

12-2011

Defining, Executing and Visualizing Representative Workflows in a Retail Domain

May Zeineldin

University of Arkansas, Fayetteville

Follow this and additional works at: <http://scholarworks.uark.edu/etd>



Part of the [Graphics and Human Computer Interfaces Commons](#)

Recommended Citation

Zeineldin, May, "Defining, Executing and Visualizing Representative Workflows in a Retail Domain" (2011). *Theses and Dissertations*. 261.

<http://scholarworks.uark.edu/etd/261>

This Thesis is brought to you for free and open access by ScholarWorks@UARK. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of ScholarWorks@UARK. For more information, please contact scholar@uark.edu, ccmiddle@uark.edu.

**DEFINING, EXECUTING AND VISUALIZING
REPRESENTATIVE WORKFLOWS IN A RETAIL DOMAIN**

**DEFINING, EXECUTING AND VISUALIZING
REPRESENTATIVE WORKFLOWS IN A RETAIL DOMAIN**

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Computer Science

By

May Zeineldin
October 6 University
Bachelor of Science in Computer Science, 2006

December 2011
University of Arkansas

ABSTRACT

Our lives are filled with routine activities that we do more or less on auto-pilot such as driving to work and cooking. This thesis explores a workflow representation as a way to represent such activities of daily living. The domain of a retail store environment is used. Workflows are initially expressed in a structured English representation, then translated into a Petri net notation and implemented in mix of Petri nets, Lua, and C so that the resulting workflows can be displayed as the actions of collections of avatarbots (avatars controlled by programs) in a 3D virtual world, Second Life. One aim of the thesis is to explore a range of retail workflows to begin to understand the range of mundane workflows at least in one area of daily living, shopping. We observe that, at a leaf level, our workflows consist of observable behaviors (implemented by action verbs like chatting, face and body animations, and actions like go to, pick up, and carry). These workflows are brittle and exception handling is needed to handle common “bugs” in a workflow. A second aim is to add metrics to our workflows so that similar workflows can be compared with different initial conditions, e.g., when a store’s floor layout is varied causing some avatars (representing customers) to walk farther and pass by more goods than other avatars (representing clerks). We conclude that representation of workflow scenarios in a virtual world platform can be helpful in illustrating, analyzing, and improving a real life environment, such as a retail store.

This thesis is approved for recommendation
to the Graduate Council.

Thesis Director:

Dr. Craig W. Thompson

Thesis Committee:

Dr. John M. Gauch

Dr. Nilanjan Banerjee

THESIS DUPLICATION RELEASE

I hereby authorize the University of Arkansas Libraries to duplicate this thesis when needed for research and/or scholarship.

Agreed _____

May Zeineldin

Refused _____

ACKNOWLEDGEMENTS

I thank my thesis advisor, Dr. Craig Thompson, for his great ideas, advice, support, and guidance in helping me to complete this thesis. I also thank my thesis committee members, Dr. John Gauch and Dr Nilanjan Banerjee, for their time and help in reviewing this thesis document.

I thank my parents, my husband and my son for supporting me and providing me with a lovely environment to work in throughout my Masters.

TABLE OF CONTENTS

1. Introduction.....	1
1.1 Context.....	1
1.2 Problem.....	2
1.3 Thesis Statement.....	3
1.4 Approach.....	4
1.5 Organization of this Thesis.....	5
2. Background	6
2.1 Key Concepts.....	6
2.1.1 Activities of Daily Living.....	6
2.1.2 Scripting Languages and Workflow Representations.....	7
2.1.3 Petri Nets.....	9
2.1.4 Virtual Worlds	11
2.1.5 Second Life.....	13
2.2 Related Work.....	14
3. Retail Workflows in Virtual World.....	17
3.1 Workflow Simulations in the Retail Store.....	17
3.1.1 Retail Workflow #1: Shopping in a Retail Store.....	18
3.1.2 Retail Workflow #2: Waiting for the Changing Room.....	20
3.1.3 Retail Workflow #3: Checking Product Size and Ordering from Another Store	22
3.1.4 Retail Workflow #4: Loss Prevention.....	23

3.1.5 Retail Workflow #5: Shopping and Dining in the Retail Store	25
3.2 Methodology	26
3.2.1 Methodology for exploring a new domain.....	26
3.2.2 Methodology for Implementing Individual Workflow Simulations.....	27
3.2.3 Methodology for Evaluating Alternative Floor Plan Layouts in the Retail Store	28
3.3 Implementation	29
3.3.1 Structured English Workflows.....	29
3.3.2 Lua Scripting and the Petri Net Interpreter.....	31
3.3.3 Layout Configurations Using Workflow Simulation.....	36
4. Results and Discussion.....	38
4.1 Results.....	38
4.1.1 Results for Workflow Simulation	38
4.1.2 Results for Evaluating Alternative Configurations in the Retail Store.....	39
4.2 Discussion and Observations	41
4.2.1 Level of Effort to Build a Domain and a New Workflow	41
4.2.2 The Range of Workflows that can happen in a Retail Store.....	44
4.2.3 A Need for Better Workflow Representations and Composition	47
5. Conclusions.....	52
5.1 Summary	52
5.2 Potential Impact	53
5.3 Future Work.....	54
References	56

Appendix A – Retail Sales WorkflowS Described in English	59
Appendix B – Retail Sales Workflow XML Files.....	62
Appendix C – Retail Sales Workflow Lua Files.....	109

LIST OF FIGURES

Figure 1: Petri net components: places, transitions, tokens	10
Figure 2: Sequential routing.....	10
Figure 3: Parallel routing	11
Figure 4: Conditional routing.....	11
Figure 5: The “University of Arkansas” island in Second Life	12
Figure 6: The University of Arkansas Retail Store in Second Life	12
Figure 7: The customer picks up a shirt.....	19
Figure 8: The customer asks the clerk for an available changing room	19
Figure 9: The customer at the checkout counter talking with the clerk.....	20
Figure 10: The customer waits for an available changing room.....	21
Figure 11: Customer asking for her size	22
Figure 12: The clerk checks for the customer’s size.....	23
Figure 13: Sensor detection at the shop door.....	24
Figure 14: The clerk checks the receipt	24
Figure 15: Shopping and dining.....	25
Figure 16: Petri net showing overall workflow	35

LIST OF TABLES

Table 1: Snap Shot of Database	39
Table 2: Total distance for first configuration	40
Table 3: Total distance for second configuration	40

1. INTRODUCTION

1.1 Context

The *Everything is Alive* (EiA) project at the University of Arkansas (<http://vw.ddns.uark.edu>) is focused on developing a consistent architecture for pervasive computing applications using information technologies like databases, ontologies, workflows, and 3D virtual worlds and device technologies like smart phones, Kinects, RFID, and sensors. The EiA project presumes that every real world object can become a smart, semantic, networked object. For instance, if a real world object is given an RFID tag, and if RFID readers are added to smart phones, then cell phone users can discover nearby objects and their associated APIs (stored in a knowledge cloud) and can then be used to query or command these objects. Just by walking around, cell phone users could collectively build a shared model of the world (using virtual world technology), themselves acting as Web search spiders collecting locations and information from objects they pass by. A 3D virtual earth could provide a useful mirror-world model enabling users' avatars to virtually go anywhere and see and interact with their remote real/virtual world surroundings. Kinects could be used to capture not only models of places but also recognize sequences of events. Over the past several years, several student projects have explored aspects of the EiA vision (as documented on the project web site), and the somewhat immature virtual world technical community is itself beginning to understand its role in pervasive model building [1].

One of the remaining grand challenge problems in artificial intelligence is the ability to collect and organize common knowledge. Central to the EiA architecture is an ontology of concrete things and their associated behaviors (together covering a significant part of common

knowledge). Regarding the ontology of concrete things, Eguchi and Thompson [2] identified a collection of protocols that make any object increasingly a smart semantic object. Eno and Thompson [3] showed how to collect object data from 3D virtual worlds and analyze it to augment existing concrete object ontologies.

Less is known about to represent and collect common knowledge about the associated behaviors. Eguchi [4] showed how to use a Kinect to recognize objects by shape and also by function. The latter capability associates a single activity (like a human in a sitting position with an object like a chair) and demonstrates a simple scenario of activity learning in which the Kinect is trained separately trained to recognize walking, skipping, and running and then used to recognize sequences of these activities. Finally, Perkins and Thompson [5, 6] developed a workflow representation and visualization capability that operates to animate avatars in a 3D virtual world.

1.2 Problem

This thesis is related to the latter problem of how to represent common human behaviors and what they might be useful for. At a notional level, this thesis begins by pointing out that our lives are filled with routine activities that we do unconsciously such as combing one's hair, driving to work, and cooking. There are also conscious activities that require focused problem solving such as preparing a class, repairing a broken item, and learning to cook. The artificial intelligence field [7] describes *planning* as an activity characterized by goal statements, rules, and facts; and using a search strategy (like Prolog's depth first search), a system can mechanically solve for the goal statement to see if it is true or under what variable bindings it is true. After searching, a proof encodes a solution. The result of planning can be called a *plan*.

One well-known artificial intelligence researcher Roger Schank termed such plans *scripts* [8]. In this thesis, we call such plans *workflows* because we will re-use workflow representations developed over the past fifteen years to represent such plans. Again, at a notional level, the active process of planning is involved in acquiring a new activity, but, once learned, the activity can be routinized and executed more-or-less on auto-pilot (unless new variations are encountered requiring re-planning).

1.3 Thesis Statement

This thesis focuses on workflows, how to represent them, how to design them, and some of their uses. Objectives of this thesis are:

- to model workflows using a 3D virtual world to understand how to represent workflows visually;
- to assess how complex the collection of workflows covering common retail activities is and understand the level of effort needed to build new workflows;
- to define and execute a representative collection of inter-related workflow simulations in one domain (retail sales was selected) to understand how many workflows might “cover” the domain;
- to begin to understand some elements of a workflow design theory including how workflow relates to planning, exception handling and options; and
- to determine the relative cost of one workflow over another to determine how virtual world workflow simulations can be used to simulate, measure differences, and compare alternative store layout designs as an approach to helping to design a layout using a tool that end-users would readily understand.

1.4 Approach

A workflow is like a recipe or a program: it contains a collection of ingredients and a collection of steps. The ingredients include a location, a collection of props, and a collection of roles. In this thesis, the location or domain is a retail store. The props include the store layout, furnishings, equipment like clothes racks and a checkout counter, and goods like apparel for sale. Roles include the customer and sales person. Within the retail environment, a collection of workflows are defined corresponding to different use cases in software: the customer browses, tries on clothes, purchases, shop lifts, and the sales person assists in some of these tasks.

Based on an earlier work by Keith Perkins and Craig Thompson [5, 6], our team developed a method of defining workflow simulations in the 3D virtual world Second Life. The easier part is to use Second Life to build the props for a new domain. The more challenging part is to define a behavior specification language. Perkins developed a Petri net representation to define workflows. This representation improved our earlier linear representation because it accommodates simultaneous actions within a workflow graph (via splitting and joining operations).

This thesis adds to our understanding of workflows. It explores a different domain (retail sales whereas Perkins considered a single hospital Catheter Lab operation). The domain of retail sales was selected because it is an easy-to-understand domain that is part of everyone's activities of daily living experience.¹ It covers a richer collection of workflows in its domain. It considers

¹ A drawback of retail sales is that it is such a familiar domain that the reader may not appreciate the amount of effort involved in representing workflows in this area. This is generally true of "activities of daily living" (described in section 2.1.1), yet these sorts of activities are also an area where computers so far play less of a role in our lives because they lack the common sense knowledge involved in representing these kinds of domains.

workflow exceptions. Finally, it considers metrics on workflows to enable comparison of workflow alternatives.

1.5 Organization of this Thesis

Chapter 2 covers background concepts related to activities of daily living, scripting languages and workflows, Petri net, virtual worlds, and the 3D virtual world Second Life, and covers research related to this thesis. Chapter 3 describes workflows implemented to cover a collection of retail store domain simulated in the 3D virtual world Second Life; describes the methodology used in implementing workflow simulations and the methodology needed for evaluating alternative floor plan configurations in the retail store; and how the workflows are implemented using Petri nets and the Lua scripting language. Chapter 4 describes results and observations. Chapter 5 includes conclusions, potential impact, and future work.

2. BACKGROUND

2.1 Key Concepts

This thesis depends on a basic understanding of activities of daily living, scripting and workflow, Petri nets, virtual worlds, and Second Life, as described in this section.

2.1.1 Activities of Daily Living

In the health care field, the term *activities of daily living* (ADL) is used to refer to basic tasks that a person should do everyday routinely and alone without depending on another person. These include bathing, dressing, eating, walking, and toileting [9]. Other broader health care ADLs include brushing teeth, taking medication every day, working, and shopping. Performing ADLs can grow more difficult with age and the elderly or people with disabilities may need either another human being or mechanical devices to help them [10]. Memory aids like digital cameras and mobile phones can be used to record memories or create alerts to remind people about appointments, personal occasions, and meetings [11].

In one interesting experiment using smart devices and monitoring, Park and Kautz [12] used multi view cameras (two narrow field of view and two wide field of view cameras) and radio-frequency identification (RFID) in the form of a bracelet reader. They placed RFID tags on various household items, furniture, appliances, and then monitored a home user. By gathering trace data from RFID and from cameras, they could record sequences of leaf level events and then translate them into higher level events. For instance, a sequence taking place in the morning, starting with opening the cupboard, getting a bowl, getting a spoon, getting cereal,

pouring cereal into the bowl, pouring milk, and finally eating cereal with spoon could be recognized as eating breakfast.

