Proceedings of the 2015 Winter Simulation Conference
L. Yilmaz, W. K. V. Chan, I. Moon, T. M. K. Roeder, C. Macal, and M. D. Rossetti, eds.

A CLOUD SOFTWARE SYSTEM FOR VISUALIZATION OF GAME-BASED LEARNING DATA COLLECTED ON MOBILE DEVICES

J. Robert Jones
Osman Balci
Anderson Norton

Departments of Computer Science and Mathematics
Virginia Polytechnic Institute and State University (Virginia Tech)
Blacksburg, Virginia 24061, USA

ABSTRACT

Digital game-based learning is a type of gameplay with a set of defined learning outcomes. Such gameplays are typically instrumented to collect data for assessing the learning outcomes. However, when the gameplay and data collection take place on a mobile device such as iPad, it becomes very difficult for a teacher to view the collected data on dozens of mobile devices used by students. This paper presents a cloud software system (CSS) under the client-server architecture to remedy this problem. We developed CSS using the Java platform, Enterprise Edition (Java EE) with IBM WebSphere Application Server, IBM DB2, and MongoDB. We also developed an educational iPad game called Taffy Town. Game-based learning data are collected on the iPad during the Taffy Town gameplay and are transmitted to our CSS over the Internet. Players (students) and teachers can login and view dynamically created visualizations of the collected learning data.

1 INTRODUCTION

In the past few years, we have seen a huge technical migration from stationary computers to ubiquitous mobile devices. More students have access to smartphones and tablets. In 2012, 39% of students in grades 6 through 8 use smartphones for homework and 31% use tablets for homework (Khadaroo 2012). The mobile device ubiquity enables educators to craft new tools that utilize this technology to deliver a better educational experience. One of those tools is the employment of digital educational games in various subjects. The number of published articles in seven technology-based learning journals from 2006 to 2010 has increased four times greater than the previous five years (Hwang and Wu 2012).

Apple (2015a) states that over 80,000 educational apps exist on the Apple App Store. It has been acknowledged that utilizing educational apps can help students learn complex subjects like fractions in a more engaging and fun manner. Some apps offer a variety of assessments and show how well the students are performing various tasks during gameplay.

Another emerging technological interest is big data, a massive collection of data, which traditional data applications cannot effectively handle. The goal of this field of research is to discover methods to analyze these large sets of information to produce some type of business intelligence. The first IEEE conference on Big Data was held in 2014 with 259 paper submissions (IEEE 2014). IEEE advocates that big data is transforming science, engineering, medicine, healthcare, finance, business, and ultimately society itself.

Unfortunately, the union of big data solutions and education has had a rocky road. In 2012, a group of educators created inBoom, an open-source computer system that stored student data in a secure, common

format that enabled schools to control what was collected and who had access to it. Less than two years later, inBloom has been abandoned due to student privacy concerns (C 2014).

With this evident transition into large bodies of students playing and interacting with digital apps, how can we analyze large datasets of gameplay data with big data analytics?

This paper investigates the ability to capture what players are doing during gameplay. This gameplay data can then be used to produce visualizations using big data techniques. The result is not only assessment information, but also building a better understanding of what students do to get those results.

This paper is organized as follows. After an introduction in Section 1, Section 2 describes the cloud software system, which we developed. The Taffy Town educational iPad game we developed to collect and transmit game-based learning data is presented in Section 3. Section 4 describes the dynamic visualization of game-based learning data. Concluding remarks are given in Section 5.

2 CLOUD SOFTWARE SYSTEM

Cloud software is the kind of software that runs on a server computer under the control of an *Application Server* software and used by a user over a network (cloud) typically by using a web browser (Schutt and Balci 2015). This section describes the Cloud Software System (CSS), which we developed for dynamic visualization of game-based learning data collected on a mobile device such as iPad (Jones 2014).

We engineered CSS by using the Java platform, Enterprise Edition (Java EE) (Oracle 2015) under the client-server architecture as shown in Figure 1 with five tiers: (1) client tier, (2) web tier, (3) business tier, (4) data mapping tier, and (5) data source tier. We used the following software products in developing CSS: IBM Rational Application Developer Integrated Development Environment (IDE) (IBM 2015a), IBM WebSphere Application Server (IBM 2015b), IBM DB2 Relational Database Management System (IBM 2015c), and MongoDB (MongoDB 2015).

