Towards Intelligent Semi-automated Game Generation in SimSYS: A Component-based Approach

**UTDCS-0?-13**

**The University of Texas at Dallas**

**MU-CTL-0?-13**

**Marquette University**

**August 2013**

**Kaleb Breault, Marquette University**

**Ricardo Daconceicao, The University of Texas at Dallas**

**Carsten Locke, The University of Texas at Dallas**

**Chris Mojica, The University of Texas at Dallas**

**Kendra M. L. Cooper, The University of Texas at Dallas**

**C. Shaun Longstreet, Marquette University**

# Abstract

The use of serious educational games has many advantages, offering immersive, engaging and fun environments that require deep thinking and complex problem solving within a construct of overcoming obstacles and challenges. Developing new games, however, to support broad and rapidly evolving disciplines has remained time consuming, expensive, and requiring the expertise of game designers, software developers, software engineering educators, and players. Here, an intelligent semi-automated component-based engineering approach for generating serious educational games is proposed, which enables educators to rapidly and independently develop their own games across diverse educational topics. A little more detail here later

# Table of Contents

Abstract ii

Table of Contents iii

List of Figures vi

List of Tables vii

1 Introduction 1

2 Overview of the Intelligent Semi-automated Game Generation Approach 3

2.1 Interactive Wizard User Interface 3

2.2 Create Component-based Game Engine 4

2.3 Repository 5

3 Background: Analytic Hierarchy Process 7

4 Game Components 10

4.1 Character Layer 11

4.2 Theme Layer 11

4.3 Lesson Layer 14

4.4 Challenge Layer 16

4.5 Locale Layer 18

4.6 Subject Layer 20

5 Create Component-based Game Engine Modules 22

5.1 Select Game Components 22

5.2 Create (Load) Game Objects 23

5.3 Compose Game Components 24

5.4 Output Game 25

6 Tool Support (…still needs work…) 26

6.1 Select Game Components Module 26

6.2 Create Game Components Module 26

6.3 Compose Game Components Module 26

6.4 Output Game xxx 26

7 Validation 33

7.1 Test Case #1 33

7.1.1 Description 33

7.1.2 Input 33

7.1.3 Expected Output 33

7.1.4 Actual Output 33

7.1.5 Test Status 35

7.2 Test Case #2 35

7.2.1 Description 35

7.2.2 Input 36

7.2.3 Expected Output 36

7.2.4 Actual output 36

7.2.5 Test Status 38

7.3 Test Case #3 38

7.3.1 Description 38

7.3.2 Input 38

7.3.3 Expected Output 39

7.3.4 Actual Output 39

7.3.5 Test Status 41

7.4 Test Case #4 41

7.4.1 Description 41

7.4.2 Input 41

7.4.3 Expected Output 42

7.4.4 Actual Output 42

7.4.5 Test Status 43

8 Conclusions and Future work 44

References 46

# List of Figures

Figure 2: A short, small example of AHP 8

Figure 3 An actual example of the AHP algorithm in action 9

Figure 4 Component Overview 11

Figure 5 Character Component Chip 11

Figure 6 Theme Component Chip 12

Figure 7 Lesson Component Chip 15

Figure 8 Challenge Component Chip 16

Figure 9 Locale Component Chip 18

Figure 10 Subject Component Chip 20

# List of Tables

[Table 1 Characters XML Example 11](#_Toc362371220)

[Table 2 Theme XML Example 12](#_Toc362371221)

[Table 3 Lesson XML Example 15](#_Toc362371222)

[Table 4 Challenge XML Example 16](#_Toc362371223)

[Table 5 Locale XML File Example 19](#_Toc362371224)

[Table 6 Subject XML Example 21](#_Toc362371225)

[Table 7 Select Game Components Algorithm 22](#_Toc362371226)

[Table 8 Create Game Component Objects Algorithm 23](#_Toc362371227)

[Table 9 Compose Game Components Algorithm 24](#_Toc362371228)

[Table 10 Output Game Algorithm 25](#_Toc362371229)

# Introduction

Educational infrastructures face significant challenges including the need to rapidly, widely, and cost effectively introduce new or revised course material; encourage the broad participation of students; support traditional and emerging “classrooms” (e-learning, distributed classrooms), and address changing student motivations and attitudes. The course material needs to address learning objectives, which span subject specific content and transferable skills, such as collaboration, critical thinking, creative thinking, problem solving, reasoning abilities, learning to learn, professionalism, and decision making.

Serious educational games (SEGs) have significant pedagogical potential as they provide immersive, engaging and fun environments that require deep thinking and complex problem solving within a construct of overcoming obstacles and challenges [3][5][11][13]. They create interactive student-centered environments rather than a passive content-centered classroom environment. This allows students to create a personalized learning experience, progressively incorporating new knowledge and scaffolding it into what they already know. Because each student is able to engage course-based material at his or her own pace; underprepared or at-risk students can focus on needed skills at their convenience. Feedback is frequent and immediate, thereby reinforcing mastery of fundamental skills required for advancing further into the game. Integrating games into curriculum creates a highly motivating learning environment; it draws on students’ sense of fantasy and amusement; it is self-directed, appealing to individual student’s curiosity; and it is a continuous challenge wherein any existing tasks or knowledge that appears incomplete, inconsistent or incorrect motivates a student to foster deeper levels of learning.

There are many potential benefits to adopting games into curricula, but where can they be found? As with other software applications, they can be acquired off-the-shelf (e.g.,[8]), by modifying an existing game (e.g., [7][10]), or by developing new ones. Games are complex applications; developing new ones has been time consuming, expensive, and has required substantial expertise from diverse stakeholders: game developers, software developers, educators, and players. Research in the game community to improve this situation with semi-automated game generation approaches is just beginning to receive attention (refer to the related work discussion in Section IV).

The component-based software engineering community continues to invest substantial effort to support the timely, cost effective development of large-scale, complex systems [4][12]. At a very high level, this community considers software development as a problem involving the selection and composition of re-usable, high quality components, like assembling “lego” blocks. The components have gone through a resource intensive development process to specify, design, implement, test, and document; they have comprehensive interface descriptions including a specification of their functional and non-functional capabilities. The non-functional description includes quality of service attributes (e.g., performance). The specification, selection, and composition of components have received considerable attention. For example, formal notations, which offer a means to specify components concisely and unambiguously, such as XML, fuzzy logic, first-order logic as well as architectural description languages and coordination languages have been explored.

A component-based software engineering approach for rapidly developing serious educational games across diverse educational topics does not appear to be available in the literature. Here, an intelligent semi-automated component-based serious educational game generation approach, ISEGCB, is presented, with a focus on the game component models and intelligent selection. Based on a meta-model presented in earlier results [ref], six game component models are defined in the ISEGCB (theme, locale, subject, characters, lesson, and challenge), which cut across the game. The components are viewed as layers of a game; they have dependencies in order to generate a consistent game. For example, characters may appear throughout the game: across the game introduction, in a challenge midway through, and the wrap-up. The game components are specified in XML, a standard, formal notation that is considered straightforward to understand and load in tool support with established libraries.

Repositories are formed from collections of these components and a multi-criteria decision-analysis algorithm called the Analytic Hierarchy Process (AHP) is used to select one of each component to create a game. The AHP algorithm uses a system of weighted criteria along with intrinsic values on choices to find the choice with the best overall score. As a requirement of the AHP algorithm, there is a separate set of metadata, in XML format, that is used to describe each of the components. Additionally, there are some inter-component dependencies that are addressed in the form of an input wizard. This wizard homogenizes the input so that a coherent game is made from a set of basic questions about how the final game should look.

A few sentences here on the selection approach xxx metadata xxx inter-component dependencies xxx the analytic hierarchy process (AHP [ref] xxx yyy zzz. The components are assembled into games using and output as game specifications in XML. The games generated are in the SimSYS [6] format.

The structure of this report is as follows. Section 2: Overview of the Intelligent Semi-automated Game Generation Approach, Section 3: Background: Analytic Hierarchy Process, Section 4: Game Components, Section 5: Component Selection Approach, Section 6: Tool Support, Section 7: Validation

# Overview of the Intelligent Semi-automated Game Generation Approach

The SEGCB approach is organized into three layers (Fig. 1): interactive wizard interface; create components based serious educational game engine; and the repository.

