Abstract—This study is a work in progress that aims to design and implement a smart learning environment based on virtual reality technology to aid the teaching and learning of programming concepts. The paper followed the approach of designing and modelling of requirement specification for the intended smart platform. This modelling approach is desirable in satisfying activities that engender the design and prototyping of the smart learning environment based on the design science research method. The study discusses the proposed architecture of the system, modelled the system with UML, presents a scenario-based model for teaching and learning of programming concepts, and connect the outcome to the future research.

Keywords—Smart learning environment, UML, programming, computing education, virtual reality, VR

I. INTRODUCTION

Programming education has become a mainstream subject taught in schools and higher education institutions [1] [2] [3] [4]. Teaching and learning of programming concepts have attracted scholars’ interest in the recent past [5]. Many studies try to investigate students’ performance in programming courses [6] [7]. These studies show that understanding programming concepts such as functions, loops, and recursion have become difficult for students of computer science [8] [9]. It is even more difficult for novices with little or no programming background because of the abstract nature of these programming concepts [10]. Anyango & Suleman [11] investigated programming topics that lecturers consider most difficult to comprehend by the students. Their study revealed that recursion is the most perceived difficult topic for novices pursuing computer science degree. Efforts to teach these programming concepts for better understanding have been made. For example, Henderson & Romero [12] studied how to teach recursion through a functional programming language. However, these challenges persist, as revealed by Piteira & Costa [13]. Nowadays, the proliferation of new technology presents the opportunity to provide educational tools that can enhance teaching/learning of these programming concepts. To this end, this study is focused on designing a smart learning environment (SLE) to provide better ways for teaching/learning of programming concepts by leveraging the latest technologies such as virtual reality (VR); mobile technology—smartphones, wearables; and ambient intelligence—GPS sensors. Smart learning environment is a technology-enhanced teaching and learning approach [14] [15], introduced as a higher level of the digital environment to support teaching and learning [16]. Although there is no consensus definition of a smart learning environment [17], scholars discussed the characteristic features that make SLE "smart" and what differentiate it from traditional learning environments [18]. These characteristics include adaptivity, context-awareness, ubiquity, personalization, automatic assessment, personalized feedback, motivation, and social-awareness. According to Agbo et al. [19], a smart learning environment is “an enhanced context-aware ubiquitous learning system that leverages social technologies, sensors and wireless communication of mobile devices to engage learners in hands-on experiences and present contents in a stimulating form; capable of connecting the learning community, increasing awareness of the physical environment, tracking and providing learning support”. One of the emerging technologies that support the development of SLE is the VR, which creates an environment that seems realistic but not normally experienced [23]. VR environment can provide truly personalized learning by adapting to sensed data from the users' senses and physical environment. Nowadays, smartphones, VR headset (e.g., Gear VR), MOGA pro controller, and many other emerging hardware supports the deployment of VR systems for user’s interaction and visualization experience.

In the bid to develop the SLE prototype to support the teaching of basic programming concepts, this study focuses on eliciting the requirements and modelling the proposed system using the Unified Modelling Language (UML). The requirement definition phase is a key component in the design science research (DSR) method [20]. The DSR method, according to Johannesson & Perjons [20], involves five phases, which include problem explication, requirements definition, prototype design, prototype demonstration, and artifact evaluation. These DSR phases allow for iterative refinement of problems, solutions, and approaches until the desired outcome is achieved. After the requirement specification, being the focus of this study, we intend to conduct a co-creation process of the prototype involving the researcher and expected users.

In requirement definition, scholars propose different notations [21]. One of these notations is natural language, which is too ambiguous, imprecise, and lack visualization that aid the understanding of complex concepts. The UML is one of the formal notations that is commonly used as a standard for modelling requirement specification [22]. UML is a standard object-oriented modelling language which allows for visualising, specifying, and documenting software systems. This paper employs UML to design a scenario-based requirement definition of a smart learning environment for learning programming concepts within a VR environment. Similar studies that use UML modelling tools for engineering software systems exist, for example, Iordan and Panoiu [23].
However, the use of UML for designing a smart learning environment that is based on VR technology needs to be explored; hence, the focus of this paper.

This paper is organized as follows: section two presents the proposed VR-based smart learning environment architecture; section three presents the UML models of the intended system; section four describes a scenario of puzzle game for learning programming concepts in a VR-based SLE; and section five briefly discusses the study contributions, conclusions, and the future research.

