-competitive strategies and only 4% with poor-competitive ones. Compared to Figure 6(b) and 6(c), the proportions of students with well-competitive strategies declined along with the engagement level (50% in G_{mid} and 23% in $G_{inactive}$). Students with a higher engagement level produce more competitive strategies.

To assess the quality of the strategies submitted, the following two quality attributes are taken into account: (1) Code structure organization: whether a student defined methods or fields in organizing the code structure of the strategy; and (2) Runtime exception avoidance: whether the submitted strategy performed correctly without any runtime exceptions during execution. The final submitted strategies can thus be categorized into three different groups: *good-quality*, *mid-quality*, and *poor-quality*. A *good-quality* strategy has proper code structure organization and is well implemented to avoid runtime exceptions. A *mid-quality* strategy performs well in either of those two quality attributes. The *poor-quality* strategy is both poor in organizing code structure and in avoiding runtime exceptions. Figure 7(a)-(c) summarizes the proportions of different quality categories in the three groups. The data show that students with higher engagement levels submit strategies with higher product quality. The proportions of *good-quality* in G_{active}, G_{mid}, and $G_{inactive}$ are 37%, 19%, and 4%, respectively. The high proportion of *poor-quality* in $G_{inactive}$ (65%) also indicates that inactive students only focused on developing a workable strategy, without considering the quality.

Summary and discussion

The evaluation results, which are presented along with different stages of the assignment practice, indicate the effectiveness of the WebHat platform. Monitoring student status through WebHat enables a timely identification of students' difficulties. By providing dynamic service adjustment, teaching staff can immediately offer corresponding online assistance. In this study, several quantitative monitoring attributes, such as the successful execution ratio and the number of competitive strategies, can confirm student engagement during assignment practice. This study shows that student difficulties in developing workable strategies can be identified at the early stage of the assignment. Corresponding Error MSG service can be immediately provided to assist in error removals. As shown in the results, the Error MSG service enhances the successful execution ratio of submitted strategies.

The enhancement of student engagement at the early stage leads to the success of the assignment practice. Students are promoted to start their assignment development before day 5 and a number of competitive strategies are developed before day 10. In this situation, the Top Player service can further motivate students in the strategy improvement before final tournament. After releasing Top Player service, the number of daily submissions is highly increased. Finally, the evaluation results show a positive correlation between engagement level and final performance. The effectiveness of WenHat platform is confirmed in enhancing student engagement for the achievement of assignment objectives.

Even though the results are positive and encouraging, two limitations must be clarified. First, the subject assignment practice may not be representative of programming assignment practice in general. This study demonstrates the effectiveness of supporting dynamic service adjustment in the assignment practice designed by integrating specific teaching methods, assignment practice using game (Jiau et al., 2009) and assignment evaluation by a competition (Lawrence, 2004; Guerreiro & Georgouli, 2008). However, the validity of the results may be impaired for other assignment types like traditional programming practice in a problem-solving style (Deek et al., 1999; Daly & Horgan, 2004). Second, the current implementation of WebHat platform mainly supports evaluation on specific input and output of a program. For other assignment practice, the development of additional evaluation services, such as the measurement of code quality, may be required. Such extension is adapted by the design of WebHat platform. Teaching agility is supported during assignment practice using WebHat.

Conclusion

Ensuring that students are engaged in a programming assignment is critical to achieve teaching objectives in computer science education. This paper introduces WebHat platform which applies a service-oriented technique to improve assignment practice of e-learning systems. WebHat supports continuous student status monitoring and dynamic evaluation service adjustment. Teaching staff can therefore assist students promptly, actively and directly during a programming practice. The utilization of WebHat is presented along with an example assignment. The

results of the study show: (1) The dynamic evaluation service adjustment encourages students to start assignment development early and to keep evolving their assignment product in the expected direction. (2) Students with higher engagement levels in WebHat submit better final assignment products with regard to both functionality and quality. The results clearly demonstrate the effectiveness of WebHat in enhancing student engagement to achieve assignment objectives.

WebHat introduces a service-based mechanism to support dynamic evaluation service adjustment and has been applied in *Introduction to Computer Science* course for several semesters in the Department of Electrical Engineering, National Cheng Kung University, Taiwan. Currently, evaluation service development and adjustment in WebHat are still dependent on experience of teaching staff. In the future, the evaluation services and their management will be further patterned in WebHat for specific assignments to accumulate experience of successful teaching. WebHat therefore can provide complete support from assignment design to practice for programming teaching.

References

Allevato, A., Thornton, M., Edwards, S. H., & Pérez-Quiñones, M. A. (2008). Mining data from an automated grading and testing system by adding rich reporting capabilities. In R. S. J. Baker et al. (Eds.), *Proceedings of the 1st International Conference on Educational Data Mining* (pp. 167–176). Retrieved from http://www.educationaldatamining.org/EDM2008/uploads/proc/17_Allevato_23.pdf

Blikstein, P. (2011). Using learning analytics to assess students' behavior in open-ended programming tasks. In P. long et al. (Eds.), *Proceedings of the 1st International Conference on Learning Analytics and Knowledge* (pp. 110-116). Banff, AB, Canada: ACM.

Cano, M.-D. (2011). Students' involvement in continuous assessment methodologies: A case study for a distributed information systems course. *IEEE Transactions on Education, 54*(3), 442–451.

Daly, C., & Horgan, J. M. (2004). An automated learning system for Java programming. *IEEE Transactions on Education, 47*(1), 10–17.