**Repository (game assets, game components, game scripts)**

XML

Game Component Collections

Game Assets (image files, audio files, educational material)

XML

Game Scripts

**Create Component-based Game Engine**

**Game Script Output**

Theme

Locale

Subject

Character

Lesson

Challenge

…

**Interactive Wizard User Interface**

**Game**

**Composition**

**Component Object Creation**

**Component Selection**

<?xml version="1.0" encoding="UTF-8" standalone="true"?>

<!-- -->

<!-- Top Level Game -->

<!-- Test Game 1 contains 1 act, 1 scene and 2 screens -->

<!-- -->

[<game>](file:///C:\Users\GameSpecifications\Test%20Game%20descriptions\Test%20Game%201\TestGame_1_XML.xml)

<!-- -->

<!-- The game contains a default player who is rewarded with 5000 points after playing the game -->

<!-- -->

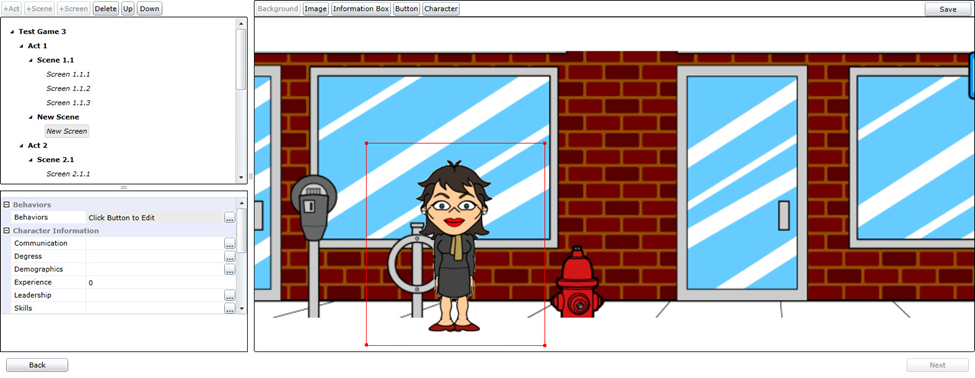
[<gameElement xsi:type="**PLAYER**" xmlns:xsi="**http://www.w3.org/2001/XMLSchema-instance**">](file:///C:\Users\GameSpecifications\Test%20Game%20descriptions\Test%20Game%201\TestGame_1_XML.xml)<location>OSR</location><size>LARGE</size>

<!-- -->

<!-- The profile of the player starts here -->

<!-- -->

[<profile>](file:///C:\Users\GameSpecifications\Test%20Game%20descriptions\Test%20Game%201\TestGame_1_XML.xml)<name> Default Player </name><type> Protagonist </type><skills> Software Engineering, Project Management, Configuration Management </skills><experience> 0 </experience><communication> Good </communication><leadership> Good </leadership><teamwork> Good </teamwork><demographics> Male, Caucasian </demographics><degrees> NONE </degrees></profile>



*Components*

*+ meta-data*

Wizard support

* Create new game
* Preview (graphics, audio)
* Edit/Save/Load

<?xml version="1.0"?>

[<Challenge><LessonChallenges><LessonChallenge>](file:///C:\Users\IntSemi-automatedGameGenerationComponent\Samples\Sample%20Game%20XMLs\Challenge.xml)<Buttons> </Buttons>[<ChallengeOptions><ChallengeOption>](file:///C:\Users\IntSemi-automatedGameGenerationComponent\Samples\Sample%20Game%20XMLs\Challenge.xml)<ChallengeOptionType>BUTTON</ChallengeOptionType><ButtonLocationType>CHALLENGE\_1</ButtonLocationType>[<Reward>](file:///C:\Users\IntSemi-automatedGameGenerationComponent\Samples\Sample%20Game%20XMLs\Challenge.xml)<Points>-2147483646</Points></Reward><Text>2</Text><Timer/><TransitionType>ADDITIONAL</TransitionType>[<AdditionalScreens><AdditionalScreen xsi:type="**failureScreen**" xmlns:xsi="**http://www.w3.org/2001/XMLSchema-instance**"><Buttons><Button>](file:///C:\Users\IntSemi-automatedGameGenerationComponent\Samples\Sample%20Game%20XMLs\Challenge.xml)<ButtonLocationType>NEXT</ButtonLocationType><Text>Retry</Text><Timer/><TransitionType>CURRENT\_CHALLENGE</TransitionType></Button></Buttons>[<LOCharacters><Character>](file:///C:\Users\IntSemi-automatedGameGenerationComponent\Samples\Sample%20Game%20XMLs\Challenge.xml)<CharacterType>VILLIAN</CharacterType><Timer/></Character>[<Character>](file:///C:\Users\IntSemi-automatedGameGenerationComponent\Samples\Sample%20Game%20XMLs\Challenge.xml)<CharacterType>HERO</CharacterType><Timer/></Character></LOCharacters>

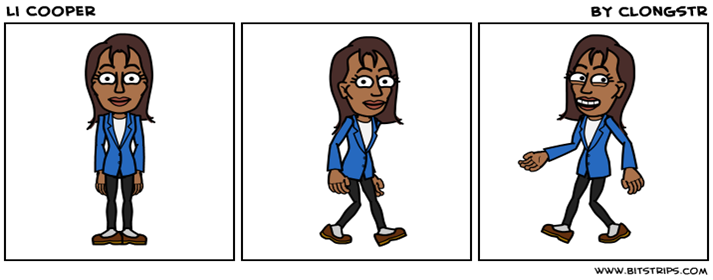
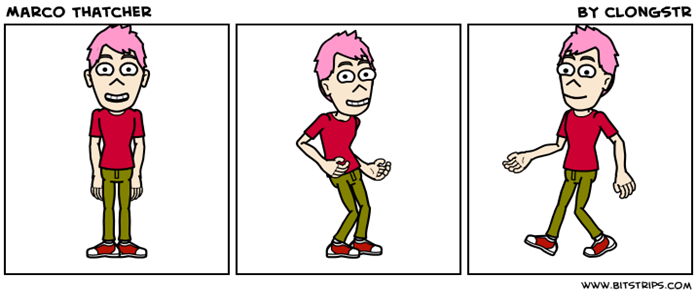
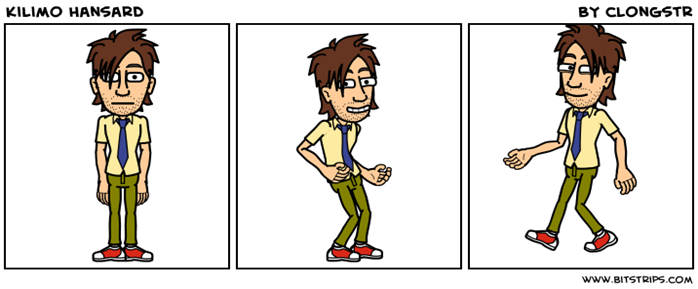
<?xml version="1.0"?>

[<Challenge><LessonChallenges><LessonChallenge>](file:///C:\Users\IntSemi-automatedGameGenerationComponent\Samples\Sample%20Game%20XMLs\Challenge.xml)<Buttons> </Buttons>[<ChallengeOptions><ChallengeOption>](file:///C:\Users\IntSemi-automatedGameGenerationComponent\Samples\Sample%20Game%20XMLs\Challenge.xml)<ChallengeOptionType>BUTTON</ChallengeOptionType><ButtonLocationType>CHALLENGE\_1</ButtonLocationType>[<Reward>](file:///C:\Users\IntSemi-automatedGameGenerationComponent\Samples\Sample%20Game%20XMLs\Challenge.xml)<Points>-2147483646</Points></Reward><Text>2</Text><Timer/><TransitionType>ADDITIONAL</TransitionType>[<AdditionalScreens><AdditionalScreen xsi:type="**failureScreen**" xmlns:xsi="**http://www.w3.org/2001/XMLSchema-instance**"><Buttons><Button>](file:///C:\Users\IntSemi-automatedGameGenerationComponent\Samples\Sample%20Game%20XMLs\Challenge.xml)<ButtonLocationType>NEXT</ButtonLocationType><Text>Retry</Text><Timer/><TransitionType>CURRENT\_CHALLENGE</TransitionType></Button></Buttons>[<LOCharacters><Character>](file:///C:\Users\IntSemi-automatedGameGenerationComponent\Samples\Sample%20Game%20XMLs\Challenge.xml)<CharacterType>VILLIAN</CharacterType><Timer/></Character>[<Character>](file:///C:\Users\IntSemi-automatedGameGenerationComponent\Samples\Sample%20Game%20XMLs\Challenge.xml)<CharacterType>HERO</CharacterType><Timer/></Character></LOCharacters>