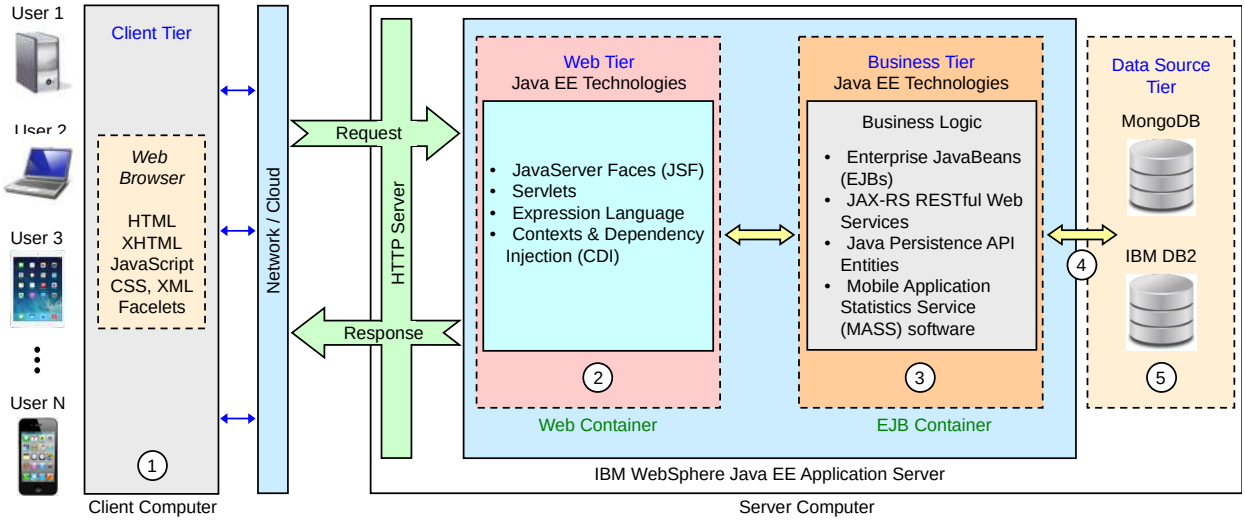

Figure 1: Java EE-based cloud software system's client-server architecture.

2.1 Tier 1: Client Tier

Students, teachers, and the CSS administrator make up the client tier. Students play our Taffy Town educational game on an Internet-connected iPad. During the gameplay, data are collected and sent to the CSS server computer over the Internet. Teachers and the CSS administrator connect to the server computer by using a web browser on an Internet-connected desktop, laptop or mobile device and see the graphical visualizations of the game-based learning data collected during the gameplay for all students.

2.2 Tier 2: Web Tier

The web tier consists of a web container, which performs two major tasks: (1) controlling and mediating, and (2) server-side preparation of the presentation before it is sent to the client computer. The Java EE technologies employed at this tier include the following: JavaServer Faces (JSF), Servlets, Expression Language, and Contexts and Dependency Injection (CDI).

2.3 Tier 3: Business Tier

The business tier consists of two components: (1) a cloud software application, and (2) three RESTful web services (Fielding 2000).

The *cloud software application* enables registered players to log in to view various dynamic visualizations of their game-based learning data collected and transmitted by the Taffy Town iPad educational game. A player creates an account with a username and password before playing Taffy Town and that account is used to log in. Additionally, a built-in CSS administrator account is provided to log in and obtain visualizations of aggregated game-based learning data across all registered Taffy Town players. These visualizations are shown without identifying who the player is to protect player's privacy.

The *three RESTful web services* consist of authentication web service, user web service, and statistics web service. The *authentication web service* provides a mechanism for Taffy Town to authenticate players when logging in to play the game. The *user web service* enables players to create accounts, modify account information, and retrieve specific user data. The *statistics web service* is responsible for processing and creating graphical visualizations of the game-based learning data collected during Taffy Town gameplay.

2.4 Tier 4: Data Mapping Tier

The Java Persistence API entities are used at this tier to provide CRUD (create, retrieve, update, delete) operations on the data that resides at the Data Source tier.

2.5 Tier 5: Data Source Tier

Two databases are used at this tier. IBM DB2 Relational Database Management System (IBM 2015c) is used for storage and retrieval of user account data. MongoDB (MongoDB 2015) is used for storage and retrieval of game-based learning data collected and transmitted by the Taffy Town iPad educational game.

3 TAFFY TOWN EDUCATIONAL GAME

The Taffy Town digital educational game is developed to (a) engage students in game-based learning of fractions, (b) collect data about the player's game-based learning and interactions during gameplay, (c) communicate with CSS, and (d) send the collected data to CSS over the Internet. It is created to run on the iPad and iPad Mini mobile devices by using Apple's Xcode IDE, iOS Software Development Kit, and 2D game engine and framework called Sprite Kit (Apple 2015b).

