Paper ID #30501

Integrating Role-Playing Gamification into Programming Activities to Increase Student Engagement

Mr. Zhiyi Li
Prof. Stephen H Edwards, Virginia Tech

Stephen H. Edwards is a Professor and the Associate Department Head for Undergraduate Studies in the Department of Computer Science at Virginia Tech, where he has been teaching since 1996. He received his B.S. in electrical engineering from Caltech, and M.S. and Ph.D. degrees in computer and information science from The Ohio State University. His research interests include computer science education, software testing, software engineering, and programming languages. He is the project lead for Web-CAT, the most widely used open-source automated grading system in the world. Web-CAT is known for allowing instructors to grade students based on how well they test their own code. In addition, his research group has produced a number of other open-source tools used in classrooms at many other institutions. Currently, he is researching innovative for giving feedback to students as they work on assignments to provide a more welcoming experience for students, recognizing the effort they put in and the accomplishments they make as they work on solutions, rather than simply looking at whether the student has finished what is required. The goals of his research are to strengthen growth mindset beliefs while encouraging deliberate practice, self-checking, and skill improvement as students work.

Integrating Role-Playing Gamification into Programming Activities to Increase Student Engagement

Abstract

A number of gamification approaches have been used to encourage greater student motivation and engagement in the classroom. This paper examines a gamification strategy that is less common in the classroom, despite its prevalence in successful games: role playing. Role playing games (RPGs) use a combination of character traits, experience points, and character leveling to illustrate how a character evolves and grows stronger as the character progresses through the game. By providing strong connections between the player's behavior and choices and how their in-game character develops, RPGs encourage players to identify with and develop a sense of ownership over their in-game character or persona. In order to encourage adoption of positive student habits and encourage belief in positive values of student behavior, we describe how RPG elements such as experience points, leveling, and character traits can be adapted for use in computer programming activities. By providing feedback on time management behavior, use of incremental software development practices, and consistent self-checking of one's work, gamification elements can be strategically structured to reinforce specific behaviors and to communicate that specific student practices or work habits are valued. By structuring RPG character traits to convey an implicit model of "desirable" or "strong" student behavior, students can use their character development status to track how well they are modeling these behaviors. Further, by providing a way for a student to identify with an empathize with an "in-assignment" representation of their personal behavior, RPG elements promote greater student ownership over their own experiences and greater personal investment in improving their work habits. We rely on existing measures of student effort as they develop their solutions that are intended to measure productive work the student invests, and to promote growth mindset beliefs. We show how these measures can be used to provide gamification-based rewards for target behaviors, such as time management choices, incremental development, and self-checking. Finally, we show how these elements can be embedded in an automated grading tool to provide a platform for embedding RPG-like experiences in assignment feedback. We apply these techniques to a historical data-set of student activities including 257 students to verify the feasibility and suitability of the design.

Keywords: Assessment, RPGs, XPs, Levels, Traits(Characters), Growth Mindset, Engagement

1 Introduction

Feedback information in automatic grading systems such as Web-CAT[9] usually aids students when learning programming. Examples of such information include a score of program correctness, percentage of test coverage, and hints for software errors, etc. However, this information is machine generated, performance-based, impersonal, and often negative. Such feedback can easily discourage, frustrate, or even turn off students, especially novice programmers. These negative effects may lead students to belief in a *fixed mindset*–believing intelligence and ability is something you're either born with, or without[7]. *Fixed mindset* belief can have serious effects in computer science education such as low CS major enrollment/retention rate, and stereotype threat to minority/female students, etc[2].

One way to counteract these effects is to design automated feedback that encourages belief in *growth mindset*—believing that ability and skill can be improved through practice and hard work[7]. Edwards et al. designed and implemented a suite of fifteen indicators to reflect students' progress and effort based on students' submissions[8]. These indicators span different aspects of students programming activities and measure positive trends of students' effort. Another example effort is that Goldman developed daily missions tasks based on these indicators in Web-CAT. Students were provided the opportunities to accomplish daily missions tasks to win rewards such as extra submission energy[11].

Studies indicate that gamification can motivate and engage students in their learning process[20][16]. Especially Toth et al. integrated Role-Playing Game (RPG) elements into computer science education experiences to improve the learning environment[24]. Inspired by their work and previous effort to improve feedback in automatic grading tools, we demonstrate a strategy for incorporating Role-Playing Games (RPGs) elements into automated grading feedback to help student learning. The research question is how can we motivate students with these new RPG elements, and whether this RPG-based strategy is feasible and rational. We first designed RPG elements such as Experience Points (XPs) and levels as rewards and recognition of progress as students work. Then we designed RPG-style character traits to reflect expected programming behavior patterns we want students follow. Examples of these expected behavior patterns are good time management, incremental development, etc. In the design scenario, the values of these integrated RPG elements improve gradually and visible as students make progress in working on their assignments. Positive information such as earning experience points (XP), level up, increase characters(traits) value were conveyed to students to encourage and motivate students.

Then we validate the design of these RPG elements against a historical CS2 programming data set including 257 students and 981 separate submission sequences. The design constraints and details such as traits range value change, level numbers, and earning experience points (XP) numbers are decided or tuned in the simulation.

Together with previous efforts, the main goal of these new RPG elements is to provide an integrated, multi-pronged strategy to engage students and convey positive information to encourage them towards the belief in a *growth mindset*–intelligence and ability can be improved by practice and hard work.

The remaining of this paper is structured as follows. Section 2 discusses related work to improve feedback in Web-CAT and applications of gamification in education area. Section 3 is to show the design of RPG elements and how to integrate them into the feedback in Web-CAT. Section 4 is evaluation part to demonstrate the simulation of our designed RPG elements with a historical programming data set. Section 5 is conclusions and future work.