In this thesis, we broaden the scope of the term activities of daily living to include any mundane activity: planning a trip; scheduling a meeting; driving to the airport; and so on. While some activities of daily living are mundane (tying one's shoes), people with disabilities may have problems with these; on the other hand, for trained individuals like physicians, a catheter heart operation can be mundane so the term is situationally defined.

2.1.2 Scripting Languages and Workflow Representations

Many gaming and virtual worlds support scripting languages to program interactions among avatars and objects. Popular scripting languages include Python and Lua. Only a few gaming environments appear to support workflow [13, 14]. While scripting languages are useful for expressing basic operations (and we use Lua for this purpose in this thesis), they are still programming languages, and we are interested in higher level languages that are more declarative, which we will term workflow languages. The reason is, we believe there is a better chance to develop and analyze reusable patterns for activities of daily living using this kind of higher level representation.

Workflow is well researched area including:

- a literature on scientific workflows that enable collections of researchers to share and process data sets [15];
- industrial tools such as Extract-Transform-Load (ETL) tools that enable organizations to process huge data sets on grids of nodes. For example, the Axiom grid processes workflows with hundreds of tasks that operate on millions of records on grids with

thousands of nodes. The National Geospatial-Intelligence Agency similarly processes maps and thematic data as inputs through dozens of processing steps to create specialty maps for defense, intelligence, and other government purposes.

- human workflows used in distributed decision making approval processes. A university registrar might define distributed workflows to insure that advising occurs before a student can register and that the student is taking classes that do not conflict. Hospitals, insurers, and many other industry segments use this sort of workflow.

As can be seen from these examples, some workflow tasks are automated, for instance, gathering data from a database or transforming data in some way. Other workflow tasks require inputs or approvals by humans.

In the mid-1990s, an industry group, the Workflow Management Coalition (WfMC), developed a standard description of workflow including a standard glossary and also a reference model [16]. The WfMC defined a Workflow Management System as “*A system that defines, creates and manages the execution of workflows through the use of software, running on one or more workflow engines, which is able to interpret the process definition, interact with workflow participants and, where required, invoke the use of IT tools and applications*” [17]. That is, a workflow management system has software components that control and manage the workflow while running and also control the workflow interaction with the users. Since that time, several workflow languages have been proposed, implemented and some even standardized, including Yet Another Workflow Language (YAWL) [17], Business Process Execution Language (BPEL) [18], and XML Process Definition Language (XPDL) [19], the last standardized by WfMC.

In this thesis, we will extend the use of the term workflow to describe a sequence of behavior (tasks, actions, steps) that happens involving a group of persons and objects.

Synonyms for workflow include script, scenario and vignette [20]. Although the term workflow is often used to cover graphs of automated steps, human workflows involve actions by people in roles. So, for the purpose of this thesis, we consider mainly workflow as sequences of observable behaviors in humans. As will be described in more detail later, a workflow in this sense can be viewed as a recipe consisting of ingredients and steps. The representation of the workflow ingredients can include a scene or domain, entities (e.g., avatars, robots, smart objects) that acts in roles (e.g., doctor, nurse, customer, clerk) as well as other ingredients, e.g., building, furniture, equipment, and supplies. Workflow steps are procedures that involve the actors that use the equipment to perform some action and workflows can be represented in UML, via Petri nets, and in other representational formats [6].

2.1.3 Petri Nets

Petri nets can be used to define workflows. Petri nets were developed by Carl Adam in 1962 in his PhD dissertation. A Petri net is a mathematical tool used to represent events in a discrete system [21]. A Petri net is like a flow chart. It is represented in the form of diagram that consists of places (represented as circles) that are connected to transitions (represented as rectangles) through arcs (represented as arrows). Places with arcs that are connected to a transition are called input places while places with arcs coming from a transition are called output places [22]. Transitions represent actions, and places represent conditions or state that should meet before an action can happen. Arcs can connect places and transitions (they do not directly connect a place to another place and also they do not connect between a transition and another transition).

If a place has a token inside it, once transition occurs, the token is removed from the input place to the output place as shown below.

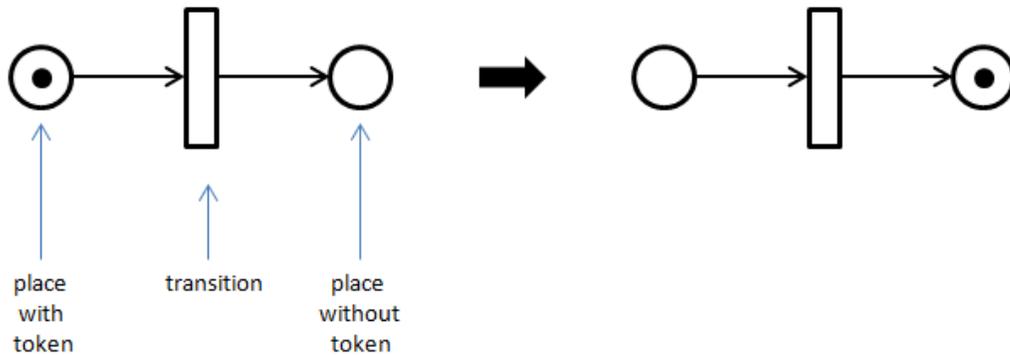


Figure 1: Petri net components: places, transitions, tokens

A place can have more than one token, with each token representing a specific object that a condition will hold. And there can be more than one arc, where each arc will represent a different token. For each Petri net diagram, there is a start place and an end place. There are different routes from the start point to the end point as shown below – figures from [2-4]:

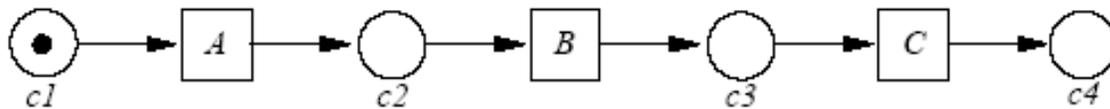


Figure 2: Sequential routing

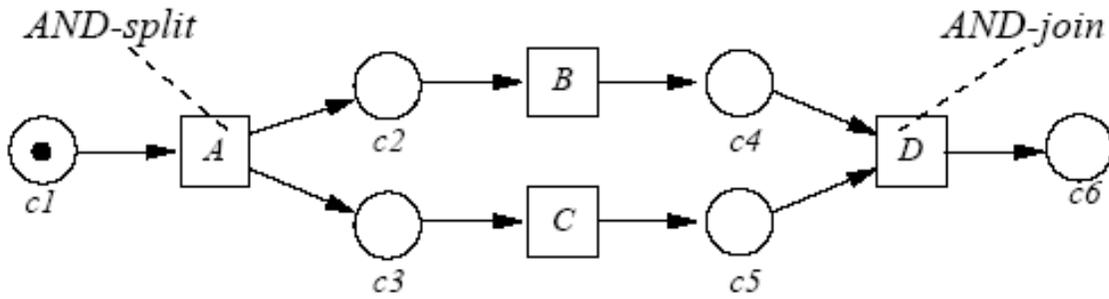


Figure 3: Parallel routing

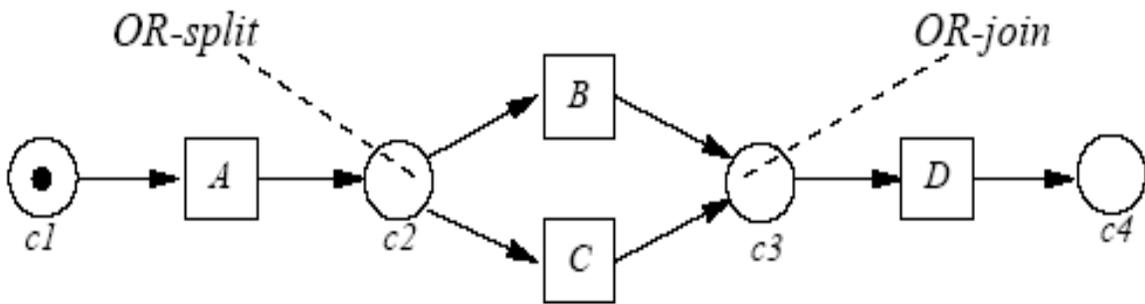


Figure 4: Conditional routing

2.1.4 Virtual Worlds

Figure 5 below shows a view of “University of Arkansas” island in Second Life, a 3D virtual world. Figure 6 shows the layout of the virtual world retail store that is used as an example in this thesis.

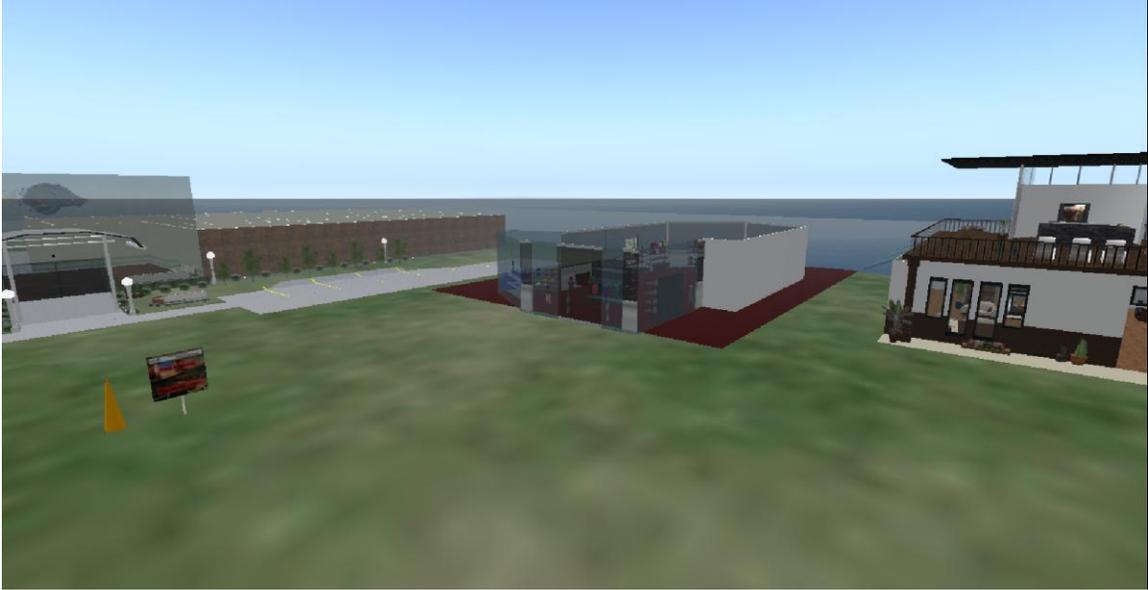


Figure 5: The “University of Arkansas” island in Second Life



Figure 6: The University of Arkansas Retail Store in Second Life

Generally, a virtual world is a distributed computer-based environment accessed by users world-wide who can interact with each other and create objects and use them [24]. Virtual world

use 3D computer graphics. Users are represented in the form of avatars that interact with the objects on the virtual world using mouse and keyboard movements [25]. In some virtual worlds, avatars can hear to sound including other nearby avatars speaking or they can communicate by chatting publically or privately using chat channels; they can build objects; and the objects can have scripts that act triggered by events.

Virtual worlds have different uses. One use is gaming – early gaming worlds include *Wolfenstein 3D* (1992), *Doom* (1993) and *Quake* (1996) [25]. Other uses of virtual worlds involve so-called “serious games” in which virtual worlds are used for visualization, education, training, and simulation in a wide variety of domains including museums, health care, manufacturing, and stores. Another use is collaboration via online meetings and online university courses. Another kind of use is socialization and community building, where different people from different cultures can chat together in their virtual living room [26].

For this research, we had different avatars that interact with each other through talking, moving, carrying, and exchanging objects. Also, we created objects that were scripted with events related actions like being touched, taken, worn, sat on. Some objects were sensitive to being nearby and others had sensor and sound effect.

2.1.5 Second Life

Second life is a 3D virtual world that was launched in 2003 by Linden Lab. It is the largest virtual world with more than 13 million registered users [27]. Residents of Second Life can interact with each other in the form of avatars. Inside Second Life, they can meet other avatars, communicate together through chatting or with voice enabled so they can talk to each other [27]; they can socialize; and they can exchange services between each other. They can

teleport to other locations in Second Life and navigate by walking, running, or flying. Avatars can also build objects and script them using Linden Scripting Language to define actions. Users can upload textures, animations, or sounds into Second Life.

Second Life has an internal currency called Linden dollars (L\$) which can be used to buy, sell, or rent land, objects or services with other users. The Linden dollar can be purchased using US or some other currencies on the LindeX exchange provided by Linden Lab [28].

Second Life uses a client/server architecture. The total number of regions is about 31,000 regions running on 8,000 servers [29]. Each region has its own simulator which handles everything that happens in the region like the objects on the region, the scripts on the region, and the avatars movements. A region keeps track of where everything is, sends locations of objects to viewers on client machines, and sends updates to viewers as needed.

The client viewer, like a Web browser, is the viewport into Second Life on the user's computer. The user can log on to the Second Life using his/ her avatar personality to create objects, move around, run, fly, put scripts on objects, and chat with friends. All these actions of the avatar are transmitted to all the client viewers on the same region and nearby.

