Chapter 1
Engineering Adaptive Serious Games Using Machine Learning

Michael A. Miljanovic, Jeremy S. Bradbury

Abstract The vast majority of serious games (SGs) do not feature any form of machine learning (ML), however, there is a recent trend of developing SGs that leverage ML to assess learners and to make automated adaptations during game play. This trend allows serious games to be personalized to the learning needs of the player and can be used to reduce frustration and increase engagement. In this chapter we will discuss the development of new ML-based SGs and present a generalized model for evolving existing SGs to use ML without needing to rebuild the game from scratch. In addition, to describing how to engineer ML-based SGs, we also highlight five common challenges encountered during our own development experiences, along with advice on how to address these challenges. Challenges discussed include: selection data for use in an ML model for SGs, choosing game elements to adapt, solving the cold start problem, determining the frequency of adaptation, and testing that an adaptive game benefits from learning.

Key words: adaptation, machine learning, personalized learning, serious games.

1.1 Introduction

Serious games (SGs), also known as educational games, are games designed with a purpose other than entertainment. Most commonly, these are video games that have been made to help aid with learning in a variety of contexts, including science, healthcare, business, and more [15]. Developers of SGs have a difficult challenge: to create a product that is first and foremost a game that not only engages players, but equally importantly improves their understanding and competency with non-game content. Game-based learning (GBL) using SGs differs from the related field of gamification [1], where game elements such as badges, points, and avatars are applied to non-game contexts. Using these definitions, an application such as Duolingo would not be considered an SG since it was not designed as a game, but it would be considered an example of gamification because it includes elements such as

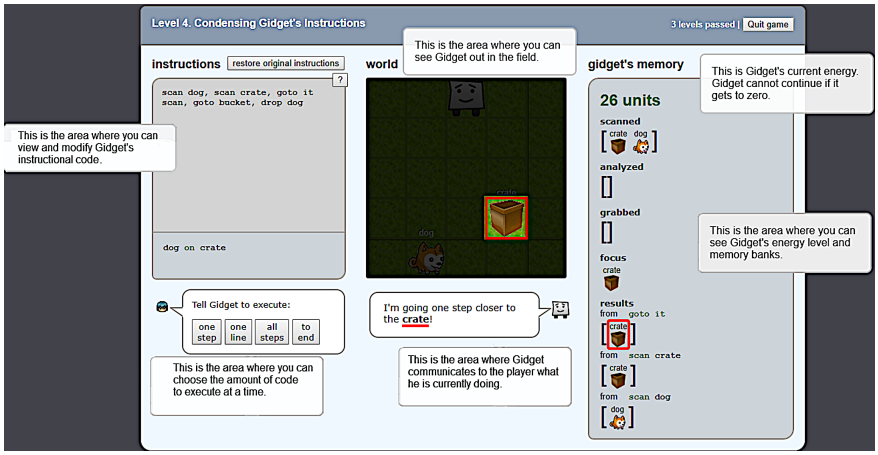

Fig. 1.1 The GidgetML game–an ML-enhanced version of the Gidget programming game that personalizes the game play and game elements to meet the learning needs of the player

achievements and a leaderboard to encourage players to use the app frequently. On the other hand, IBM's CityOne is an SG designed to help players learn about transportation, environmental, and logistical issues that are relevant to city leaders and businesses. The distinction between these applications is important–Duolingo seeks to use gamification to improve engagement, while CityOne is intended to be intrinsically motivating by its nature of being a game.

The target audience for SGs can include any demographic, but the players of these games are considered learners in the context of the game's educational environment. One of the greatest challenges of serious games is learner assessment, which often takes the form of stealth assessment [13] in order to avoid disrupting the game play experience.

The importance of learner assessment is even more significant for the purposes of adaptation. Adaptive SGs are a new form of SG that use learner data collected before or during game play to modify the game automatically [14]. This strategy can be used to reduce the difficulty level for players who are struggling with the game's content, or to increase the challenge for players who have demonstrated a high level of competence. The concept of dynamic difficulty adjustment has already been implemented in entertainment games like Left 4 Dead, which has a "Director" that constantly adjusts the game as it assesses the player's performance.

Automatically assessing learning is a particularly difficult task, which has led to the use of techniques such as machine learning (ML) to analyze learners based on data provided to a player experience model [3]. A variety of ML algorithms might be appropriate for this purpose, but each presents its own challenges. Different learning models require various forms of data, and it can be challenging to determine what data points are relevant to assessing learner competency.