2 Background and Related Work

2.1 Previous Effort to Improve Feedback

In order to improve feedback mechanism in assessment tools such as Web-CAT, encourage students with positive information especially lead to growth mindset belief, Recently effort were made. A suite of fifteen indicators were designed and implemented to reflect students' progress and effort based on their program submissions[8]. These fifteen indicators compose of seven general purpose and eight software testing indicators. The main purpose of these fifteen indicators is to measure students' positive trend of effort by comparing difference between current submission with previous submissions. The suitability of these fifteen indicators were verified by a two semester historical CS2 programming data set. Goldman [11] developed daily missions in feedback to encourage students based on these indicators. Students can accomplish a small set of tasks in daily time period (24 hours). These daily mission tasks were designed based on a subset of indicators. Once students accomplish daily tasks, they will be awarded extra submission energy—more submission opportunities than regular in a limited time period. Mukund et al. [19] designed a virtual teaching assistant (TA) named Maria in Web-CAT to help students alleviate negative emotions toward programming. Maria also give motivating or encouraging comments to move students towards growth mindset trend. Kazerouni et al. [14 15] designed and implemented metrics to reflect students' expected programming behaviors such as time management and incremental development, based on log information collected by Eclipse plugin *DevTracker*.

2.2 Applications of Gamificaton in Education Area

Gamification is defined to apply game design elements in non-game contexts[5]. Gamification in education is an approach to increase learners' motivation and engagement by incorporating game design elements into educational environments [6]. In Gamification of education, game design elements may include: avatar, badges, leaderboards, levels, points, and virtual goods, etc. Goehle applied video game elements levels and achievements with online homework program *WeBWorK* to increase student motivation and engagement[10]. Li et al. developed a tutorial system named GamiCAD for first-time AutoCAD users. They applied gamified components challenging levels, rewards in form of bonus levels and bonus missions, etc to increase student engagement[17]. Studies examine the impact of gamification on students' learning activities in different perspectives such as performance, engagement, behavior, motivation, participation, and retention, etc[6 13 4 12]. However, gamification in education does not always work and sometimes even has negative effects. Their effect vary in different educational learning contexts [4].[23].

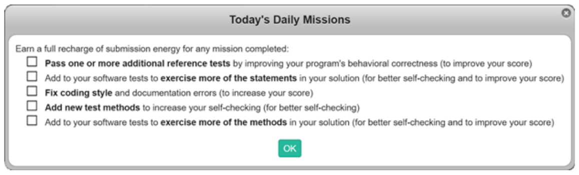
3 Design

Our main objective is to augment current feedback in Web-CAT with RPG elements to encourage growth-mindset belief and encourage positive behavior patterns such as time management. Inspired by gamification education work[6 13 4 12], we have several candidate RPG components to select from, such as: leaderboards, scores, rewards, avatars, progress bars, experience points/levels etc. However, we should be careful to select and design appropriate RPG elements to avoid performance-based game elements leading to students' *fixed mindset* or stereotype threat. For example, Christy showed leaderboards in the classroom may have unexpected effects. They may lead to stereotype threat for female students, based on women's math performance information[3]. Initially we design three game components: experience points (XP)/levels, and traits (characters) together towards our goal. However, these RPG elements were purposely designed individualized and to avoid competition among students. Students can only visualize and compare their individualized performance with their own historical performance in feedback and not compare with their peers. Since similar to leaderBoard, RPG competition among students may frustrate lagged students to lead *fixed mindset* trend. One of characters of students with fixed mindset is they feel threatened by the success of others [1].

3.1 Experience Points (XP) and Levels

In RPGs, experience points (XP) and levels are often used to reward players and demonstrate their progress through the game. Players earn XP and level up by accomplishing tasks such as defeating enemies, overcome obstacles, pick up trophies, etc. We design XP and levels in a similar way but in an educational context.

The first design consideration is whether XP/levels are applicable for only a single course or transferable across multiple courses. In RPGs, XP/levels are not transferable between games since different games have different mechanisms. However, XP/levels inside one game can be consecutive in different phases since the mechanism are similar. For example, Dungeons & Dragons has three tiers for levels: ”Heroic” for levels 1–10, ”Paragon” for levels 11–20, and ”Epic” for levels 21–30[25]. In our university, the computer science curriculum is designed to teach students in ascending difficulty levels. Freshman students normally take introductory programming courses like CS1. Then sophomore and junior students level up to take intermediate data structures and architecture courses. Finally senior students accomplish advanced level courses such as system/database/network courses and capstone project courses. In RPGs, experience points (XP) are used to quantify a player's (or character's) progression through the game. XP can be implemented in different ways. Level-based progression XP are widely applied: Players win enough XP as rewards to reach next higher level[27]. Players in the next level will have increased ability. We want to design level-based XP to reflect students' progress through their courses. However, we want to avoid associating XP directly with performance-based criteria such as students' assignment scores, since this may cause unexpected negative effects. A suite of indicators that assess students' progress and effort based on their submissions[8] are a possible candidate measure for XP. However, the triggered indicator information is hidden from students. If we associate indicators with XP/levels directly, students are not aware of them and cannot associate XP changes with indicators' triggered information. We have to somehow connect XP

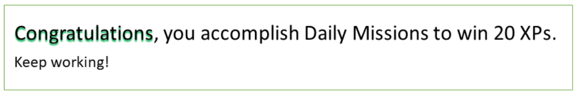


Today's Daily Missions

Earn a full recharge of submission energy for any mission completed:

- ☐ **Pass one or more additional reference tests** by improving your program's behavioral correctness (to improve your score)
- ☐ Add to your software tests to **exercise more of the statements** in your solution (for better self-checking and to improve your score)
- ☐ **Fix coding style** and documentation errors (to increase your score)
- ☐ **Add new test methods** to increase your self-checking (for better self-checking)
- ☐ Add to your software tests to **exercise more of the methods** in your solution (for better self-checking and to improve your score)

Figure 1: An example of daily missions interface, from Goldman et. al's work[11]



Congratulations, you accomplish Daily Missions to win 20 XPs.
Keep working!