2.2 Related Work

Second Life does not provide a higher level way to represent workflows. This thesis builds on recent past work by our research team as described below.

Over the past few years, our research team at University of Arkansas has implemented several workflow simulations using the Second Life virtual world. Demos for these workflow simulations are available on our research project's website at <http://vw.ddns.uark.edu/index.php?page=media>. An early demonstration shows robot patients in

hospital beds while robot nurses attend to their needs. When a robot patient reports it is hungry, cold, or in pain, a robot nurse responds by leaving the hospital ward and returning with a tray of food, a warming blanket, or pain medications, respectively. This demonstration used a Robot Command Language developed by Nick Farrer [30]. That command language, implemented entirely in Second Life's Linden Scripting Language, enabled a user to *rez* (create) a robot and command it to go to a waypoint location, pick up an object, carry an object, or put down an object. A built-in path finding algorithm was used to find a path from the robot's present location to a named location. In Second Life, robots are object built from Second Life prims and are different than avatars built from meshes. As part of a companion project that the author worked on, we build a parallel capability to demonstrate a workflow using avatarbots, that is, avatars controlled by program code instead of users. That project involved a workflow in a Catheter Lab and showed doctors, nurses, an anesthesiologist and a patient during a heart catheterization procedure. Our team developed an English workflow, then a Prolog implementation (overly simple), then manually translated that to programming steps. A major limitation of that system was, there was no explicit workflow representation, just a program that controlled the avatars. Also, each avatar in turn completed a step – there were no simultaneous activities in that initial “workflow”. Our team also built a restaurant ordering-and-serving workflow and a workflow for the balcony scene in Romeo and Juliet. These workflows are viewable on the above mentioned project website.

The explicit workflow modeling capability used in this thesis depends on an implementation developed by Keith Perkins in his Master's thesis [6]. Perkins identified that an existing Petri net implementation could be used to represent the workflow steps. He used a popular gaming scripting language Lua [31, 32] instead of Linden Scripting Language to define

individual workflow script operations. Keith's workflows took place in a health care setting at the "Hogspital" on University of Arkansas island.² Perkins re-used the same heart catheterization operation workflow but this time represented the workflow via Petri nets, Lua and C# software. The heart catheterization implemented included all the steps in a real world operation, based on a script from Dr. Fran Hagstrom based on her experiences at Washington Regional Hospital. However, Perkin's workflows did not handle exceptions that can happen during the workflow. For example, no alternative scenarios were programmed in case something unexpected occurred during the operation, like dropping a scalpel. This current thesis is based on the work of Perkins but it models a different domain, a retail store. It covers a wider variety of workflows to understand workflow interactions. Several English workflow scenarios (between customers and clerks) for a retail store were developed in English and using Petri nets. These workflows were simulated and a mechanism to handle exceptions was incorporated. Finally, an additional mechanism to add metrics to a workflow was developed so results of workflows under different initial conditions could be compared.

² The University of Arkansas mascot is a razorback hog; hence the name of the hospital on University of Arkansas island in Second Life.

3. RETAIL WORKFLOWS IN VIRTUAL WORLD

This chapter covers a description of representative workflows in the domain of a retail store that were implemented in Second Life (section 3.1); the methodology used to represent retail workflows (section 3.2); and the workflow implementation based on Petri nets and Lua (section 3.3). The section is organized in this order because it is easier to first see examples of workflows, and then understand the design methodology and the implementation.

Before beginning, it is worth pointing out that, like much of software, workflows have a definition and the definition can be executed. The workflow definition defines a range of possible behaviors and an execution trace provides information on an individual execution instance. An execution trace can be viewed in Second Life as a running simulation, recorded as a video, or captured an execution trace that contains e.g., a list of script step instances that were executed.

3.1 Workflow Simulations in the Retail Store

As mentioned in section 2.2, the Everything is Alive project has previously built individual workflows in various domains – in the health care domain, a robot nurses and patients workflow and also a Catheter Heart Operation workflow; in the restaurant domain, an ordering- and serving food workflow; and in the domain of theater, the balcony scene from Romeo and Juliet. In that earlier work, the projects did not develop a range of workflows representing different “business processes” within a single domain.

Below are described a collection of retail store workflow simulations that were defined and implemented in the Second Life including:

- Shopping in Retail Store
- Waiting for the Changing Room
- Checking Product Size and Ordering from Another Store
- Loss Prevention
- Shopping and Dining in Retail Store

These workflows are described in more detail below. YouTube videos demonstrating these workflows are available at the following links:

- A combination of Shopping in Retail Store and Waiting for the changing room workflows: http://www.youtube.com/watch?v=_KY5LwFy-4s&feature=player_embedded
- Checking Product Size and Ordering from Another Store: <http://www.youtube.com/watch?v=2bBUCrbd67Y>
- Loss Prevention: <http://www.youtube.com/watch?v=OS4HDefzMJY>
- Shopping and Dining in Retail Store: <http://www.youtube.com/watch?v=yWsiI2-GEew>

3.1.1 Retail Workflow #1: Shopping in a Retail Store

In the “Shopping in a Retail Store” workflow, a customer (avatarbot) enters the store and starts looking at the products, picks up some products (shirt and pants) as shown in Figure 7, walks over to the clerk (another avatarbot), and asks the clerk for a changing room to try the products; the clerk takes the customer to the available changing room as shown in Figure 8; and the customer enters the changing room. After a while, the customer comes out and goes to the check out as shown in Figure 9, buys the products and takes them in a bag and then leaves the store.



Figure 7: The customer picks up a shirt



Figure 8: The customer asks the clerk for an available changing room



Figure 9: The customer at the checkout counter talking with the clerk

This first workflow encodes a main business process use case of shopping wherein a customer enters a store, selects an item, tries it on, decides to purchase, purchases, and leaves with the purchase.

3.1.2 Retail Workflow #2: Waiting for the Changing Room

Sometimes, something goes wrong with a workflow so something like exception handling is needed. In Workflow #2, another customer enters the store and takes a shirt, goes to clerk and asks for an available changing room. The clerk checks the changing room and informs the customer that currently there is not one available. The clerk asks the customer to have a seat until the changing room becomes available. Figure 10 shows the second customer waiting for the changing room. Once the changing room is available, the clerk indicates to the customer that

the changing room is free, the customer, enters the changing room, and, after a while, the customer comes out and goes to the check out desk, purchases the shirt, and leaves.



Figure 10: The customer waits for an available changing room

The second workflow can occur simultaneously with the first so that both customers and the clerk are in the store at the same time interacting with each other. In addition to illustrating simultaneous workflows, this second workflow is like the first except that it also illustrates an exception that occurred when the changing room (a scarce resource) was unavailable resulting in the second customer waiting for the scarce resource to be available.

This exception was added to become part of workflow #1 by adding a conditional step wherein the clerk first checks if the changing room is available and if not then asks the customer to have a seat until it is available.

3.1.3 Retail Workflow #3: Checking Product Size and Ordering from Another Store

In this workflow, which is also an exception to workflow #1, a customer enters the store and checks the available products. The customer finds a desirable shirt but does not find her size, takes the shirt to the clerk, and asks the clerk if the store has her size for that shirt (see Figure 11). The clerk takes the shirt from the customer, checks if the store has that size (see Figure 12); then returns back to the customer telling her that this store does not have her size; and then offers to look for the same size in different stores and order it for the customer. The customer agrees. The clerk then finds the size in another store and orders it for the customer. Finally, the customer thanks the clerk and leaves the store.



Figure 11: Customer asking for her size



Figure 12: The clerk checks for the customer's size

3.1.4 Retail Workflow #4: Loss Prevention

Another kind of exception to workflow #1 occurs if a customer tries to take an item out of the store without paying for it. This workflow illustrates security monitoring inside the retail store to prevent shoplifting. In this workflow, the customer enters the store, looks through the products, picks a shirt and a pair of pants, moves to the clerk and asks for a changing room. The clerk guides the customer to the changing room, the customer enters the changing room and, after a while, she comes out, moves to the check out desk and picks sunglasses and pays for all the products she chooses except for the sunglasses. At the door of the store, there are sensors on the gates which beep as the customer leaves the store with the unpaid sunglasses as shown in Figure 14. The clerk comes to the customer and asks for the receipt as shown in Figure 15. The customer gives it to him and the clerk finds that the customer did not pay for the sunglasses. The clerk then asks the customer if she wants to purchase the sunglasses, and the customer returns

back to the check out desk, pays for the sunglasses and leaves the store. The sensors on the gates of the store emulate sensors used to keep real life stores secure.



Figure 13: Sensor detection at the shop door



Figure 14: The clerk checks the receipt

3.1.5 Retail Workflow #5: Shopping and Dining in the Retail Store

Some retail stores provide in-store dining so a “dining in a retail store” workflow was added to the collection of workflows – see Figure 15. In the dining workflow scenario, the customer enters the store, moves to the dining area and sits down. The waiter comes to her, takes her order, prepares the food, and serves the food to the customer. The customer eats her food, and the waiter takes the tray and gives the customer the check. The customer pays the check and leaves the store.

During the execution of the dining workflow simulation, the workflow for shopping inside the retail store is also being executed concurrently. The objective of adding the dining workflow was to show that more than one workflow can run at the same time and to show that more than two avatars can be at the same area doing their tasks at the same time. Each avatar was doing their tasks without interfering with the other avatars.

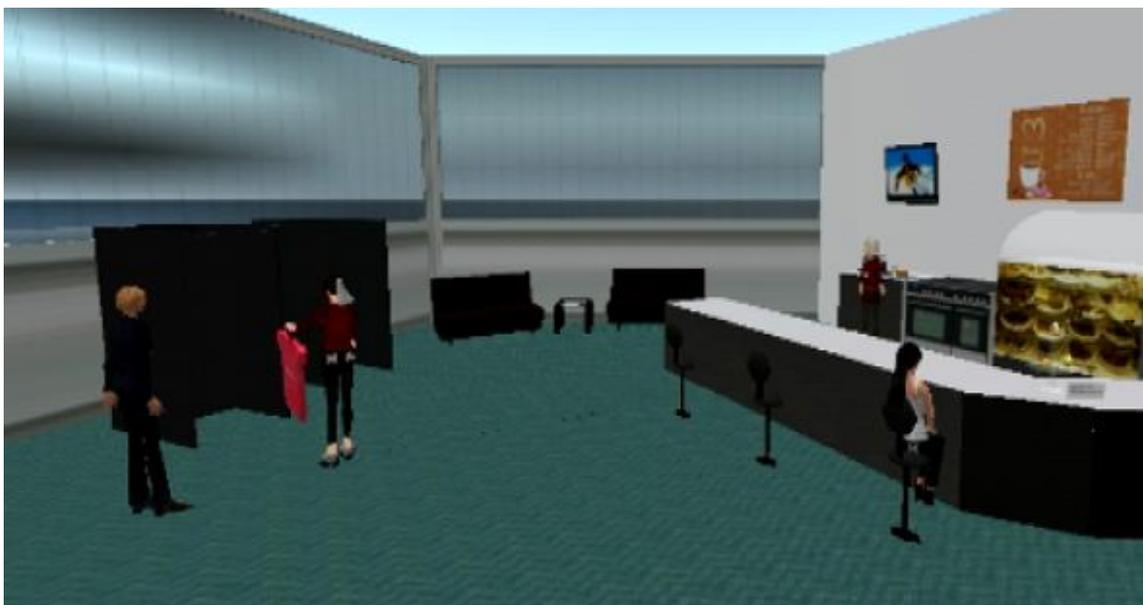


Figure 15: Shopping and dining

3.2 Methodology

Three methodologies are described: one to identify workflows in a new domain; a second to describe each workflow; and a third to add metrics to a workflow so two workflows can be compared.

3.2.1 Methodology for exploring a new domain

This methodology involves picking a domain related to activities for daily living and identifying the workflows within that domain.

Step 1: Pick a domain. People's days are spent doing routine things – getting up, going places, working, going to school, shopping, going out to eat at restaurants, being entertained, and going back home at day's end to repeat the steps the next day. We will term these various loci of activities *domains*. We could pick any of these to expand on to understand how complex that domain is. We picked the **retail domain** because it is familiar, easy to reason about, and can be considered an activity of daily living about which most adults are familiar.

Step 2: Identify what sorts of things we know about the domain. After selecting a new domain, a brainstorming step is to identify what we know about the domain. This can result in a list of kinds of places and objects that are related to the domain; a listing of the kinds of human roles; a list of representative workflows that account for most of the activities in the domain; and a list of exception workflows in case something goes wrong. As a result of this step, a representative collection of workflows is identified for simulation. The workflows that we selected for the retail domain were listed in section 3.1. The roles we identified were clerk and customer. The “ingredients” of the workflows are seen in the workflow simulations as earlier described.

3.2.2 Methodology for Implementing Individual Workflow Simulations

Each workflow simulation was implemented based on the following steps. The actual English descriptions of the retail workflows from Step 1 are shown in Appendix A. The actual Lua and Petri net representations of workflows are shown in Appendix B. Reviewing these may be helpful in understanding the steps below.