This chapter will discuss some of the challenges that come from engineering adaptive SGs, and to support our discussion of engineering adaptive serious games

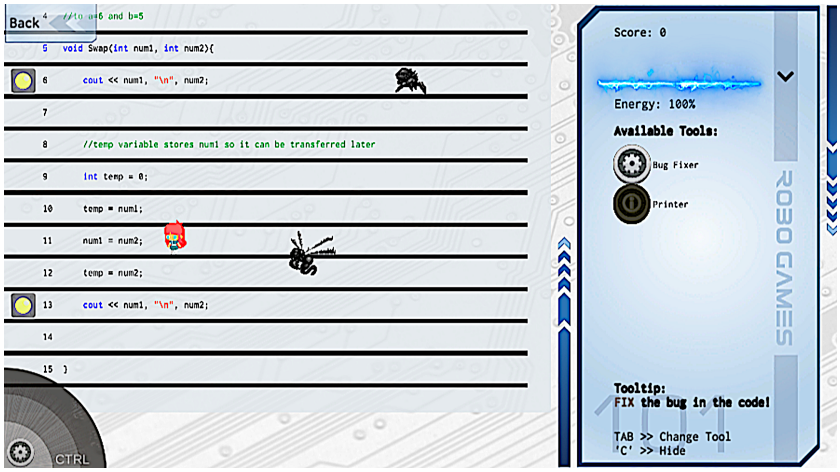

Fig. 1.2 The RoboBug game–an ML-enhanced program debugging game that personalizes the game play and the hint system to meet the learning needs of the player

we will use two case studies throughout this chapter to illustrate and provide context to the issues with using ML for adaptation. Both case studies are examples of SGs for computer science education; however, the lessons learned can be generalized to other domains.

The first case study is an SG called GidgetML [12], which is a modified version of the serious programming game Gidget created by Michael Lee and Amy Ko [5, 6]. The game requires players to correct bugs in code to navigate the Gidget character and provide instructions to complete various tasks on a two-dimensional grid map (see Fig. 1.1). GidgetML uses an ML algorithm that takes information about failed player attempts and code efficiency to categorize players into different competency categories. It then modifies subsequent levels of the game based on the player's competency–players demonstrating higher levels of competence start each level with additional bugs in their starter code and stricter limitations on how efficient their solution must be to be accepted as correct.

The second case study is an SG called RoboBUG [9], which was designed to help players gain familiarity with various techniques for debugging source code. Players navigate their character through lines of code akin to navigating a text editor, and must identify lines of code that contain semantic or logical errors (see Fig. 1.2). Two adaptive variants of RoboBUG were created, each using data including time on task and failure counts to categorize players [8]. One adaptive variant modified the game's code and obstacles to increase or decrease challenge by obfuscating bugs or making them easier to see, while the second variant provided an increasing number of hints to players depending on their assessed competency level.

1.2 Data Models

With a traditional non-game-based learning assessment, it can be difficult to make judgments about a learner's competence–questions can be phrased poorly, or the answers to a multiple-choice question might be deduced by context clues. However, in an educational game, the multitude of actions taken by a player over time can be examined in a variety of different ways in order to gauge their performance. For example, one might consider how much idle time elapses during game play, how much time a player spends tinkering with different parts of a puzzle, or how many mistakes are made before the player is able to progress in a game. These data features and others can be sampled across a game play session and used to predict a player's level of competence based on a well-designed game play model.

The theoretical basis for modelling player behavior in an SG lies in competence-based knowledge space theory (CbKST) [4]. This theory distinguishes between assessment problems/activities and the desired competencies that pertain to them. A given assessment problem might have a number of different solutions that pertain to different competencies, and similarly any given competence might be able to provide a solution to many different assessment problems. In addition, there are some competencies which are composed of other competencies, representing a skill that can only be developed after others have already been learned.

As an example, consider the assessment problem of summing a list of numbers using a program. One solution might involve iterating through each element of the list, adding that number to an initial sum of zero, and examining the sum after all elements have been added. This solution would require the learner to demonstrate competency in arithmetic and iteration; however, this is not the only way to solve the problem. Another solution could be a recursive approach to adding the numbers together; although this solution might be equally correct, it requires an understanding of recursion that is not necessary to solve the problem. Therefore, we see that many assessment problems may be linked to many different competencies, and that not all of these links are equal–clearly the ability to sum two numbers is required, but knowledge of iteration and recursion are optional.

1.2.1 Game Task Model

In the context of educational games, most game "tasks" are intended to foster the development of or test a player for specific competencies. Thus, to assess any player, one must first examine each task (which is a form of the assessment problem) to determine the linked competencies that are prerequisites or otherwise associated. This can be particularly difficult if there are multiple different approaches that might be suitable for overcoming the task, in which case, it is necessary to identify the approach selected by the player. However, restrictions placed upon the player may be used to limit the space of solutions that could be used to solve a problem. For example, in a block-based coding game, removing any "loop" blocks would force

the player to rely on a recursive solution to solve the problem, and eliminate the competency relationship between the task and iteration.

Thus, the first necessary step for applying adaptivity to an educational game is to consider the tasks in the game and determine the relevant competencies for each. This means identifying the necessary prerequisite competencies needing to attempt a solution for a task, and the desired outcomes from completing the task. Ideally, the first tasks in the game should require as few prerequisites as possible (or zero), and subsequent tasks should build upon what has been learned from the previous ones. If there are any gaps in competencies between tasks, then this should be addressed by the introduction of additional tasks or some other way to help the player develop the necessary prerequisite. For example, if a task requires a prerequisite competency that the player would only learn if they completed an optional game play task, then that task should either be made mandatory, or another task should be added that can compensate for the missing exercise.