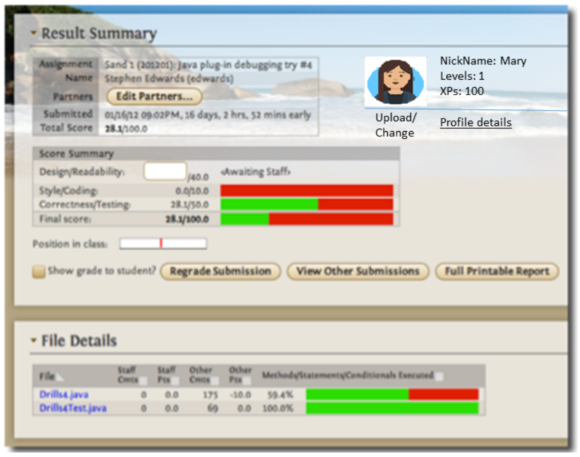
Figure 2: An example of notification to a student, after a daily mission task accomplished.

with students' visible and understandable gains. Fortunately, Goldman et al.[11] designed daily missions in Web-CAT to challenge students with small, achievable goals, based on a subset of indicators' triggered information during each 24-hour time period. The daily mission tasks span aspects of software design, implementation, and testing activities when students work on their assignments. Examples of daily missions are improving code modularity, improving documentation, and increasing self-testing, etc. Each day, a randomly selected set of 5 sub-tasks are presented to students to accomplish. If students accomplish daily missions, they will be awarded with perks that give extra submission opportunities for their assignment for that day. Figure 1 shows a daily missions interface[11]. We design XP to work together with this daily missions interface: rewarding students through an XP increase, such as 20 XP points, when they accomplish a daily missions task. Figure 2 shows the designed feedback dialog box to students when they accomplish a daily missions task.

Once we designed XP, we continue to design levels. Recall that the objective of XP/level design is to reward students to show their progress and recognize their effort. The naive strategy is to divide XP range by levels evenly. e.g., XP from 1 to 100 is specified level 1, XP from 101 to 200 is specified as level 2, and so on. However, this evenly division strategy is boring and don't have effect we desired. In current RPG games such as Dungeon & Dragon, Pokemon, and Final Fantasy, as players progress, it becomes harder to reach next level: more XP increase are needed. These RPG games apply exponential or quadratic XP increase to level up. We design exponential next level function similar to Final Fantasy as[22]:

```
function nextLevel(level)
    local exponent = 1.5
    local baseXP = 10
    return math.floor(baseXP*(level^exponent))
end
```

Parameter *exponent* defines the difficulty between levels and how difficulty increases as level up. If exponent is 1, the level change aren't getting increasingly harder. If exponent is very large ¿= 5, the difficulty of level change will be very hard. We tune and find these parameters in next level



Result Summary

Assignment	Sand 1 (201201): Java plug-in debugging try #4
Name	Stephen Edwards (edwards)
Partners	Edit Partners...
Submitted	01/16/12 09:02PM, 16 days, 2 hrs, 52 mins early
Total Score	28.1/100.0

NickName: Mary
Levels: 1
XPs: 100
Upload/ Change
Profile details

Score Summary		
Design/Readability:	/40.0	<Awaiting Staff>
Style/Coding:	0.0/10.0	
Correctness/Testing:	28.1/50.0	
Final score:	28.1/100.0	

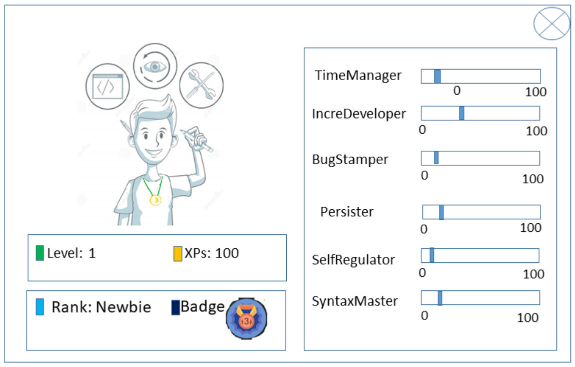
Position in class:

Show grade to student? Regrade Submission · View Other Submissions · Full Printable Report

File Details

File	Staff Cmts	Staff Pts	Other Cmts	Other Pts	Methods/Statements/Conditionals Executed
Drills4.java	0	0.0	175	-10.0	59.4%
Drills4Test.java	0	0.0	69	0.0	100.0%

Figure 3: Web-CAT feedback main interface



Level: 1 · XPs: 100
Rank: Newbie · Badge

TimeManager	0	100
IncreDeveloper	0	100
BugStamper	0	100
Persister	0	100
SelfRegulator	0	100
SyntaxMaster	0	100

Figure 4: Student profile with traits(characters)

function.

3.2 Students Traits(Characters)

In Role-Playing Games (RPGs), players' attribute normally is a data to define how far to reach certain fictional ability or character. e.g., in Dungeons & Dragons games player has six attributes: Strength, Intelligence, Wisdom, Constitution, Dexterity, Charisma[26]. In educational game design, progress bar is often used to reflect students' status in learning process[16]. Inspired by RPGs and progress bar, we design students' traits(characters) to reflect expected programming behaviors we want to students work towards. Similar to experience points(XP) and levels, students traits value is designed in range from 0 to 100 and spans five CS courses and each course with four assignments. For each assignments students are expected to get 5 points traits value gain at most. Figure 4 demonstrates six traits (characters) we designed in student profile interface. We design student profile interface can be accessed from detail button in top right corner of main feedback interface in Web-CAT. Student profile interface shows basic information of a student such as avatar, XP/Levels, rank, badges, and traits(characters).

The six students traits are:

- **Time Manager**: students' efficient time management behavior pattern.

  It is based on metric *Working Early and Often* in Kazerouni et al work[15]. In their work, Metric *Working Early and Often* was originally defined as mean and median of time period of between students' edit events and project due time, captured by Eclipse Plugin *DevEventTracker* when students submit their assignments. The larger the metric *Working Early and Often* 'value, the better students' time management and avoidance of procrastination.

- **Incremental Developer**: students work their assignment incrementally.

  Similarly, this trait is also based on metrics from Kazerouni et al. work[14]. They designed three metrics: *Incremental Checking*, *Incremental Test Checking*, and *Incremental Test Writing* to reflect students' incremental effort. All these three metrics are also based on events captured by *DevEventTracker*, but in the range of intervals between events.