<?xml version="1.0"?>

[<Challenge><LessonChallenges><LessonChallenge>](file:///C:\Users\IntSemi-automatedGameGenerationComponent\Samples\Sample%20Game%20XMLs\Challenge.xml)<Buttons> </Buttons>[<ChallengeOptions><ChallengeOption>](file:///C:\Users\IntSemi-automatedGameGenerationComponent\Samples\Sample%20Game%20XMLs\Challenge.xml)<ChallengeOptionType>BUTTON</ChallengeOptionType><ButtonLocationType>CHALLENGE\_1</ButtonLocationType>[<Reward>](file:///C:\Users\IntSemi-automatedGameGenerationComponent\Samples\Sample%20Game%20XMLs\Challenge.xml)<Points>-2147483646</Points></Reward><Text>2</Text><Timer/><TransitionType>ADDITIONAL</TransitionType>[<AdditionalScreens><AdditionalScreen xsi:type="**failureScreen**" xmlns:xsi="**http://www.w3.org/2001/XMLSchema-instance**"><Buttons><Button>](file:///C:\Users\IntSemi-automatedGameGenerationComponent\Samples\Sample%20Game%20XMLs\Challenge.xml)<ButtonLocationType>NEXT</ButtonLocationType><Text>Retry</Text><Timer/><TransitionType>CURRENT\_CHALLENGE</TransitionType></Button></Buttons>[<LOCharacters><Character>](file:///C:\Users\IntSemi-automatedGameGenerationComponent\Samples\Sample%20Game%20XMLs\Challenge.xml)<CharacterType>VILLIAN</CharacterType><Timer/></Character>[<Character>](file:///C:\Users\IntSemi-automatedGameGenerationComponent\Samples\Sample%20Game%20XMLs\Challenge.xml)<CharacterType>HERO</CharacterType><Timer/></Character></LOCharacters>

…



*Existing*

*Standards,*

*Course material*



Figure 1: Overview of the Component-based Serious Educational Game Generation Approach

## Interactive Wizard User Interface

The User Interface provides wizards to acquire information from the game developer; preview the game generated; and edit the game generated as needed. The information acquired from the game developer is used to characterize and create a useful game. The information includes: game subject area; target player audience; intended context to use the game; level of difficulty; length; and the style of the game. For example, a teacher for grade 4 students in WI, U.S.A. may need a short game to use in his class as an early part of the lesson. The teacher may characterize the game needed as follows:

* subject area: basic algebra
* target player: grade 4 student in WI, U.S.A.
* use of the game: in class time
* level of difficulty of the game: easy
* length of the game: short
* style of the game: cartoon
* educational standard: WI state standards for grade 4 mathematics

The wizard provides the options the game developer can select from; the current prototype uses dropdown menus.

For example, the user can choose from subject areas (basic algebra, reading), levels of difficulty (easy, moderate, difficult), length of game (short, moderate long), target player (grade or experience level). As additional repositories of subject area assets are added to the repository, the options can be readily updated. In the future it will be interesting to explore alternative interface techniques, natural language processing techniques for speech or text input.

Using the information acquired, the game developer can create the game; the Create Component-based Serious Educational Game Engine is used to accomplish this. The teacher can preview the generated game, edit it if desired, and save the game.

## Create Component-based Game Engine

The Create Component-based Serious Educational Game Engine is a pipeline of four modules: component selection; component object creation; game composition; and game script output. Currently, the search, ranking, and selection are done manually for the component selection; the object creation, composition, and game script output modules have been prototyped.

The Component Selection module uses the characterization information provided by the game developer to search, rank, and select a collection of game components that are used to assemble a game. It does so by implementing an Analytic Hierarchy Process (AHP) algorithm to weigh the possible choices with the characterization information. The AHP algorithm is not used to give a perfectly correct answer, it is used to find a component that fits the search criteria as best as possible given restrictions on scope of the Repository. The game components are retained in the Repository; there are collections of theme, locale, subject, character, lesson, and challenge components specified in XML. Meta-data about the collections are used in the search; these are also represented in XML. The output of this module is a collection of the selected game components’ names (e.g., file names); this collection is used by the Component Creation module.

The Component Creation module uses the collection of component names provided to it retrieve the components from the Repository and convert them into a collection of working objects. The component creation is in a separate module in order to encapsulate the capability to load the XML files from the Repository. The output of this module is a collection of working objects; this collection is used by the Game Composition module.

The Game Composition module assembles a game out of the components by using an overall game structure as a wireframe. The games are organized as a sequence of Acts (intro act, internal acts, outro act); the acts have screens. For example, an internal (learning) act has screens to introduce, present, and wrap-up lessons and challenges. The acts are built in order: Intro Act, Learning Act(s), then the Outre Act. Once all the acts are created, they are added to the game’s wireframe.

Once the game is built it can be passed to the game script output module, which exports the game into the desired format. Currently, the game is output as an XML file in the SimSYS game specification format.

## Repository

The repository contains the re-usable game components, game assets, and game scripts. The game components, described in more detail in Game Components, are: challenge, characters, lesson, locale, subject, and theme. The challenge entity represents the challenge presented at the end of the lesson and is currently limited to a multiple choice quiz of varying length. The characters entity contains information about what names and assets should be used for each of the four supported character types (hero, villain, player, alternate). The lesson entity encompasses a single lesson which directs the student to learn some objective. The locale entity holds all information describing the location of the experience such as the background, character positions, background and foreground object positions and locations of where text can be displayed. This allows other entities to simply reference a named location, such as a character’s speech bubble, and provide the accompanying text. The subject entity describes the overall subject that is being taught and any introductory text. The theme is the story aspect of the game and contains not only the information used to build the intro and outro acts, but also story snippets, which surround each learning act to progress the story throughout the learning experience.

The game asset collection in the repository includes graphic images, audio files, and existing standards and course material. The graphic images are organized by characters, props, backdrops, and interactions; images are in the standard .png format, 200x300 pixels. For each character, there are 56 images to provide a variety of poses (standing, sitting, walking, talking, facing different directions (left, right, straight ahead). Images of props include furniture, easels, podiums, clocks, computers, phones, and so on. The backdrops provide a background image for a setting. For example, backdrop images for offices, classrooms, meeting rooms, medieval castles, forests, outer space, and so on are stored for re-use. The interaction images include information boxes, conversation bubbles, and buttons. Audio files include sound effects and music; these are stored in standard .wav or .mid files format. Existing standards and course material are also stored in the repository as a valuable reference material. Course material may be in presentation slides, lesson notes, course books, homework assignments, examinations; this ad-hoc collection may be stored in a wide variety of file formats.

The game script collection in the repository includes the games generated. Games can be created, played, and then improved (edited/saved) as needed. The games are organized by broad subject area (professional development, science, arts and humanities, medicine, business, and so on). A taxonomy for serious educational games will be explored to improve this categorization.

# Background: Analytic Hierarchy Process

The AHP is a structured system for making complex decisions with multiple criteria. It was developed by Thomas Saaty in the 70s and has been It begins with the creation of a hierarchy of the criteria. Then after this is established, the programmer then compares the elements to each other two at a time, assigning values to every element in relation to all the other elements. [11] The intrinsic values of the choices are then given weight by these elements and the choice with the highest overall score is chosen. It is an established approach that has been applied to ranking and selection problems such as deciding how best to reduce climate change [2], selecting university faculty [5], and decisions regarding the location of offshore manufacturing plants [1].

An example of AHP is deciding on a car to buy from a list and having different priorities on different aspects of the car such as safety, Cost, seating/storage, warranty. The searcher gives each of these four attributes a “weight” in relation to each other. For instance, safety is twice as important as cost, and cost is a third as important as seating/storage etc, etc.( see Fig 1) Each of the alternative decisions are given a score for their attributes and a weighted score is then calculated using the “weights” of the attributes. For example, if a car scores a 7/10 in safety it would earn 70% of the possible points for that category. The weighted scores are then added up and the contestant is given an overall score. The contestants are then sorted into a list with the highest/ best choice at the top. For slight randomization, we randomly generate a number, if it is divisible by three take the highest weighted component, else continue onto the next component with a new random number. (The best choice is the default case if a number divisible by 3 is never generated).