1.2.2 Player/Learner Model

Once there is a suitable model for game tasks, it then becomes possible to create a model for the player based on CbKST. A player's behavior in-game can be logged for both post-game assessment as well as in-game assessment, although the latter presents a greater challenge due to the time sensitivity of the evaluation and incompleteness of the data. A best estimate of a player's competency should be gleaned based on their performance on tasks, whether successful or unsuccessful. Game play data has a high degree of granularity, and while some games may be suited to simply differentiating between successful and unsuccessful attempts at a task, it is often preferable to examine specifically how a player attempts to complete a task, and evaluate the player based on both their approach and whether or not it was successful. Two students who both succeed or both fail at a task will not necessarily be at an equal level of competency, considering their performance and how efficient or how close they were to solving the problem, as well as how easily they were able to reach that solution can provide further insight as to their competency level.

1.3 A Generalized Methodology for Evolving Existing Serious Games to Use ML

To support the development of new adaptive SGs, we have devised a methodology to guide the process of adding automatic adaptation to existing non-adaptive SGs [10]. The methodology has four key phases (see Fig. 1.3):

1. **Identify** a potential adaptive game
2. **Model** game play tasks and the ability of learners

3. **Build** ML and supporting functionality into the existing code base.
4. **Evaluate** the benefits of the adaptive modifications.

1.3.1 Identify

A number of technical and learning factors should be considered when identifying if a serious game is appropriate for adding adaptation via ML.

The three most important technical factors are source code availability, software quality and game playability. First, the source code will need to be publicly available. This is a non-issue if the game was written by the same developers who are extending it; however, if the developer implementing adaptation is someone other than the original developer then it is necessary to ensure that the software license for the chosen game allows for modification and redistribution. Second, the serious game needs to be of significant quality and robustness to support a planned redevelopment. The quality of the software can be assessed by reviewing available software artifacts including documentation, source code and issue tracking data. Third, the playability of the game should also be considered and existing playability studies are an asset.

The main learning factors that need to be considered are:

- *Learning outcomes:* Adapting the learning content of a game requires a clear understanding of the required knowledge, topics, and learning outcomes that are present in the original game.
- *Learner experience and demographics.* Making informed decisions about adapting the chosen game requires detailed knowledge about the learners who will play the game. Learners of various age groups may respond differently to in-game adaptations. Furthermore, knowledge about the level of experience of the game's audience is needed in order to make good decisions about how to adjust learning content, including accommodation for an audience with no experience. Special consideration should also be given to adapting for learners of diverse educational

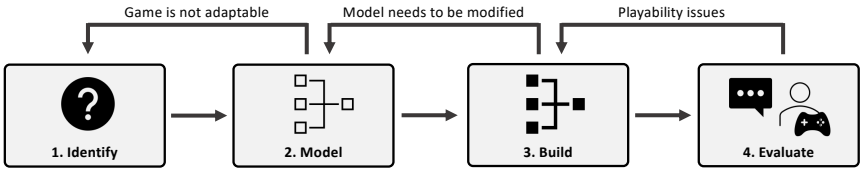

Fig. 1.3 An Overview of our methodology for evolving adaptive serious games. Once a game has been identified for adaptation, a model and plan is developed that connects the game play tasks with a method for assessing learners. After the model has been created, the adaptation functionality is built into the existing code base. Finally, the new adaptive serious game should be evaluated to determine its efficacy (e.g., learning, engagement) and the evaluation results should be compared with the efficacy results of the original non-adaptive serious game.

backgrounds outside of computer science. Finally, we suggest choosing games that are inclusive in order to reach a diversified audience of learners.
- *Learning evaluation:* In order to properly evaluate the final adaptive serious game in phase four, it is best to choose an existing game that has already been evaluated with respect to learning. The existing evaluation can serve as a baseline in assessing the learning benefits of adaptation later.

1.3.2 Model

Once a game has been selected, the next step is to determine how viable it is to make it adaptive. This is based on the ability to create an accurate model of a game play task as well as the ability to model learners using the collected data.

Game play tasks can be deconstructed into several key features that are relevant for evaluating learning. In alignment with CbKST, each task should be associated with one or more competencies that are prerequisites; in other words, the task cannot or should not be completed unless the learner has demonstrated (and been given opportunity to do so) competency in those prerequisites. The successful completion of a task should provide evidence to both prerequisite and other associated competencies, so that learners can be evaluated based on the tasks they have completed. In addition to associated competencies, tasks also are targets for adaptation, and so the parameters of the task that can be changed should be taken into account. For instance, a task might have a time limit, feedback/hints, or other constraints that can be adjusted based on the perceived level of player competency. If a game is unable to adjust these task features (or has no tasks at all, such as a non-structured "open world" game), it may not be well suited for adaptation.