II. PROPOSED ARCHITECTURE OF SMART LEARNING ENVIRONMENT

This section introduces the proposed smart learning environment architecture (see Figure 1). In the system architecture, the input data from the learner environment are acquired automatically or specified by the learner at every learning session. These input data, for example, learner’s geolocation, temperature, noise level, motion state, etc., are used by the system to support the learner intelligently.

For instance, providing recommendations on how to learn, when to learn, and what to learn at a given time can be helpful to the learner based on the contextual information the system is able to receive. Besides, input data from the user's environment, his/her prior knowledge, if it exists, is also determined and is used to control the internal behavior of the system to achieve a highly adaptive and personalized learning experience.

The VR-based architecture for the smart learning environment aims to immerse learners in a VR environment in terms of visualization and interactions. VR is a technology that creates an environment that seems realistic but not normally experienced [24]. The application of VR has been studied by researchers in different scenarios [25]. This study draws motivation from Bouali et al. [26] to explore VR for teaching and learning of programming concepts. Thus, the system is designed based on Hwang [27] recommendation for designing a smart learning environment, and a brief explanation follows:

a. The data layer provides different types of input data required to manipulate the system behavior. For example, a first-time learner is required to provide information regarding his/her learning objectives, previous knowledge, other related profile data that the system may need to provide a more personalized learning experience. Additionally, data from learners’ environments could come from their devices’ sensors to adapt learning to suit learners’ contextual needs.

b. The logic layer contains modules that collectively provide a smart learning experience. These modules will include instructions and program logics. Further, a 3D modeling of the game is expected to take place in this layer. Additionally, the use of C# programming language and the Unity game engine will be employed for the scripting of the game. The components, objects, and event-driven scripting that the Unity game engine provides are useful in creating simple games for learning.

c. The presentation layer is a 3-dimensional virtual environment with a head-mounted sensory display. With the smartphone placed in front of a headset, learners can control the behavior of the game object within the virtual environment by using a hand controller/joystick. While learning is taking place, learning objects (LO) automatically adapt to suit the learner based on a set of rules for sequencing LO and outcome of learning assessment carried out during the session. Similarly, sensed data from the device could also influence the behavior of the system in real-time at the presentation layer.

III. UML DESIGN OF SMART LEARNING ENVIRONMENT FOR TEACHING PROGRAMMING CONCEPTS

This section presents a UML design of an underway smart learning environment to teach programming concepts (i.e., variables, functions, and recursion). The use of UML to model educational tools is significant in order to achieve the design phase of the DSR, as endorsed by other authors [22] [28].

The use case diagram in Figure 2 shows the functionalities of the system (system requirements) where a learner is able to...
view and take lessons that are designed to teach basic programming concepts, practice quizzes at different stages of learning, and automatic assessment is done to evaluate learning outcome. Learners receive personalized feedback and recommendations based on the learning outcome. The lessons are game-based, which are modelled to contain LO. The outcome from the learner’s assessment controls the sequencing of LO. If a learner is evaluated to perform the below satisfactory score, the SLE automatically recommend LOs that simplify the lesson before advancing to the next higher level LO.

IV. A SCENARIO-BASED SMART LEARNING ENVIRONMENT FOR TEACHING PROGRAMMING CONCEPTS

Generally, the intended VR-based SLE is expected to hold a few puzzle games that depict real-world problems. While trying to provide solutions to these problems, the player visualizes programming concepts of function and recursion in an interactive manner with the game element for optimal engagement. In other words, during the gameplay, we try to demonstrate these programming concepts to aide player’s understanding. For example, we have presented a scenario in this paper called the river crossing puzzle (a computational thinking concept). Aside from teaching students how to acquire problem-solving skills and algorithmic thinking, the river crossing puzzle has been identified to teach programming concepts such as functions and recursion [29]. River crossing puzzle presents a problem where a farmer has got a number of items to be taken across the river. The available boat is able to carry the farmer and only one item at a time. Some combination of these items cannot mutually co-habit in the absence of the farmer. Hence, the farmer ensures that all the items cross the river unharmed.

To solve this problem, the farmer would need to create and call some functions within the gameplay environment and pass parameters to them. For example, function to get an object within the game and place it inside the boat; function to move the object across the river; etcetera will be instantiated and called during the game session. Similarly, some repetition will occur that may require a function to recall itself (recursion). All of these processes happening in the gameplay will demonstrate the practicality of function and recursion and explanation of each event within the gameplay be given to the learner by a game agent.