In Taffy Town gameplay, the player is tasked to save Taffy Town from taffy-eating aardvarks known as the "Varks". The player, as part of the Taffy Task Force (TTF), patches holes in various town buildings with vark-resistant taffy. The player observes Taffy Town and when a building is attacked, the player stops the attack by tapping the building image. This presents the player with a piece of taffy, hole graphic, and a fraction question. The player needs to answer the presented question by manipulating the taffy to patch the hole. The game allows manipulation of taffy by moving, stretching, cutting, gluing, deleting extra pieces, and painting taffy pieces. After the player transforms the presented taffy, s/he attempts to patch the hole. If the player responds to the question correctly, the hole is patched. Otherwise, the hole shakes signaling that the patch is unsuccessful. Regardless of correctness, the game transitions back to viewing Taffy Town in anticipation for another attack by the Varks.

As the player answers questions, information about each taffy manipulation is sent to CSS. The data collected includes various facets of information about the question and all transformations performed on the taffy. In order for this data to correctly map to the current player, users are required to create and login to their account before playing. Authentication and account creation features are offered to the user with pop-up windows to ensure that their experience is contained within the game.

After the player launches the Taffy Town game app on an iPad, s/he creates an account or signs in to an existing account. Then, in random intervals, a building is designated as attacked by displaying a dust cloud image on top of the building image as shown in Figure 2. In order to stop an attack, the player must tap the image of the attacked building. If the player delays tapping an attacked building, more buildings can be attacked. The player is not penalized for neglecting to tap an attacked building; however, the game does depend on this user interaction to transition to the workspace scene.

Figure 2: Attacked taffy town buildings.

The workspace scene, shown in Figure 3, is where the core interaction takes place. In this scene, the player manipulates the piece of taffy in order to answer a question by "patching" the hole. The scene is divided into two areas, an image of the damaged building on the left and a grid with a piece of taffy on the right. The core elements of the workspace scene, depicted in Figure 3, are described below:

1. *Fraction Question:* A randomly generated fraction question, which must be answered by the player by manipulating the piece of taffy. Figure 3 displays the fraction question "This piece of taffy is ¼ of the hole the Varks made. Make the whole!"
2. *Hole:* The black image with irregular shape in Figure 3 represents the hole the aardvarks ate and corresponds to the fraction question displayed. For the question displayed in Figure 3, the player is required to transform the taffy image from ¼ to a whole to be able to patch the hole.
3. *Taffy:* The square image on the right in Figure 3 represents the piece of taffy that can be manipulated by gestures on the iPad.
4. *Paint Brush:* The paint brush tool shown in Figure 3 is used to color the taffy pieces.