In addition to being able to create models of tasks, there must be sufficient available data sources to create an accurate model of the learner. The primary source for modeling learner competency comes from their completion of tasks. The number of successful attempts and failures on tasks associated with specific competencies generates a significant amount of data that can help classify the learner and predict their level of competency in different areas. In addition to successes and failures, the degree to which a learner is successful or unsuccessful provides more useful information than a binary feature. For example, the speed at which a learner completes tasks, or the quality of their solution, might help distinguish mid-competency learners from those with high competency. It is important, however, to separate data that reflects competency in learning versus competency in games–for example, it may be the case that faster completion of tasks only indicates that the learner is better at playing games, but does not have a higher level of competency than a slower learner that has higher-quality solutions. The degree to which data can inform a learner model has a significant impact on whether or not a game can accurately assess the learner, and therefore make good decisions about how and when to adapt.

1.3.3 Build

This phase includes integrating key modelling functionality into the existing code base, for example, logging player behavior (if not already present), initializing the learner assessment model, and then applying an adaptation strategy.

Learner-specific adaptation requires a constant gathering and assessment of learner information. The data gathered is categorized based on what it is measuring and how often the measurements can take place . For example, data may be gathered during a task, between a task or between game play sessions.

There are different options that a developer might consider for initializing a player's assessment model. This is a challenging part of the build phase and will be discussed in detail in Sect. 1.4.3. The key consideration is to determine how to initialize the SG until enough data is collected to adapt to an individual player.

Once data is being logged and the SG initialization has been established, the developer can proceed to apply the previously chosen adaptive strategy. This involves increasing restrictions on steps, work, errors, and time for players who have demonstrated high competence and are seeking a greater challenge. Conversely, these restrictions should be reduced for players who exhibit low competence in order to accommodate their needs and reduce the level of challenge.

1.3.4 Evaluate

One of the challenges with serious game development is the need for accurate and reliable evaluation. One benefit to our approach of evolving existing serious programming games is that many have existing studies that can be replicated and reproduced for the adaptive versions, thus allowing us to evaluate the benefits of the adaptive modifications by comparing the study results from the original and adaptive versions of a game. In cases where the original version of a serious game did not have an evaluation, we recommend following best practices which may include questionnaires, skill tests, interviews and controlled experiments.

1.4 Challenges in Engineering Adaptive Serious Games

Engineering adaptive serious games presents a number of unique challenges that are not present in the development of non-adaptive SGs. In this section, we will discuss five of the most common challenges faced by SG developers planning to add adaptation to existing SGs:

- **Selecting data for use in an ML model for SGs.** The selection of user data can have a huge impact on the success of ML as a method for adapting to a learner's needs in SGs. We will describe how to select the relevant features from

potentially dense game play data based on assessing the data with respect to a learner's competence and understanding of a learner's mindset.

- **Choosing game elements to adapt.** Modern video games use player performance data to modify game play difficulty, and this can also be used in the context of SGs to modify game play elements that facilitate learning. For example, this might include changing the frequency or verbosity of in-game hints, or modifying game play tasks based the abilities of the learner.
- **Solving the cold start problem.** Many ML methods require an initial data set for training, but at the beginning of game play, there is often little or no information available. We will discuss different strategies for addressing the cold start problem as well as how the generation of synthetic data sets can assist with making ML features viable sooner.
- **Determining how frequently to adapt.** It is possible to adapt an SG between game sessions, between game levels or even during a game play task. One issue with determining frequency is that frequent adaptations can be computationally expensive, may frustrate the user, and can even negatively impact learning. Another issue is that infrequent adaptations, while mitigating performance issues, can be too late to handle frustrated players who fail to complete a task that is not suited for their level of competence. We will share our experience with selecting the adaptation frequency based on an analysis of the game play and the target learners of the game.
- **Evaluating that an adaptive game benefits learning.** The best practices in playtesting non-adaptive SGs are insufficient to provide confidence in the efficacy of ML-based SG features. Therefore, additional testing on top of traditional playtesting is needed. We will discuss our experiences with testing an ML-based SG including how to identify the relevant metrics and how to assess the results to determine if the ML adaptations are working correctly.

1.4.1 Challenge #1: Selecting Data

How do you decide what SG data to select for use in an ML model?

As with many machine learning applications, it is generally the case that the data available is not going to perfectly suit the needs of the ML model (see Fig. 1.4 for examples of available game data). For example, a player who fails a task that requires an understanding of a particular learning concept is not necessarily incompetent at that concept–human error must be considered. Thus, a probabilistic approach is best to account for cases where a player makes a careless mistake or when some tasks vary in their difficulty. It is likely the case that a player who can succeed at multiple challenging tasks is competent in the prerequisite competences, even if they occasionally fail simpler tasks based on the same competences.