- **Bug Stamper**: students' ability to find and fix program bugs.

  This trait *Bug Stamper* is designed to base on the indicator *Increasing Correctness*. For each submission, we calculate sum of the indicator *Increasing Correctness* triggered in current and historical submissions. Then apply normalization to get the value of the trait *Bug Stamper*.

- **Persistence**: students' continuous effort or work even facing on difficulties and obstacles.

  We define measurable students' persistence pattern in two phases: Phase 1: Non-progress. Students make sequential submissions showing no-progress(stuck) based on measurable parameters such as indicator *Increasing Correctness*[8]; Phase 2: Breakthrough. Students finally make progress in coming submission reflected by obvious positive change in measurable parameters. The frequency of persistence pattern in students is used for this student trait.

- **Syntax Mastery**: Students' proficiency to find and correct syntax errors.

  Syntax errors can be detected in program compile and execution time. Students' program will not compile and execute correctly until syntax errors were fixed. Normally students fix syntax errors by compile and test their solutions with IDEs before make their submissions in Web-CAT. But in some conditions students make submissions even their program cannot compile or execute, especially when project deadline is approaching or testing time. Web-CAT can compile and execute students' solutions. *Syntax Mastery* can be measured directly in Web-CAT as ratio of submissions of program of successfully compile and execute to all submissions for each assignment.

- **Self-Regulator**: students' strategy to work their assignments wisely.

  Self-regulation is originally defined as the ability to be aware of one's thoughts and actions and evaluate how well they are moving towards a goal [21]. In the contexts of students' learn programming process, in order to accomplish a program solution, self-regulation can be composed of multiple phases activities such as reiterating problem, search for analogous problems, implement and evaluate solutions, etc [18]. One of self-regulation element is monitoring process towards a solution. Web-CAT main feedback interface provides rich information to let students trace and compare the change of their submissions. Currently We design Self-Regulator as how often students check Web-CAT feedback interface in their whole submission period and how much parameter score change between adjacent submissions.

These six students traits are designed to monotonous increasing and span multiple courses and assignments. Student can visualize them in profile interface when they are working on their assignments.

3.3 Automatic Integration of RPGs into Web-CAT

We design RPG elements XP, levels and traits(characters) together and integrate them into feedback mechanism in automatic grading system web-CAT. When students work on their assignment to make submissions in Web-CAT, in the backend, Web-CAT automatically monitor triggered indicators' information in their submissions. Based on these triggered indicators' information, RPG elements were calculated. Students can earn experience points(XP), level up, improve their traits value as they make progress when working assignments. On top-right corner of main Web-CAT main feedback interface, there is a profile button. When students click the profile button, as shown in Figure 3. Students can view profile interface when they are work on their assignments, as shown in Figure 4. The profile interface basically shows ranking information reflected by XP, levels, traits value. At the same time, students will also be provided daily missions opportunity to improve their status demonstrate by RPG elements value.

4 Evaluation of Suitability

4.1 Evaluation of Experience Points and Levels

Before integrating the designed XP/levels into feedback in Web-CAT, we evaluated these new components' rationalization and feasibility, with a historical students program cohorts. Recall in design part XP/Levels span through multiple courses instead of a single one. Let's say five courses in total. However, we only have one historical data set available, collected from two semesters of CS2 Data Structure and Algorithm course at XXX XXX. In this cohort total 257 students made 981 separate program solutions with total 20,363 submissions. Each students made approximate 4 assignment submissions. We simulate to have five sequential courses of data set by applying one copy of our available data set for each five individual courses. Although the difference between these five sequential courses such as difficulty levels may exist, they are ignored in our evaluation.

We designed students can win XP through multiple ways: either change their behaviors in expected patterns such as more efficient time management, or accomplish daily missions tasks. In order to simplify our simulation, the former case is left for future and currently we only consider the latter case: students were provided daily missions tasks opportunities and accomplish them to win XP. A student can earn 20 XP once they finish a daily missions task.

Table 1 shows statistic results for XP earned at the end of each five courses in our

Table 1: Statistic results for XP distribution at the end of each course in simulation.

name	mean	sd
End of course 1	100.69	52.53
End of course 2	201.46	102.28
End of course 3	302.38	150.94
End of course 4	402.99	201.66
End of course 5	505.21	250.85

Table 2: Level/XP distribution in courses 1-5 in simulation

Course1		Course2		Course3		Course4		Course5	
Level	XP	Level	XP	Level	XP	Level	XP	Level	XP
1	10	6	150	9	270	12	415	15	580
2	30	7	185	10	320	13	470	16	640
3	50	8	230	11	365	14	525	17	700
4	80	9	270	12	415	15	580	18	765
5	110								
6	150								

Notes: XP value is upper bound students can reach in corresponding level.

simulation.

Recall in the design part we use exponential level up function to calculate levels. We tuned level up function parameters *baseXP* and *exponent* value correspondingly based on statistic results of XP in tables 1, especially mean values of each five courses. We find *BaseXP* value 10, *exponent* value 1.5 meet with our design goal. We calculate the distribution of levels/XP in courses 1-5 in simulation, as shown in table 2.

From table 2 we can see students' levels gradually span courses 1-5 to reach up to level 18. Students level up approximate 4 levels for each five courses.

We also plot a histogram of students numbers percentage vs levels at the end of each five courses, as shown in figure 5. The horizontal axis is level number, vertical axis is percent of students number. From figure 5 we can see as students work from course 1 to course 5, major students gradually move from low levels to high levels.