Deek, F. P., Turoff M., & McHugh, J. A. (1999). A common model for problem solving and program development. *IEEE Transactions on Education, 42*(4), 331–336.

Edwards, S. H., & Pérez-Quiñones, M. A. (2008). Web-CAT: Automatically grading programming assignments. In J. Amillo et al. (Eds.), *Proceedings of the 13th AnnualConference on Innovation and Technology in Computer Science Education* (pp. 328-328). Madrid, Spain: ACM.

Fenwick, J., Norris, C., Barry, F., Rountree, J., Spicer, C., & Cheek, S. (2009). Another look at the behaviors of novice programmers. In S. Fitzgerald et al. (Eds.), *Proceedings of the 40th ACM Technical Symposium on Computer Science Education* (pp. 296–300). Chattanooga, TN, USA: ACM.

Fernández Alemán, J. L. (2011). Automatic assessment in a programming tools course. *IEEE Transactions on Education, 54*(4), 576-581.

Fredricks, J. A., Blumenfeld, P. C., & Paris, A. H. (2004). School engagement: Potential of the concept, state of the evidence. *Review of Educational Research, 74*(1), 59–109.

Funabiki, N., Fukuyama, Y., Matsushima, Y., Nakanishi, T., & Watanabe, K., (2012). An improved Java programming learning system using test-driven development method. In S. I. Ao et al. (Eds.), *Proceedings of the International MultiConference of Engineers and Computer Scientists* (pp. 1-6). Hong Kong, China: Newswood Limited.

Funabiki, N., Nakanishi, T., Amano, N., Kawano, H., Fukuyama, Y., & Isogai, M. (2010). Software architecture and characteristic functions in learning management system "NOBASU." *Proceedings of the International Symposium of Applications and the Internet* (pp. 109-112). Retrieved from http://doi.ieeecomputersociety.org/10.1109/SAINT.2010.24

Guerreiro, P., & Georgouli, K. (2008). Enhancing elementary programming courses using e-learning with a competitive attitude. *International Journal of Internet Education, 10*, 1–6.

Hansen, S., & Eddy, E. (2007). Engagement and frustration in programming projects. In I. Russell et al. (Eds.), *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education* (pp. 271–275). Covington, Kentucky, USA: ACM.

Hulls, C. C. W., Neale, A. J., Komalo, B. N., Petrov, V., & Brush, D. J. (2005). Interactive online tutorial assistance for a first programming course. *IEEE Transactions on Education, 48*(4), 719–728.

Jiau, H. C., Chen, J. C., & Ssu, K.-F. (2009). Enhancing self-motivation in learning programming using game-based simulation and metrics. *IEEE Transactions on Education, 52*(4), 555–562.

Law, K. M. Y., Lee, V. C. S., & Yu, Y. T. (2010). Learning motivation in e-learning facilitated computer programming courses. *Computers & Education, 55*(1), 218-228.

Lawrence, R. (2004). Teaching data structures using competitive games. *IEEE Transactions on Education, 47*(4), 459–466.

Kelleher, C., & Pausch, R. (2005). Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys, 37*(2), 83–137.

Munns, G. & Woodward, H. (2006). Student engagement and student self-assessment: The REAL framework. *Assessment in Education: Principles, Policy and Practice, 13*, 193–213.

Murphy, C., Kaiser, G., Loveland, K., & Hasan, S. (2009). Retina: Helping students and instructors based on observed programming activities. In S. Fitzgerald et al. (Eds.), *Proceedings of the 40th ACM Technical Symposium on Computer Science Education* (pp. 178–182). Chattanooga, TN, USA: ACM.

Norris, C., Barry, F., Fenwick, J., Reid, K., & Rountree, J. (2008). ClockIt: Collecting quantitative data on how beginning software developers really work. In J. Amillo et al. (Eds.), *Proceedings of the 13th Annual Conference on Innovation and Technology in Computer Science Education* (pp. 37–41). Madrid, Spain: ACM.

Papazoglou, M. P. (2003). Service-oriented computing: Concepts, characteristics and directions. *Proceedings of the 4th International Conference on Web Information System Engineering* (pp. 3–12). doi: 10.1109/WISE.2003.1254461

Papazoglou, M. P., Traverso, P., Dustdar, S., & Leymann, F. (2008). Service-oriented computing: A research roadmap. *International Journal of Cooperative Information Systems*, *17*(2), 223–255.

Pisan, Y., Richards, D., Sloane, A., Koncek, H., & Mitchell, S. (2003). Submit! A web-based system for automatic program critiquing. In T. Greening and R. Lister (Eds.), *Proceedings of the 5th Australasian Conference on Computing Education* (pp. 59–68). Adelaide, Australia: Australian Computer Society.

Spacco, J., Hovemeyer, D., Pugh, W., Emad, F., Hollingsworth, J. K., & Padua-Perez, N. (2006). Experiences with marmoset: Designing and using an advanced submission and testing system for programming courses. In R. Davoli et al. (Eds.), *Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education* (pp. 13–17). Bologna, Italy: ACM.

Verdú, E., Regueras, L. M., Verdú, M. J., Leal, J. P., Castro, J. P., & Queirós, R. (2011). A distributed system for learning programming on-line. *Computers & Education, 58*(1), 1-10

Wu, Y.-C., & Jiau, H. C. (2012). A monitoring mechanism to support agility in service-based application evolution. *SIGSOFT Software Engineering Notes, 37*(5), 1–10.