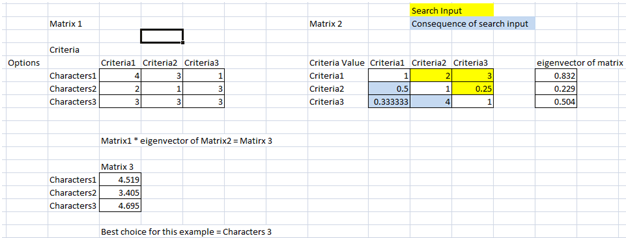


Figure 2: A short, small example of AHP

Matrix1 is the list of choices and their criteria score. Matrix 2 shows the relations of each criteria in respect to each other. In this example, criteria 1 is twice as important as criteria 2 and three times more important than criteria 3, additionally criteria 3 is four times as important as criteria 2. Given these weights this makes Characters3 the best choice even though Characters1 scored better in the most important category.

A sample test case from our AHP algorithm has many more criteria and choices to choose from. For instance here is an example from a test case searching for a relevant Subject XML file:

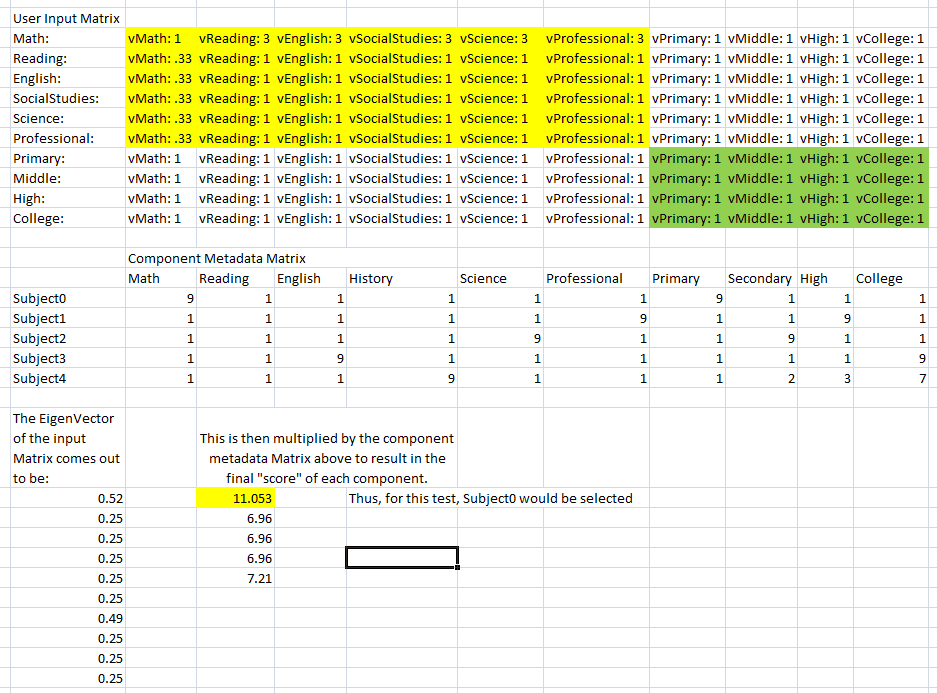


Figure 3 An actual example of the AHP algorithm in action

The top matrix is an example of the inputs and the component’s weights while the middle matrix is the list of choices and their values for the given criteria.

# Game Components

The overall structure of the game is broken down into six different layers, or components. Each different component consists of two XML files. One file is the Meta-data for searching capabilities and the other being the core and functionality of the component. Each game consists of seven different types of screens:

* Intro Screen
* Story Intro Screen
* Lesson Screen
* Challenge Screen
* Failure Screen
* Story Outro Screen
* Outro Screen

Each screen requires the following tasks before the screen can be considered complete:

* Place Backdrop
* Characters must be:
  + Provide
  + Selected
  + Placed
* Textboxes must be:
  + Created
  + Placed
* Generic Interaction Objects must be:
  + Created
  + Placed
* Decorations (Optional)
  + Created
  + Placed
* Misc. (Optional)
  + Timer
  + Reward
  + Movement

In this section, we will be discussing what tasks each component handles.

(Figure 4 is still under construction)

Figure 4 Component Overview

## Character Layer

Available Characters

Characters

Character

Meta-data

Retrieve

Character Core

*- Role*

*- Name*

Figure 5 Character Component Chip

The character component is accessed by the theme component, lesson component, and the challenge component. It provides the characters that will be populating the game.

Table 1 Characters XML Example

|  |
| --- |
| <Characters>  <Alt>  <Directory>character\_1</Directory>  <Name>Chareeeesa</Name>  <Prefix>char1</Prefix>  </Alt>  <Hero>  <Directory>character\_19</Directory>  <Name>Sir Solvesalot</Name>  <Prefix>char19</Prefix>  </Hero>  <Villain>  <Directory>character\_22</Directory>  <Name>Calcutron</Name>  <Prefix>char22</Prefix>  </Villain>  <Player>  <Directory>character\_10</Directory>  <Name>Siva</Name>  <Prefix>char10</Prefix>  </Player>  </Characters> |

## Theme Layer

Place Generic Interaction Props

Theme

Theme

Meta-data

Retrieve

Place Backdrop

Theme Core

Intro/Outro

*- Select Generic Interaction Props*

*- Select Characters*

*- Create Textboxes*

*- Character Movement (in the works)*

Place Textboxes

Place Set Decoration Props

Select Characters

Create Textboxes

Story Intro/Outro

Figure 6 Theme Component Chip

The theme component is unique in that it creates the Into Screen and the Outro Screen completely with the help of only the character component. It does all of the following for the Intro Screen and Outro Screen:

* Selects Characters provided by the Character Component
* Selects Game Interaction Props
* Creates Textboxes
* Places Characters to a specified location on the screen.

All of the other screens require the locale to complete the screen.

Table 2 Theme XML Example

|  |
| --- |
| <Theme>  <IntroScreens>  <IntroScreen>  <Background>Backdrops\Home Office.png</Background>  <ThemeButtons>  <entry>  <key>NEXT</key>  <value>  <Height>50.566680908203125</Height>  <LocX>581.83331298828125</LocX>  <LocY>332.75</LocY>  <PathToAsset></PathToAsset>  <Text>Continue</Text>  <Width>157.86669921875</Width>  <behavior>  <behaviorType>TRANSITION\_BEHAVIOR</behaviorType>  <displayName>Transition Behavior</displayName>  <Transition>NEXT\_SCREEN</Transition>  <trigger>Click</trigger>  </behavior>  <name>Next</name>  </value>  </entry>  </ThemeButtons>  <GameObjects>  <GameObject>  <Height>78.316665649414062</Height>  <LocX>327.14999389648438</LocX>  <LocY>123.69999694824219</LocY>  <PathToAsset>Props\GenericInteraction\TextBubble.png</PathToAsset>  <Width>247.89999389648438</Width>  </GameObject>  </GameObjects>  <InformationBoxes>  <InformationBox>  <Height>59.20001220703125</Height>  <LocX>346.88330078125</LocX>  <LocY>128.63331604003906</LocY>  <Text>Today's lesson was difficult. I wonder if Gooble can help!</Text>  <Width>210.28340148925781</Width>  <name>Today's lesson was difficult. I wonder if Gooble can help!</name>  </InformationBox>  </InformationBoxes>  <ThemeCharacters>  <entry>  <key>PLAYER</key>  <value>  <Height>292.91667175292969</Height>  <LocX>209.36663818359375</LocX>  <LocY>130.48335266113281</LocY>  <Text></Text>  <Width>171.433349609375</Width>  <CharacterAssetType>WALK\_RIGHT\_BEHIND</CharacterAssetType>  <CharacterType>PLAYER</CharacterType>  </value>  </entry>  </ThemeCharacters>  </IntroScreen>  (…) –More IntroScreens  <OutroScreens>  <OutroScreen>  (…) –Mimics The Intro Screen  </OutroScreen>  </OutroScreens>  <ThemeStories>  <ThemeStory>  <StoryIntroScreens>  <StoryIntroScreen xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xsi:type=*"themeStoryScreenIntro"*>  <Buttons>  <Button>  <ButtonLocationType>NEXT</ButtonLocationType>  <Text>Continue</Text>  <Timer></Timer>  <TransitionType>NEXT\_SCREEN</TransitionType>  </Button>  </Buttons>  <LOCharacters>  <Character>  <CharacterType>PLAYER</CharacterType>  <MovementType></MovementType>  <Timer></Timer>  </Character>  (…)  </LOCharacters>  <InformationBoxes>  <InformationBox>  <TextType>PLAYER</TextType>  <Text>Who are you?</Text>  <Timer></Timer>  </InformationBox>  (…)  </InformationBoxes>  </StoryIntroScreen>  </StoryIntroScreens>  <StoryOutroScreens>  <StoryOutroScreen xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xsi:type=*"themeStoryScreenOutro"*>  (…) –Mimics the StoryIntroScreen  </StoryOutroScreen>  </StoryOutroScreens>  </ThemeStory>  </ThemeStories>  </Theme> |

