THE DESIGN OF MULTIPLAYER SIMULATIONS FOR ONLINE GAME-BASED LEARNING

Matti Kariluoma, Brian M. Slator, Bradley Vender, Otto Borchert, Guy Hokanson, Peng Yan
North Dakota State University
Fargo, North Dakota, USA
{matti.m.kariluoma, brian.slator, bradley.vender, otto.borchert, guy.hokanson, peng.yan}@ndsu.edu
Bob Cosmano
WoWiWe Instruction Co LLC
Fargo, North Dakota, USA
cosmano@wowiwe.net

ABSTRACT
A software architecture that facilitates interactive multiplayer simulations, intelligent software tutors, and immersive virtual environments is presented. The implementation employs a hybrid selection of techniques mainly related to a "restful" server strategy. The paper concludes with an implementation of the software architecture: Virtual Cell, a three-dimensional environment for teaching the processes and structures of Cellular Biology.

KEY WORDS
Architectures for Educational Technology Systems, Game-based and Simulation-based Learning, Immersive Virtual Environments, Interactive Learning Environments

1 Introduction

Genuine collaboration among peers is an underutilized pedagogical tool in the domain of serious (educational) games. There are few multiplayer serious games due to the complexity of designing, creating, and supporting them [1].

This paper describes an architecture designed to facilitate the creation of collaborative multiplayer simulations for use in serious games.

1.1 Background

Immersive Virtual Environments (IVEs) provide students with an opportunity to assume the role of a scientist, using the tools, processes, and terminology of a discipline within a simulation. For example, a geology game allows students to "model" geologists, using rock picks, scratch plates, and describing strike and dip measurements. This role playing allows students to build their knowledge using constructivist principles [2, 3].

Within the simulation, targeted and pedagogically sound goals and tasks keep students in the Zone of Proximal Development (ZPD) [4] where material is not so hard that it is difficult to understand and not so easy that students need barely pay attention.

Multiplayer collaborative simulations provide many other benefits to players as well. Anderson [5] cites a number of studies noting that massively multiplayer online games (a type of multiplayer collaborative simulation) increase cognitive skill, appeal to multiple learning styles, and provide multiple modes of interaction with learning materials [6, 7, 8, 9, 10]. Students also enjoy MMORPG based lessons [11, 12].

The World Wide Web Instructional Committee (WWWIC) at North Dakota State University (NDSU) is an ad hoc committee of faculty, staff, and students working to advance education through the use of IVEs [13]. WWWIC has implemented educational IVEs for a wide range of scientific disciplines [14, 15, 16, 17]. These IVEs are designed using a common set of guidelines [18]:

- Collaborative and multi-user
- Constructed with the help of content experts
- Strong cognitive and pedagogical features to assist the players in learning the content
- Consistent evaluation of learning outcomes

JavaMOO [19], inspired by LambdaMOO [20], is a framework developed by WoWiWe Instruction Co. in collaboration with NDSU for building virtual worlds used for educational purposes. The JavaMOO framework consists of:

- A server capable of running a large simulation and managing the interactions of many players
- An application programmers' interface (API) for building an application-specific client that:
  - Presents the state of the environment
  - Allows players to interact with the simulation
  - Allows players to interact with each other
- A method of communication between client and server that is extendable for application needs

The remainder of this paper describes this framework: the requirements that drove the development, the overall architectural framework of the platform, and a specific implementation – the Virtual Cell game.

2 Requirements for Educational Multiplayer Simulations

Reuter [1] lists a number of requirements for a successful collaborative multiplayer simulation:

- Give realistic and logical reasons for collaboration
- Require equal contribution by all players
- Minimize waiting times
- Promote communication
- Include actions to coordinate

During the development of WWWIC simulations, other practical requirements emerged including:

- Allow multiple players to view the simulation
- Allow multiple players to view each other's actions
- Reset simulations for new groups of players
- Facilitate player-to-player communication
- Assign tasks to foster cooperation/competition
- Guide players through their tasks
- Remediate players when they lose their way
- Assess player's learning progress

3 An Architecture for Educational Multiplayer Simulations

Previous attempts at collaborative multiplayer simulations had a number of drawbacks. The game state could only be delivered as text, and many messages were needed to describe scenes in ongoing simulations. We also found that only ≈ 20 players could be in a room before the simulation became unusable.