- **Step 3:** The workflow was described in the form of a structured English description. The first part of the workflow description declares the domain and props needed for the workflow. The second part of the workflow declares the steps of the workflow (called script operators) and the control flow among them. See section 3.3.1 and Appendix A.
- **Step 4:** The objects, scripts, and avatar animations declared in Step 3 were built in Second Life. Examples of objects needed in the retail workflow scenarios are (the store's floor plan, changing room, the checkout counter, shelves, and products) and scripts were added to them if needed. Examples of scripts include scripting the changing room so that the customers can open and close the door and scripting the couch so that customers can sit on it. Example avatar animations include avatar animations include picking up, carrying and dropping objects. Finally, chat-to-speech converters were implemented so that avatars can communicate using speech messages.
- **Step 5:** The individual workflow script steps were implemented using Lua scripts and the functions of the avatarbot control program. In Lua, tasks were assigned to each avatar such as Move to location, Attach shirt to avatar, Move to changing room. At the end, each avatar will have a set of activities that it can accomplish. When Lua alone is used to execute the tasks, without using the Petri nets, the steps are performed in

sequential order, so, each avatar does not execute its tasks unless the earlier tasks performed by the other avatars were finished. When only using Lua workflows, Lua only runs a single task at a time. See section 3.3.2.

- **Step 6:** When Petri nets are included, Lua workflow simulations were implemented using Petri nets. By using Petri nets, more than one action can occur at the same time unlike the Lua script alone where steps have to occur consecutively. See section 3.3.3.

3.2.3 Methodology for Evaluating Alternative Floor Plan Layouts in the Retail Store

After the basic ability to run a collection of workflows was completed (described in the previous section), a second methodology was used to compare alternative floor plan layouts in the retail store:

- **Step 7:** The objects in the Second Life that were scripted using LSL to capture <X, Y, Z> coordinates and a time stamp for each avatarbot as it moved within the store.
- **Step 8:** A PHP program was implemented to link the scripted avatarbots in the Second Life to an external database. The PHP program exports avatarbot positions and timestamps and stores them in a database, creating a track history.
- **Step 9:** The total distance travelled by each avatarbot is calculated using a MATLAB subroutine. The MATLAB program the avatarbot track history and calculates the distance travelled by an avatarbot between way points A and B using the standard Pythagorean formula:

$$\text{distance}(A, B) = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2} \quad (3.1)$$

The total distance travelled by each avatarbot was then calculated by summing the segment distances along the path that the avatar traveled and the total time is calculated by summing the time taken for each segment.

- **Step 10:** The retail store configuration was changed, the workflow simulation was re-executed, and the metric data was collected for the new configuration. (Note: for each configuration, the simulation data was run ten times and averaged.) See section 3.3.4.
- **Step 11:** The layout configuration with the shortest distance and time for the customer was selected as preferred.³

3.3 Implementation

Most of the steps above are self-explanatory but some require additional detail. Section 3.3.1 describes the structured English representation of workflows (step 3). Section 3.3.2 describes how the workflow simulations are implemented using scripted objects in Second Life, an avatarbot control program, and workflow control programs (steps 4-5). Lua, Petri nets and C# are used to implement the workflow simulations.

3.3.1 Structured English Workflows

Some methodologies for designing representations begin with English representations and then refine them into more declarative representations. An example in the database field is the Entity-Relationship (E-R) data model which represents a database schema as a collection of

³ In chapter 4, we will discuss other metrics we could have collected and other objective functions.

Entities and Relationships, both with attributes. A common approach to designing E-R diagrams is to begin with English sentences about a domain to be represented and extract the nouns to become entities and the verbs to become relationships [33]. This is a manual translation and design process that humans can use to design database schemas.

A similar approach can be used to initially capture workflows in a structured English format. At a high level, a workflow can be viewed as a recipe containing ingredients and a procedure (or subprocedures). The ingredients declare the domain and props needed for the workflow (the entities or objects) and the procedure declares the steps (which can be referred to as script operators) of the workflow (called script operators) and the control flow among them.

A given domain contains:

- objects that can be grouped into user defined categories like buildings, vehicles, equipment, goods, and supplies. Some of the objects within a domain will need to be scripted using Second Life's Linden Scripting Language.
- roles like customer and salesperson or doctor, nurse, patient, and anesthesiologist and role assignments of named avatarbots to roles. Associated with avatars are animations and associated with roles may be uniforms.
- workflows that are named and that each consist of a step of script steps

For the domains and workflows in this thesis, all entities are observables in the sense that an observer can sense them by seeing them, hearing them, and so on. All the objects in a workflow are observables in this sense. So too are avatarbot primitive operations which include operations like Go to <waypoint>, Pick up <object>, Carry <object>, Put down <object>, Chat <string>, and Animate <avatar> <animation>.

In the retail store domain, objects that needed to be built included: the building walls and windows; the checkout counter, the changing room, and the dining area; furnishings including chairs and couches; equipment including racks and shelves to hang products and the RFID sensor at the door; and many kinds of products including shirts, pants, skirts, hats, glasses, and watches. Some of the objects needed to be scripted using the Second Life LSL scripting language. For example the changing room was scripted so that the avatars can open and close the door. A chat2voice script was added so avatar communications can use a robotic male or female voice. The products were scripted manually so that avatars can hold or drop products and move while carrying them. Animations were added for exchanging money between customer and clerk during checking out. In the workflow simulations there were audio voices between customer-clerk talking to each other.

Examples of workflows represented in structure English appear in Appendix A.

3.3.2 Lua Scripting and the Petri Net Interpreter

Workflows were added using Lua, Petri nets and C#. In this thesis, this process is described at a high level because workflow definition was the subject of Keith Perkins' Master's Thesis [6] and this thesis is not about building workflows but rather using them.

Lua is a light-weight scripting language widely used as a scripting language for video games [31, 32]. Lua has a semantics largely borrowed from Scheme and a syntax borrowed from Modula, CLU, C++, SNOBOL and AWK. It can be embedded in other applications using its C API.

In this thesis research, the operators in the English steps were implemented using Lua scripting. Each avatarbot has its own Lua engine and can execute the Lua scripting using the

command *luacall*. At the same time, the Lua script accesses the C# Avatarbot object that encloses the Avatarbot executing script. C# is called in the Lua script by using the command *docmd* which allows Lua to start executing the commands.

Initially, in this project, we used Lua, not just to program workflow script steps but also to program flow of control from one script step to the next. Later, we used Petri nets for the latter purpose. Lua scripting executes the commands step by step and cannot execute more than one step at the same time. Therefore, we upgraded our implementation so that Lua was used to execute individual workflow script steps but a Petri net was used as a control structure to provide control among steps because the Petri nets can execute more than one step at the same time. Also the Petri net can execute more than one workflow simulation at the same time.

We started by implementing the workflow scenarios using Lua script and the avatarbot function on the Second Life. The following code is a snippet of code for Lua:

```
{
moveto 116.83 61.56 28.07 tele @ mayzein Resident
moveto 115.81 30.85 28.11 tele @ clerk123 Resident
moveto 101.75 51.32 28.10 tele @ customer123 Resident
}

@ mayzein Resident
regequip Store
movewp <110.65,61.36,28.07><110.04,52.17,28.11>
pause 2
touch <<Store_shirt_7>>
pause 1

@ clerk123 Resident
turnto 180
forward
pause 1
```

There are two ways to assign a task to avatar. The first way is represented in the form of a block that contains the commands between curly braces “{}”; this means that the commands

are executed in parallel. The symbol “@” terminates the list of command arguments and is followed by the name of the avatar that will do the task. Blocks can contain at most one single command (task) for each avatar and once the avatar finishes his task, he has to wait till the end of the block until all avatars finish their tasks. The second way for assigning tasks to avatars is putting “@” followed by the avatar name and the block of commands (tasks) for the avatar. This is performed when more than one command is assigned to each avatar. “regequip Store” will check for the objects that can be touched and can be attached to avatars inside the Store. “touch <<Store_shirt_7” is a command for the avatar to touch shirt 7 and attach it to his hand.

The next step, involved implementing the workflow simulation using Lua script and Petri nets. By using Petri nets more than one avatar can work at the same time to accomplish set of commands. The following is a snippet of code for Lua using Petri nets and a diagram illustrating the Petri nets:

For Customer1:

```
function init()
    return "T00_Poscustomer, poscust||" ..
           "T02_Enterfitting, enterfit||" ..
           "T04_Enterroom, enterroom||" ..
           "T07_Checkout, checkout||" ..
           "T12_Actioncheckout, action||" ..
           "T13_Done, done"
end

function enterroom(taskname)
    doCmd("say 1313 open")
    doCmd("moveto 115.81 26.83 28.13")
    doCmd("say 1313 open")
    doCmd("pause 5")
    callBack(myName, "okay||" .. taskname)
end
```

For Customer2:

```
function init()
    return "T05_Enterstore, enterstore||" ..
           "T08_Sitdown, sit||" ..
           "T11_Stand, stand||" ..
           "T14_Finishcheck, finish||" ..
```

```

        "T16_Pay,paid"
end

function enterstore(taskname)
    doCmd("movewp <103.16,57.53,28.10>
           <109.38,57.61,28.10>
           <110.08,52.67,28.11>")
    doCmd("pause 1")
    doCmd("touch <<Store_shirt_2>>")
    doCmd("movewp <107.43,50.49,28.11>
           <112.25,31.92,28.11>")
    doCmd("turnto 300")
    doCmd("forward")
    doCmd("pause 1")
    doCmd("play TALK")
    doCmd("say primaudio|voice=4|
           text=Hi,Can I enter the fitting room?")
    doCmd("pause 6")
    callBack(myName, "okay||" .. taskname)
end

```

As explained in section 2.1.3, a Petri net can be represented in the form of flow charts and block diagrams. It consists of places that are connected to transitions through arcs. Arcs have an integer weight which defines the number of tokens that passes through them after a transition is fired. By default, arcs have a weight of one and transitions have weight of one. When more than one transition occurs within a single place this means that some actions are occurring. Once the first transition fires to start an action, this means that a token in this place has occurred. Once the action is completed, the token is moved to another place. This process continues until the last transition is fired and the end is reached.

In this thesis, each avatarbot has its own Lua script which contains various sets of functions that describe the tasks that each avatar should do. These functions are initialized using Petri nets in the form of transitions. When transition fires this means that it starts doing the actions, then the rest of the transitions take place until the Petri nets reach the last transition. More than one avatarbot can have sets of actions that can fire at the same time in parallel. The objective of implementing the workflow scenarios using Lua scripts and Petri nets is executing

more than one action in parallel and executing more than one workflow scenario at a certain instant.

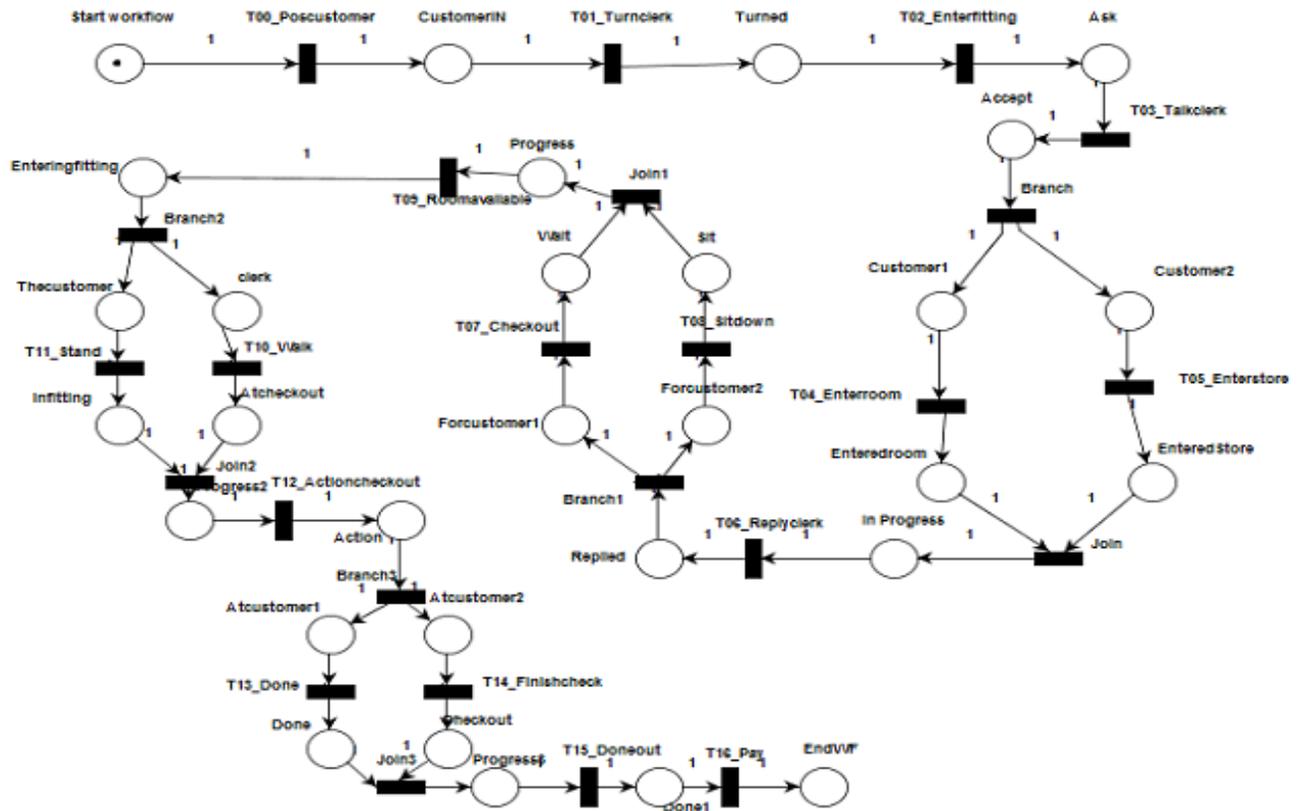


Figure 16: Petri net showing overall workflow

Figure 16 illustrates how the Petri net works and each transition is assigned a number, mn , followed by the name of the task. A Branch means ‘split’ where more than one avatar will have commands to be accomplished. Once all tasks are performed, the Join transition will be activated.