Depending on the genre of game and the design of the task, some data features may not be relevant to the assessment of the player. For example, one should be wary

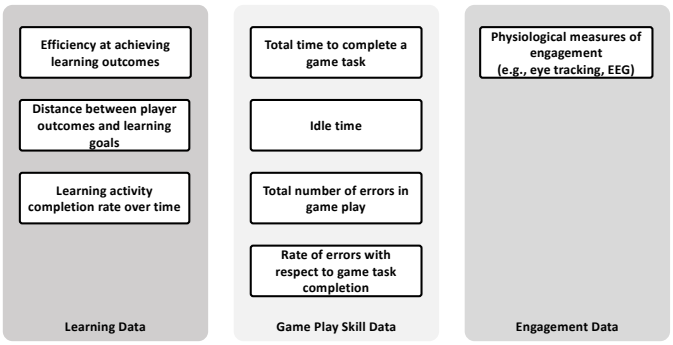

Fig. 1.4 Examples of serious game data available for use in adaptation

of using time elapsed as a metric of competency if the task itself is untimed and there is a possibility that the player may sit idle because they have opted to physically step away from the game for some time period. It can be difficult to distinguish between idle time and time spent thinking about a problem, but generally given the fast-paced nature of games (compared to traditional learning activities), one can expect that a lengthy delay is likely the case of the player being absent. This should be accounted for when automating any form of data collection from game play, as it will likely cause a significant effect on the data set if a 10-minute break from the game is measured as 10 minutes thinking about a problem. A player taking a long time to think about something, but still sitting and playing the game, is likely to consider interacting with some part of the environment, and games that present obstacles or hazards for the player to interact with while they play may mitigate their ability to idle.

Finally, the data provided for all the models must be used to make an assessment of the player. One approach to this is to use an unsupervised machine learning algorithm to compare the player to others who have previously completed the game. An unsupervised algorithm is a good choice because there is no way to say with certainty how competent any given player is based on their performance; we can only make estimates at best. An advantage to this approach is the ability to separately assess the existing data set and categorize each player based on their demonstrated competency. This way, the player can be assigned a label (such as low, medium, or high competency) and that label used to make determinations about the best form of adaptation. In the next section, we will discuss what can be adapted based on this information.

In GidgetML, the data collected from each task included the number of failed attempts at a solution as well as the number of "steps" needed to complete the task. Each player was then categorized into different levels of competency in debugging based on a K-means clustering machine learning algorithm. Using this algorithm, each of the three clusters was labeled as low, medium, or high based on the average performance of the players in that cluster, and each player in that cluster was assigned that label. Then, when the model had to account for a new player, it would repeat

the K-means clustering algorithm and use the old labels to determine the player's categorization based on the players with whom they shared a cluster.

1.4.2 Challenge #2: Game Elements

How do you select what game elements to adapt?

With an accurate model of tasks and players, a game can determine when it is necessary to adapt to the learner. However, there remains the question of what to adapt in a task–specifically, which game elements should be changed and how should they be adjusted.

It is first important to distinguish between game elements conducive to learning, and game elements relevant only to game play and entertainment. For example, a change in the amount of hints and feedback provided to the player is likely to impact their learning, while increasing the number of obstacles that require manual dexterity to avoid may only serve to make the game more challenging if those obstacles are not related to any of the game's competencies. It may be tempting to increase the difficulty of a game by adding more obstacles or using stricter restrictions on time. However, the audience for a learning game will include players who are skilled at games but lacking the desired competencies, as well as players who demonstrate high levels of competency but are less familiar with video games. There are different ways to challenge each player, and if the goal of adaptation is to create an experience that finds a balance between challenge and skill, then the way in which players are challenged will vary based on their skill level.

An important purpose of adaptation is to facilitate learners who are struggling with difficult content. This is why one of the most valuable forms of adaptation comes in the form of feedback. This can vary from hints about how to approach a task differently, or even demonstrations about how to attempt to solve a problem that the learner may not be familiar with. The administration of feedback must also be carefully timed; presenting instructions at the start of a task might be ignored, but giving a hint to a player immediately after they make an unsuccessful attempt at an action is likely to immediately affect what the player will do next.

GidgetML modified only two game elements using its adaptive settings. The first modification was a change to the sample solution provided to players at the start of each task–players who demonstrated a high level of competency were given obscure code with many errors, while players at a low level were given code that might be mostly correct with only a few mistakes. The second modification changed the restrictions on what would be an acceptable solution to a given task. For high-competence players, only a very efficient or perfect solution would be accepted as correct, while low-competence players would be able to submit less efficient solutions to pass a level.

RoboBUG was developed to have two different types of adaptation. The first was a change to game play obstacles and code, using an approach similar to GidgetML