These limitations led to the use of external webservers for content delivery and external databases to store learning assessments. This became too hard to maintain and deploy in larger projects [19, 21].

The performance issues involved with supporting multiple scenes with many simultaneous players necessitated the development of additional constraints.

We place these additional restrictions on the Java-MOO framework [Figure 1]:

- **Non-Authoritative State Synchronization.** The server should not try to calculate the full state of the simulation, especially the size and shape of players and their collisions.
- **Common Interface.** The client and server should communicate using a message format that can be parsed by many programming languages.

3.1 Non-Authoritative State Synchronization

Many of these requirements imply state synchronization between clients. How do we keep many players in a shared view of the simulation, especially when the simulation might update many times a second?

Initial attempts to simulate state were implemented exclusively on the server, with each client merely viewing the simulation, much like the current Second Life architecture [22]. This places untenable load on a server when high definition/high resolution 3D graphical objects are needed for the sake of authentic visualization and learning.

A non-authoritative approach [23] to the state synchronization problem was taken to compensate for these drawbacks. This approach places handling of player input, calculation of physics, and event interactions locally on each client. The final results are then sent to the server for dissemination to other clients.

Representational State Transfer (REST) [24] Application Programming Interfaces are used to transmit state between the clients and the server. This provides scalability for the virtual environment, allowing many more players in the same room than our previous attempts.

3.2 Common Interface

To support client and server interaction, a common communication interface must be developed. JSON [25] (JavaScript Object Notation, a "lightweight data-interchange format") was selected because it provides a programming-language-agnostic messaging format that has library support in many major languages.

This affords writing clients in a programming language different than that of the server, typically a programming language more suited for graphical clients as opposed to the programming languages typically used to write servers.

4 Virtual Cell

Virtual Cell is an immersive three-dimensional environment for teaching the processes and structures of Cellular Biology. It currently contains real-time simulations of the electron transport chain, the photosynthesis light reactions, and osmosis.

In this implementation, as in most systems of this type, the client shows the player graphical objects in a 3D

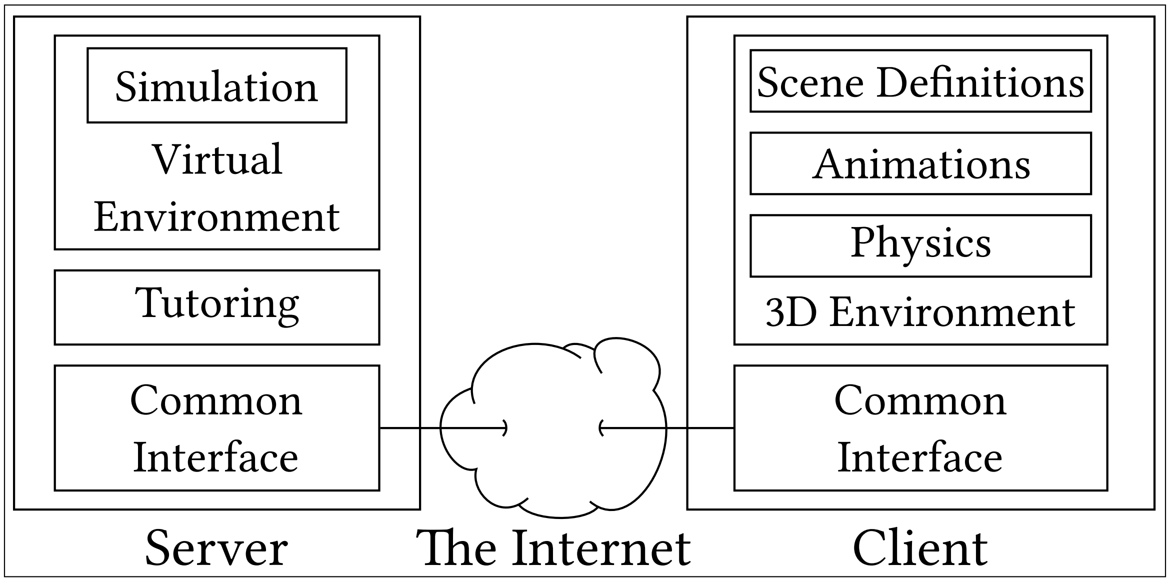

Figure 1. System Diagram. The server maintains a persistent virtual world and facilitates multiplayer collaboration. The client presents a view to the player and sends the player's input (including collisions between objects in the scene) to the server.

scene, and the server keeps track of positions of players and items in the context of each room.