A learner can decide to continuously learn by progressing from one level to another or pause to continue at a later time. The system records the learning progress and evaluates the learner in order to provide personalized feedback, hints, and supports. Similarly, we have presented the sequence diagram of the SLE platform to further show detailed interaction that happens between the actors (users) and the objects (characters/assets). Figure 4 depicts the different sequences of interactions that take place during a particular use case instance. The actor triggers the app after connecting various devices such as Google Cardboard and smartphone. The system initiates a session and responds to the actor with a welcome message and identifies the learner’s status, environment, and context. Actors are then redirected to proceed to the virtual learning environment and can control the environment with an input device. As learning progresses at each level, the system evaluates the learner's score. If the score is satisfactory, the system automatically redirects the learner to the next stage of the lesson; otherwise, the system provides hints and personalized support that allows the learner to gain better knowledge.

Figure 3 is a UML activity model that illustrates how the SLE platform immerses a learner within a VR environment by transforming the learner from the basic concept of programming represented as learning object into a more difficult concept. The diagram shows activities within the system and the flow of control that takes place from one activity to another. For example, a new learner is identified and prompted to provide some profiling data that may be useful to the system to make adaptive and personalized learning possible.

A learner can decide to continuously learn by progressing from one level to another or pause to continue at a later time. The system records the learning progress and evaluates the learner in order to provide personalized feedback, hints, and supports. Similarly, we have presented the sequence diagram of the SLE platform to further show detailed interaction that happens between the actors (users) and the objects (characters/assets). Figure 4 depicts the different sequences of interactions that take place during a particular use case instance. The actor triggers the app after connecting various devices such as Google Cardboard and smartphone. The system initiates a session and responds to the actor with a welcome message and identifies the learner’s status, environment, and context. Actors are then redirected to proceed to the virtual learning environment and can control the environment with an input device. As learning progresses at each level, the system evaluates the learner's score. If the score is satisfactory, the system automatically redirects the learner to the next stage of the lesson; otherwise, the system provides hints and personalized support that allows the learner to gain better knowledge.

IV. A SCENARIO-BASED SMART LEARNING ENVIRONMENT FOR TEACHING PROGRAMMING CONCEPTS

Generally, the intended VR-based SLE is expected to hold a few puzzle games that depict real-world problems. While trying to provide solutions to these problems, the player visualizes programming concepts of function and recursion in an interactive manner with the game element for optimal engagement. In other words, during the gameplay, we try to demonstrate these programming concepts to aide player’s understanding. For example, we have presented a scenario in this paper called the river crossing puzzle (a computational thinking concept). Aside from teaching students how to acquire problem-solving skills and algorithmic thinking, the river crossing puzzle has been identified to teach programming concepts such as functions and recursion [29]. River crossing puzzle presents a problem where a farmer has got a number of items to be taken across the river. The available boat is able to carry the farmer and only one item at a time. Some combination of these items cannot mutually co-habit in the absence of the farmer. Hence, the farmer ensures that all the items cross the river unharmed.

To solve this problem, the farmer would need to create and call some functions within the gameplay environment and pass parameters to them. For example, function to get an object within the game and place it inside the boat; function to move the object across the river; etcetera will be instantiated and called during the game session. Similarly, some repetition will occur that may require a function to recall itself (recursion). All of these processes happening in the gameplay will demonstrate the practicality of function and recursion and explanation of each event within the gameplay be given to the learner by a game agent.

Figure 3 is a UML activity model that illustrates how the SLE platform immerses a learner within a VR environment by transforming the learner from the basic concept of programming represented as learning object into a more difficult concept. The diagram shows activities within the system and the flow of control that takes place from one activity to another. For example, a new learner is identified and prompted to provide some profiling data that may be useful to the system to make adaptive and personalized learning possible.

A learner can decide to continuously learn by progressing from one level to another or pause to continue at a later time. The system records the learning progress and evaluates the learner in order to provide personalized feedback, hints, and supports. Similarly, we have presented the sequence diagram of the SLE platform to further show detailed interaction that happens between the actors (users) and the objects (characters/assets). Figure 4 depicts the different sequences of interactions that take place during a particular use case instance. The actor triggers the app after connecting various devices such as Google Cardboard and smartphone. The system initiates a session and responds to the actor with a welcome message and identifies the learner’s status, environment, and context. Actors are then redirected to proceed to the virtual learning environment and can control the environment with an input device. As learning progresses at each level, the system evaluates the learner's score. If the score is satisfactory, the system automatically redirects the learner to the next stage of the lesson; otherwise, the system provides hints and personalized support that allows the learner to gain better knowledge.