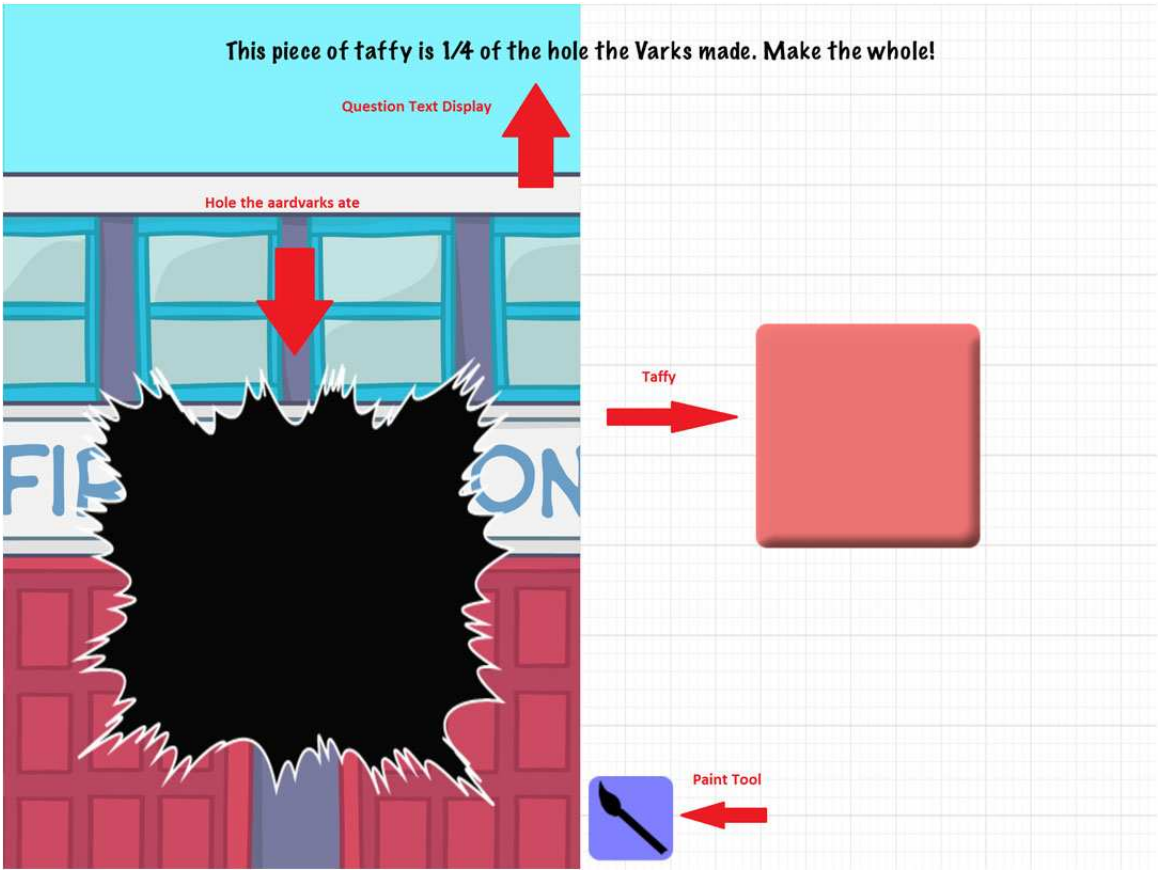

Figure 3: Taffy town workspace scene.

Each time the workspace scene is displayed, a random fraction question is generated for the player to respond to. Two types of questions are asked: (1) "The Varks ate a hole that is n/d the size of this piece of taffy. Change the taffy to patch the hole!", and (2) "This piece of taffy is n/d of the hole the Varks made. Make the taffy as a whole!". The question types differ with regard to the target object in which the fraction applies. In type 1, the fraction is applied to the hole. Conversely, in type 2 the fraction is applied to the taffy piece. Each question type can be either a proper or improper fraction.

Taffy pieces can be moved around on the screen by placing a single finger on the targeted taffy and dragging it around. The taffy follows the finger all around the screen. When the player stops touching the taffy, it stops in its current location. The player is allowed to move taffy pieces to any place on the screen, including over the hole.

A player can slice or partition a piece of taffy by swiping, using a single finger, through the taffy piece. The player must start on the grid, swipe through the taffy, and end on the grid in a single motion. The taffy piece is divided on a successful swipe either horizontally or vertically. Diagonal slicing is not supported. Figure 4 shows several pieces of taffy after horizontal and vertical slices.

A player can stretch or shrink a piece of taffy horizontally or vertically by placing two fingers at opposite ends of it and correspondingly perform "pinch open" or "pinch close" gestures. A player can copy a taffy piece by double tapping it with one finger. The copied taffy piece can be pasted by double tapping the desired grid location with one finger. A taffy piece can be deleted by double tapping with two fingers. Double tapping the paint brush icon with one finger displays the color palette as shown in Figure 5. Tapping a color in the palette sets the paint brush color. Then, dragging the paint brush and dropping it on top of a taffy piece colors that piece with the color of the brush.