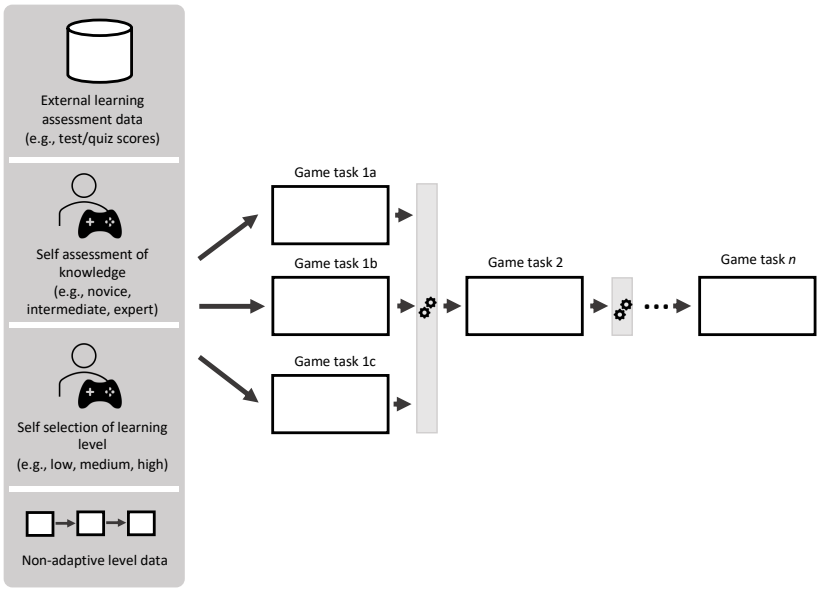

Fig. 1.5 Alternative approaches to addressing the cold start problem in adaptive SGs

for obscuring the code presented to players. RoboBUG also featured game obstacles that would hinder the player when navigating through the code, which were changed to be faster and more inconvenient for players at higher levels of competence. The second form of adaptation was the introduction of hints and feedback presented to players as they interacted with the code or failed to complete a level. Players at low levels of competence would receive frequent feedback and advice on how to understand the code in a task, while players at higher levels would receive such hints infrequently or not at all.

1.4.3 Challenge #3: Cold Start

How do you solve the cold start problem in adaptive SGs?

The cold start problem occurs when for new users "...the system does not have information about their preferences in order to make recommendations" [7]. Specifically, for adaptive SGs there is insufficient information about a player's learning needs to make a recommendation on the starting state of the SG.

It may be tempting to resolve the cold start problem by providing the game with existing external data about players, such as their grades or results on a pre-game quiz (see Fig. 1.5). However, this has a number of drawbacks. Firstly, there are issues of privacy in tracking the behavior of players while associating their data with

grades–this could potentially be used to identify players who would otherwise wish to remain anonymous. There is also the question of whether or not the grades or quiz results are actually reflective of the player's competence. The cold start problem is not solved if the data provided is not accurate, and it could be the case that a player's grades or quiz scores are not reflective of the same competencies associated with the game's tasks.

One approach that can be considered is simply ignoring the cold start problem altogether. With this approach, there will be significant errors in the assessment of players for whom little game play data has been collected, but the model should become more accurate as the size of the data increases over time. A way to mitigate the issue of initial errors in assessment is to limit the degree to which adaptation happens in the earlier parts of the game. For example, the first tasks in a game might have little variation between them in order to account for the potential errors in assessment, while later tasks might have a large range of difficulty levels as there is a greater level of confidence in a player's actual level of competency.

An alternative solution is to use an earlier part of the game as the initial data set for the model, and only begin to adapt game play once the player has reached a certain point. The introductions to many games feature tutorials or guides to demonstrate for players how they interact with the game's environment. It may be the case that these tutorials and guides are suitable for data collection, but unsuitable for any kind of adaptation. In such a scenario, the way in which a player demonstrates the competencies targeted by the tutorial should be used as the initial data provided for machine learning. Thus, it is better for these tutorials to allow for a high degree of player agency, as opposed to tutorials which provide explicit step-by-step instruction on how to proceed.

GidgetML handled the cold start problem by only introducing adaptive game play halfway through the game's tasks. The first half of the game was an extended tutorial to introduce each game play feature one at a time, and it was only after these tasks were completed that GidgetML would begin to adapt to select new tasks. This meant that GidgetML had access to many levels worth of game play data that could be used to predict performance from each player.

1.4.4 Challenge #4: Adaptation Frequency

How frequently should you adapt in an SG?

Aside from the questions of how and what to adapt, there is also the question of when. Frequent adaptation can have the benefit of quickly addressing issues of frustration experienced during game play, which might otherwise lead to players abandoning the game from believing they lack the skill to play. However, more advanced machine learning algorithms may lead to issues of computational performance in the game, particularly as data sets grow and when the number of features is large.

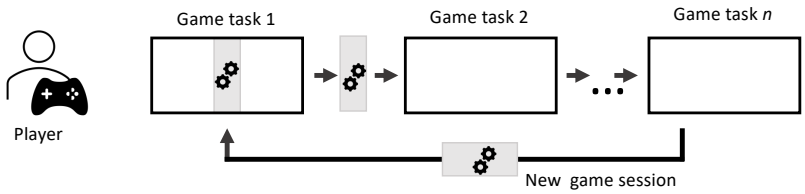

Fig. 1.6 Adaptation frequency–within tasks, between tasks and between sessions

	Computational cost of adaptation	Time to adaption	Quantity of game data available
Within tasks	●●●	●	●
Between tasks	●●	●●	●●
Between sessions	●	●●●	●●●

Fig. 1.7 Trade-offs between different adaptation frequencies

Adaptation frequency can be separated into three categories: between session, between task, and during task (see Fig. 1.6). Between session adaptation occurs when the algorithm only runs after the player has finished playing, presumably to pick up the game at a later time. Between task adaptation uses the data from each task completed and, in combination with the existing data set, adjusts the subsequent task based on the results of the game's algorithm. During task adaptation takes into account the player's current behavior and, based upon the task as well as the data sets, determines whether or not (and how) the task should be adjusted while the player is attempting to complete it.

