Chapter 3
Software Engineering for Dynamic Game Adaptation in Educational Games

Vipin Verma, Ashish Amresh, Tyler Baron, and Ajay Bansal

Abstract Educational games and game-based assessments have evolved over the past several years and are continuing to evolve. They promote student engagement in the learning process by creating an interactive environment where they can learn in a fun and challenging way. This gives them the potential to yield diagnostic information to educators and feedback to students. During the game play process, game-based assessment (GBA) can be used to assess the learning imparted by the game to the students. A common strategy for GBA has been to utilize surveys and built-in quizzes to measure student learning during the game play. However, this impacts students' attention negatively as they need to change their attention from game play to the assessment and back. Stealth assessment provides a natural alternative for assessment of learning without breaking the delicate flow of engagement. It aims to blur the lines between assessment and learning by weaving them together within the game. Stealth assessment uses game play interaction data to build inferences about student performance and learning. As an advantage, it provides ways to assess hard-to-measure constructs such as learning proficiency, critical thinking, persistence, and other twenty-first-century skills. Designing and developing an educational game takes time, and repeating the process for every new content or concept can be inefficient. The authors provide a framework called content-agnostic game engineering (CAGE) that can be used to create multiple learning contents within a single game by reusing already developed educational game mechanics. CAGE helps reduce time for creating an educational game by building content-agnostic mechanics that could be used across multiple content topics. It does so by separating the game into three components of mechanics, content, and student modeling that operate independently. Additionally, stealth assessment can be integrated into CAGE as a part of the student model and can

V. Verma (✉) · T. Baron · A. Bansal
Arizona State University, Tempe, AZ, USA
e-mail: tjbaron@asu.edu; Ajay.Bansal@asu.edu

A. Amresh
Northern Arizona University, Flagstaff, AZ, USA
e-mail: amresh@asu.edu

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023

K. M. L. Cooper, A. Bucchiarone (eds.), *Software Engineering for Games in Serious Contexts*, https://doi.org/10.1007/978-3-031-33338-5_3

also be content agnostic as a way to demonstrate the advantages of adopting a CAGE-based development framework. While CAGE can work with multiple content domains, it cannot work with every domain. The limit is decided by how the game mechanics are implemented. In this chapter, we discuss the software practices to implement CAGE architecture and ways to embed stealth assessment in a content-agnostic way.

Keywords Content agnostic · Stealth assessment · Dynamic adaptation · Bayesian network · Facial tracking

3.1 Introduction

Educational assessment has evolved over the past several years from traditional pen-and-paper tests to forms such as game-based assessment and continues to evolve. It must provide feedback to learners and diagnostic information to teachers [1]. Game-based learning offers an interactive environment for the students to learn in a fun and challenging way while keeping them engaged in the learning process. Game-based assessment (GBA) offers a way to assess them while they are interacting with the game. GBA may consist of built-in quizzes and surveys to assess the student learning while they are playing. However, such methods tend to break the students' attention from learning to complete the assessment. Stealth assessment is a way to assess the learners while they are playing the game without breaking their flow, eventually blurring the lines between learning and assessment by integrating them [2]. It utilizes the data generated during the game play to make inferences about the student learning and performance at various grain sizes. Further, it can be used to assess skills such as creativity [3], systems thinking [4], persistence [5], and other twenty-first-century skills, which are needed for success today [6]. These skills in general are hard to assess by traditional means of assessment.

3.1.1 Research Background

Evidence-centered design (ECD) is an instructional theory that assessments should be focused on evidence-based arguments [7]. Namely, that a student's understanding of the material can be measured by their reaction to changes in observable variables. Making these changes is something that GBL can do well as dynamically changing the problem during learning is easier with GBL than most traditional teaching methods. The issue that arises with this is the need to ensure that the change flows naturally within the game system and does not break the player's flow or distract from the learning process [2]. It is here that stealth assessment techniques can ensure that these measurements do not require overt actions that remind the player that they are currently being evaluated on their understanding of the given

topic. Many methods of stealth assessment are also helpful in measuring student reactions to unobservable variables as these are harder to detect but also important to measure in order to gauge student understanding [7]. Finally, it is important that the learning assessment be well integrated into the game itself and not stand out as a separate experience [8]. Not only would that break student immersion, but games that integrate the two are both more effective teaching tools and generally better liked by students using them.

3.1.2 Motivation

Presently, serious games are designed with an emphasis on the educational content of the game, which is a good approach but has some issues [9]. Such games are usually not as engaging when compared to the entertainment games, as the educational content of the game takes precedence over entertainment while designing the game. Further, the game is strongly connected to the learning content, due to which the programming code, game mechanics, game design, and learning assessment cannot be effectively re-used for teaching another content. This issue is compounded by the complex level of observation needed to assess students' reactions to non-observable variables in order to properly assess student learning [7]. The setup required to do these measurements causes the measures themselves to also be strongly connected to the learning content. This creates a scenario where readopting existing games to other educational content becomes extremely difficult as the game mechanics need to be rebuilt and the learning assessment must be completely revisited as well. The authors delineate the software framework called content-agnostic game engineering (CAGE) that can be used to alleviate this problem. It uses content-agnostic mechanics across a set of learning contents to develop the game. Further, [10] observed that the explicit assessment of the learning content in the form of tests and questionnaires can impact the player engagement in a negative way. Therefore, [1] integrated stealth assessment in the CAGE framework. It was built in a content-agnostic manner so that the assessment can be re-used across multiple content domains and thus contribute to reduction in the overall game development time to a considerable extent.