## Lesson Layer

Lesson

Lesson

Meta-data

Retrieve

Textboxes

Select Characters

Lesson

Core

*- Character Movement (in the works)*

*- Timer*

Figure 7 Lesson Component Chip

The lesson component provides what characters and text boxes will be on each screen. They will then be placed on the screen by the locale component. This is also where a movement feature will be implemented in the future.

Table 3 Lesson XML Example

|  |
| --- |
| <Lesson>  <LessonScreens>  <LessonScreen>  <Buttons>  <Button>  <ButtonLocationType>NEXT</ButtonLocationType>  <Text>Continue</Text>  <Timer></Timer>  <TransitionType>NEXT\_SCREEN</TransitionType>  </Button>  </Buttons>  <LOCharacters>  <Character>  <CharacterType>HERO</CharacterType>  <Timer></Timer>  </Character>  <Character>  <CharacterType>PLAYER</CharacterType>  <Timer></Timer>  </Character>  </LOCharacters>  <InformationBoxes>  <InformationBox>  <Text>To perform the addition operator you must add together the operands. EX: 2+4=6</Text>  <TextType>ALT1</TextType>  <Timer></Timer>  </InformationBox>  </InformationBoxes>  </LessonScreen>  </LessonScreens>    </Lesson> |

## Challenge Layer

Challenge

Challenge

Meta-data

Retrieve

Textboxes

Select Characters

Challenge

Core

*- Timer*

*- Reward*

*- Transition Type*

*- Character Movement (in the works)*

Figure 8 Challenge Component Chip

The challenge layer selects the characters and provides the textboxes that will again be placed on the Challenge screens and Failure Screen (Addiction Screen in XML) by the locale component. This is also where the timer and reward features will be handled. Movement on all challenge screens will be implemented here as well.

Table 4 Challenge XML Example

|  |
| --- |
| <Challenge>  <LessonChallenges>  <LessonChallenge>  <Buttons>  </Buttons>  <ChallengeOptions>  <ChallengeOption>  <ChallengeOptionType>BUTTON</ChallengeOptionType>  <ButtonLocationType>CHALLENGE\_1</ButtonLocationType>  <Reward>  <Points>-2147483646</Points>  </Reward>  <Text>2</Text>  <Timer></Timer>  <TransitionType>ADDITIONAL</TransitionType>  <AdditionalScreens>  <AdditionalScreen xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xsi:type=*"failureScreen"*>  <Buttons>  <Button>  <ButtonLocationType>NEXT</ButtonLocationType>  <Text>Retry</Text>  <Timer></Timer>  <TransitionType>CURRENT\_CHALLENGE</TransitionType>  </Button>  </Buttons>  <LOCharacters>  <Character>  <CharacterType>VILLIAN</CharacterType>  <Timer></Timer>  </Character>  <Character>  <CharacterType>HERO</CharacterType>  <Timer></Timer>  </Character>  <Character>  <CharacterType>PLAYER</CharacterType>  <Timer></Timer>  </Character>  </LOCharacters>  <InformationBoxes>  <InformationBox>  <Text>You answered incorrectly!</Text>  <TextType>ALT1</TextType>  <Timer></Timer>  </InformationBox>  <InformationBox>  <Text>Don't worry! You can try again.</Text>  <TextType>HERO</TextType>  <Timer></Timer>  </InformationBox>  </InformationBoxes>  <ChallengeOptions />  <Timer></Timer>  </AdditionalScreen>  </AdditionalScreens>  <ChallengeOptionType>BUTTON</ChallengeOptionType>  </ChallengeOption>  <ChallengeOption>  (…)  </ChallengeOption>  </ChallengeOptions>  <LOCharacters>  <Character>  <CharacterType>HERO</CharacterType>  <MovementType></MovementType>  <Timer></Timer>  <Character>  <CharacterType>PLAYER</CharacterType>  <Timer></Timer>  </Character>  </Character>  </LOCharacters>  <InformationBoxes>  <InformationBox>  <Text>You can do it!!</Text>  <TextType>HERO</TextType>  <Timer></Timer>  </InformationBox>  <InformationBox>  <Text>Using Algebra solve:</Text>  <TextType>CHALLENGE\_QUESTION</TextType>  <Timer></Timer>  </InformationBox>  <InformationBox>  <Text>2 + 4</Text>  <TextType>CHALLENGE\_DESCRIPTION</TextType>  <Timer></Timer>  </InformationBox>  </InformationBoxes>  <Timer>60</Timer>  </LessonChallenge>  </LessonChallenges>  </Challenge> |

## Locale Layer

Locale

Local

Meta-data

Retrieve

Place Backdrop

Place Generic Interaction Props

Locale

Core

*- Select Generic Interaction Props*

*- Select Set Decoration Props*

Place Set Decoration Props

Place Textboxes

Figure 9 Locale Component Chip

The locale component provides the backbone for all the screens excluding the intro screen and outro screen. As I mention before both of those screens are created completely by the theme and character components. The locale places the backdrop, generic interaction props, decoration props, and textboxes on the screen at a specified location. The textboxes and characters are not created by the locale, only placed. The interaction props and decoration props are created here however and then placed on the screen. This is done for the

Table 5 Locale XML File Example

|  |
| --- |
| <Locale>  <LocaleScreens>  <entry>  <key>LESSON</key>  <value>  <Background>Backdrops\Medieval.png</Background>  <Buttons>  <entry>  <key>BACK</key>  <value>  <Height>1</Height>  <LocX>1</LocX>  <LocY>1</LocY>  <Width>1</Width>  </value>  </entry>  (…) –Additional Buttons  </Buttons>  <Characters>  <entry>  <key>PLAYER</key>  <value>  <Height>91.26666259765625</Height>  <LocX>135.98329162597656</LocX>  <LocY>313.01666259765625</LocY>  <Width>104.21666717529297</Width>  <CharacterAssetType>RIGHT\_POINT\_NO</CharacterAssetType>  <CharacterType>PLAYER</CharacterType>  <Movements>  <Movement>  <AnimationSequences>  <AnimationSequence>WALK\_RIGHT\_OPEN</AnimationSequence>  <AnimationSequence>WALK\_LEFT\_BEHIND</AnimationSequence>  </AnimationSequences>  <EndX>136</EndX>  <EndY>313</EndY>  <MovementType>WALK\_ONTO\_SCREEN</MovementType>  <Speed>1</Speed>  <StartX>1</StartX>  <StartY>313</StartY>  </Movement>  <Movement>  <AnimationSequences>  <AnimationSequence>WALK\_RIGHT\_BEHIND</AnimationSequence>  <AnimationSequence>WALK\_LEFT\_OPEN</AnimationSequence>  </AnimationSequences>  <EndX>1</EndX>  <EndY>313</EndY>  <MovementType>WALK\_OFF\_SCREEN</MovementType>  <Speed>1</Speed>  <StartX>136</StartX>  <StartY>313</StartY>  </Movement>  </Movements>  </value>  </entry>  (…) –Additional Characters  </Characters>  <GameObjects>  <!-- Player text box -->  <GameObject>  <Height>143.68331909179688</Height>  <LocX>36.083301544189453</LocX>  <LocY>194.00001525878906</LocY>  <PathToAsset>Props\GenericInteraction\Regular Speech 2.png</PathToAsset>  <Width>155.40000534057617</Width>  </GameObject>  (…) –Additional Game Objects  <InformationBoxes>  <entry>  <key>PLAYER</key>  <value>  <Height>46.866683959960938</Height>  <LocX>61.36663818359375</LocX>  <LocY>224.21665954589844</LocY>  <Width>84.48333740234375</Width>  </value>  </entry>  (…) –Additional Information Boxes  </InformationBoxes>  </value>  <key>CHALLENGE</key> (…) –Mimics the Lesson Format  <key>FAILURE</key> (…) –Mimics the Lesson Format  <key>LESSON\_STORY\_INTRO</key> (…) –Mimics the Lesson Format  <key>LESSON\_STORY\_OUTRO</key> (…) –Mimics the Lesson Format  </LocaleScreens>  </Locale> |

## Subject Layer

Subject

Subject

Meta-data

Retrieve

Subject

Core:

Figure 10 Subject Component Chip

The subject currently stands independent of all the components. This component will be used in the future to reduce hardcoded titles in the Intro and Outro textboxes to allow each theme component to be interchangeable.