4.2 Evaluation of Students' traits

Six students traits were designed to encourage expected programming patterns we want students follow. Similar to experience points (XP) and levels, we run simulation to validate these traits value and range change, to find appropriate threshold value and any constraints. We applied same historical CS2 programming dataset used before in experience points(XP) and levels

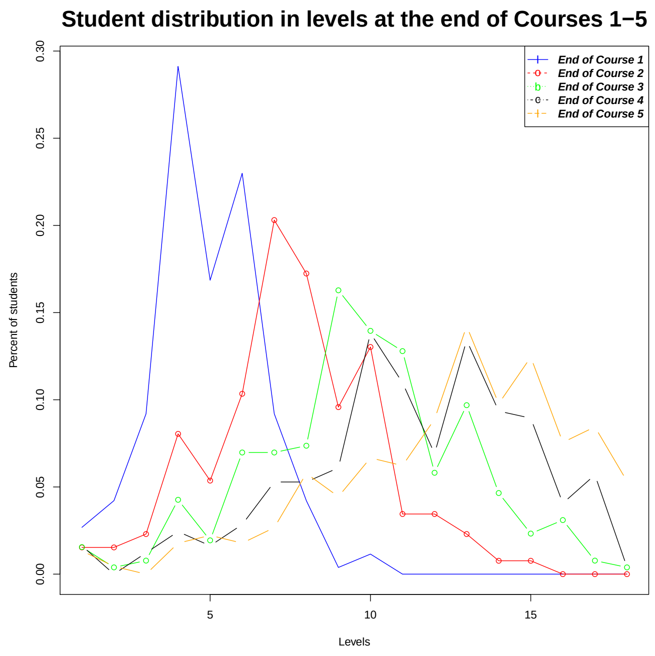


Figure 5: Histogram of students distribution in five courses

simulation.

For traits *Time Manager*, *Increment Developer*, and *Bug Stamper*, we simulate to generate snapshot of traits value gain change for each students submissions, based on indicators triggered information and time information. In each submission, students make some traits gain if they make progress. In simulation 257 students take 5 continuous courses and each course include 4 assignments. Figure 6-8 show the snapshots of traits gain at the end of five courses for these three traits, in coarse-grain perspective. Students traits value gradually increases as they work from course 1 until course 5. TimeManager trait gains is low, compared with IncrDeveloper and BugStamper.

In addition to coarse-grained perspective of these three traits gains with mean and standard deviation shown in Figure 6-8. We also plot fine-grained perspective of these three traits gains with students distribution at the end of each five courses, as shown in Figure 9-11.

For traits *Persistence*, we want to validate rationality of parameters for students non-progress(stuck) phase: how many submissions and how much time period. We run simulation with a historical two semester CS2 programming data set including 257 students made by 981 separate submission sequences. In this simulation, we select an example criteria to reflect students make non-progress or breakthrough: Indicator *Increasing Correctness* which shows students' solution in current submission passed more instructor-provided reference tests than before[8]. If Indicator *Increasing Correctness* does not change significant in current submission compared adjacent previous submission, we treat current submission in non-progress(stuck) status; otherwise, students make breakthrough. From common sense, the duration time of *Persistence* pattern cannot be too short or too long. We limit non-progress time period in a range