3.1.3 Chapter Outline

This chapter will target the emerging software practices in GBA and game adaptation. It will cover three main points: conducting stealth assessment, the CAGE framework, and the student model. Within stealth assessment, this chapter will elaborate upon how it has been used to assess twenty-first-century skills, which are hard to measure otherwise, while keeping the learners engaged with the learning process [11]. Stealth assessment includes techniques like mouse and touch-tracking

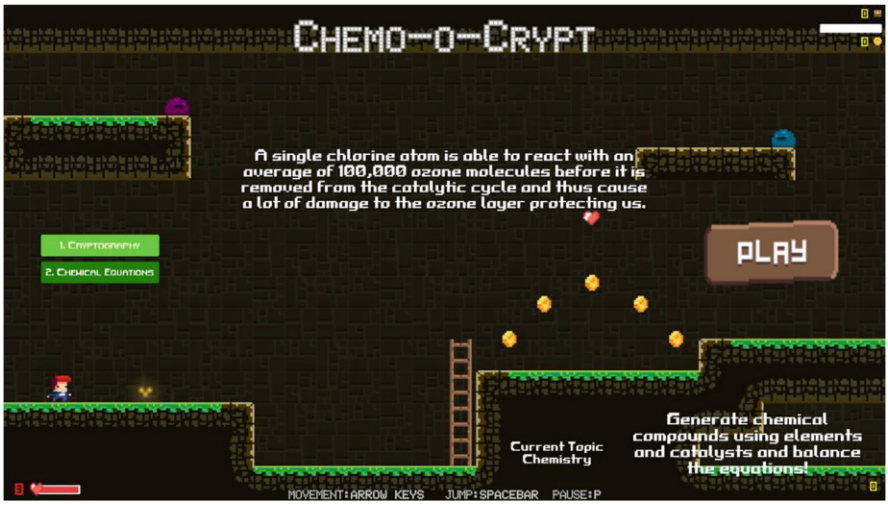

Fig. 3.1 A game developed using CAGE architecture

and how they have been used to measure the students' cognitive load without explicitly asking it from the learners [12]. The CAGE section will incorporate how a single assessment design can be used in the assessment of multiple content domains. Figure 3.1 shows a game developed using the CAGE architecture developed by Baron [10]. This game teaches cryptography and chemistry, one at a time to its learners. So, a single game is made with an effort of two and targets two content domains at once. Similarly, a single assessment strategy is embedded in the game that targets assessment of both the content domains. We delineate the process to implement a content-agnostic game and examine its feasibility for multiple serious games creation with an embedded assessment. We also probe into the effectiveness of game adaptation within the CAGE framework. Finally this chapter will discuss the inclusion of the student model, representing the learning state of a student at any point of time. This will include modeling the student's learning state using their performance parameters and how it can be used to dynamically adapt the game during run-time to accommodate the learning style and knowledge state of the student [13].

3.2 Stealth Assessment

A process that involves using data to determine if the learning goals are met or not is called assessment [14]. It is an equally important process when compared to designing the game mechanics and learning content of the game [1]. Stealth assessment is a technique in which the assessment is directly woven into the fabric

of the game such that it is undetectable. It can be used to make inferences about the player performance by utilizing the data generated during the game play [15]. It has been used for the assessment of many twenty-first-century skills such as systems thinking [4], persistence [5], creativity [3], problem-solving skills [16], team performance [17], and causal reasoning [18]. With the help of large volumes of data generated during the game play, stealth assessment can be used to identify the process which a player follows to solve a problem in the game [12]. Other assessment techniques, such as survey questionnaires can break the flow of learners and disengage them from the learning process. However, stealth assessment can replace these obtrusive assessment techniques to keep their flow intact [19]. It can also be used to provide immediate feedback to the learners during their game play session, which they can immediately incorporate into their game play [20]. The real-time feedback can further be used to dynamically adjust the game difficulty in real time to the level appropriate for the learner [21].

3.2.1 Stealth Assessment Implementation Techniques

There are a number of ways that can be used to implement stealth assessment in serious games. In this section, these methods are discussed along with their software implementation techniques.

3.2.1.1 Mouse and Touch-Tracking

This technique can be used in games that employ a computer mouse or a touchscreen device to play. It has been used for the assessment of memory strength [22], positive and negative emotions [23], gender stereotypes [24], cognitive load [25], and numerical representation [26]. In this technique, the mouse or touch coordinates are tracked as the user moves their cursor across the screen. It can reveal the hidden bias or the intent of the user while they make a decision to reach a certain target.

To implement mouse-tracking in Web-browser-based games, *mousemove* event is available in the built-in Web APIs for JavaScript [27]. This event is fired whenever mouse is moved across a target element. Further, the *pageX* and *pageY* can be used to get the X and the Y coordinates of the point where the *mousemove* event was triggered [28]. Both *pageX* and *pageY* are available to use as a part of the *MouseEvent*. These X and Y coordinates can then be used to trace the path of the mouse as it is dragged across the screen by the user, which is used to assess user's intent or inherent bias in their thought process. Below is an example code snippet that tracks mouse movement within an html id element called *toBeTracked* and reads their x and y coordinates:

```
$('#toBeTracked').mousemove(trackMouseMovement);
function trackMouseMovement(event) {
  var eventDoc, doc, body;
  var areaHeight = $('#toBeTracked').height();
  if (event.pageX == null && event.clientX != null) {
        eventDoc = (event.target &&
          event.target.ownerDocument);
        eventDoc = document || eventDoc;
        doc = eventDoc.documentElement;
        body = eventDoc.body;
        event.pageX = event.clientX +
          (doc && doc.scrollLeft ||
            body && body.scrollLeft || 0) -
          (doc && doc.clientLeft ||
            body && body.clientLeft || 0);
        event.pageY = event.clientY +
          (doc && doc.scrollTop ||
            body && body.scrollTop || 0) -
          (doc && doc.clientTop ||
            body && body.clientTop || 0 );
  }
  // get the x-coordinate of the user
  var userX = event.pageX -
    $('#toBeTracked').offset().left;
  // get the y coordinate of the user
  var userY = event.pageY -
    ($('#toBeTracked').offset().top + areaHeight);
}
```