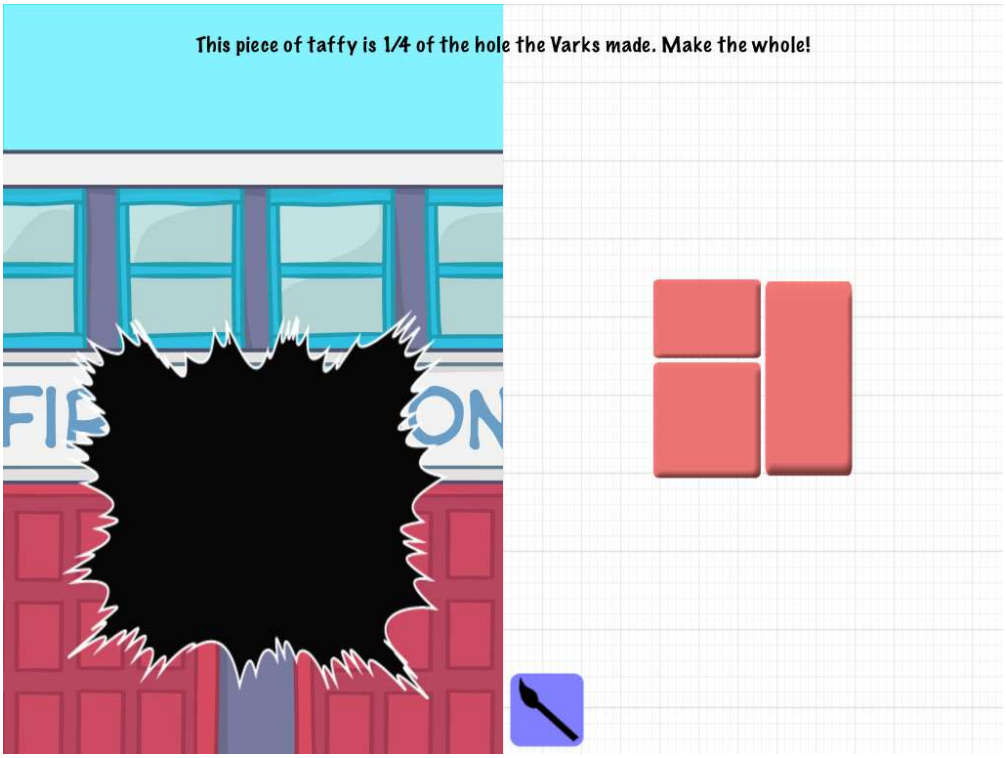

Figure 4: Horizontal and vertical slicing of taffy pieces.

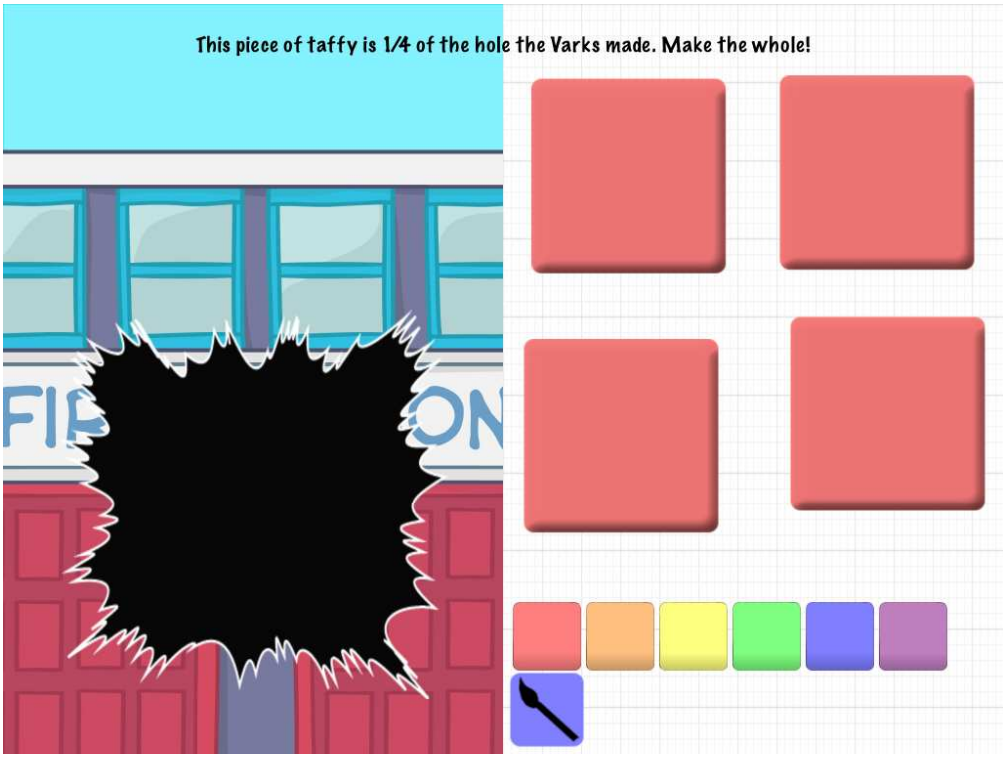

Figure 5: Stretching, shrinking, and coloring taffy pieces.

Multiple taffy pieces can be combined or "glued" together by dragging a piece of taffy TD into another piece of taffy TS. As the pieces touch, they are combined into a single larger piece of taffy TG. Since taffy of various sizes can be glued together, the game performs the following algorithm to glue the taffy pieces. If the heights of TD and TS are equal to each other, then the width and height of the combined taffy are computed as illustrated in Figure 6.