4.1 Simulations

Electron Transport Chain Simulation. In this simulation, many copies of the electron transport chain (ETC) are placed in the cristae of a mitochondrion [Figure 2]. For pedagogical purposes, the complexes that compose the ETC are placed in linear order of their function.

Players observe the complexes as they move protons and electrons through the ETC. All of the models, animations, and physics are located on the client. Each complex is a separate game object. The server maintains a list of complexes in the mitochondrion room and sends the list to the client. The client uses a matching algorithm [Figure 5] to decide how many ETCs should be shown and their placement in the 3D scene such that their positions are consistent across all clients.

Players then perform tests and shoot objects at the complexes. The client informs the server if a collision occurs, and the server returns the new game state [Figure 6].

The objective of this simulation is to watch and test for the complex in each chain that is failing to perform its function. The player then finds a replacement complex and places it into its position in the chain.

Photosynthesis Simulation. Modeling of photosynthesis reactions is much the same as our ETC simulation. The complexes involved in photosynthesis are again placed linearly, this time embedded in the thylakoid membrane inside a chloroplast [Figure 3]. Photons striking the photosystem complexes are depicted as red streaks of light. The same matching algorithm [Figure 5] used in the ETC allows the player to interact with the photosynthesis complexes.

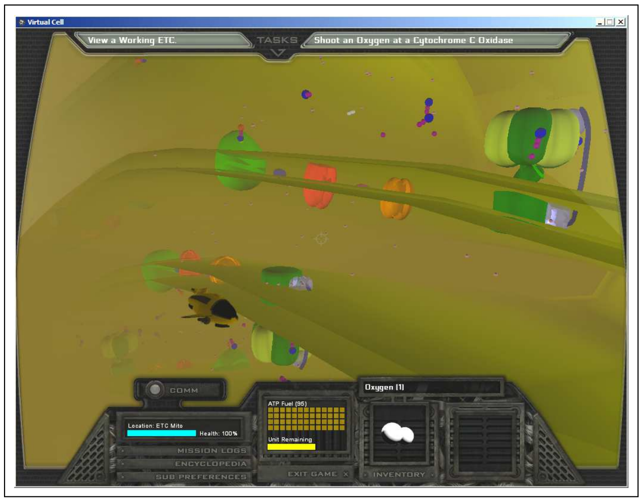

Figure 2. The Electron Transport Chain (ETC) module. In the foreground are two ETCs embedded in the interior of a mitochondrion and a collaborator (submarine, lower-left quadrant). Players examine, probe, diagnose, and finally repair any faulty complexes encountered.

While the ETC simulation deals with each set of complexes independently, allowing players to replace damaged complexes, the photosynthesis simulation requires players to identify missing components within the entire simulation. To model this, we introduce tests that are performed on the chloroplast room the complexes inhabit. The players then drop the missing components (water, light, etc.) into the room in order to repair the entire chloroplast.

Osmosis Simulation. ETC and photosynthesis simulate processes inside cell substructures. The osmosis simu-