Unity3D, a game engine popularly used by the game developers, also has some built-in ways to get the mouse coordinates [29]. They have a *mousePosition* method within the Unity's input system that returns a vector corresponding to the user's mouse position [30]. Unreal Engine, another popular game engine, also exposes a way to get the mouse position [31]. However, it should be noted that excessive mouse-tracking can affect the game performance in a negative way and could even crash the system. A high temporal resolution will consume a lot of computer memory. For example, it is less expensive to collect mouse tracking data every 250 ms as compared to 100 ms. Further, it depends on the target computer memory configuration as well. A game with mouse-tracking temporal resolution of 200 ms that works as expected on a given computer system may crash when deployed on a system with lower available memory as maintaining all of the data about the mouse position will add up quickly.

3.2.1.2 Emotion Tracking

The VisagelSDK [32] from Visage Technologies and Affdex [33] are two readily available software solutions that can be used to recognize human emotions by utilizing the Facial Action Coding System [34]. These solutions can be used for offline or real-time analysis of human faces. For offline analysis, a recording of the facial stimulus is required, while real-time analysis needs a Web camera integrated with the application that can send the facial stimulus as an input to the software. Affdex can detect various emotions and expression [35], and this output can then be used to detect if a person is bored or frustrated during the game play. As an example, Baron [10] used an algorithm to categorized the observed emotions into states of boredom, flow, and frustration using the following algorithm:

- If all the emotions are below the threshold, then the player is classified in a BORED state unless they were in a state of FLOW previously.
- If any of the emotions is above the threshold, then the player is in a non-bored state.
  - If anger is above the threshold and happiness is below the threshold, then the player is classified to be in a FRUSTRATION state.
  - If surprise is above the threshold and sadness is below the threshold, then the player is classified to be in a FLOW state.
- If the above rules fail, then the player is classified to be in a state called NONE.

In another example, [36] used Affdex output data to predict the states of boredom, flow, and frustration during game play using binary logistic regression. This prediction can be used to adapt the game or content difficulty in real time [1]. Their results suggested that it was easier to predict boredom and flow, compared to detecting frustration. Below are some of the prediction equations that their analysis revealed:

$$ln(Flow/NonFlow) = -0.84 + (0.4 \times Fear) + (0.09 \times Happiness) + (-0.074 \times Sadness)$$

$$ln(Boredom/NonBoredom) = -1.24 + (-1.13 \times Fear) + (-0.38 \times Happiness) + (0.15 \times Sadness)$$

$$ln(Frustration/NonFrustration) = 1.85 + (-0.02 \times Attention) + (-0.03 \times BrowFurrow) + (0.02 \times EyeClosure) + (-0.067 \times LipPress) + (-0.03 \times LipPucker) + (0.03 \times LipSuck)$$

3.2.1.3 Player Data-Tracking

A large amount of background data can be collected when a video game is played. Such data may include (but not limited to) time spent on task, total time played, player death count, player score, and quiz response. Any data that can be attributed to an observed variable can be gathered in a log file, which can be used for online or offline analysis. These variables can also be used in conjunction with each other

for dynamic adaptation. For example, [1] used player score and their emotions to change the game difficulty in real time according to the following rules:

- Step up the game difficulty only when they have achieved 50% of max achievable score and are detected as being bored.
- Step down only when their score is less than 20% of max achievable score and are detected as being frustrated.
- If they're in Flow, don't change anything, and wait a while before checking again.

3.2.1.4 Bayesian Modeling

This approach utilizes the conditional dependence of several variables on each other, to create a probabilistic graphical model of the system. A Bayesian network for knowledge tracing is shown in Fig. 3.2 [13]. This model can be used to make inferences about the player learning based on the other observed variables. The model consists of a two-quiz sequence with four parameters called prior knowledge P(L), slip rate P(S), guess rate P(G), and learn rate P(T). Prior knowledge is gauged using diagnostic tests, while guess rate accounts for the probability of guessing despite not possessing the knowledge. Slip rate represents the probability of answering incorrectly (slipping) even when a skilled student actually knows the correct answer. The learn rate accounts for the probability that the learning will occur in the next level, based on the learning from the previous levels.