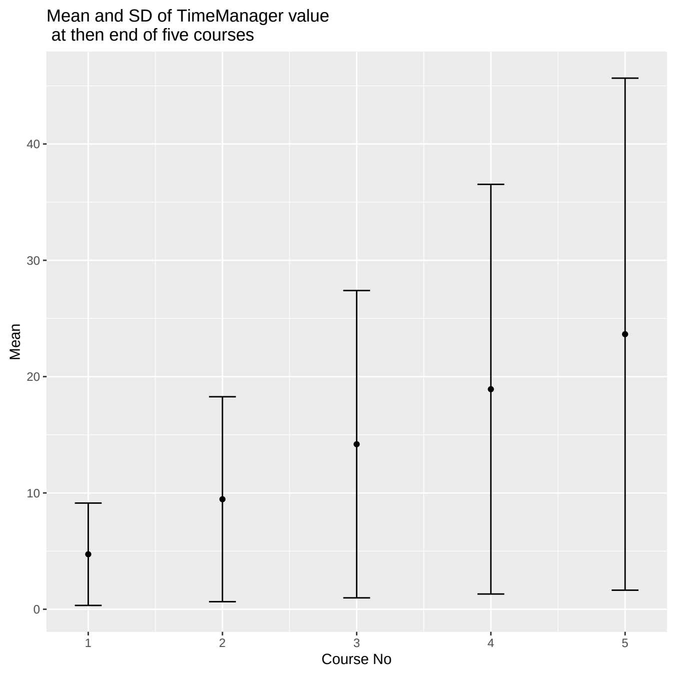


Figure 6: Mean and SD for TimeManager trait gains at the end of five courses

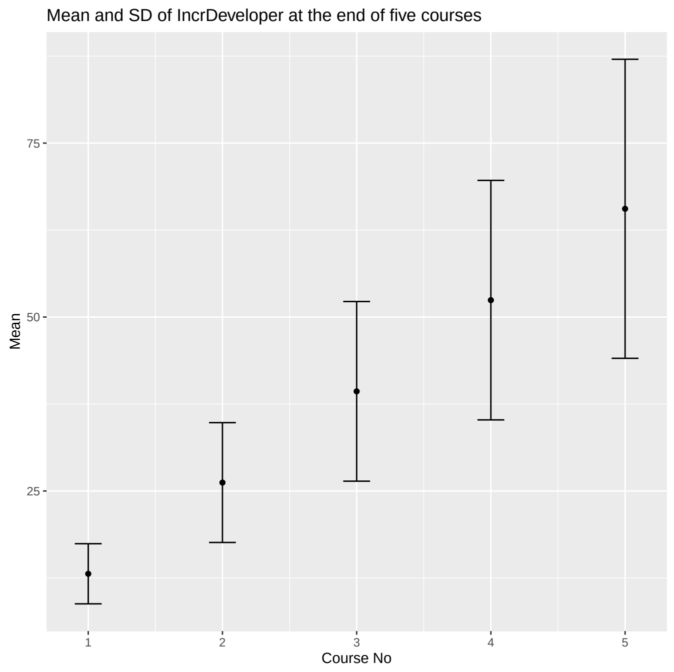


Figure 7: Mean and SD for IncrDeveloper trait gains at the end of five courses

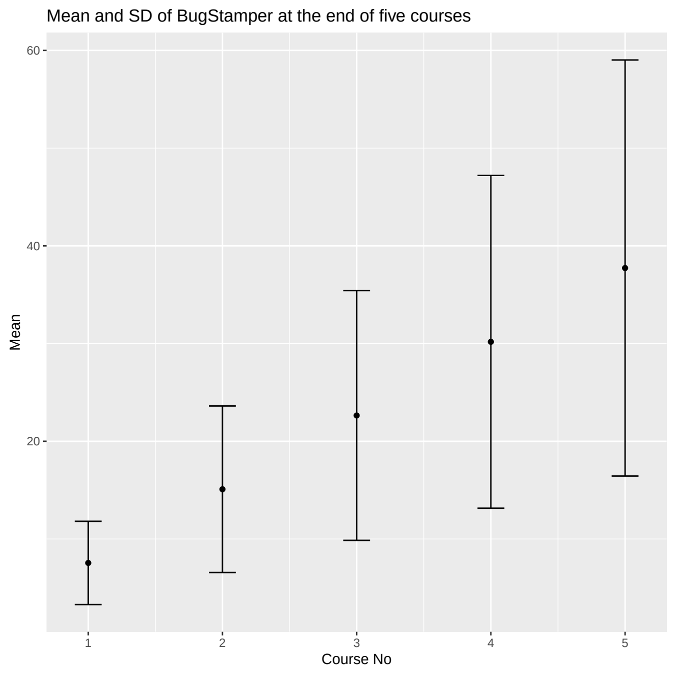


Figure 8: Mean and SD for BugStamper trait gains at the end of five courses

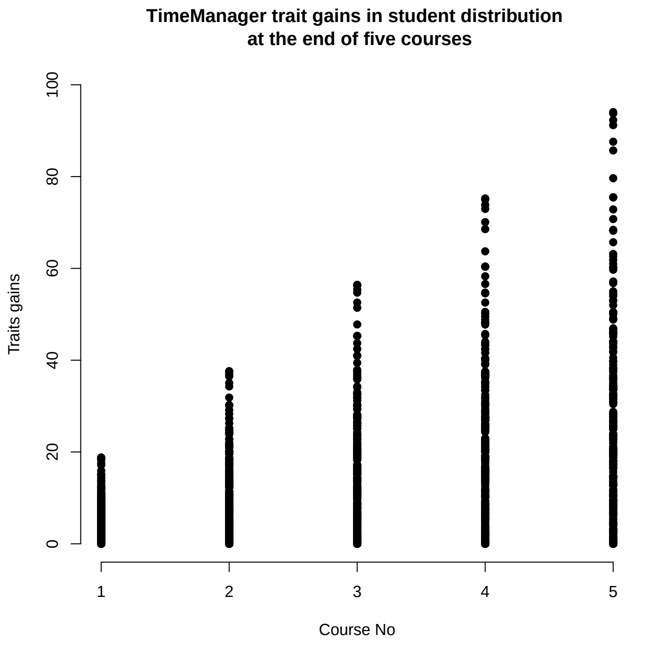


Figure 9: TimeManager trait gains in students distribution at the end of five courses

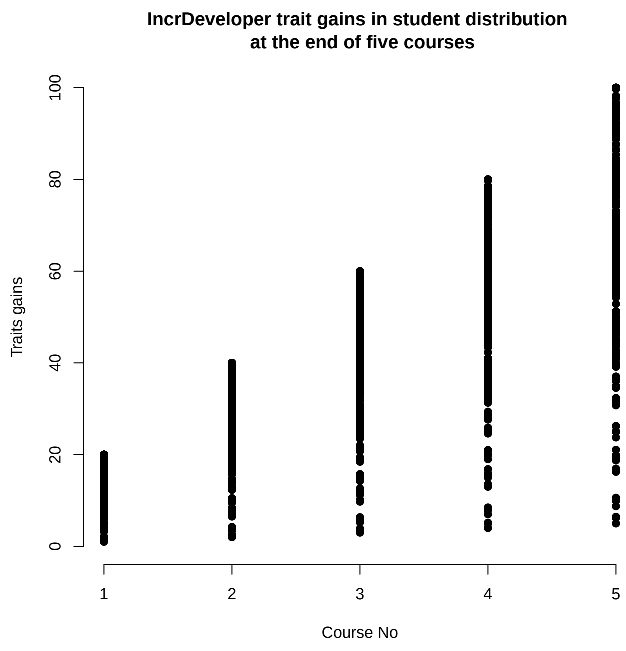


Figure 10: IncrDeveloper trait gains in students distribution at the end of five courses

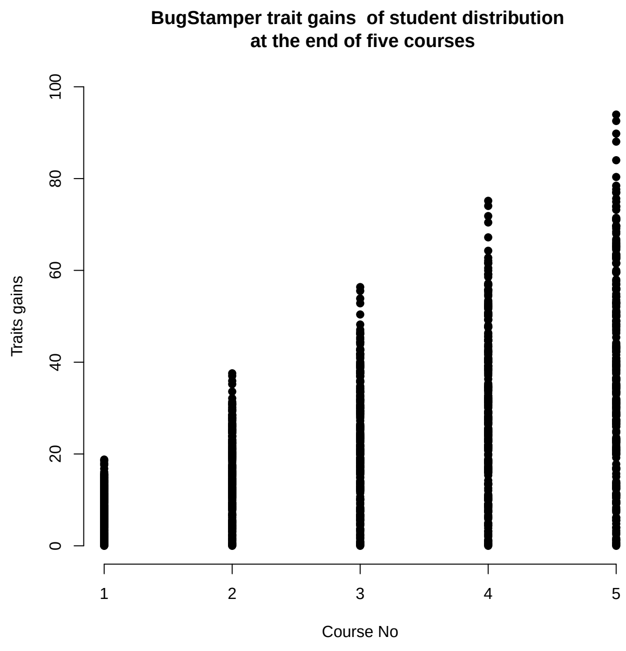


Figure 11: BugStamper trait gains in students distribution at the end of five courses