For the snipped code, the function “init” will initialize the transitions that customer1 and customer2 have to accomplish. Both customer1 and customer2 have two separate programs. For example, for customer1 in the “function enterroom”, the avatar will enter the changing room. The function docmd(“say 1313 open”) will let the avatar open and close the door of the changing room. While customer1 is entering the changing room, customer2 will enter the store using the function “enterstore”. He will enter the store, pick up a shirt, and move to the clerk to check for an available changing room. This snipped code illustrates how more than single avatar can accomplish their tasks at the same time.

In this thesis, examples of workflows represented in Petri nets and Lua scripting appear in Appendix B and C respectively.

3.3.3 Layout Configurations Using Workflow Simulation

After implementing the workflow simulations, another goal was to calculate the efficiency of alternative floor layout configurations of the retail store. There could be many measures for a best store layout. For present purposes, the efficiency measure of a layout configuration is defined to be the measured distance and time the avatars need to travel in the different floor plan layouts for a given workflow. This measure optimizes for efficiency of customer shopping and/or clerk movement around the store. Other possible measures might include time spent browsing and ease of finding the items a customer is interested in; dollars

spent per customer; amount of interactions with sales personnel; amount of time customers have to wait on others, for changing rooms and at checkout; whether the customer finds the items they want in the store; and general customer satisfaction with the entire shopping experience so they might return to this store. This thesis only considers the distance measures.

For a given floor plan layout, the objects, shelves, hangers, products, changing rooms, and the check out desk were laid out inside the retail store. Next the workflow simulations for the retail store were executed and the distance and the time each avatar consumed to perform his or her tasks was calculated. In a next step, the layout of the retail store was modified by changing the locations of the objects inside it. The workflow simulations were rerun again and the total distance and time each avatar consumed to finish his tasks re-calculated. The total distances and times were compared in both configurations to judge which configuration is the most efficient. The objective of evaluating alternative layout configurations is to optimize distance for customers, so they do not walk long distance looking for products, or walk long distance to see the sales product.

4. RESULTS AND DISCUSSION

4.1 Results

4.1.1 Results for Workflow Simulation

The basic result is that we can use Second Life to visually simulate a collection of workflows that together describe several representative use cases of activities within a retail store. YouTube videos demonstrating these workflows are available at the following links:

- A combination of Shopping in Retail Store and Waiting for the changing room workflows: http://www.youtube.com/watch?v=_KY5LwFy-4s&feature=player_embedded
- Checking Product Size and Ordering from Another Store: <http://www.youtube.com/watch?v=2bBUCrbd67Y>
- Loss Prevention: <http://www.youtube.com/watch?v=OS4HDefzMJY>
- Shopping and Dining in Retail Store: <http://www.youtube.com/watch?v=yWsiI2-GEew>

A secondary result involves the implementation. Initially, in simulating the “Shopping in the Retail Store” workflow, only Lua scripts were used to represent not just script steps but also the control among steps. Since Lua did not support threads, actions occurred consecutively and each avatar waited until the others finished their tasks to complete its tasks. In this implementation, the workflow required a total of 3:00 minutes to run to completion. After replacing Lua-based control with Petri net-based control, more than one action can occur at the same time and the avatars can work in parallel. The re-implemented workflow required a total of

2:08 minutes to run to completion. Not only faster, the Lua script and Petri net implementation more closely resembles real life scenarios where several actions typically act in parallel.

4.1.2 Results for Evaluating Alternative Configurations in the Retail Store

By selecting the “Shopping in the Retail Store” workflow simulation as an example and attaching the tracking script to each avatar, the X and Y coordinates of each avatar were recorded in an external database during the workflow simulation. Below is a snap shot of the avatar track history database:

	AvatarID	Time	ID	location	X	Y
<input type="checkbox"/>	2	2011-10-28 10:27:49	1037	<106.52599, 41.76902, 28.18526>	106.526	41.769
<input type="checkbox"/>	4	2011-10-28 10:27:55	1038	<118.32614, 45.58671, 28.82196>	118.326	45.5867
<input type="checkbox"/>	4	2011-10-28 10:27:56	1039	<116.31070, 45.99615, 28.13973>	116.311	45.9962
<input type="checkbox"/>	3	2011-10-28 10:27:56	1040	<112.44159, 43.12058, 28.17821>	112.442	43.1206
<input type="checkbox"/>	3	2011-10-28 10:27:57	1041	<109.76779, 41.36260, 28.17821>	109.768	41.3626
<input type="checkbox"/>	3	2011-10-28 10:27:58	1042	<107.15346, 39.64431, 28.17821>	107.153	39.6443
<input type="checkbox"/>	3	2011-10-28 10:27:59	1043	<104.34121, 38.28962, 28.17821>	104.341	38.2896
<input type="checkbox"/>	3	2011-10-28 10:28:00	1044	<103.96254, 41.11449, 28.17820>	103.963	41.1145
<input type="checkbox"/>	4	2011-10-28 10:28:04	1045	<114.90077, 44.37617, 28.18480>	114.901	44.3762

Table 1: Snap Shot of Database

The field “Avatar ID” is an ID assigned to each avatarbot: customer 1 has ID 2, clerk has ID 3, and customer 2 has ID 4. The field “ID” is the primary key of the table. The field “Time” is the time stamp. The field “location” shows the <X, Y, Z> coordinates for the avatars. The field “X” and the field “Y” contain the extracted values from the field “location”.

MATLAB was used to calculate the total distance traveled by each avatar.

In a first store layout configuration, the changing room and the couches were placed at the far end of the store. The total distance travelled by each avatar and the total time were measured as follows (an average of ten runs through the simulation):

Avatar Name	Total Distance	Total Time
Customer 1	83.26 meters	2:00 minutes
Customer 2	101.98 meters	3:00 minutes
Clerk	24.88 meters	2:00 minutes

Table 2: Total distance for first configuration

The store was rearranged so that the changing room and the couches were placed in the middle of the store. The total distance travelled by each avatar and the total time were measured as follows (an average of ten runs through the simulation):

Avatar Name	Total Distance	Total Time
Customer 1	64.67 meters	1:30 minutes
Customer 2	81.23 meters	2:00 minutes
Clerk	16.58 meters	1:30 minutes

Table 3: Total distance for second configuration

The purpose of evaluating alternative configurations in the retail store is to compare different store layouts to see which is best. Assuming that minimizing distance traveled is desirable and ignoring any other criteria, this simulation shows that it is better to place the changing room and the couches in the middle of the store to minimize the distance travelled by each avatar.

4.2 Discussion and Observations

4.2.1 Level of Effort to Build a Domain and a New Workflow

As described in section 3.3.1, one step in designing a new domain is to identify the ingredients (objects) needed. Typically all workflows for a domain can make use of the same family of ingredients.

Level of effort to build domain objects

For each object needed, the object must be designed in Second Life which may involve building the object from primitive 3D graphical objects called prims and then importing textures appropriate for the object. Some objects require scripting so they will behave in a certain manner. Some avatars may require uniforms. For some workflows, they may require animations. In Second Life, the average prim count for objects is around 5. It can take between 5 minutes to an hour or more to create an object depending on the object's complexity. Once an object is created, it can be stored (de-rezzed), accessed (rezzed), and copied instantly. Second Life does not provide much capability for importing object models from outside the environment but does permit sharing or selling objects within that virtual world. This can mean that for a new domain, it is expensive to build the ingredients. But once built, they are easy to move or reuse.

An estimate of the time to construct the retail store's ingredients follows:

- Building the layout of the store 240 min
- Changing room and checkout counter 180 min
- Computers and Keyboards 90 min
- Shelves 120 min

- Shirts 120 min
- Pants 120 min
- Shorts 90 min
- Sunglasses 90 min
- Skirts 120 min
- Bags 90 min
- TOTAL 1260 min

Level of effort to build domain objects

For a given workflow, the level of effort involves building the Petri net and also the Lua scripts. Typically, building a workflow is iterative, a portion is developed, timing is added, and the workflow segment is executed. If something fails or the workflow timing or animations look wrong, then redesign occurs, followed by retesting.

An estimate of the time to construct the retail store's ingredients follows:

- Shopping in Retail Store 420 min
- Waiting for the Changing Room 300 min
- Checking Product Size and Reordering 420 min
- Loss Prevention 480 min
- Shopping and Dining in Retail Store 720 min
- TOTAL 2340 min

Level of effort to rearrange the retail store and change the workflows

For rearranging the retail store, the efforts involved changing the location of the objects inside the store. For example the locations of the changing rooms and the couches were varied. The effort for changing in the workflow involved changing the $\langle X, Y, Z \rangle$ coordinates of the changing room and the couches in the Lua scripts. Also, the $\langle X, Y, Z \rangle$ coordinates for each avatar had to be modified as their waypoints were changed to the new waypoints after rearranging the store.

An estimate of the time to rearrange the retail store and change in the workflows follows:

- Changing the location of products in the retail store 120 min
- Changing in the workflow (Lua script) 180 min
- TOTAL 300 min

Improvements to reduce the time to populate domains with objects

One question we could ask is, are graphical representations of the real world feasible and if so, how complex would they be. While cartoonish, the Second Life / Open Simulator virtual world models can recognizably represent the ordinary human-scale world of trees, buildings, vehicles, and products. Stored in a relational database management system, the entire representation for Open Simulator uses around 25 database tables [34]. Increasingly, gaming technology is creating more and more realistic graphical models. Still, we can speculate that graphical schemes that provide reasonable representations of the real world may not be considerably more complex.

Several improvements to the current system could make it much faster and easier to populate a domain with object models. For domains where 3D models already exist, the time to

build object models is just the time to identify them, “rez” them, and place them. If object models were available in open repositories like Google 3D Warehouse and if suppliers routinely provided 3D models along with their products (which organizations like Wal-Mart or Home Depot could mandate), then it would not be necessary to build models manually. Virtual worlds like Open Simulator already allow cut and paste operations for copying object models from 3D object repositories into and out of that open source virtual world. In this way, the cost to populate a retail store or similar domains might eventually become very small. Furthermore, if future improvements to digital cameras and Kinect-like technologies eventually provide image2model representations, those could also help populate 3D object models.

Capturing routine workflows seems to be the bigger challenge. This thesis demonstrates a manual method. Additional developers could encode many routine workflows, given time. Probably, avatars operated by humans could provide sequences of steps that would be recorded providing a keyboard macro-like recording capability. Technologies like the Kinect might be employed to learn individual activities and sequences of activities [35]. Still, there are open questions related to how to represent workflows, discussed in the next sections.

4.2.2 The Range of Workflows that can happen in a Retail Store

Section 5.1 described five representative workflows that illustrate use cases that can happen in a retail store. To what extent is this a complete set?

Additional Retail Store Workflows

Consider the following additions to this collection:

- an inventory workflow in which the store’s products are logged in a database

- a re-stocking workflow in which the store's products are replenished
- a customer selects items based on customer-specific criteria
- a customer is buying items for someone else so does not use the changing room
- a customer is in a hurry so does not use the changing room
- an item return workflow in which a customer returns an item
- a workflow in which friends shop together and discuss their product selections
- a workflow in which some items are tagged by barcode and others by RFID tag
- a workflow in which a customer becomes angry with a clerk
- a workflow in which a customer cannot find a clerk

These workflows could all be added manually to the current system with modest effort. The inventory and restocking workflows might benefit from an inventory database, an idea explored in an earlier Master's thesis by Alex Ahmad [36]. Some of the workflows might require additional knowledge structures, e.g., a criteria list associated with a customer to identify products they want and a feature list associated with products to identify product attributes, along with a best fit algorithm. The store being out of stock for some items can trigger other workflows like a re-order or re-stock workflow and can affect other knowledge structures like a record of daily sales.

Additional workflows that could take place in retail store

Next consider the following candidate workflows:

- a workflow in which a customer receives a cell phone call while shopping
- a workflow in which two customers discuss the big game or the weather
- a workflow in which one customer bumps into another and says Excuse me.

- a workflow in which a parent brings a small child shopping and the child plays among the racks or cries when the parent disciplines the child.
- a workflow in which a fire at the mall forces evacuation of the store

These workflows may happen in a retail store but are largely incidental to the domain of shopping. They might be part of a general interaction domain that can overlap with shopping.

Next consider the following candidate workflows:

- a customer browses, tries on, browses, tries on, ... and finally four hours later selects among the clothing
- a customer selects every item in the store for purchase
- on Black Friday, when the store opens a hundred avatarbots are waiting at the door to browse and make purchases

These workflows are relevant to retail stores but indicate either that normal shopping behavior is violated by overly repetitive behaviors or they may violate capacity planning extremes – but it is not clear where to draw the line.