There are solutions available for modeling Bayesian networks in many programming languages. Bayes Server [37] is one such software used by Verma et al. [38] in their study. It has programming APIs available in C#, Python, Java, R, MATLAB, Excel, Apache Spark, and JavaScript. The C# API can be easily integrated to a Unity3D game program [39] for dynamically generating Bayesian networks, setting

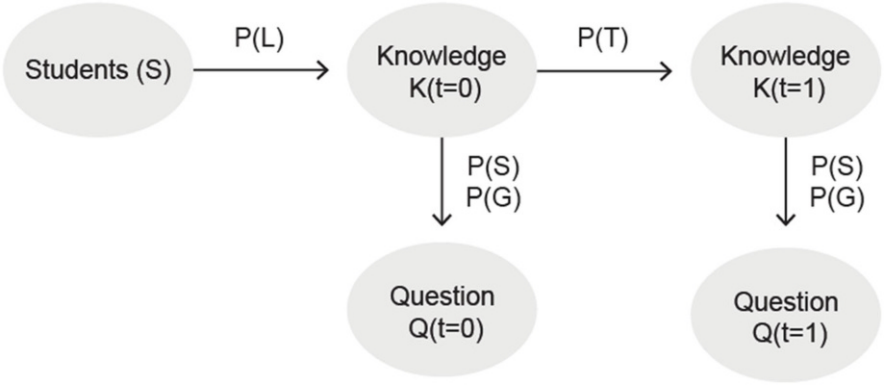

Fig. 3.2 Bayesian network depicting knowledge tracing model [13]

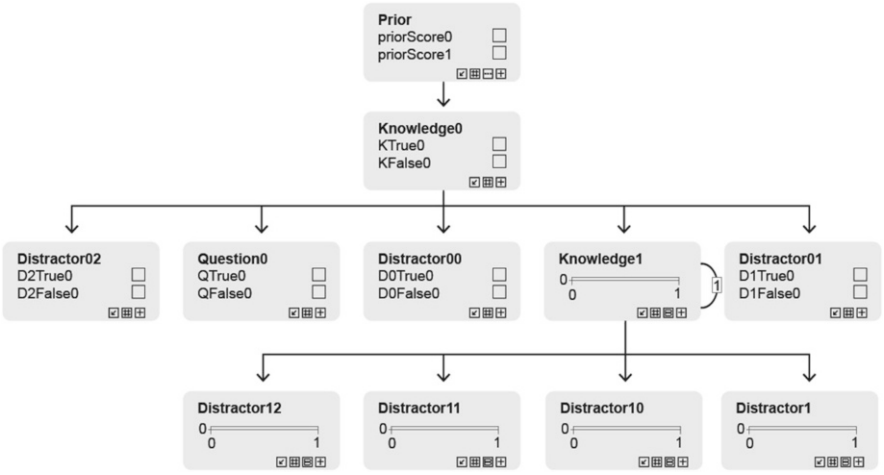

Fig. 3.3 Sample dynamic Bayesian network generated using Bayes Server

evidence as well as querying the network at any point in time. Figure 3.3 shows a dynamic Bayesian network that was programmed in Bayes Server.

3.2.1.5 Educational Data Mining

This technique uses a large number of data mining methods, which can be utilized to find out the patterns in bulk data captured during educational game play sessions [40]. It has been used for stealth [36, 41] as well as non-stealth measurements [42]. A given data mining technique applies to a given problem based on the assumptions and type of data. Therefore, the method should be carefully chosen based on the assumptions related to data.

3.3 Endogenous and Exogenous Games

The terms endogenous and exogenous games refer to how interconnected the game mechanics and the learning content are [8]. The more connected they are, the more the player both learns and enjoys the gaming experience [43]. Both levels of integration present their own problems from the perspective of game design, educational design, and game mechanics engineering.

3.3.1 Exogenous Games

The more exogenous a game is, the less of a connection there is between what the player is doing in the game and the content being taught [8]. Take as an example a math learning game where the player flies a spaceship through an asteroid field, destroying asteroids for points. At the end of the level, they are presented with a math quiz that must be completed in order to advance to the next level. This game would be considered completely exogenous. There is no connection between the game mechanics, flying the ship and shooting the asteroids, and the math content being taught. In particular, a player's ability to demonstrate skill at flying the ship and destroying the asteroids is not at all an indicator of how they will perform on the math quiz. Likewise, getting a high score on the math quiz does not in any way help the player fly the ship better on the next level [44].

Despite these drawbacks, exogenous games are more common in the classroom [44]. This is in part because they are easier from a design standpoint. When designing the game mechanics, the designers do not need to consider the educational content at all. These are often game designers who are focused and experienced at making entertaining gaming experiences, but often not at making educational content [10]. Likewise, the educational designers are experts at designing educational content and scaffolding the learning process, but are unfamiliar with what makes a quality gaming experience. With an exogenous game, these two parties can remain within their respective areas of expertise without needing to seek compromise with each other.

The impact exogenous design has on the actual construction of the game software is harder to judge [10]. When the game is truly exogenous, there is a need to develop two completely different experiences. The first is for the game itself and the mechanics that the player engages with, and the second is for the learning portion of the game. One benefit of this is that these types of educational experiences are often very straightforward, often taking the form of quizzes [44]. These then should be relatively simple to construct from a software perspective. The game mechanics in exogenous games are also often simpler, since they do not need to link with the educational content at all. This means that exogenous games require the time and effort to build two completely different gaming environments within the same project, with little space for overlap. However, these two pieces of the software are usually simpler to build than would often be required for a single combined code base.