Computationally, it is easiest to adapt between game play sessions, but for games that do not have significant replay value or are unlikely to be played for more than one session, this approach may not be useful. Between task adaptation can serve to adjust tasks based on previous behavior, and allow the game to plan out the player's path to help adjust for any shortcomings in desired competencies. However, it does not provide a solution to players struggling with an immediate task, and cannot adjust to compensate unless the player finally completes the task or fails it. Although the approach of adapting during a task offers a solution to this problem, it has its own issues, namely those of performance as well as the risk of overfitting data depending on the algorithm being used (see Fig. 1.7).

GidgetML and RoboBUG made use of between-task adaptations in order to select from three different versions of subsequent tasks in the game. Since the elements of the game that were modified were exclusively restrictions on the task's acceptance criteria or increased vagueness of the sample code or sample solution, there would not be any elements available to modify during the task. In addition, since the games were designed to be completed in a single session, it would not be possible to make use of between session adaptation in an effective way.

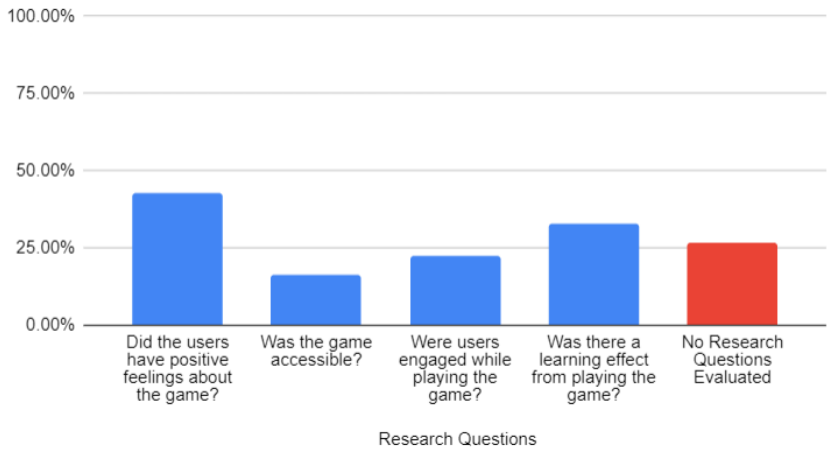

Fig. 1.8 Questions evaluated by studies of serious programming games [11]

1.4.5 Challenge #5: Evaluating Learning

How do you evaluate that an adaptive SG benefits learning?

Perhaps the most challenging of all issues facing developers of educational games is the ability to evaluate their efficacy, both for entertainment and engagement as well as their value for learning. Although engagement and learning may be correlated, it is possible for a game to be significantly engaging while providing little learning value, or vice versa.

Historically, educational games are not frequently given thorough evaluations for their value in learning (see Figs. 1.8 and 1.9) [11]. Many games developed by researchers and gaming companies are only tested for their functionality, and not tested to see if the target audience will gain any significant improvement in their development of any competencies. Part of the reason for this is that it is difficult to ascertain whether or not a game is an effective learning tool. Adaptive and non-adaptive games are equally capable at logging game play data, but there is no established set of best practices for evaluating the learning effect of educational games.

One approach to testing is to consider the use of a post-game evaluation, such as a questionnaire, to determine how much the player understands after playing the game. This was the approach used for RoboBUG–players would complete a pre-game questionnaire that tested their knowledge of the learning competencies, then play the game, then complete the same test again to see if they would change their answers. However, this has several issues–firstly, players may simply not wish to complete the assessment if there is no incentive to do so. Secondly, a pre-game questionnaire is necessary in order to compare the results of the questionnaires, and such tests are an inconvenience that may encourage players to not bother playing the

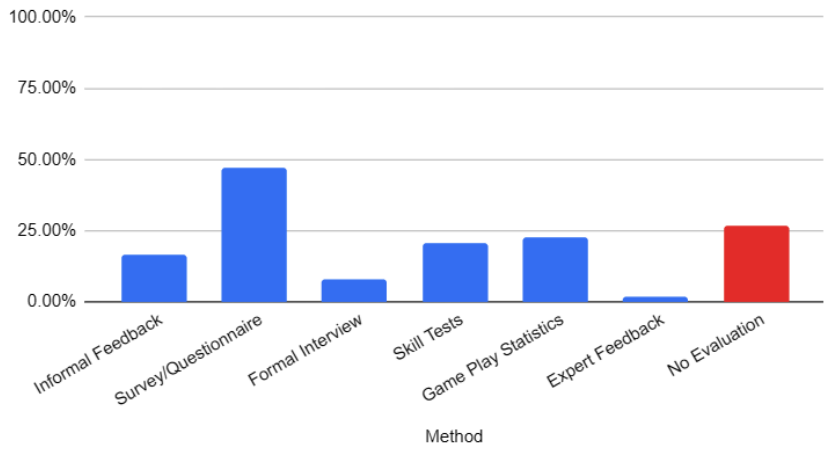

Fig. 1.9 Methods used in the evaluation of serious programming games [11]