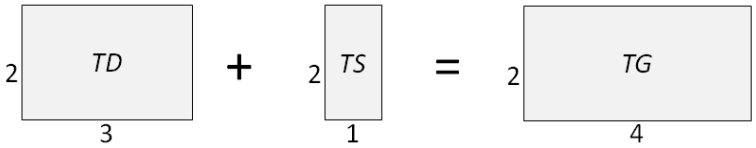

Figure 6: width(TG) = width(TD) + width(TS) and height(TG) = height(TD) = height(TS).

If the widths of TD and TS are equal to each other, then the width and height of the combined taffy are computed as illustrated in Figure 7.

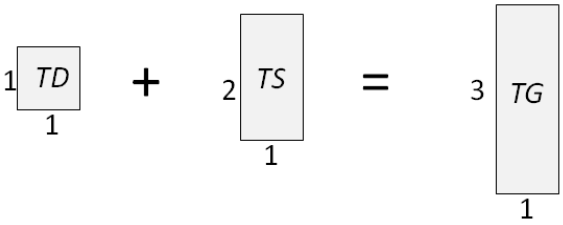

Figure 7: width(TG) = width(TD) = width(TS) and height(TG) = height(TD) + height(TS).

If the widths and heights of TD and TS are different from each other, then the width and height of the combined taffy are computed as illustrated in Figure 8.

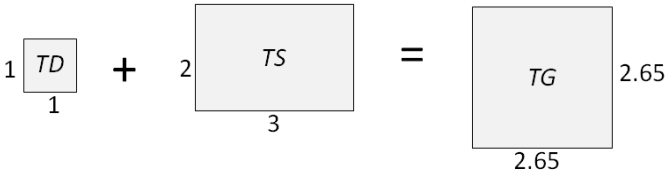

Figure 8: width(TG) = height(TG) = $\sqrt{}$(area(TD) + area(TS)).

In combining two pieces of taffy with different colors, the color of the dragged taffy is chosen as the color of the final taffy piece.

4 VISUALIZATION OF GAME-BASED LEARNING DATA

The user's dashboard features three distinct data visualizations: (1) actions stacked bar chart, (2) accuracy pie chart, and (3) question type distribution stacked bar chart (Murray 2013). We utilized D3.js and crossfilter.js, two JavaScript libraries in order to display these graphs in real-time.

D3.js is a library that allows HyperText Markup Language (HTML) documents to be manipulated based on a given dataset. It uses HTML, Scalable Vector Graphics (SVG), and Cascading Style Sheets, allowing a data-driven approach in creating and manipulating various Document Object Model (DOM) elements (Bostock 2014).

Crossfilter.js is a multi-dimensional JavaScript library for exploring large multivariate datasets in the browser. It is created by Square to allow merchants to dissect payment history datasets (Square 2012). It allows websites to manipulate, reduce, and filter extremely large datasets.

A four-step process is employed in conjunction with D3.js and crossfilter.js in order to generate each SVG graph. The *statistics web service* is polled every thirty seconds in order to load new information. If the dataset is stale, the following steps are triggered causing the charts to animate with the new information.

1. *Dataset Retrieval:* The statistics web service is called to pull down a set of data specific to the chart being generated. This is called "raw player game data".
2. *Crossfilter Transformation:* The pull downed dataset is then grouped, aggregated, and filtered on various indices specific to the chart being generated. This set of data is referred to as "filtered player game data".
3. *Graph Transformation:* Each graph expects the data to be in a specific format. This step in the process allows the filtered player game data to be transformed into a format D3.js understands. Data after this transformation is referred to as "transformed player game data".
4. *Graph Rendering:* Transformed player game data is loaded into custom D3.js scripts that generate the graph in the SVG format and displays it on the dashboard.

The three distinct data visualizations are described below.

4.1 Actions Stacked Bar Chart

Figure 9 illustrates the first type of data visualization displayed on the far left of the dashboard. This graph correlates actions performed by a player and accuracy of the player's answer. The graph's components are described below (SAS 2015).

1. X-Axis: Represents the actions (e.g., glue, slice, copy) performed by the player.
2. Y-Axis: The total number of times the action was performed during the entire Taffy Town gameplay.
3. Groups (Bars): Each group or bar is composed of correct and incorrect subgroups stacked.
4. Legend: Distinguishes between the correct and incorrect subgroups by using different colors.