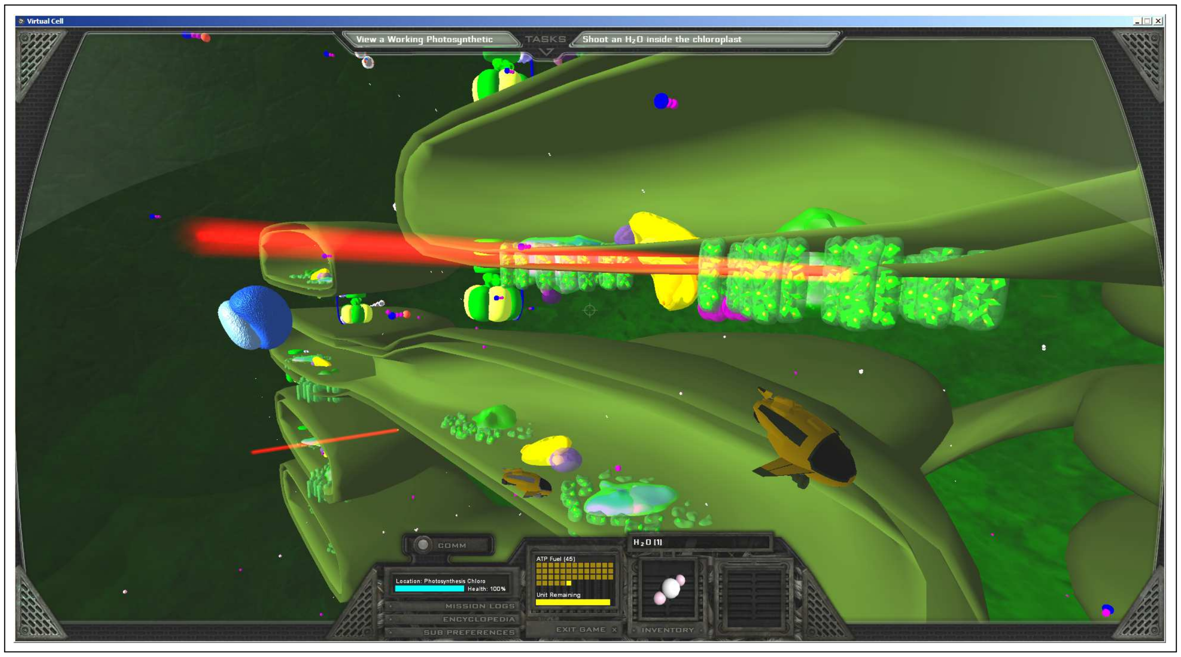

Figure 3. The Photosynthesis module. The player is inside a chloroplast, along with two other players (submarines). Two photons (red streaks of light) are about to strike two photosystem complexes. Players must diagnose the missing component in the chloroplast as a whole. For example, if too little light is present, photon torpedoes are deployed to stimulate the reaction.

lation models a still larger system: the flow of water in the entire cell. An animation of a shrinking cell with blockages (that prevent the flow of water) was implemented on the client [Figure 4]. As the player repairs the problem, the server updates a numeric "health" for the cell. The client uses this health to update the shrinking cell animation.

In order to calculate the health of the cell, the server monitors the contents of the cell room as well as a subset of the rooms that link to it (i.e. the rooms that represent the inner contents of objects in the cell). Players use the tests developed for the photosynthesis simulation to discover which substance needs to be dropped in which room to improve the health of the cell.

4.2 Intelligent Tutors

The tutoring (guidance and remediation) in the Virtual Cell [26] uses a goal system adaptable to players of all abilities. The tutoring is located entirely on the server. Players are assigned tasks and guidance in accordance with their learning goals. Task difficulty is linked with player proficiency such that they remain in the ZPD [4].

Individualized tutoring is provided to players who are unable to complete their tasks. The player's history is recorded on the server, and later searched for patterns indicative of confusion. Two types of tutoring are employed: blind and orientated tutoring. Blind tutoring is applied when no pattern can be found, or the pattern found indicates a chance mistake. Orientated tutoring is used when a found pattern suggests the confusion of two similar topics.

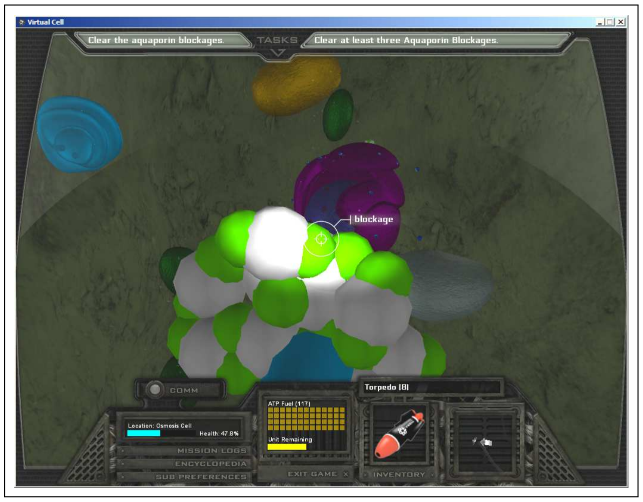

Figure 4. The Osmosis module. The player is examining a plasmodesma blockage (foreground) which is preventing water from flowing into the cell (background). The player must fire concussive torpedoes to remove the blockage.