Table 6 Subject XML Example

|  |
| --- |
| <Subject>  <IntroText>Algebra</IntroText>  <Subject>Algebra Adventures</Subject>  </Subject> |

# Create Component-based Game Engine Modules

## Select Game Components

Your algorithm work here

// few sentences here, refer to Table 7

Table 7 Select Game Components Algorithm

|  |
| --- |
| **Input:** xxx.  **Output: yyy**.  … |

The selection of the components is now done through a graphical user interface consisting of several basic questions on how the game should look. Currently, the GUI is in its beta stages, and therefore is not very complex or attractive. However, it will be easily expandable, in both the options/answers to questions and the number of questions themselves. This is just the uppermost layer however, and the real selection of component pieces happens on a much lower level.

The input from the aforementioned GUI/wizard is analyzed after the submit button is hit, and the program proceeds to fill out an “Input Matrix” for each game component. These matrices can be loosely interpreted as block matrixes (a matrix composed of multiple sub-matrices), with each “block” consisting of one set of criteria, for instance, the Characters Input Matrix consists of three 2x2 blocks, one for gender age and dress with their diagonals aligned from top left to bottom right, making the Characters Input Matrix 6x6. After these matrices are all filled in, they are passed back into the main algorithm. From there, each matrix is analyzed and the eigenvector is extracted, this is a matrix with the same row numbers as the original, however only one column.

In parallel, the main algorithm collects the metadata from all of the eligible game components. The metadata consists of scores of the given component for each input criteria option. It stores this data in large matrixes, one for each component, called its Component Search Space. Each Search Space, as with the “Input Matrix,” has its own size with the number of rows equal to the number of eligible components, and number of columns equal to the number of criteria/metadata for that component. In our example, the Characters component, there are currently four different versions to choose from and six metadata criteria scores per choice, therefore the Characters Search Space is 5x6.

These two matrices, the eigenvector and the Search Space, are then multiplied together to make a new matrix with each of the eligible components to a row and only one column. The values in this one column are each of the components “scores.” The component with the highest score is the best suited to the search criteria.

To address the question of dependencies among the components, the GUI level of abstraction and analyzing of initial search criteria allows the program to take the inputs and reshape them to fit better with the available components to make coherent games. This option was chosen over limiting search space based on previous component choices, as this could potentially leave the user with no remaining search space.

## Create (Load) Game Objects

// few sentences here, refer to Table 8

Table 8 Create Game Component Objects Algorithm

|  |
| --- |
| **Input:** A map of string to string representing the association of layers to their respective filenames in the repository.  **Output:** Layers, which is an object containing all layers or components used to build the game.  function loadXmlComponents  layers : Layers;  jaxbContext : JAXBContext;  file : File;  unmarshaller : Unmarshaller;  **for** layer ϵ layers – {lesson, challenge}  jaxbContext ← JAXBContext.newInstance(layer.class)  unmarshaller ← jaxbContext.createUnmarshaller();  file ← File(xmlFiles.layer);  layers.layer ← unmarshaller.unmarshal(file);  jaxbContext ← JAXBContext.newInstance(lesson.class)  unmarshaller ← jaxbContext.createUnmarshaller();  lessons : Lesson[];  **for** lessonFile ϵ xmlFiles.lessons  lesson : Lesson;  file ← File(lessonFile);  lesson ← unmarshaller.unmarshal(file);  lessons ← lessons ∪ {lesson};  jaxbContext ← JAXBContext.newInstance(challenge.class)  unmarshaller ← jaxbContext.createUnmarshaller();  challenges : Challenge[];  **for** challengeFile ϵ xmlFiles.challenges  challenge : Challenge;  file ← File(challengeFile);  challenge ← unmarshaller.unmarshal(file);  challenges ← challenges ∪ {challenge};  learningActs : LearningAct[];  **for** (lesson ϵ lessons) && (challenge ϵ challenges)  learningAct: LearningAct;  lessonActs : LessonAct[];  lessonAct : LessonAct;  lessonAct.lessonScreens ← lesson;  lessonAct.challengeScreens ← challenge;  lessonActs ← lessonActs ∪ {lessonAct};  learningAct.lessonActs ← lessonActs;  learningActs ← learningActs ∪ {learningAct};  layers. learningActs ← learningActs; |

## Compose Game Components

// few sentences here, refer to Table 9

Table 9 Compose Game Components Algorithm

|  |
| --- |
| **Input:** The layers object containing all entities with all dependencies set.  **Output:** A Game object containing the built and assembled game.  function buildGame  game : Game;  game ← layers.getStructure().createGame();  **Input:** A Game object with a complete game and a filename where the game should be exported.  **Output:** An xml file representing the game which is written to the disk.  function exportGame  jaxbContext : JAXBContext;  jaxbContext ← JAXBContext.newInstance(Game.class);  marshaller : Marshaller;  marshaller ← jaxbContext.createMarshaller();  marshaller[Marshaller.JAXB\_FORMATTED\_OUTPUT] ← true;  file : File;  file ← new File(exportFilename);  **call** marshaller.marshal(game, file);  **Input:** All inputs are dependencies.  **Output:** A Game object representing the created game.  function createGame  acts : Act[];  screens : ScreenNode[];  screens ← theme.getIntro();  acts ← acts ∪ createActFromScreens(screens);  **for**(int i = 0; i < locale.getLearningActs().size(); i++)  screens ← locale.getAct(i);  acts ← acts ∪ createActFromScreens(screens);  screens ← theme.getOutro();  acts ← acts ∪ createActFromScreens(screens);  game : Game;  game.acts ← acts;  **call** wireUpActs(acts);  **return** game;  **Input:** The learning act id, and the screen type.  **Output:** A list of ScreenNode which represents the screens.  function buildScreens  lessonScreens : ScreenNode[];  currentScreen : UUID;  nextScreen : UUID;  currentScreen ← UUID.randomUUID();  themeStory : ThemeStory;  themeStory ← theme.getThemeStories()[learningActId];  themeStoryScreen : BaseScreen[];  **if** (screenType == ScreenType.LESSON\_STORY\_INTRO)  themeStoryScreen ← themeStory.getIntro();  **else**  screenTransitions[TransitionType.END\_OF\_STORY] ← currentScreen;  themeStoryScreen ← themeStory.getOutro();  **for** screen ϵ themeStoryScreen  nextScreen ← UUID.randomUUID();  lessonScreens ← lessonScreens ∪ buildScreen(learningActId, screen, localeScreens[screenType], currentScreen, nextScreen);  currentScreen ← nextScreen;  **if** (screenType == ScreenType.LESSON\_STORY\_INTRO)  screenTransitions[TransitionType.BEGINNING\_OF\_LESSON] ← nextScreen;  **return** lessonScreens |

## Output Game

// few sentences here, refer to Table 10

Table 10 Output Game Algorithm

|  |
| --- |
| **Input:** A Game object with a complete game and a filename where the game should be exported.  **Output:** An xml file representing the game which is written to the disk.  function exportGame  jaxbContext : JAXBContext;  jaxbContext ← JAXBContext.newInstance(Game.class);  marshaller : Marshaller;  marshaller ← jaxbContext.createMarshaller();  marshaller[Marshaller.JAXB\_FORMATTED\_OUTPUT] ← true;  file : File;  file ← new File(exportFilename);  **call** marshaller.marshal(game, file); |

# Tool Support (…still needs work…)

// move UML class diagrams here

Nb. Organize the class diagrams for the different tool modules around the 4 pipelined components in the architecture???