With additional thinking, we could list more retail store workflows. Even so, if retail shopping only requires, say, 20-50 workflows, then it may not be daunting to list dozens to hundreds of domains each with tens of workflows to provide an approximation of how many workflows might cover activities of daily living. Potentially, a small army of workflow developers could manually define a useful starting set of workflows to cover many routine activities of daily living.

4.2.3 A Need for Better Workflow Representations and Composition

Our project has made some progress understanding workflows but has identified some additional problems that still need to be resolved.

Some initial problems with workflows were resolved with Petri nets

When our Everything is Alive project first began to explore workflows, the first generation solutions were limited in significant ways. One limitation was that the workflows were written entirely procedurally in a hardcoded brittle manner. If a nurse dropped a scalpel, the doctor probably would not notice and would proceed with the operation without it, including performing incisions! Another limitation was a single linear flow workflow meant that activities happened in sequence, never in parallel. This limitation also meant that one thread of control (the god thread) issued commands to each avatar:

Doctor: Chat “Nurse, please hand me the scalpel.”

Nurse: PickUp scalpel. Animate HoldOutHand.

Nurse: Chat “Doctor, here is the scalpel”

Doctor: PickUp scalpel.

That is, the central workflow issues commands to avatars which executed each in turn. The avatars did not contain their own flows of control. Another limitation was that there were no conditional steps and alternate paths through the workflow. With the introduction of Petri nets, we were able to support parallel and conditional workflows.

Some problems with workflows that this thesis helped improve our understanding

This thesis explored some additional aspects of workflow design:

- the thesis provides an idea of how complicated a domain is in terms of number of workflows and in terms of effort to build domain objects and workflows
- this thesis illustrates how workflows can be used to represent exceptional situations in which some workflow step fails and some “fix” is required, like waiting for the changing room to be free or forgetting to pay for an item before leaving the store.
- this thesis provided a way to log avatar track histories and how to use that information to analyze alternative floor plan layouts for the retail store.

Improvements still needed to our understanding of workflows

Some additional limitations, not yet resolved, still limit our workflows:

- Some steps in our workflows are brittle. For instance, the customers do not really engage in unrestricted browsing; rather they browse to specific apparel and purchase those items. Also, the customer never takes a variable number of products into the changing room. Also, the customer always takes the sunglasses without paying, never another product. Turning workflow steps into sub-workflows would solve this problem at the expense of spending more time refining individual actions (not our current highest priority).
- Our Petri net program does not yet support subprocedures. If an avatarbot made a cake, it would be more modular to define workflows for the cake and frosting separately.
- Our workflow representations only record leaf level observable objects and behaviors but do not record the reasons for actions – the why. A deeper planning representation would be more useful as a deeper representation for reasoning about situations and identifying new solutions based on rules. One should be able to ask an avatarbot why they did some

action and determine the higher level objective. One might also associate English descriptions with each avatar step to be able to explain or describe the steps.

- In our current workflows logs, we just record avatar locations over time. We could record additional information, including the entire trace of each action in the workflow. This log could later be queried and analyzed to provide “after action” reports useful in comparing different workflows in areas beyond alternative layout configurations.

Need for a Design Theory for Workflows

In the database area, there is a theory based on functional dependencies for modeling tables. That theory tells when to split a complex table into subtables. (It is interesting to note that we are unaware of any similar theory for the design of objects in an object oriented program though one might expect the results from the database might carry over.)

It would seem that the workflow area might also be subject to a design theory. It appears that workflows are composable: workflows can be conjoined sequentially and one workflow can be embedded with the step of another. This is like the rules of structured programming. If workflow steps were further defined to have pre- and post-condition, then a follow-on workflow could only occur if a predecessor workflow had satisfied the preconditions of the follow-on workflow. Also, workflows seem similar to threads in that the same avatarbot can progress multiple threads by time-division multiplexing so that steps in one workflow could be interleaved with steps in another workflow and, if an actor needed to perform a next step, they could select from any next steps available. This would explain how a doctor who asks for the scalpel might then ask the anesthesiologist if he had played golf on the weekend.

Another way to consider how to grow a complex of workflows is to consider a large collection of traces of workflows. Individual workflows might be viewed as sentences in a language where the terminals were observable actions of a given workflow. Then, a grammar induction step might attempt to build a collection of workflows from the sentences.

Along the same lines, it appears that a simple workflow like our Workflow #1 accounts for a typical path that involves going to a store to purchase items (by selecting them, trying them on, and buying them). Then considering each step, we can find exceptions (like Workflows #2, 3, and 4) that must be accounted for. That is, if something goes wrong in the main workflow, then alternative workflows are added to correct the situation.

The above observations are not yet a design theory of workflows but do constitute some thoughts on how to begin to develop such a theory.

Simulations in a Virtual World – are they the best approach?

In section 3.3.4, we compared two floor plan layouts in the virtual world and found one improved on the other in terms of distance traveled and time spent within the simulation. It is important to ask – is there a better way to discover this result?

It certainly seems that building alternate simulations in a virtual world is cheaper than building them in the real world.

With a few minutes additional reflection, we observe that a virtual world simulation mirrors the real world, but possibly not abstracting as much as might be useful to derive a comparison of alternative floor plan layouts. For instance, a graph algorithm that abstracts physical space might not be better, or a spreadsheet, or a closed form analytic solution that aimed at providing an optimal placement. Each of these might be constructed in a fraction of the time it

might take to reach the same conclusion we reached. This is an entirely valid comparison. It may be that a board of directors or a homeowner could visualize alternatives better in a virtual world, but it may also be that it is a somewhat expensive way to get specific answers. The jury is still out on just when a 3D virtual world simulation is appropriate in comparing alternative layouts.

5. CONCLUSIONS

5.1 Summary

In section 1.3, we listed objectives of this thesis. Below, we indicate how we accomplished these objectives:

A first objective of the thesis was to model workflows using a 3D virtual world to understand how to represent workflows visually. In section 3.1 we described the visualization of the workflows in Second Life. In section 3.2 we described a methodology for modeling workflows. In section 3.3 we described how to represent the workflows using Petri nets and Lua scripts.

A second objective of the thesis was to assess how complex the collection of workflows covering common retail activities is and understand the level of effort needed to build new workflows. In section 4.2.1 we discussed the level of effort to build workflows to cover the retail store domain.

A third objective of the thesis was to define and execute a representative collection of inter-related workflow simulations in one domain (retail sales was selected) to understand how many workflows might “cover” the domain. In section 3.1, we described the representative selection of workflows, and we provided more detail in Appendices A, B and C. In section 4.2.2, we discussed the range of workflows that can happen in a retail store.

A fourth objective of the thesis was to begin to understand some elements of a workflow design theory including how workflow relates to planning, exception handling and options. In section 4.2.3 we discussed some initial ideas on a design theory for workflows.

A fifth objective of the thesis was to determine the relative cost of one workflow over another to determine how virtual world workflow simulations can be used to simulate, measure differences, and compare alternative store layout designs as an approach to helping to design a layout using a tool that end-users would readily understand. In section 3.2.3 we discussed evaluating alternative configurations in the retail store. In section 4.2.3 we discussed that a workflow log could contain all actions in the workflow, available for additional analyses.

5.2 Potential Impact

At present, computers do not have extensive representations for common sense knowledge. A workflow is an abstracted model of real world behavior meant to capture steps in everyday activities. If computers can be told or can learn how to represent activities of daily living, that is, the routine activities that humans engage in, then computers can become more useful in our lives. Already, GPS systems that know where we are going can plan routes for us and guide us along those routes. If computers track buses and know that we want to catch a bus, they can advise us when to leave work to rendezvous with the bus. If a computer can watch a heart catheterization operation and knows what to expect, they could monitor what step was ongoing and what step to expect next and generate detailed descriptions of that particular operation that could be used in after action reports or analysis to improve patient survivability. Or they could train novice physicians in what to do in exceptional cases when something goes wrong.

5.3 Future Work

The following list describes enhancements needed to better improve our understanding of workflows:

- Our present Petri net implementation does not include a subprocess facility and instead models all workflows in a single large workflow. A subprocess capability is needed. In addition, we need a way to parameterize workflows.
- We need to deconstruct complex workflows into operators with preconditions and develop a planner that use goals to reason about situations. Some previously solved plans are likely to be like our current workflows.
- We need to understand how the system can observe a step or a few steps in a workflow to recognize which existing workflow is being executed.
- We need to understand how a new user can be trained to learn a workflow. How can the workflow be modified to help guide a new user especially when exceptions occur.
- Another enhancement involves the recording of a whole collection of workflows in a repository and later using “after action” reporting to replay specific scenarios or use data mining to find “interesting events” in a workflow.
- Another enhancement would be to recognize partial workflow sequences and induce that the activities are part of higher level workflows. Often, at a glance, we can determine that someone is brushing their teeth or teaching in a classroom or shopping.
- Another enhancement is to understand what breaks when we increase the number of avatars to tens or hundreds of avatars in the same workflow.

- In this work we optimized only the distance and the time consumed by each avatar. A future enhancement can investigate alternative objective functions such as exposure of avatars to more items in the store and the optimum location for advertisements.

REFERENCES

- [1] C. Thompson, "Virtual World Architectures," Guest Editor Introduction, Special Issue on Virtual World Architectures, *IEEE Internet Computing*, Sept/Oct 2011, Authored Call for Papers which appeared in *IEEE Internet Computing*, July/August 2010.
- [2] A. Eguchi, C. Thompson, "Towards a Semantic World: Smart Objects in a Virtual World," *International Journal of Computer Information Systems and Industrial Management Applications (IJCISIM)*, Vol 3, 2011, pp 905-911. URL: <http://www.mirlabs.org/ijcisim>
- [3] Josh Eno, Craig Thompson, "Virtual and Real-World Ontology Services," Special Issue on Virtual World Architectures, *IEEE Internet Computing*, Sept/Oct 2011.
- [4] A. Eguchi, "Object Recognition based on Shape and Function," B.S. Thesis, Computer Science and Computer Engineering Department, University of Arkansas, Fayetteville, December 2011.
- [5] K. Perkins, C. Thompson, "Workflow in a Virtual World," *Proceedings of the X10 Workshop on Extensible Virtual Worlds*, IBM Island in Second Life, March 29-30, 2010. URL: <http://vw.ddns.uark.edu/X10/content/X10--papers.htm>.
- [6] K. Perkins, "Workflow Simulation in a Virtual World," Master's Thesis, Computer Science and Computer Engineering Department, University of Arkansas, Fayetteville, May 2011.
- [7] B. Coppin, *Artificial Intelligence Illuminated*, Jones/Bartlett Publishers, Sudbury, MA, 2004.
- [8] Scripts, Accessed September, 2011, URL: [http://en.wikipedia.org/wiki/Scripts_\(artificial_intelligence\)](http://en.wikipedia.org/wiki/Scripts_(artificial_intelligence))
- [9] Activities of Daily Living, Accessed September, 2011, URL: http://en.wikipedia.org/wiki/Activities_of_daily_living
- [10] Measuring Activities of Daily Living, Accessed September, 2011, URL: <http://aspe.hhs.gov/daltcp/reports/meacmpes.htm>.
- [11] A. Al-Shiha, S. Dlay, and W. Woo, "Design an Artificial Mirror System as an Interaction Method for the Life Logging System," *Proceedings of the 2010 International Conference on User Science and Engineering (i-USER)*, pp. 128-132, Shah Alam, Malaysia, December 2010.

- [12] S. Park and H. Kautz, "Hierarchical Recognition of Activities of Daily Living using Multi-Scale, Multi-Perspective Vision and RFID," *Proceedings of the Fourth International Conference on Intelligent Environments (IE 08)*, Seattle, Washington, July 2008.
- [13] W. van der Aalst, "Making Work Flow: On the Application of Petri Nets to Business Process Management," *Proceedings of the 23rd International Conference on the Application and Theory of Petri Nets*, Adelaide, Australia, June 2002. URL: <http://books.google.com/books?id=QvATm4v3zoYC>
- [14] R. Brown, A. Lim, Y. Wong, S. Heng, and D. Wallace. "Gameplay Workflow: A Distributed Game Control Approach," *Proceedings of the 2006 International Conference on Game Research and Development (CyberGames '06)*, Murdoch University, Australia, 2006, pp. 207-214.
- [15] C. Thompson, D. Korsmeyer, "Internet Access to Scientific Data," *Special Issue of IEEE Internet Computing*, Volume 9, Number 1, January-February 2005. pp. 17-19.
- [16] Workflow Terminology and Glossary, Doc. No. WFMC-TC-1011, Version 3.0, Workflow Management Coalition, 1999, Accessed September, 2011, URL http://www.wfmc.org/standards/docs/TC-1011_term_glossary_v3.pdf.
- [17] Yet Another Workflow Language (YAWL), URL: <http://en.wikipedia.org/wiki/YAWL>
- [18] Business Process Execution Language, URL: <http://en.wikipedia.org/wiki/BPEL>
- [19] XML Process Definition Language (XPDL), URL: <http://en.wikipedia.org/wiki/XPDL>
- [20] Workflow, Accessed September, 2011, URL: <http://en.wikipedia.org/wiki/Workflow>.
- [21] Petri nets, Accessed September, 2011, URL: <http://www.labri.fr/perso/anca/FDS/Pn-ESTII.pdf>.
- [22] Petri net, Accessed September, 2011, URL: http://en.wikipedia.org/wiki/Petri_net.
- [23] Routing within a Petri net, Accessed September, 2011, URL: <http://www.tonymarston.net/php-mysql/workflow.html>.
- [24] Virtual World, Accessed September, 2011, URL: http://en.wikipedia.org/wiki/Virtual_world.