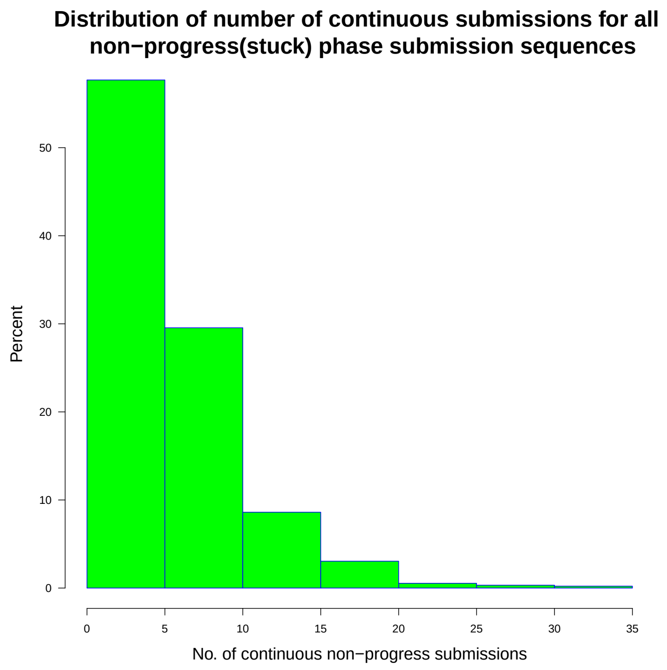


Figure 12: Histogram of continuous submission numbers in non-progress phase

at least 30 minutes and no more than daily work time (8 hours).

We calculate distribution of non-progress submission numbers and time period, as shown in figure 12 and 13.

Figure 5 shows histogram of percent of submissions vs No. of continuous submissions among all candidate non-progress submissions. Students normally make 20 to 50 submissions in a single assignment. Percent of submissions decreases as No. of continuous submissions increases. Major 80 percent qualified non-progress submissions with non-progress(stuck) phase have No. of continuous submissions less than 10. We can use a value range from 5-10 as parameter for No. of continuous non-progress submissions.

Figure 6 shows histogram of percent of submissions vs time period of non-progress phase. We can see percent of submissions decreases as time period of non-progress phase increases. Major 80 percent qualified non-progress submissions is within 200 minutes(around 3 hours).

We have not validate traits *Self-Regulator* and *Syntax Master*. We leave these three traits' evaluation in future once we collect data in classroom after we deploy these traits in Web-CAT.

5 Conclusions

This paper demonstrates initial work to design RPG elements character traits, experience points(XP), and levels together to integrate them into feedback of an automatic program assignment grading tool. These RPG elements together were validated by using a data set of historical programming data to assess its rationality and feasibility.

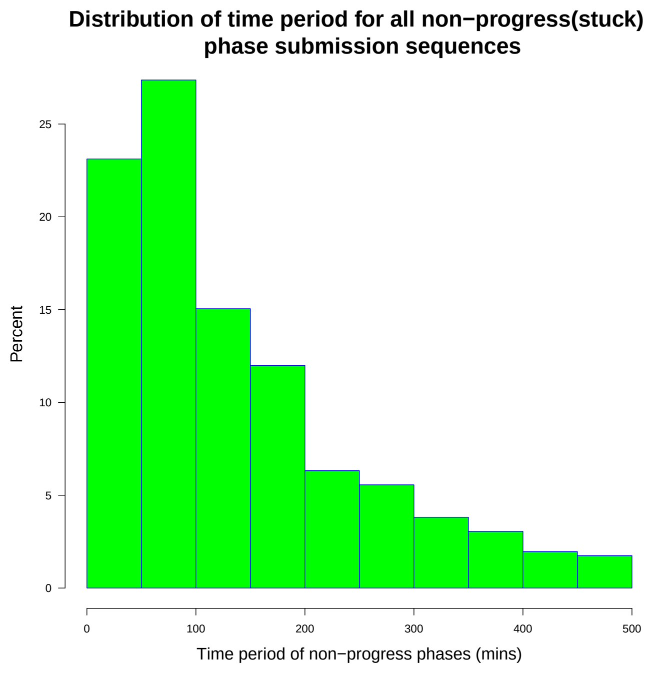


Figure 13: Histogram of time period in non-progress phase

Together with prior work to improve feedback in automatic assessment tools, these RPG elements aim to communicate visible information when students make progress on their assignments, with the goal of engaging students and leading students towards growth mindset beliefs and adoption of positive development behaviors.

Among these RPG elements, six characters(traits) were chosen to lead students towards expected behavior patterns such as time management and incremental developments, self-checking, etc. experience points(XP) and levels are designed as visible rewards and reflect a student's status when they make progress such as accomplishing daily missions tasks or demonstrating positive changes in targeted behavioral traits. All these RPG elements were designed span through multiple courses and assignments.

We validate four of six traits with a historical CS2 programming data set to find appropriate threshold parameters relate character traits value gain. In similar way we also validate the design of XP and levels with same historical CS2 programming data set. The simulation spans through multiple courses and multiple assignments. The simulation helps us to find appropriate threshold parameters value in these RPG elements design. Simulation results mainly meet our design expectation for these RPG elements.

We plan to implement and deploy these new designed RPG elements in CS1/CS2 courses to conduct a usability study for these elements in classroom.