## Select Game Components Module

## Create Game Components Module

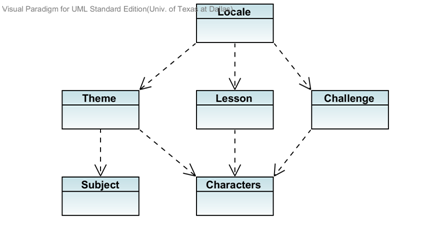
## Compose Game Components Module

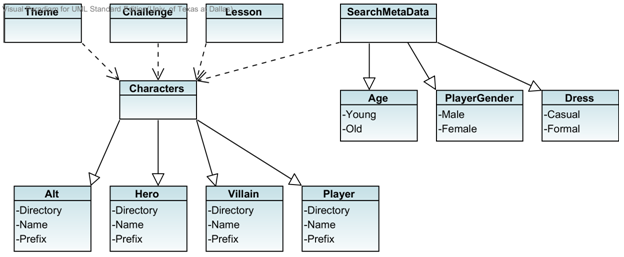
## Output Game xxx

Summarize the development effort. Module by module

The following diagrams are a different approach on describing how the components interact together.

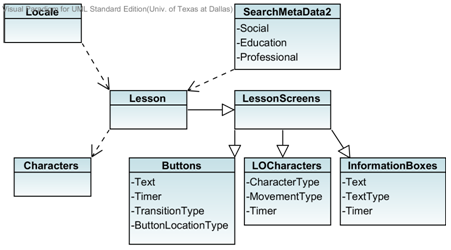
The diagram below is a high level view of how the components merge to form the desired game. This diagram will be reference throughout while we discuss each layer individually. The search meta-data is not demonstrated in this diagram but it will be visible in the individual components. The meta-data for each component will be stored in a separated XML file. In order to implement the AHP algorithm each element will be given a physical number that will be fed into the algorithm for calculation.





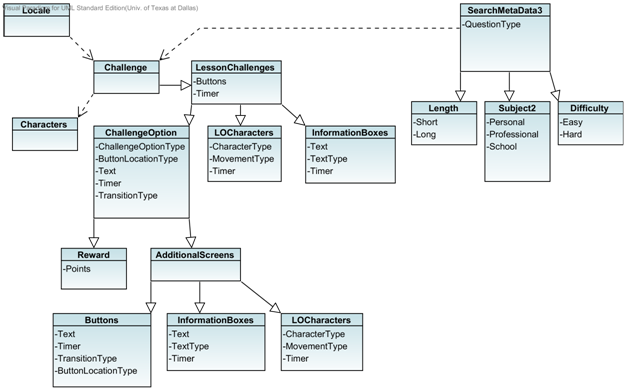
The character layer contains names and assets called from the repository that can be used for each of the four supported characters. The four characters are: Alternative (Alt), Hero, Villain, and Player. The character component is directly used by the theme, lesson, and challenge components. Their dependency can be seen in both the diagram above and the layer overview.

As seen in the diagram, the character component contains 3 different search elements and 6 total sub-elements. Age, PlayerGender and Dress allows user to find the perfect set of characters they prefer. And older crowd can focus on dress and age while the younger generation may want to pick their characters based on PlayerGender. Either way these categories provide a quick and easy way to find what you are looking for.



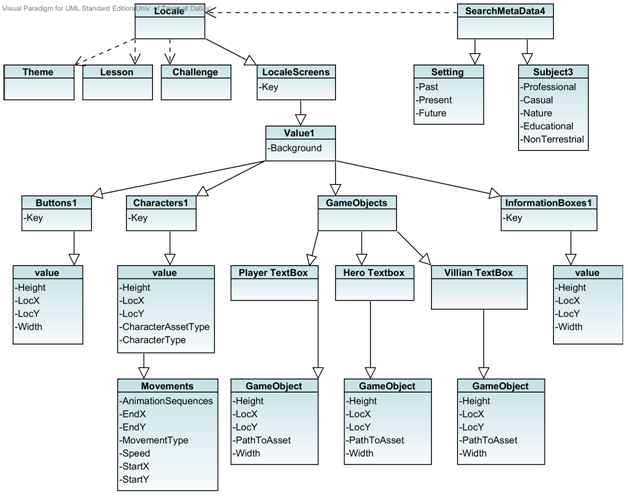
The lesson component encompasses a single lesson which directs the student to learn some objective. All objects populating the screen are provided by this component. Such objects include: buttons, textboxes, and characters. The locale will then call this component in order to piece together the lesson screens. (See locale for more detail)

The lesson meta-data is currently very general. We currently do not have enough games to be more specific with our search. For testing purposes we have the 3 broad elements you see in the diagram: Social, Education, and Professional.



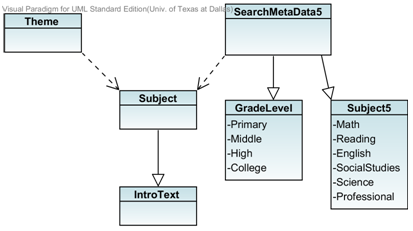
Represents the challenge presented at the end of the lesson and is currently limited to a multiple choice quiz of varying length. Like the lesson component, this component populates the challenge screens and failure screens (Additional Screens) with buttons, textboxes, and characters. The locale component then brings it all together to complete the screen. This is also where points and a timer can be handled.

The meta-data for this component contains length, subject, and difficulty. All of which have subcategories which are assigned number values as mentioned before. This component, like the lesson, is still very general because we are in early stages of the xml component repo. As more challenges are created more search elements can be added.



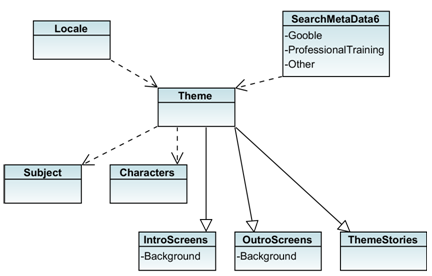
The locale component is what ties everything together. It provides the background and dialog bubbles for the lesson screens, challenge screens, failure screens, introstory screens, and outrostory screens. The locale is also responsible for the location and movement or animation of all the objects (characters, textboxes, dialog bubbles, buttons) that are created by the individual components for the same screens (lesson, challenge, failure, introstory, outrostory screens).

Locale meta-data is characterized by a specific setting and subject. The current subcategories for setting are: Past, Present, Future. The subcategories for the subject consists of: Professional, Casual, Nature, Education, Non Terrestrial.

****

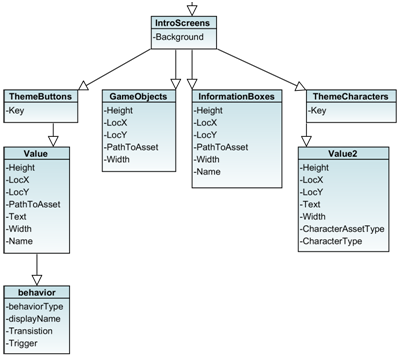
The subject component describes the overall subject that is being taught and any introductory text. The theme component is supposed to be dependent on this component however it still needs to be implemented.

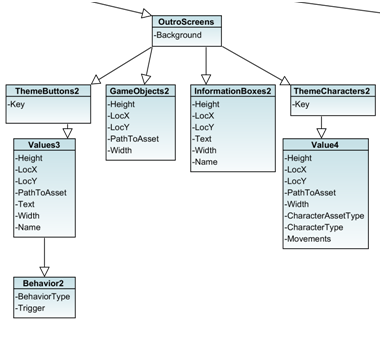
Meta-data for this component is currently very broad because our component repository is still very small. Someone looking for a game over a specific topic will need to be able to search for more than just the subject or grade level. This component’s meta-data will be updated at a future date.

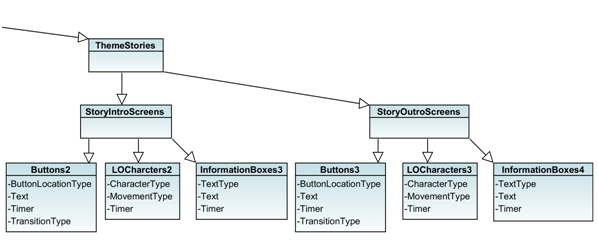
****

The theme layer contains the story aspect of the game. This component provides the game objects for the introstory screens and the outrostory screen. These two screens are then put together by the locale. The intro and outro screens are fully built from this theme component. The game objects are created and then placed on the background that is also provided by this component. Due to the size of the Theme component it is presented in three diagrams. The first of which is an overall look at the component and then the IntroScreens, OutroScreens, and ThemeStories are expanded more below.