3.3.2 Endogenous Games

The counterpart to exogenous games is endogenous games, where there is a strong connection between the game mechanics and the educational content [8]. This means that there is at least some connection between what the player does in the

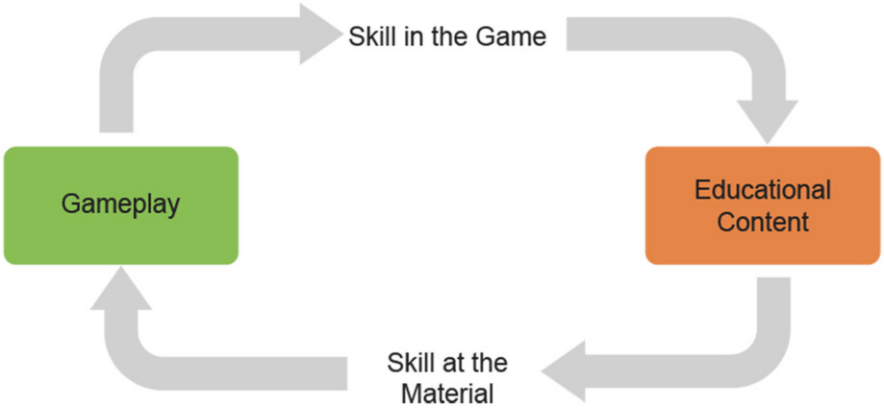

Fig. 3.4 Link between game mechanics and educational content

game and their skill in the educational content. Taking the previous example game, it is possible to make it more endogenous by adding two mechanics. The first would be to link skill at the math quizzes to the game play, and the second would be a way to reciprocate that link. Adding a mechanic where doing well on the math quiz would empower the player's spaceship would be a first example of a way to tie the educational content to the game play. The player's math skills would now have a direct effect on the game play. Likewise, adding a mechanic where destroying asteroids would earn the player hints on the math quizzes would be one possible way to link the game play to the quiz. With this mechanic in place, a player who does better in the spaceship portion of the game would be more likely to do well on the math quiz. This example is still not very endogenous [44], but this highlights a starting point for creating these links to create endogenous game mechanics. Figure 3.4 shows these links between the mechanics and educational content and how they need to feed into each other to create a cycle.

An example of a highly endogenous game [8] would be Typing of the Dead (Typing) by Sega [45]. In this game, players automatically walk between locations and are approached by zombies. Each zombie has a word displayed on them, and the player must type the word displayed, and their character will shoot the zombie. If the player is too slow, the zombie will damage them, and they will eventually die if this happens too many times in one level. In this case, the mechanics of how the player plays the game and what they are being taught are completely interlinked [44]. The goal of this game is to teach the player how to type and to increase their typing speed. The player must type quickly in order to avoid being killed in the game, showing a link between the game play and the learning content. At the same time, having a higher level of skill at typing will transfer to the player getting further in the game than another player with a lower level of skill would. This creates the ideal situation for an endogenous game where the player will learn the most effectively [43].

Endogenous games are less common than exogenous games, despite being more effective as teaching tools because they are harder to make for a variety of design reasons [44]. Many of the benefits that made exogenous games easy to design are reversed in this situation, making the design process much more difficult. If the game play and educational content are to be tightly linked, then it is necessary for the game designers and educational designers to work together. The game mechanics cannot be designed without consideration for the educational content, and likewise the educational content must take into account what the player will be doing in the game in order to decide how to present the content. Making these priorities line up with each other can take a considerable amount of time out of the development process.

The way in which endogenous design impacts the software structure is easier to judge than with exogenous games [10]. Due to their nature, exogenous games required the creation of two almost completely separate experiences with little overlap. Endogenous games by their nature then would instead consist of one larger integrated experience. The immediate benefit of this is that it removes the problem of creating almost two separate code structures with little overlap. The downside is that this one structure is likely much more complex than the other two would have been. Endogenous mechanics are often more complex due to the need to integrate the learning material directly. This both makes them more difficult to implement and also can force the way you design your high-level code structure. Having the educational content wrapped around the mechanics can make that material difficult to edit and possibly require mechanics changes as well. Learning assessment, bug tracing, and fixing can also be more complicated due to this connection. Endogenous design has been shown to be the better approach for teaching [43], but these issues make building them a challenge [10]. What would be best would be a way to create endogenous mechanics that are strongly linked in game play, but not in actual code. This is the principle which is followed in the CAGE architecture to develop the game rapidly while keeping the player engagement intact for learning purposes.

3.4 CAGE Architecture

The CAGE architecture can be used to develop games and assessments that can cater to multiple learning contents [9, 10]. It employs content-agnostic endogenous mechanics that are tied to game play and therefore allows creation of multiple games with an effort, which would be used to create a single game. CAGE follows a component-based architecture [10] shown in Fig. 3.5. It consists of three main components: the mechanics component, the content component, and the student model.

The mechanics component corresponds to the in-game mechanics, such as jumping, walking, sliding, or other actions players take in order to play and progress in the game. In CAGE, this component is designed to be content-agnostic and endogenic so that it can be used with multiple learning contents. The content