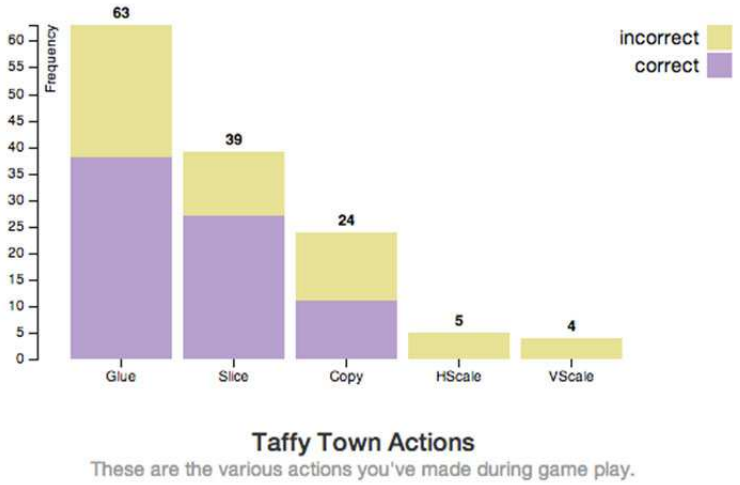

Figure 9: Stacked bar chart depicting various actions performed by a Taffy Town player.

4.2 Accuracy Pie Chart

Figure 10 illustrates the second type of data visualization displayed in the middle of the dashboard. It shows the percentage of correct answers in the form of a pie chart. The pie chart consists of two slices. The green slice represents the percentage of correct answers. The other slice shows the percentage of incorrect answers. The exact percentage of correctness number is shown in the middle of the pie chart.

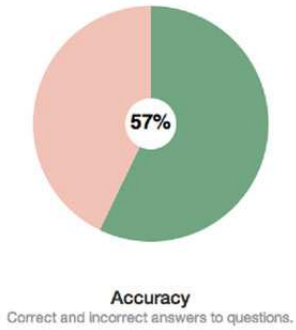

Figure 10: Pie chart showing the distribution of correct and incorrect answers.

4.3 Question Type Stacked Bar Chart

Two types of questions are asked during the Taffy Town gameplay as described in section 3. In order to understand which type of question leads to more correct answers, Figure 11 shows the relationship between question type and accuracy. The question type chart is composed of several components described below.

1. X-Axis: Represents the type of question answered by the player.
2. Y-Axis: The total number of times the question was answered by a player.
3. Groups (Bars): Each group or bar is composed of correct and incorrect subgroups stacked on top of each other.
4. Legend: Distinguishes between the correct and incorrect subgroups by using different colors.

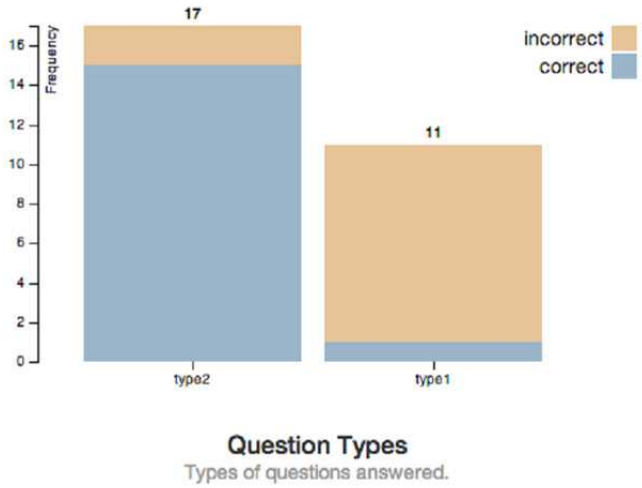

Figure 11: Stacked bar chart correlating question type and accuracy.

5 CONCLUDING REMARKS

A research report on gaming in education claims that "digital games (1) are built on sound learning principles, (2) provide more engagement for the learner, (3) provide personalized learning opportunities, (4) teach 21st century skills, and (5) provide an environment for authentic and relevant assessment" (McClarty et al. 2012). Assessment of game-based learning poses significant technical challenges for researchers, teachers, and administrators. A student plays an educational game on a separate computer (e.g., mobile device – iPad, iPod Touch, iPhone; laptop or desktop) in a classroom or at home. Integrating all of the learning assessment data collected during gameplays on different computers and automatically generating a learning assessment report for the teacher to use remains to be a serious technical challenge. This paper describes a client-server solution as a contribution to addressing this technical challenge.

REFERENCES

Apple. 2015a. "iPad in Education." https://www.apple.com/education/ipad/apps-books-and-more/, Accessed August 2, 2015.

Apple. 2015b. "iOS Dev Center." https://developer.apple.com/devcenter/ios/, Accessed August 2, 2015.

Bostock, M. 2015. "D3.js Overview: D3 Data-Driven Documents." http://d3js.org/, Accessed August 2, 2015.