- [25] What is a Virtual World, Accessed September, 2011, URL: <http://www.wisegeek.com/what-is-a-virtual-world.htm>.
- [26] What is a Virtual World, Accessed September, 2011, URL: <http://www.virtualworldsreview.com/info/whatis.shtml>.
- [27] Seven Things You Should Know About Second Life, Accessed September, 2011, URL: <http://www.educause.edu/ELI/7ThingsYouShouldKnowAboutSecon/163004>.
- [28] Second Life, Accessed September, 2011, URL: http://en.wikipedia.org/wiki/Second_Life.
- [29] Server Architecture – Second Life Wiki, Accessed September, 2011, URL: http://wiki.secondlife.com/wiki/Server_architecture.
- [30] N. Farrer, "Second Life Robot Command Language," Tech Report, CSCE Department, University of Arkansas, Fayetteville, AR, 22 February 2009. URL: <http://vw.ddns.uark.edu/content/2009-02--SL-Robot-Command-Language-v0--Nicholas-Farrer.doc>
- [31] About Lua, Accessed September, 2011, URL: <http://www.lua.org/about.html>.
- [32] Lua 5.1 Reference Manual, Accessed September, 2011, URL: <http://www.lua.org/manual/5.1/manual.html>.
- [33] M. Kifer, A. Bernstein, and P. Lewis, *Database Systems: An Application-Oriented Approach*, 2nd Edition, Addison Wesley, 2006.
- [34] C. Dempewolf, G. Stafford, S. Williams, "Mapping the OpenSimulator Architecture," Term Project for CSCE 5013 Software Architecture, Computer Science and Computer Engineering Department, University of Arkansas, Fayetteville, AR, Spring 2011.
- [35] A. Eguchi, "Object Recognition Based on Shape and Function," B.S. Honors Thesis, Computer Science and Computer Engineering Department, University of Arkansas, Fayetteville, AR, December 2011.
- [36] A. Ahmad, "Modeling Healthcare Logistics in a Virtual World: the Database Connection," M.S. Thesis, CSCE Dept., University of Arkansas, May 23, 2009.

APPENDIX A – RETAIL SALES WORKFLOWS DESCRIBED IN ENGLISH

Following is the English text describing the retail workflows. It describes the scene, ingredients, roles, and the sequence of workflow tasks.

Workflow domain:
Retail

Scene:
Retail Store

Roles:
Sales Clerk 1 ;aids shopper, completes purchase transaction
Sales Clerk 1 ;serves food, completes payment
Shopper 1 ;browses, may try on apparel, may purchase
Shopper 2 ;browses, may try on apparel, may purchase
Shopper 3 ;browses food menu, may eat, pay

Equipment:
Store Layout
Clothing Racks
Clothing of various kinds including shirts, pants, sunglasses
Changing Room with door
Checkout Counter
Cash Register
Waiting Area (couches)
Dining Area
Food of various kinds
Trays for Serving Food
Dining Cash Register

Workflow #1 - Shopping in Retail Store

- 1) Shopper 1 enters Store, browses products, and pick up shirt
- 2) Shopper 1 asks sales clerk about the changing room
- 3) Sales clerk tells shopper 1 changing room is available and he can enter
- 4) Shopper 1 enters changing room and tries the shirt
- 5) Shopper 1 leaves changing room and moves to check out and drops the shirt
- 6) Sales clerk moves to the cash register
- 7) Sales clerk asks shopper 1 if that's all for him today
- 8) Shopper 1 replies saying yes thanks
- 9) Shopper 1 takes the bag and leaves the store

Workflow #2 - Waiting for the Changing Room

- 1) While step 3 and 4 are executing in Workflow #1, Shopper 2 enters Store, browses products, and picks up shirt
- 2) Shopper 2 asks sales clerk about the changing room
- 3) Sales clerk tells shopper 2 the changing room is not available now, have a seat and he will let him know when it is available
- 4) Shopper 2 sits on couch

- 5) When Shopper 1 leaves changing room, Sales clerk tells shopper 2 the changing room is available
- 6) Shopper 2 enters changing room and tries the shirt
- 7) Shopper 2 leaves changing room and moves to check out and drops the shirt
- 8) Sales clerk asks shopper 2 if she finds everything all right
- 9) Shopper 2 replies saying yes thanks
- 10) Shopper 2 takes the bag and leaves the store

Workflow #3 - Checking Product Size

- 1) Shopper 1 enters Store, browses products and pick up shirt
- 2) Shopper 1 moves to cash register and asks sales clerk for his shirt size
- 3) Sales clerk takes the shirt and search for shopper 1 size
- 4) Sales clerk return back to shopper 1 and tells him that he don't have his size
- 5) Sales clerk offers to check for shopper 1 size in another store
- 6) Shopper 1 tells sales clerk yes please
- 7) Sales clerk finds the shirt in another store and reserves it for shopper 1
- 8) Shopper 1 thanks sales clerk and leaves the store

Workflow #4 - Loss Prevention

- 1) Shopper 1 enters Store
- 2) Sales clerk welcomes shopper 1
- 3) Shopper 1 asks sales clerk for shirts and pants
- 4) Sales clerk shows shopper 1 their location
- 5) Shopper 1 browses products and pick up shirt , sunglasses and pants
- 6) Shopper 1 asks sales clerk for changing room
- 7) Sales clerk shows shopper 1 the changing room
- 8) Shopper 1 enters changing room and tries the shirt and the pants
- 9) Sales clerk moves to cash register
- 10) Shopper 1 moves to check out and drop shirt and pants
- 11) Sales clerk takes money from shopper 1
- 12) Shopper 1 takes the bag and leaves Store
- 13) While leaving the store, the sensor on the doors of the store will beep
- 14) Shopper 1 waits for sales clerk to check his reset
- 15) Sales clerk tells shopper 1 that he forgot to pay for sunglasses
- 16) Shopper 1 returns to cash register with sales clerk and pays for sunglasses
- 17) Shopper 1 leaves the store

Workflow #5 - Shopping and Dining in Retail Store

- 1) Shopper 1 enters Store, moves to food corner and have a seat
- 2) Sales clerk 1 tells shopper 1 the menu of the day and ask him for his order
- 3) Sales clerk 1 takes the order and starts preparing the food
- 4) While step 3, Shopper 2 enters Store, browses products and pick up shirt
- 5) Shopper 2 asks sales clerk 2 about the changing room
- 6) Sales clerk 2 tells shopper 2 changing room is available and he can enter
- 7) While step 6, sales clerk 1 serve the food to shopper 1

- 8) Shopper 1 starts eating the food
- 9) Shopper 2 enters changing room and tries the shirt
- 10) While step 6 through step 7 are executing, Shopper 3 enters Store, browses products and pick up shirt
- 11) Shopper 3 asks sales clerk 2 about the changing room
- 12) Sales clerk 2 tells shopper 3 the changing room is not available now, have a seat and he will let him know when it is available
- 13) Shopper 3 sits on couch
- 14) While step 12 and 13 are executing, sales clerk 1 takes the empty tray and charge shopper 1
- 15) Shopper 1 pays the reset and leaves the store
- 16) Shopper 2 moves to check out and drops the shirt
- 17) Sales clerk 2 tells shopper 3 the changing room is available
- 18) Shopper 3 enters changing room and tries the shirt
- 19) Sales clerk 2 moves to the cash register
- 20) Sales clerk 2 asks shopper 2 if that's all for him today
- 21) Shopper 2 replies saying yes thanks
- 22) Shopper 2 takes the bag and leaves the store
- 23) Shopper 3 moves to check out and drops the shirt
- 24) Sales clerk 2 asks shopper 3 if he finds everything all right
- 25) Shopper 3 replies saying yes thanks
- 26) Shopper 3 takes the bag and leaves the store

APPENDIX B – RETAIL SALES WORKFLOW XML FILES

For the Retail Sales workflows, the main configuration file is *retailwf.xml* and the Petri net definition file is *retailwf_PN.xml* for shopping in Retail Store. They are listed below. See Figure 16 for a graphical representation of this Petri net. Note: *retailwf_PN.xml* has been stripped of all elements that are used only by the Pipe2 editor, and contains only the elements needed for the Petri net interpreter.

retailwf.xml

```
<workflow>
  <wfname>RetailWF</wfname>
  <initfunc>init</initfunc>
  <petrinet>
    <filename>retailwf_PN.xml</filename>
    <lastplace>EndWF</lastplace>
    <lastplacecnt>1</lastplacecnt>
    <lastplaceoper>ge</lastplaceoper>
    <logoutfinished>>true</logoutfinished>
  </petrinet>
  <bots count="3">
    <bot>
      <fname>mayzein</fname>
      <lname>Resident</lname>
      <passw>password</passw>
      <role>Customer1</role>
      <startloc>116.83 61.56 28.07</startloc>
    </bot>
    <bot>
      <fname>clerk123</fname>
      <lname>Resident</lname>
      <passw>password</passw>
      <role>Clerk</role>
      <startloc>115.81 30.85 28.11</startloc>
    </bot>
    <bot>
      <fname>customer123</fname>
      <lname>Resident</lname>
      <passw>password</passw>
      <role>Customer2</role>
      <startloc>101.75 51.32 28.10</startloc>
    </bot>
  </bots>
</workflow>
```

retailwf_PN.xml

```
<?xml version="1.0" encoding="iso-8859-1"?>
<pnml>
  <net id="Net-One" type="P/T net">
    <tokenclass id="Default" enabled="true" red="0" green="0" blue="0"/>
    <place id="Accept">
      <graphics>
        <position x="645.0" y="120.0"/>
      </graphics>
      <name>
        <value>Accept</value>
      <graphics>
        <offset x="32.0" y="-11.0"/>
      </graphics>
    </name>
    <initialMarking>
      <value>Default,0</value>
      <graphics>
        <offset x="0.0" y="0.0"/>
      </graphics>
    </initialMarking>
    <capacity>
      <value>0</value>
    </capacity>
  </place>
  <place id="Action">
    <graphics>
      <position x="255.0" y="420.0"/>
    </graphics>
    <name>
      <value>Action</value>
      <graphics>
        <offset x="7.0" y="35.0"/>
      </graphics>
    </name>
    <initialMarking>
      <value>Default,0</value>
      <graphics>
        <offset x="0.0" y="0.0"/>
      </graphics>
    </initialMarking>
    <capacity>
      <value>0</value>
    </capacity>
  </place>
</net>
</pnml>
```

```

</place>
<place id="Ask">
  <graphics>
    <position x="705.0" y="60.0"/>
  </graphics>
  <name>
    <value>Ask</value>
    <graphics>
      <offset x="26.0" y="-14.0"/>
    </graphics>
  </name>
  <initialMarking>
    <value>Default,0</value>
    <graphics>
      <offset x="0.0" y="0.0"/>
    </graphics>
  </initialMarking>
  <capacity>
    <value>0</value>
  </capacity>
</place>
<place id="Atcheckout">
  <graphics>
    <position x="150.0" y="345.0"/>
  </graphics>
  <name>
    <value>Atcheckout</value>
    <graphics>
      <offset x="74.0" y="-3.0"/>
    </graphics>
  </name>
  <initialMarking>
    <value>Default,0</value>
    <graphics>
      <offset x="0.0" y="0.0"/>
    </graphics>
  </initialMarking>
  <capacity>
    <value>0</value>
  </capacity>
</place>
<place id="Atcustomer1">
  <graphics>
    <position x="210.0" y="510.0"/>
  </graphics>

```

```

<name>
  <value>Atcustomer1</value>
  <graphics>
    <offset x="12.0" y="0.0"/>
  </graphics>
</name>
<initialMarking>
  <value>Default,0</value>
  <graphics>
    <offset x="0.0" y="0.0"/>
  </graphics>
</initialMarking>
<capacity>
  <value>0</value>
</capacity>
</place>
<place id="Atcustomer2">
  <graphics>
    <position x="285.0" y="510.0"/>
  </graphics>
  <name>
    <value>Atcustomer2</value>
    <graphics>
      <offset x="67.0" y="-7.0"/>
    </graphics>
  </name>
  <initialMarking>
    <value>Default,0</value>
    <graphics>
      <offset x="0.0" y="0.0"/>
    </graphics>
  </initialMarking>
  <capacity>
    <value>0</value>
  </capacity>
</place>
<place id="Checkout">
  <graphics>
    <position x="285.0" y="600.0"/>
  </graphics>
  <name>
    <value>Checkout</value>
    <graphics>
      <offset x="59.0" y="-1.0"/>
    </graphics>
  </name>

```

```

</name>
<initialMarking>
  <value>Default,0</value>
  <graphics>
    <offset x="0.0" y="0.0"/>
  </graphics>
</initialMarking>
<capacity>
  <value>0</value>
</capacity>
</place>
<place id="clerk">
  <graphics>
    <position x="150.0" y="255.0"/>
  </graphics>
  <name>
    <value>clerk</value>
  <graphics>
    <offset x="41.0" y="-5.0"/>
  </graphics>
</name>
<initialMarking>
  <value>Default,0</value>
  <graphics>
    <offset x="0.0" y="0.0"/>
  </graphics>
</initialMarking>
<capacity>
  <value>0</value>
</capacity>
</place>
<place id="Customer1">
  <graphics>
    <position x="600.0" y="255.0"/>
  </graphics>
  <name>
    <value>Customer1</value>
  <graphics>
    <offset x="23.0" y="-12.0"/>
  </graphics>
</name>
<initialMarking>
  <value>Default,0</value>
  <graphics>
    <offset x="0.0" y="0.0"/>