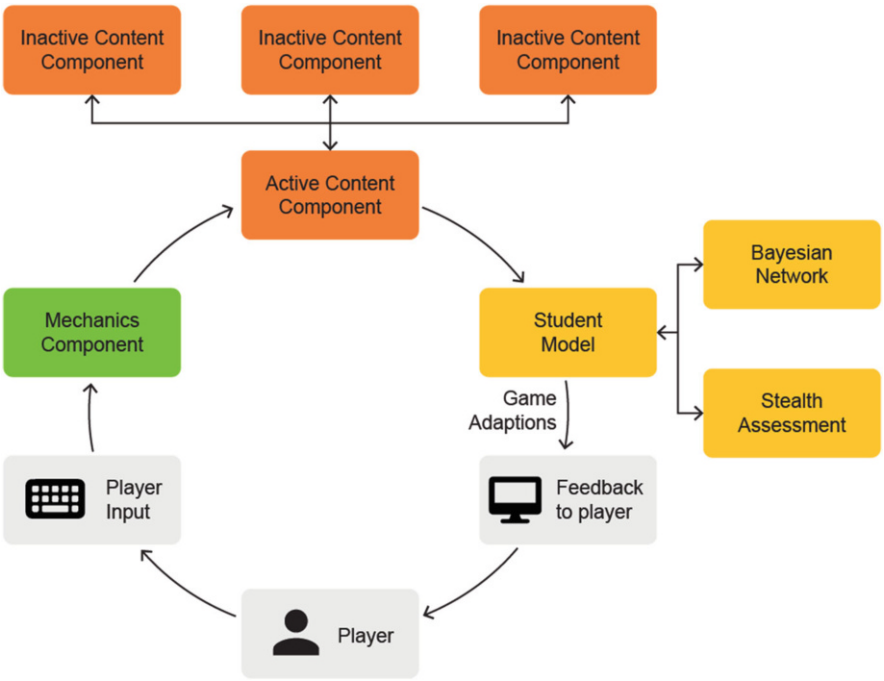

Fig. 3.5 Component-based CAGE architecture

component corresponds to one of the learning contents that the game is expected to teach. A CAGE game may consist of several learning components, but only one of them will be active at any point in time. The mechanic component passes the player action as input to the content component, which then evaluates the action as right or wrong. This assessment is then passed to the student model, which acts as an accumulator for the analysis to build a model of the student's understanding. The student model is also designed to be content-agnostic so that it can be re-used across multiple learning content domains, by containing multiple models to track understanding on the different topics.

The student model can be used for summative or formative analysis. As a summative analysis, it can be used to provide the play-through summary of the player, such as total time they took to learn the content, number of failures, success, scores, etc. As a formative analysis, it can be used to inform the game play session as well as the player feedback. When used for feedback, it can be used to provide hints or corrective actions to the players. When the student model is used to inform game play, it can make the game adjust dynamically based on the player's current skill level. This dynamic adaptation could be altering the game environment, the difficulty of the game or learning content, or all of these together.

There are several ways to make the game content-agnostic. In [10], one such process was using hooks for the game objects to communicate with each other.

Hooks are the generic messages that are generated by the game mechanics whenever an event occurs in the game, such as the player jumping across a chasm. The mechanics component passes these hooks to the content component, which uses them if they are relevant for that content component. Hooks is an abstract base class that implements polymorphism. Each content then implements their own version of the hooks, overriding only the methods that they need. Below is an example in C# in the Unity3D engine for a hook that corresponds to the player taking an action.

```
using UnityEngine;
using System.Collections;
/// <summary>
/// The hook used when you need to pass
/// actions to the content class or its derivatives
/// </summary>
public class ActionHook : Hook
{
    public string action { get; private set; }

    public ActionHook(string ac)
    {
        type = HookType.Action;
        action = ac;
    }

    public override string ToString()
    {
        return "Action performed: " + action;
    }
}
```

Another method to make the game content-agnostic is to use a database or JSON files to keep the content data. All the data for a particular learning content type will reside in their own JSON files. Players will be given the option to choose the learning content at the beginning of the game. When they later change the learning content, the active JSON files are swapped to use the one that the player selects. For example, consider the below piece of code showing two different content classes that inherit a common base class called Content. The source JSON file name is changed based on the content selected by the user.

```
public class Chemistry : Content
{
    public Chemistry(string name, string description,
    string jsonFile)
    {
        this.name = "Chemistry";
        this.description = "Balance the equations!";
```

```
        this.jsonFile = "Chemistry.json;
    }

}

public class Cryptography : Content {
    public Cryptography(string name, string description,
    string jsonFile)
    {
        this.name = "Cryptography";
        this.description = "Encode/decode the data!";
        this.jsonFile = "Cryptography.json";
    }
}
```

3.4.1 Student Model

A student model could be used to model various states of the student, such as their emotional state, gaming skill, and knowledge. It is an accumulator model that can collect the data from player interactions and mine it to store useful information about the student. This information could be used to gauge their behavior as well as learning and anything else that might be hidden from an external observer. Any stealth assessment technique discussed in Sect. 3.2.1 can be used to create a student model. These techniques can be combined with each other to create a more complex student model that can be used to inform about multiple aspects of a student. For example, it can be used to measure their game play proficiency and how it affects their learning performance governed by their cognitive and emotional states. This model can then be used to provide feedback and remediation to students. Additionally, it can be used to dynamically adapt the game play to create an optimal learning environment suitable for their needs.

3.4.2 Stealth Assessment in CAGE

As indicated in Sect. 3.4.1, various ways of stealth assessment can be implemented at once to create the desired assessment in the game and generate a student model. The code snippet below illustrates how the predictive equations from [36] were used with the algorithm implemented by Baron [10] to create a stealth assessment within a CAGE game.

```
private AffectiveStates determineState()
{
  AffectiveStates state = AffectiveStates.None;
  //probs is an array containing probability values
  //e.g. probs[ATTENTION] is probability of
  //player being in the state of attention
  float boredom = (float)(-8.44 + (0.07 *
        probs[ATTENTION]) +
        (0.02 * probs[BROWFURROW]) +
        (0.06 * probs[BROWRAISE]) +
        (0.02 * probs[INNERBROWRAISE]) -
        (0.028 * probs[MOUTHOPEN]) -
        (0.03 * probs[SMILE]));

  boredom = Mathf.Exp(boredom);
  boredom = boredom / (1 + boredom);

  float flow = (float)(1.5 - (0.02 *
    probs[ATTENTION]) - (0.025 * probs[EYECLOSURE]) -
        (0.037 * probs[INNERBROWRAISE]) +
        (0.02 * probs[LIPPUCKER]) -
        (0.02 * probs[LIPSUCK]) +
        (0.02 * probs[MOUTHOPEN]) +
        (0.08 * probs[SMILE]));

  flow = Mathf.Exp(flow);
  flow = flow / (1 + flow);

  float frustration = (float)(1.85 -
    (0.02 * probs[ATTENTION]) -
    (0.03 * probs[BROWFURROW]) +
        (0.02 * probs[EYECLOSURE]) -
        (0.067 * probs[LIPPRESS]) -
        (0.03 * probs[LIPPUCKER]) +
        (0.03 * probs[LIPSUCK]));

  frustration = Mathf.Exp(frustration);
  frustration = frustration / (1 + frustration);

  if (frustration > boredom)
  {
    if (frustration > flow)
      state = AffectiveStates.Frustration;
    else
```

```
        state = AffectiveStates.Flow;
    }
    else
    {
      if (boredom > flow)
        state = AffectiveStates.Boredom;
      else
        state = AffectiveStates.Flow;
    }
    return state;
}
```