The theme meta-data theoretically can be interchangeable with all the components. The component adds creativity and appeal to the game. The current searchable elements are Gooble, Professional Training, and Other.

****

****

****

# Validation

N test cases (at least)

One test case per test game to generate

* At this time, all the test cases fail

Each test has overview description (purpose of the test), input, expected output, actual output, status

## Test Case #1

### Description

Test Case #1 is the output of the search algorithm when “No Preference” is selected for each of the input questions.

### Input

None- No Preference for all questions

### Expected Output

The default game:

Characters0  
Lesson0  
Challenge0  
Locale0  
Subject0  
Theme0

### Actual Output

As Expected

|  |  |
| --- | --- |
| Original Space Game Story Intro Screen. |  |
| Test Case #1 Intro Screen |  |
| Original Math Game Challenge Screen |  |
| Test Case #1 Challenge Screen |  |

### Test Status

Pass

## Test Case #2

### Description

We will try to recreate the default game by searching its characteristics instead of relying on it being the default.

### Input

Player Gender: Female

Character Age: Young

Character Dress: Casual

Game Theme: Gooble

Game Subject: Math

Game Setting: Natural

Challenge Difficulty: Easy

### Expected Output

The Default game:

Characters0  
Lesson0  
Challenge0  
Locale0  
Subject0  
Theme0

### Actual output

Exactly as expected.

|  |  |
| --- | --- |
| Original Space Game Story Intro Screen. |  |
| Test Case #1 Story Intro Screen |  |
| Original Math Game Lesson Screen |  |
| Test Case #2 Lesson Screen |  |

### Test Status

Pass

## Test Case #3

### Description

Search for a new auto-generated, yet coherent game. With a professional setting and lesson, try to recreate the “Office” pre-generated game as closely as possible.

### Input

Player Gender: Male

Character Age: Young

Character Dress: Casual

Game Theme: Workplace

Game Subject: Professional

Game Setting: Professional

Challenge Difficulty: No Preference

### Expected Output

Characters2  
Lesson1  
Challenge1  
Locale1  
Subject1  
Theme1

### Actual Output

Exactly as expected

|  |  |
| --- | --- |
| Original Workplace Game Intro Screen. Demonstrates that the game theme is the workplace. |  |
| Test Case #3 Intro Screen. The theme chosen was in fact the workplace. |  |
| Original Workplace Lesson Screen. Demonstrates professional clothing. |  |
| Test Case #1 Lesson Screen. Demonstrates that the characters selected were wearing casual clothing. |  |

### Test Status

Pass

## Test Case #4

### Description

In this test, unlike the previous two, no specific game was trying to be made. A random game was attempted with opposing setting subject and theme.

### Input

Player Gender: No Preference

Character Age: Young

Character Dress: Formal

Game Theme: Dream

Game Subject: Science

Game Setting: Natural

Challenge Difficulty: Hard

### Expected Output

Characters1  
Lesson3  
Challenge3  
Locale0  
Subject2  
Theme3

### Actual Output

Characters1  
Lesson0  
Challenge2  
Locale0  
Subject2  
Theme3

|  |  |
| --- | --- |
| Test Case #4  Intro Story Screen. Background does not match the dialog. Fail. |  |
| Game Lesson Screen. |  |
| Test Case #2 Challenge Screen does not match the lesson. Fail. |  |

### Test Status

33% fail

This test failed, but only slightly. Regardless of the expected output, the challenge and lesson need to match to make a coherent game. Further specification of search and a larger library to choose from may help. In addition, more metadata would also be useful.

// try for 6 test games for the end of the term

// please move this material into sections above

Fig. 14 Test Cast #1

Fig. 15 Test Case #2

# Conclusions and Future work

Serious educational games (SEGs) are recognized as valuable educational tools, with significant potential to provide immersive, engaging and fun environments for students across diverse domains. Developing these games, however, remains challenging. A component-based semi-automated game generation approach has the potential to support educators in rapidly creating games. Here, we have introduce our proposed SEGCB approach, which has an interactive wizard user interface, a game generation engine, and a repository of components, game scripts, and game assets. The game components proposed in this preliminary work are the theme, characters, subject, locale, lesson, and challenge. The components have been captured in XML; loaded, composed, and output as a game script [2].

The ability of mix and matching game components are now being tested and have shown great promise. As shown with a few examples in the two test cases the ability to combine different components is successful but not without limitations. One limitation with the components is that each game component should only be paired with components of a similar subject or theme. Our test cases were used without those dependencies which created a game that is not responsible. For example, a space lesson should not paired with the challenge about unrelated vocabulary. As our repository of games continue to grow so will our ability to create a variety of different games pertaining to one subject or theme.

The addition of the search class/component and XML repositories (metadata and component) have been a major focus in this term and it is progressing. Outside of the actual search feature, tools have also have been made to simplify the addition of new components along with their metadata to their respective repositories. This allow the each repository to be updated by running one tool. The search class currently takes input from hard coded text files but it is a future goal of this term is to implement a user interface for the game engine.

# References

1. Atthirawong, Walailak; Bart McCarthy (September, 2002). "An Application of the Analytical Hierarchy Process to International Location Decision-Making". In Gregory, Mike. Proceedings of The 7th Annual Cambridge International Manufacturing Symposium: Restructuring Global Manufacturing. Cambridge, England: University of Cambridge. pp. 1–18.
2. Berrittella, M.; A. Certa, M. Enea, P. Zito (January, 2007). "An Analytic Hierarchy Process for the Evaluation of Transport Policies to Reduce Climate Change Impacts". *Fondazione Eni Enrico Mattei (Milano)*.
3. Daconceicao, Ricardo, Carsten Locke, Kendra Cooper, and Shaun Longstreet. “Semi-automated Serious Educational Game Generation: a Component-based Game Engineering Approach.” IEEE CGAMES, 2013 (to appear).
4. Gee, J. P. (2007). What Video Games Have to Teach Us About Learning and Literacy. 2nd Edition, Palgrave Macmillan.
5. Grandzol, John R. (August 2005). "Improving the Faculty Selection Process in Higher Education: A Case for the Analytic Hierarchy Process" (PDF). IR Applications 6.
6. Heineman, G. T., Councill, W.T., Component-Based Software Engineering – Putting the Pieces Together, Addison Wesley, 2001.
7. Klopfer, E., Osterweil, S., & Salen, K. (2009). Moving Learning Games Forward. Massachusetts Institute of Technology, USA: The Education Arcade.
8. S. Longstreet and K. Cooper, “A Meta-model for Developing Simulation Games in Higher Education and Professional Development Training”, IEEE 17th International Conference on Computer Games, 2012.
9. Lopes, R. and Bidarra, R. (2011). Adaptivity challenges in games and simulations: a survey, *IEEE Trans.on Computational Intelligence and AI in Games*, Vol. 3, Issue 2, pp. 85-99.
10. Sanford, R. (2006). Teaching with Games: COTS games in the classroom. Innovating e-learning: Transforming Learning Experiences online conference.
11. Saaty, Thomas L. (2008-06). "Relative Measurement and its Generalization in Decision Making: Why Pairwise Comparisons are Central in Mathematics for the Measurement of Intangible Factors – The Analytic Hierarchy/Network Process". *RACSAM (Review of* the Royal *Spanish Academy of Sciences, Series A, Mathematics)* **102** (2): 251–318.
12. Scacchi, W. (2011). Modding as a basis for developing game systems. In *Proceeding of the 1st International Workshop on Games and Software Engineering*, pp. 5-8.
13. Shaffer, D. W. (2007). How Computer Games Help Children Learn. Palgrave Macmillan.
14. Szyperski, C., *Component Software*, Addison-Wesley, 2 edition, 2002.
15. Walker, A. and Shelton, B.E. (2008). Problem-based educational games: connections, prescriptions, and assessment. .*J. of Inter. Learning Res*.,vol.19, no.4, pp. 663-684.