game in the first place. Finally, and most importantly, it is completely possible that the questionnaires do not actually assess the competencies that the game will teach. A multiple-choice test, or other test that can be automatically evaluated, is ill-suited to determine whether or not a player has achieved the highest levels of Bloom's taxonomy [2]. Although it may be possible to make a test suited to determining whether or not the player can recall, understand, or apply different competencies, it is not as easy to determine whether or not a learner can analyze, evaluate, or create when limited to an automated grading system. It is also challenging to overcome this obstacle by creating a generalizable questionnaire, because any two games are unlikely to cover the exact same content.

1.5 Discussion

The development of new adaptive serious games and the addition of adaptation to existing serious games are both challenging engineering problems the require a combination of expertise in the learning domain, game development, software development and machine learning. In this chapter we have presented one approach to engineering adaptive serious games using ML. We have also discussed five of the common challenges faced in the development of adaptive SGs and provided insight and guidance based on our own experience of development adaptive SGs in the field of computer science.

The main limitations of the models and practices presented are as follows:

- All of the models and practices are based on our experience with the development of adaptive SGs for computer science. While we believe these generalize to other domains we acknowledge that the generalizability has not been fully researched.

- Our methodology for evolving existing SGs to us ML has not been independently utilized by third-party developers. We have successfully applied this methodology to add adaption to a third-party developed SG (Gidget) and to an in-house developed SG (RoboBUG) but we do not have data on the use of this methodology outside of our research group.

Our experience with the development and deployment of adaptive SGs in computer science has shown that despite the challenges that may be encountered, the use of adaptation in SGs provides an opportunity to enhance the engagement and learning of SG players by personalizing the game play to their specific learning needs.

References

1. Ilaria Caponetto, Jeffrey Earp, and Michela Ott. Gamification and education: A literature review. In *European Conference on Games Based Learning*, volume 1, page 50. Academic Conferences International Limited, 2014.
2. Mary Forehand. Bloom's taxonomy. *Emerging perspectives on learning, teaching, and technology*, 41(4):47–56, 2010.
3. Maite Frutos-Pascual and Begoñya García Zapirain. Review of the use of ai techniques in serious games: Decision making and machine learning. *IEEE Transactions on Computational Intelligence and AI in Games*, 9(2):133–152, 2015.
4. Simone Kopeinik, Alexander Nussbaumer, Michael Bedek, and Dietrich Albert. Using cbkst for learning path recommendation in game-based learning. In *20th International Conference on Computers in Education*, pages 26–30, 2012.
5. Michael J Lee and Amy J Ko. Personifying programming tool feedback improves novice programmers' learning. In *Proceedings of the Seventh International Workshop on Computing Education Research*, pages 109–116. ACM, 2011.
6. Michael J Lee and Amy J Ko. A demonstration of gidget, a debugging game for computing education. In *2014 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 211–212. IEEE, 2014.
7. Blerina Lika, Kostas Kolomvatsos, and Stathes Hadjiefthymiades. Facing the cold start problem in recommender systems. *Expert Systems with Applications*, 41(4, Part 2):2065–2073, 2014.
8. Michael A. Miljanovic. *Adaptive Serious Games for Computer Science Education*. PhD thesis, Ontario Tech University, Oshawa, ON, Canada, Oct. 2020. (Supervisor: Jeremy S. Bradbury).
9. Michael A. Miljanovic and Jeremy S. Bradbury. Robobug: A serious game for learning debugging techniques. In *Proc. of the 13th Annual ACM International Computing Education Research Conference (ICER 2017)*, pages 93–100, Aug. 2017.
10. Michael A. Miljanovic and Jeremy S. Bradbury. Making serious programming games adaptive. In *Proc. of the 4th Joint Conference on Serious Games (JCSG 2018)*, Nov. 2018.
11. Michael A. Miljanovic and Jeremy S. Bradbury. A review of serious games for programming. In *Proc. of the 4th Joint Conference on Serious Games (JCSG 2018)*, Nov. 2018.
12. Michael A Miljanovic and Jeremy S Bradbury. Gidgetml: an adaptive serious game for enhancing first year programming labs. In *Proceedings of the 42nd International Conference on Software Engineering (ICSE 2020)*, 2020.
13. Valerie J Shute. Stealth assessment in computer-based games to support learning. *Computer games and instruction*, 55(2):503–524, 2011.
14. Alexander Streicher and Jan D Smeddinck. Personalized and adaptive serious games. In *Entertainment computing and serious games*, pages 332–377. Springer, 2016.
15. Tarja Susi, Mikael Johannesson, and Per Backlund. Serious games: An overview. 2007.