3.5 Validation of the CAGE Framework

Baron [10] found that the code re-usability is a desired aspect of game creation process when developing multiple serious games, which is facilitated by the CAGE framework. This indicates that the CAGE framework was effective at creating multiple games for learning with the help of content-agnostic mechanics. He also found that the amount of programming required to create subsequent games from the code of the first game was drastically reduced. As a result, any further game creation needed lesser amount of time and effort when compared to creating the game for the first time [10]. This suggests that the CAGE framework was effective in the rapid development of educational games. However, the small number of participants for this study could limit the generalizability of findings. Further, the research was conducted in a classroom environment, and the participants were working alone instead of a team, to develop these games.

Verma [1] implemented the stealth assessment into the CAGE framework and used it to dynamically adapt the game. Verma et al. [36] found that the facial expression were a better predictor of the player states of boredom, flow, and frustration, as compared to the facial emotions. Therefore, they used facial expression to model the three states and subsequently used it to adapt the game play. However, in another study, they found that the game adaptation was only effective for players who had lower domain knowledge of the learning content [46]. For example, in a game that teaches chemical equation, the players who already are an expert in balancing chemical equations would not benefit from game adaptation. However, players who join the game play with a low prior knowledge about chemical equation balancing would benefit more. Therefore, the game adaptation should be designed primarily for the low-domain learners. A limitation of the finding was that the participants were not evenly distributed across the test and the control groups, which could have caused biased results.

Baron [10] found that it leads to reduced engagement when playing multiple CAGE games since they employ similar game play mechanics. However, game

adaptation helped in alleviating the problem, and therefore player engagement was sustained across multiple CAGE game sessions [1]. Further, the adaptation helped improve player engagement when considered independent of the CAGE framework. The adaptation was implemented with the help of a student model built using dynamic Bayesian network. Therefore, it is recommended to create a dynamic game adaptation within the educational games to keep the player motivation levels intact. This study was conducted online during the pandemic and observed 35% dropout rate, which might have been caused by the issues in the UI or bugs in the game. Further, the effect size obtained in the results was low, and therefore, these results must be interpreted with caution.

Another experiment to establish the validity of the student model indicated that it can be applied in a content-agnostic way [38]. It involved comparing the inference from the embedded student model with an external assessment and found a significant correlation between the two. However, it depends on how the assessment is implemented in the game. Therefore, it is suggested to validate your own assessment before assuming its correctness. Nevertheless, the experiment shows that an assessment that has been designed to be content agnostic can be valid for multiple content domains.

References

1. Verma, V.: Content Agnostic Game Based Stealth Assessment. PhD thesis, Arizona State University (2021)
2. Shute, V., Spector, J.M.: Scorm 2.0 white paper: stealth assessment in virtual worlds. Unpublished manuscript (2008)
3. Kim, Y.J., Shute, V.: Opportunities and challenges in assessing and supporting creativity in video games. In: Video Games and Creativity, pp. 99–117. Elsevier, Amsterdam (2015)
4. Shute, V.: Stealth assessment in computer-based games to support learning. Comput. Games Instruct. **55**(2), 503–524 (2011)
5. Ventura, M., Shute, V., Small, M.: Assessing persistence in educational games. Des. Recommendations Adapt. Intell. Tutoring Syst. Learner Model. **2**(2014), 93–101 (2014)
6. Rotherham, A.J., Willingham, D.T.: 21st-century' skills. Am. Educ. **17**(1), 17–20 (2010)
7. Mislevy, R.J., Almond, R.G., Lukas, J.F.: A brief introduction to evidence-centered design. ETS Res. Rep. Ser. **2003**(1), i–29 (2003)
8. Malone, T.W., Lepper, M.R.: Making learning fun: a taxonomy of intrinsic motivations for learning. In: Aptitude, Learning, and Instruction, pp. 223–254. Routledge, Milton Park (2021)
9. Baron, T., Heath, C., Amresh, A.: Towards a context agnostic platform for design and assessment of educational games. In: European Conference on Games Based Learning, p. 34. Academic Conferences International Limited, Reading (2016)
10. Baron, T.: An Architecture for Designing Content Agnostic Game Mechanics for Educational Burst Games. PhD thesis, Arizona State University (2017)
11. Shute, V.J., Kim, Y.J.: Formative and stealth assessment. In: Handbook of Research on Educational Communications and Technology, pp. 311–321. Springer, Berlin (2014)
12. Verma, V., Baron, T., Bansal, A., Amresh, A.: Emerging practices in game-based assessment. In: Game-Based Assessment Revisited, pp. 327–346. Springer, Berlin (2019)
13. Pardos, Z.A., Heffernan, N.T.: Modeling individualization in a bayesian networks implementation of knowledge tracing. In: International Conference on User Modeling, Adaptation, and Personalization, pp. 255–266. Springer, Berlin (2010)