4.3 Verification

The Crushinator [27] is a tool developed by WoWiWe Instruction Co. in collaboration with NDSU to verify the correctness of and load test a goal-based multiplayer simulation. A model of the goal system under investigation is written in the Unified Modeling Language (UML). This model is used by the Crushinator to construct multiple

Client: **ChangeRoom**. The player enters a room.

Server: The contents of the previous and current rooms are updated.

Client: A predefined 3D scene containing pre-placed game and non-game objects is loaded.

Client: **GetRoomContents**. The player requests the contents of the room.

Server: A list of game objects in the room, and their properties (including a URI for each object) is returned.

Client: The client reconciles its view with the server's:

- The list of predefined game objects in the scene is sorted by type and by distance from the origin point $(0,\ 0,\ 0)$.
- The list of URIs returned from the server is sorted by type and alphabetized.
- The lists are merged.
- Any remaining URIs are discarded.
- Game objects in the scene with no assigned URI are disabled.

Figure 5. Matching Algorithm.

Client: Each game object in the scene is associated with a URI obtained in [Figure 5].

Client: **GetInventory**. The player requests their inventory.

Server: A list of game objects, and their properties (including a URI for each object) is returned.

Client: **FireItem**. The player requests a game object (via its URI) be placed in the room.

Server: The game object:

- Is removed from the player's inventory
- Maintains its ownership field
- Is returned along with its properties, including a (possibly new) URI

Client: **GetSceneState**. Meanwhile, the client has been refreshing its view of the simulation.

Server: Players are informed a new game object has been added to the room.

Client: If the game object is of a predefined type, a new instance is created in the scene with the properties given by the server.

Client: **UseItem**. Meanwhile, clients are checking for collisions of game objects. When a collision occurs, the URIs of the colliding game objects are sent.

Server: The ownership field is checked against the callee. If they match, the simulation state is updated.

Figure 6. Simulation Interaction Algorithm.

paths through the goal system, a form of Model-Based Exploratory Testing (MBET). These paths are then tested and analyzed for errors, alternate outcomes, etc.

To perform load testing, a number of valid paths through the goal system are generated. A large number of automated players are created and assigned one of these paths to follow. This functionality was most helpful in choosing among alternative methods to display scenes, design simulations, and structure the goal system itself.

5 Conclusion

The JavaMOO framework provides a platform for the rapid development of multiplayer educational serious games that meet requirements for successful multiplayer collaboration. The Virtual Cell provides a rich source of experience for developing more games using the platform. The algorithms and tools outlined can be used to develop new games for a variety of different disciplines.

Scalability of the multiplayer simulations has been improved. Many more players can view the simulation (≈ 60) than in previous attempts (≈ 20).

Acknowledgement

This research was supported in part by an SBIR grant from the National Institutes of Health, National Center for Research Resources (NIH-2R44RR024779-02A1).

References

[1] C. Retuer, Authoring Multiplayer Serious Games, *8th International Conference on the Foundations of Digital Games*, 2013, 490-492.

[2] J. Dewey, *The School and Society*, (Chicago, IL: The University of Chicago Press, 1900).

[3] J. Piaget, Science of Education and the Psychology of the Child, in H.E. Gruber & J.J. Voneche (Eds.) *The Essential Piaget*, (New York: Basic Books,1977, 695-725).

[4] L.S. Vygotsky, *Mind in Society: The Development of Higher Psychological Processes*,(Cambridge, MA: Harvard University Press, 1978).

[5] B. Anderson. MMORPGs in Support of Learning,in R. Van Eck (Ed.) *Gaming and Cognition: Theories and Practices from the Learning Sciences*, (Hershey, PA: Information Science Reference, 2010, 55-80).

[6] M. Childress & R. Braswell, Using Massively Multiplayer Online Role-Playing Games for Online Learning, *Distance Education, 27(2)*, 2006, 187-196.

[7] C.J. Dede, The Future of Multimedia: Bridging to Virtual Worlds, *British Journal of Educational Technology, 38 (3)*, 1992, 535-537.

[8] H.F. O'Neil, R. Wainess, & E.L. Baker, Classification of Learning Outcomes,*Curriculum Journal 16(4)*, 2005, 455-474.

[9] R. Reigle & W. Matjka, Dying to learn: Instructional design and MMORPGs, *21st Annual Conference on Distance Teaching and Learning*, 2006.