C, K. N. 2014. "Big Data and Education: Withered inBloom." *The* Economist, April 30, http://www.economist.com/blogs/schumpeter/2014/04/big-data-and-education/, Accessed August 2, 2015.

Fielding, R. T. 2000. "Architectural Styles and the Design of Network-based Software Architectures." Doctoral dissertation, University of California, Irvine, CA.

Hwang, G.-J., and P.-H. Wu. 2012. "Advancements and Trends in Digital Game-Based Learning Research: A Review of Publications in Selected Journals from 2001 to 2010." *British Journal of Educational Technology* 43(1): E6-E10.

IBM. 2015a. "IBM Rational Application Developer for WebSphere Software." http://www-03.ibm.com/software/products/en/application, Accessed August 2, 2015.

IBM. 2015b. "IBM WebSphere Application Server." http://www-03.ibm.com/software/products/en/appserv-was, Accessed August 2, 2015.

IBM. 2015c. "IBM DB2 Relational Database Management System." http://www-01.ibm.com/software/data/db2/, Accessed August 2, 2015.

IEEE. 2014. "2014 IEEE International Conference on Big Data." Washington, DC, Oct. 27-30, http://cci.drexel.edu/bigdata/bigdata2014/, Accessed August 2, 2015.

Jones, J. R. 2014. "A Client-Server Architecture for Collection of Game-based Learning Data." M.S. thesis, Department of Computer Science, Virginia Tech, Blacksburg, Virginia, Dec.

Khadaroo, S. T. 2012. "Not Just 4 Texting: 1 in 3 Middle-Schoolers Uses Smart Phones for Homework." *The Christian Science Monitor*, Nov. 29, http://goo.gl/NqEYAH, Accessed August 2, 2015.

McClarty, K. L., A. Orr, P. M. Frey, R. P. Dolan, V. Vassileva, and A. McVay. 2012. "A Literature Review of Gaming in Education: Research Report." Pearson Education, June

MongoDB. 2015. "MongoDB: Designed to Store Data the Way Your Apps Do." https://www.mongodb.org/, Accessed August 2, 2015.

Murray, S. 2013. *Interactive Data Visualization* for *the Web*. O'Reilly Media, Inc., Sebastopol, CA

Oracle. 2015. "Java EE at a Glance." Oracle Corporation, Redwood Shores, CA, http://www.oracle.com/technetwork/java/javaee/overview/, Accessed August 2, 2015.

SAS Institute Inc. 2015. "Chart Terminology." http://goo.gl/EYWdo3, Accesed August 2, 2015.

Schutt, K., and O. Balci. 2015. "Cloud Software Development Platforms: A Comparative Overview." Technical Report, Department of Computer Science, Virginia Tech, Blacksburg, VA.

Square. 2012. "Crossfilter." http://square.github.io/crossfilter/, Accessed August 2, 2015.

AUTHOR BIOGRAPHIES

J. ROBERT JONES is a Senior Software Development Engineer / Solutions Architect for the Enterprise Integration Platform at Metropolitan Life Insurance Company (MetLife) in Raleigh, NC. He earned his M.S. degree in Computer Science from Virginia Tech in 2014. He developed an architecture for collecting game-based learning data as part of his thesis research. Bob's areas of expertise include software engineering and architecture, mobile application development, service-oriented architecture, and cloud computing. His e-mail address is jjones31@vt.edu.

OSMAN BALCI is Professor of Computer Science and Director of the Mobile / Cloud Software Engineering Lab at Virginia Tech. He received his Ph.D. degree from Syracuse University in 1981. Dr. Balci served as the founding Editor-in-Chief of two international journals: *Annals of Software Engineering*, 1993-2002 and *World Wide Web*, 1996-2000. He served as Chair of ACM SIGSIM, 2008-2010; and Director at Large of the Society for M&S International, 2002-2006. Most of Dr. Balci's work has been funded by the U.S. Navy. His current areas of expertise center on mobile / cloud software engineering, architecting system of systems, modeling and simulation, and digital game-based learning. His e-mail and web addresses are balci@vt.edu and http://manta.cs.vt.edu/balci.

ANDERSON NORTON is Associate Professor of Mathematics at Virginia Tech. He received his Ph.D. in Mathematics Education from the University of Georgia in 2004. Dr. Norton was the 2013 recipient of the Early Career Research Award from the Association of Mathematics Teacher Educators. Dr. Norton's research focuses on building models of students' mathematical knowledge and learning. His e-mail and web addresses are norton3@vt.edu and http://www.mathed.soe.vt.edu/Faculty/norton.html.