14. Chin, J., Dukes, R., Gamson, W.: Assessment in simulation and gaming: a review of the last 40 years. Simul. Gaming **40**(4), 553–568 (2009)
15. Shute, V., Masduki, I., Donmez, O., Dennen, V.P., Kim, Y.-J., Jeong, A.C., Wang, C.-Y.: Modeling, assessing, and supporting key competencies within game environments. In: Computer-Based Diagnostics and Systematic Analysis of Knowledge, pp. 281–309. Springer, Berlin (2010)
16. Shute, V., Wang, L.: Measuring problem solving skills in portal 2. In: E-Learning Systems, Environments and Approaches, pp. 11–24. Springer, Berlin (2015)
17. Mayer, I., van Dierendonck, D., Van Ruijven, T., Wenzler, I.: Stealth assessment of teams in a digital game environment. In: International Conference on Games and Learning Alliance, pp. 224–235. Springer, Berlin (2013)
18. Shute, V., Kim, Y.J.: Does playing the world of goo facilitate learning. In: Design Research on Learning and Thinking in Educational Settings: Enhancing Intellectual Growth and Functioning, pp. 359–387 (2011)
19. Chen, J.: Flow in games (and everything else). Commun. ACM **50**(4), 31–34 (2007)
20. Crisp, G.T., Assessment in next generation learning spaces. In: The Future of Learning and Teaching in Next Generation Learning Spaces. Emerald Group Publishing Limited, Bingley (2014)
21. Shute, V., Ventura, M., Small, M., Goldberg, B.: Modeling student competencies in video games using stealth assessment. Des. Recommendations Intell. Tutoring Syst. **1**, 141–152 (2013)
22. Papesh, M.H., Goldinger, S.D.: Memory in motion: movement dynamics reveal memory strength. Psychonomic Bull. Rev. **19**(5), 906–913 (2012)
23. Yamauchi, T., Xiao, K.: Reading emotion from mouse cursor motions: affective computing approach. Cognit. Sci. **42**(3), 771–819 (2018)
24. Freeman, J.B., Ambady, N.: Motions of the hand expose the partial and parallel activation of stereotypes. Psychol. Sci. **20**(10), 1183–1188 (2009)
25. Rheem, H., Verma, V., Becker, D.V.: Use of mouse-tracking method to measure cognitive load. In: Proceedings of the Human Factors and Ergonomics Society Annual Meeting, vol. 62, pp. 1982–1986. SAGE Publications Sage CA, Los Angeles (2018)
26. Faulkenberry, T.J., Testing a direct mapping versus competition account of response dynamics in number comparison. J. Cognit. Psychol. **28**(7), 825–842 (2016)
27. Element: mousemove event (2022)
28. Mouseevent.pagex (2022)
29. Unity3d (2022)
30. Unity3d: Input.mouseposition (2022)
31. Unreal engine: get mouse position (2022)
32. Visage Technologies (2022)
33. Affectiva (2022)
34. Ekman, P., Friesen, W.V.: Facial Action Coding System: A Technique for the Measurement of Facial Movement, Consulting Psychologists Press. Palo Alto, Santa Clara (1978)
35. iMotions Inc. Affectiva channel explained (2018). https://help.imotions.com/hc/en-us/articles/360011728719-Affectiva-channel-explained. Accessed 07 Aug 2022
36. Verma, V., Rheem, H., Amresh, A., Craig, S.D., Bansal, A.: Predicting real-time affective states by modeling facial emotions captured during educational video game play. In: International Conference on Games and Learning Alliance, pp. 447–452. Springer, Berlin (2020)
37. BayesServer. Dynamic bayesian networks – an introduction (2022)
38. Verma, V., Amresh, A., Craig, S.D., Bansal, A.: Validity of a content agnostic game based stealth assessment. In: International Conference on Games and Learning Alliance, pp. 121–130. Springer, Berlin (2021)
39. BayesServer. Dynamic bayesian networks c# api in bayes server (2022)
40. Scheuer, O., McLaren, B.M.: Educational data mining. In: Encyclopedia of the Sciences of Learning, pp. 1075–1079 (2012)

41. Baker, R.S.J.D., Gowda, S., Wixon, M., Kalka, J., Wagner, A., Salvi, A., Aleven, V., Kusbit, G., Ocumpaugh, J., Rossi, L.: Sensor-free automated detection of affect in a cognitive tutor for algebra. In: Educational Data Mining 2012 (2012)
42. D'Mello, S.K., Graesser, A.: Mining bodily patterns of affective experience during learning. In: Educational data mining 2010 (2010)
43. Mislevy, R.J., Oranje, A., Bauer, M.I., von Davier, A.A., Hao, J.: Psychometric Considerations in Game-Based Assessment. GlassLabGames (2014)
44. Shaffer, D.W., Squire, K.R., Halverson, R., Gee, J.P.: Video games and the future of learning. Phi delta kappan **87**(2), 105–111 (2005)
45. Typing of the dead, the description (2022)
46. Verma, V., Craig, S.D., Levy, R., Bansal, A., Amresh, A.: Domain knowledge and adaptive serious games: exploring the relationship of learner ability and affect adaptability. J. Educ. Comput. Res. **60**(2), 406–432 (2022)