[10] C. Steinkuehler & S. Duncan, Scientific Habits of Mind in Virtual Worlds, *Journal of Science Education and Technology, 17(6)*, 2008, 530-543.

[11] C. Bonka & V. Dennen, Massive Multiplayer Online Gaming: A Research Framework for Military Training and Education, *Advanced Distributed Learning (ADL) Initiative*, 2005.

[12] B. Cameron & F. Dwyer, The Effect of Online Gaming, Cognition and Feedback Type in Facilitating Delayed Achievement of Different Learning Objectives, *Journal of Interactive Learning Research, 16(3)*, 2005, 243-258.

[13] B.M. Slator, R. Beckwith, L. Brandt, H. Chaput, J.T. Clark, L.M. Daniels, C. Hill, P. McClean, J. Opgrande, B. Saini-Eidukat, D.P. Schwert, B. Vender, & A.R. White, *Electric Worlds in the Classroom: Teaching and Learning with Role-Based Computer Games*, (New York: Teachers College Press, 2006).

[14] G. Hokanson, O. Borchert, B.M. Slator, J. Terpstra, J.T. Clark, L. M. Daniels, H.R. Anderson, A. Bergstrom, T.A. Hanson, J. Reber, D. Reetz, K.L. Weis, A.R. White, & L. Williams, Studying Native American Culture in an Immersive Virtual Environment, *IEEE International Conference on Advanced Learning Technologies*,Santander, Spain, 2008, 788-792.

[15] B. Saini-Eidukat, D.P. Schwert, & B.M. Slator, Geology Explorer: Virtual Geologic Mapping and Interpretation, *Journal of Computers and Geosciences 27(4)*, 2001.

[16] B.M. Slator, & G. Farooque, The Agents in an Agent-based Economic Simulation Model, *International Conference on Computer Applications in Industry And Engineering*,Las Vegas, Nevada, 1998.

[17] A.R. White, P. McClean, & B.M. Slator, The Virtual Cell: A Virtual Environment for Learning Cell Biology, *Tenth International Conference on College Teaching and Learning*, Jacksonville, Florida, 1999,

[18] O. Borchert, L. Brandt, G. Hokanson, B.M. Slator, B. Vender, & E.J. Gutierrez, Principles and Signatures in Serious Games for Science Education, *Gaming and Cognition: Theories and Practice from the Learning Sciences*, 2010.

[19] B. Vender, O. Borchert, B. Dischinger, G. Hokanson, P. McClean, & B.M. Slator. JavaMOO Virtual Cells for Science Learning, *Virtual Immersive and 3D Learning Spaces: Emerging Technologies and Trends*, 2010, 194-211.

[20] P. Curtis, Not Just a Game: How LambdaMOO Came to Exist and What It Did to Get Back at Me, *High Wired: On the Design, Use, and Theory of Educational MOOs*, 1998.

[21] B. Dischinger, *An Architecture for the Implementation and Distribution of Multiuser Virtual Environments* (Masters Thesis, Fargo, North Dakota: NDSU, 2010).

[22] M. Rymaszewski, *Second Life: The Official Guide* (New York: John Wiley & Sons, 2007).

[23] V. Wendel, M. Babarinow, T. Horl, S. Kolmogorov, S. Gobel, & R. Steinme, Woodment: Web-Based Collaborative Multiplayer Serious Game, *Transactions on Edutainment IV LNCS 6250*, 2010, 68-78.

[24] R. Fielding, *Architectural Styles and the Design of Network-based Software Architectures* (PhD Dissertation, Irvine, California: UCI, 2000).

[25] N. Nurseitov, M. Paulson, R. Reynolds, & C. Izurieta, Comparison of JSON and XML Data Interchange Formats: A Case Study, *International Conference on Computer Applications in Industry And Engineering*, San Francisco, California, 2009, 157-162.

[26] P. Yan, B.M. Slator, B. Vender, W. Jin, M. Kariluoma, O. Borchert, G. Hokanson, V. Aggarwal, B. Cosmano, K.T. Cox, A. Pilch, & A. Marry, Intelligent Tutors In Immersive Virtual Environments, *Conference on Cognition and Exploratory Learning in Digital Age*, Fort Worth, Texas, 2013.

[27] C.J. Schaefer, *Model-Based Exploratory Testing: A Controlled Experiment* (Masters Thesis, Fargo, North Dakota: NDSU, 2013).