IV. A SCENARIO-BASED SMART LEARNING ENVIRONMENT FOR TEACHING PROGRAMMING CONCEPTS

Generally, the intended VR-based SLE is expected to hold a few puzzle games that depict real-world problems. While trying to provide solutions to these problems, the player visualizes programming concepts of function and recursion in an interactive manner with the game element for optimal engagement. In other words, during the gameplay, we try to demonstrate these programming concepts to aide player’s understanding. For example, we have presented a scenario in this paper called the river crossing puzzle (a computational thinking concept). Aside from teaching students how to acquire problem-solving skills and algorithmic thinking, the river crossing puzzle has been identified to teach programming concepts such as functions and recursion [29]. River crossing puzzle presents a problem where a farmer has got a number of items to be taken across the river. The available boat is able to carry the farmer and only one item at a time. Some combination of these items cannot mutually co-habit in the absence of the farmer. Hence, the farmer ensures that all the items cross the river unharmed.

To solve this problem, the farmer would need to create and call some functions within the gameplay environment and pass parameters to them. For example, function to get an object within the game and place it inside the boat; function to move the object across the river; etcetera will be instantiated and called during the game session. Similarly, some repetition will occur that may require a function to recall itself (recursion). All of these processes happening in the gameplay will demonstrate the practicality of function and recursion and explanation of each event within the gameplay be given to the learner by a game agent.

Figure 3 is a UML activity model that illustrates how the SLE platform immerses a learner within a VR environment by transforming the learner from the basic concept of programming represented as learning object into a more difficult concept. The diagram shows activities within the system and the flow of control that takes place from one activity to another. For example, a new learner is identified and prompted to provide some profiling data that may be useful to the system to make adaptive and personalized learning possible.

A learner can decide to continuously learn by progressing from one level to another or pause to continue at a later time. The system records the learning progress and evaluates the learner in order to provide personalized feedback, hints, and supports. Similarly, we have presented the sequence diagram of the SLE platform to further show detailed interaction that happens between the actors (users) and the objects (characters/assets). Figure 4 depicts the different sequences of interactions that take place during a particular use case instance. The actor triggers the app after connecting various devices such as Google Cardboard and smartphone. The system initiates a session and responds to the actor with a welcome message and identifies the learner’s status, environment, and context. Actors are then redirected to proceed to the virtual learning environment and can control
STEPWISE DESCRIPTION OF THE RIVER CROSSING PUZZLE

Step 1: The player selects a character (farmer) and at least three items as explained in each level of the game below.

Step 2: The player instantiates functions to collect each item into the boat from the departure terminal and drop off at the destination terminal.

Step 3: The player calls the function to move each item across the river.

Step 4: Move function is recursively called after a conditional statement is evaluated.

TWO LEVELS DESCRIPTION OF THE PUZZLE GAME

Our conceptual design of the game will be modelled using the 3D Unity game engine, where assets and objects are classified into three categories:

1. First category is a universal set of animals containing subsets {goat, wolf}, {dog, cat}, & {lion, monkey}
2. Second category is a set of farm products {banana, watermelon}
3. Third category contains sets of farmers (river-crossers)

Level 1 rules: In no order, one grouped asset is picked from the first category, one asset is picked from the second, and one river-crosser is picked from the third category, respectively. Alternatively, the arrangement of the assets can be pre-defined within the game so that players do not need to select them. These options can be agreed upon during the co-design process of the game with prospective users.

Gameplay for level 1:
The river-crosser takes selected objects across the river with the function MOVE, which can be recursively called based on the following conditions.

i. if objects (animals) cannot co-habit on their own without causing harm to each other
ii. if an object (animal or farm product) remains to be moved across the river

The task is to move all the objects across the river with a limited number of calling the function MOVE.

After completing the level 1 task, the game rewards, gives feedback/recommendations, and initiates the next level.

Level 2 rules: Three objects (animals), two objects (farm produce), and one river-crosser are selected.

Gameplay for level 2:
Rules of the gameplay in level 1 apply. However, the number of objects is increased; therefore, the game becomes harder, and the task remains to limit as much as possible, the number of the recursive call to the function MOVE in order to complete the task with the best score.
ACKNOWLEDGMENT

Work by the author, Solomon Sunday Oyelere was supported by the Jenny and Antti Wihuri Foundation.

REFERENCES