References

[1] [n.d.]. Growth Mindset Toolkit.
`http://www.transformingeducation.org/growth-mindset-toolkit`

[2] Joshua Aronson, Carrie B. Fried, and Catherine Good. 2002. Reducing the Effects of Stereotype Threat on African American College Students by Shaping Theories of Intelligence. *Journal of Experimental Social Psychology* 38, 2 (2002), 113 – 125.
`https://doi.org/10.1006/jesp.2001.1491`

[3] Katheryn Christy and Jesse Fox. 2014. Leaderboards in a virtual classroom: A test of stereotype threat and social comparison explanations for women's math performance. *Computers & Education* 78 (09 2014), 66–77.
`https://doi.org/10.1016/j.compedu.2014.05.005`

[4] Katheryn R. Christy and Jesse Fox. 2014. Leaderboards in a virtual classroom: A test of stereotype threat and social comparison explanations for women's math performance. *Computers & Education* 78 (2014), 66 – 77.
`https://doi.org/10.1016/j.compedu.2014.05.005`

[5] Sebastian Deterding, Dan Dixon, Rilla Khaled, and Lennart Nacke. 2011. From Game Design Elements to Gamefulness: Defining "Gamification". In *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments (MindTrek '11)*. ACM, New York, NY, USA, 9–15.
`https://doi.org/10.1145/2181037.2181040`

[6] Christo Dichev and Darina Dicheva. 2017. Gamifying education: what is known, what is believed and what remains uncertain: a critical review. *International Journal of Educational Technology in Higher Education* 14, 1 (20 Feb 2017), 9.
`https://doi.org/10.1186/s41239-017-0042-5`

[7] C. S. Dweck. 1999. *Self-theories: Their Role in Motivation, Personality and Development*. Taylor & Francis, Philadephia, PA, New York, NY, US: Psychology Press.

[8] Stephen Edwards and Zhiyi Li. 2016. Towards Progress Indicators for Measuring Student Programming Effort During Solution Development. In *Proceedings of the 16th Koli Calling International Conference on Computing Education Research (Koli Calling '16)*. ACM, New York, NY, USA, 31–40. `https://doi.org/10.1145/2999541.2999561`

[9] S. H. Edwards. 2004. Using software testing to move students from trial-and-error to reflection-in-action. In *Proceedings of the 35th ACM Technical Symposium on Computer Science Education (SIGCSE '04)*. ACM, New York, NY, USA, 26–30.

[10] Geoff Goehle. 2013. Gamification and Web-based Homework. *PRIMUS* 23, 3 (2013), 234–246. `https://doi.org/10.1080/10511970.2012.736451` arXiv:https://doi.org/10.1080/10511970.2012.736451

[11] Andrew Goldman. 2019. *Using Daily Missions to Promote Incremental Progress on Programming Assignments*. Master's thesis. Virginia POlytechnic Institute and State University. An optional note.

[12] Tatsuhito Hasegawa, Makoto Koshino, and Hiromi Ban. 2015. An English vocabulary learning support system for the learner's sustainable motivation. *SpringerPlus* 4, 1 (27 Feb 2015), 99. `https://doi.org/10.1186/s40064-015-0792-2`

[13] Jincheul Jang, Jason J. Y. Park, and Mun Y. Yi. 2015. Gamification of Online Learning. In *Artificial Intelligence in Education*, Cristina Conati, Neil Heffernan, Antonija Mitrovic, and M. Felisa Verdejo (Eds.). Springer International Publishing, Cham, 646–649.

[14] Ayaan M. Kazerouni, Stephen H. Edwards, T. Simin Hall, and Clifford A. Shaffer. 2017. DevEventTracker: Tracking Development Events to Assess Incremental Development and Procrastination. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '17)*. ACM, New York, NY, USA, 104–109. `https://doi.org/10.1145/3059009.3059050`

[15] Ayaan M. Kazerouni, Stephen H. Edwards, and Clifford A. Shaffer. 2017. Quantifying Incremental Development Practices and Their Relationship to Procrastination. In *Proceedings of the 2017 ACM Conference on International Computing Education Research (ICER '17)*. ACM, New York, NY, USA, 191–199. `https://doi.org/10.1145/3105726.3106180`

[16] Petros Lameras, Sylvester Arnab, Ian Dunwell, Craig Stewart, Samantha Clarke, and Panagiotis Petridis. 2017. Essential features of serious games design in higher education: Linking learning attributes to game mechanics. *British Journal of Educational Technology* 48, 4 (2017), 972–994. `https://doi.org/10.1111/bjet.12467` arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/bjet.12467

[17] Wei Li, Tovi Grossman, and George Fitzmaurice. 2012. Gamicad: a gamified tutorial system for first time autocad users. In *In Proc. of ACM UIST*. ACM, New York, 103–112.

[18] Dastyni Loksa and Amy Ko. 2016. The Role of Self-Regulation in Programming Problem Solving Process and Success. 83–91. `https://doi.org/10.1145/2960310.2960334`

[19] Mukund Babu Manniam Rajagopal. 2018. *Virtual Teaching Assistant to Support Students' Efforts in Programming*. Master's thesis. Virginia POlytechnic Institute and State University. An optional note.

[20] Michael Sailer, Jan Ulrich Hense, Sarah Katharina Mayr, and Heinz Mandl. 2017. How gamification motivates: An experimental study of the effects of specific game design elements on psychological need satisfaction. *Computers in Human Behavior* 69 (2017), 371 – 380. `https://doi.org/10.1016/j.chb.2016.12.033`

[21] R. K. Sawyer. 2006. *Introduction: The new science of learning*. The Cambridge handbook of learning sciences (1-18), New York: Cambridge University Press.

[22] Dan Schuller. 2017. How to Make an RPG: Levels. `http://howtomakeanrpg.com/a/how-to-make-an-rpg-levels.html`

[23] Steven J. Spencer, Claude M. Steele, and Diane M. Quinn. 1999. Stereotype Threat and

Women's Math Performance. *Journal of Experimental Social Psychology* 35, 1 (1999), 4 – 28. https://doi.org/10.1006/jesp.1998.1373

[24] David Toth and Mary Kayler. 2015. Integrating Role-Playing Games into Computer Science Courses As a Pedagogical Tool. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education (SIGCSE '15)*. ACM, New York, NY, USA, 386–391. https://doi.org/10.1145/2676723.2677236

[25] Wikipedia contributors. [n.d.]. Epic level. https://en.wikipedia.org/wiki/Epic_level

[26] Wikipedia contributors. 1999. Description of Attributes in RPGs. https://en.wikipedia.org/wiki/Attribute_(role-playing_games)

[27] Wikipedia contributors. 1999. Experiencepoints2019. https://en.wikipedia.org/wiki/Experience_point