```

```

    </graphics>
  </initialMarking>
  <capacity>
    <value>0</value>
  </capacity>
</place>
<place id="Customer2">
  <graphics>
    <position x="720.0" y="255.0"/>
  </graphics>
  <name>
    <value>Customer2</value>
  <graphics>
    <offset x="86.0" y="-9.0"/>
  </graphics>
</name>
  <initialMarking>
    <value>Default,0</value>
  <graphics>
    <offset x="0.0" y="0.0"/>
  </graphics>
</initialMarking>
  <capacity>
    <value>0</value>
  </capacity>
</place>
<place id="CustomerIN">
  <graphics>
    <position x="285.0" y="60.0"/>
  </graphics>
  <name>
    <value>CustomerIN</value>
  <graphics>
    <offset x="55.0" y="-15.0"/>
  </graphics>
</name>
  <initialMarking>
    <value>Default,0</value>
  <graphics>
    <offset x="0.0" y="0.0"/>
  </graphics>
</initialMarking>
  <capacity>
    <value>0</value>
  </capacity>

```

```

</place>
<place id="Done">
  <graphics>
    <position x="210.0" y="600.0"/>
  </graphics>
  <name>
    <value>Done</value>
    <graphics>
      <offset x="5.0" y="3.0"/>
    </graphics>
  </name>
  <initialMarking>
    <value>Default,0</value>
    <graphics>
      <offset x="0.0" y="0.0"/>
    </graphics>
  </initialMarking>
  <capacity>
    <value>0</value>
  </capacity>
</place>
<place id="Done1">
  <graphics>
    <position x="420.0" y="630.0"/>
  </graphics>
  <name>
    <value>Done1</value>
    <graphics>
      <offset x="50.0" y="38.0"/>
    </graphics>
  </name>
  <initialMarking>
    <value>Default,0</value>
    <graphics>
      <offset x="0.0" y="0.0"/>
    </graphics>
  </initialMarking>
  <capacity>
    <value>0</value>
  </capacity>
</place>
<place id="EndWF">
  <graphics>
    <position x="525.0" y="630.0"/>
  </graphics>

```

```

<name>
  <value>EndWF</value>
  <graphics>
    <offset x="41.0" y="-8.0"/>
  </graphics>
</name>
<initialMarking>
  <value>Default,0</value>
  <graphics>
    <offset x="0.0" y="0.0"/>
  </graphics>
</initialMarking>
<capacity>
  <value>0</value>
</capacity>
</place>
<place id="Enteredroom">
  <graphics>
    <position x="600.0" y="390.0"/>
  </graphics>
  <name>
    <value>Enteredroom</value>
    <graphics>
      <offset x="9.0" y="-1.0"/>
    </graphics>
  </name>
  <initialMarking>
    <value>Default,0</value>
    <graphics>
      <offset x="0.0" y="0.0"/>
    </graphics>
  </initialMarking>
  <capacity>
    <value>0</value>
  </capacity>
</place>
<place id="EnteredStore">
  <graphics>
    <position x="735.0" y="390.0"/>
  </graphics>
  <name>
    <value>EnteredStore</value>
    <graphics>
      <offset x="83.0" y="-5.0"/>
    </graphics>
  </name>
  <initialMarking>
    <value>Default,0</value>
    <graphics>
      <offset x="0.0" y="0.0"/>
    </graphics>
  </initialMarking>
  <capacity>
    <value>0</value>
  </capacity>
</place>

```

```

</name>
<initialMarking>
  <value>Default,0</value>
  <graphics>
    <offset x="0.0" y="0.0"/>
  </graphics>
</initialMarking>
<capacity>
  <value>0</value>
</capacity>
</place>
<place id="Enteringfitting">
  <graphics>
    <position x="90.0" y="150.0"/>
  </graphics>
  <name>
    <value>Enteringfitting</value>
  <graphics>
    <offset x="3.0" y="13.0"/>
  </graphics>
</name>
<initialMarking>
  <value>Default,0</value>
  <graphics>
    <offset x="0.0" y="0.0"/>
  </graphics>
</initialMarking>
<capacity>
  <value>0</value>
</capacity>
</place>
<place id="Forcustomer1">
  <graphics>
    <position x="360.0" y="345.0"/>
  </graphics>
  <name>
    <value>Forcustomer1</value>
  <graphics>
    <offset x="2.0" y="20.0"/>
  </graphics>
</name>
<initialMarking>
  <value>Default,0</value>
  <graphics>
    <offset x="0.0" y="0.0"/>

```

```

    </graphics>
  </initialMarking>
  <capacity>
    <value>0</value>
  </capacity>
</place>
<place id="Forcustomer2">
  <graphics>
    <position x="450.0" y="345.0"/>
  </graphics>
  <name>
    <value>Forcustomer2</value>
  <graphics>
    <offset x="54.0" y="-7.0"/>
  </graphics>
</name>
  <initialMarking>
    <value>Default,0</value>
  <graphics>
    <offset x="0.0" y="0.0"/>
  </graphics>
</initialMarking>
  <capacity>
    <value>0</value>
  </capacity>
</place>
<place id="Infitting">
  <graphics>
    <position x="75.0" y="345.0"/>
  </graphics>
  <name>
    <value>Infitting</value>
  <graphics>
    <offset x="1.0" y="1.0"/>
  </graphics>
</name>
  <initialMarking>
    <value>Default,0</value>
  <graphics>
    <offset x="0.0" y="0.0"/>
  </graphics>
</initialMarking>
  <capacity>
    <value>0</value>
  </capacity>

```

```

</place>
<place id="In Progress">
  <graphics>
    <position x="570.0" y="450.0"/>
  </graphics>
  <name>
    <value>In Progress</value>
    <graphics>
      <offset x="51.0" y="-8.0"/>
    </graphics>
  </name>
  <initialMarking>
    <value>Default,0</value>
    <graphics>
      <offset x="0.0" y="0.0"/>
    </graphics>
  </initialMarking>
  <capacity>
    <value>0</value>
  </capacity>
</place>
<place id="Progress">
  <graphics>
    <position x="345.0" y="150.0"/>
  </graphics>
  <name>
    <value>Progress</value>
    <graphics>
      <offset x="39.0" y="-5.0"/>
    </graphics>
  </name>
  <initialMarking>
    <value>Default,0</value>
    <graphics>
      <offset x="0.0" y="0.0"/>
    </graphics>
  </initialMarking>
  <capacity>
    <value>0</value>
  </capacity>
</place>
<place id="Progress2">
  <graphics>
    <position x="120.0" y="420.0"/>
  </graphics>

```

```

<name>
  <value>Progress2</value>
  <graphics>
    <offset x="68.0" y="-5.0"/>
  </graphics>
</name>
<initialMarking>
  <value>Default,0</value>
  <graphics>
    <offset x="0.0" y="0.0"/>
  </graphics>
</initialMarking>
<capacity>
  <value>0</value>
</capacity>
</place>
<place id="Progress3">
  <graphics>
    <position x="315.0" y="630.0"/>
  </graphics>
  <name>
    <value>Progress3</value>
    <graphics>
      <offset x="51.0" y="0.0"/>
    </graphics>
  </name>
  <initialMarking>
    <value>Default,0</value>
    <graphics>
      <offset x="0.0" y="0.0"/>
    </graphics>
  </initialMarking>
  <capacity>
    <value>0</value>
  </capacity>
</place>
<place id="Replied">
  <graphics>
    <position x="420.0" y="450.0"/>
  </graphics>
  <name>
    <value>Replied</value>
    <graphics>
      <offset x="-3.0" y="21.0"/>
    </graphics>
  </name>

```

```

</name>
<initialMarking>
  <value>Default,0</value>
  <graphics>
    <offset x="0.0" y="0.0"/>
  </graphics>
</initialMarking>
<capacity>
  <value>0</value>
</capacity>
</place>
<place id="Sit">
  <graphics>
    <position x="450.0" y="225.0"/>
  </graphics>
  <name>
    <value>Sit</value>
  <graphics>
    <offset x="24.0" y="-6.0"/>
  </graphics>
</name>
<initialMarking>
  <value>Default,0</value>
  <graphics>
    <offset x="0.0" y="0.0"/>
  </graphics>
</initialMarking>
<capacity>
  <value>0</value>
</capacity>
</place>
<place id="Start workflow">
  <graphics>
    <position x="75.0" y="60.0"/>
  </graphics>
  <name>
    <value>Start workflow</value>
  <graphics>
    <offset x="54.0" y="-15.0"/>
  </graphics>
</name>
<initialMarking>
  <value>Default,1</value>
  <graphics>
    <offset x="0.0" y="0.0"/>

```

```

    </graphics>
  </initialMarking>
  <capacity>
    <value>0</value>
  </capacity>
</place>
<place id="Thecustomer">
  <graphics>
    <position x="75.0" y="255.0"/>
  </graphics>
  <name>
    <value>Thecustomer</value>
  <graphics>
    <offset x="14.0" y="3.0"/>
  </graphics>
</name>
  <initialMarking>
    <value>Default,0</value>
  <graphics>
    <offset x="0.0" y="0.0"/>
  </graphics>
</initialMarking>
  <capacity>
    <value>0</value>
  </capacity>
</place>
<place id="Turned">
  <graphics>
    <position x="495.0" y="60.0"/>
  </graphics>
  <name>
    <value>Turned</value>
  <graphics>
    <offset x="25.0" y="-15.0"/>
  </graphics>
</name>
  <initialMarking>
    <value>Default,0</value>
  <graphics>
    <offset x="0.0" y="0.0"/>
  </graphics>
</initialMarking>
  <capacity>
    <value>0</value>
  </capacity>

```

```

</place>
<place id="Wait">
  <graphics>
    <position x="360.0" y="225.0"/>
  </graphics>
  <name>
    <value>Wait</value>
    <graphics>
      <offset x="22.0" y="-7.0"/>
    </graphics>
  </name>
  <initialMarking>
    <value>Default,0</value>
    <graphics>
      <offset x="0.0" y="0.0"/>
    </graphics>
  </initialMarking>
  <capacity>
    <value>0</value>
  </capacity>
</place>
<transition id="Branch">
  <graphics>
    <position x="645.0" y="180.0"/>
  </graphics>
  <name>
    <value>Branch</value>
    <graphics>
      <offset x="58.0" y="0.0"/>
    </graphics>
  </name>
  <orientation>
    <value>90</value>
  </orientation>
  <rate>
    <value>1.0</value>
  </rate>
  <timed>
    <value>>false</value>
  </timed>
  <infiniteServer>
    <value>>false</value>
  </infiniteServer>
  <priority>
    <value>1</value>
  </priority>

```

```

    </priority>
  </transition>
  <transition id="Branch1">
    <graphics>
      <position x="420.0" y="390.0"/>
    </graphics>
    <name>
      <value>Branch1</value>
    <graphics>
      <offset x="-5.0" y="35.0"/>
    </graphics>
    </name>
    <orientation>
      <value>90</value>
    </orientation>
    <rate>
      <value>1.0</value>
    </rate>
    <timed>
      <value>>false</value>
    </timed>
    <infiniteServer>
      <value>>false</value>
    </infiniteServer>
    <priority>
      <value>1</value>
    </priority>
  </transition>
  <transition id="Branch2">
    <graphics>
      <position x="90.0" y="195.0"/>
    </graphics>
    <name>
      <value>Branch2</value>
    <graphics>
      <offset x="70.0" y="9.0"/>
    </graphics>
    </name>
    <orientation>
      <value>90</value>
    </orientation>
    <rate>
      <value>1.0</value>
    </rate>
    <timed>

```

```

    <value>>false</value>
  </timed>
  <infiniteServer>
    <value>>false</value>
  </infiniteServer>
  <priority>
    <value>1</value>
  </priority>
</transition>
<transition id="Branch3">
  <graphics>
    <position x="255.0" y="480.0"/>
  </graphics>
  <name>
    <value>Branch3</value>
    <graphics>
      <offset x="16.0" y="5.0"/>
    </graphics>
  </name>
  <orientation>
    <value>90</value>
  </orientation>
  <rate>
    <value>1.0</value>
  </rate>
  <timed>
    <value>>false</value>
  </timed>
  <infiniteServer>
    <value>>false</value>
  </infiniteServer>
  <priority>
    <value>1</value>
  </priority>
</transition>
<transition id="Join">
  <graphics>
    <position x="675.0" y="450.0"/>
  </graphics>
  <name>
    <value>Join</value>
    <graphics>
      <offset x="59.0" y="11.0"/>
    </graphics>
  </name>

```

```

<orientation>
  <value>90</value>
</orientation>
<rate>
  <value>1.0</value>
</rate>
<timed>
  <value>>false</value>
</timed>
<infiniteServer>
  <value>>false</value>
</infiniteServer>
<priority>
  <value>1</value>
</priority>
</transition>
<transition id="Join1">
  <graphics>
    <position x="405.0" y="165.0"/>
  </graphics>
  <name>
    <value>Join1</value>
    <graphics>
      <offset x="39.0" y="4.0"/>
    </graphics>
  </name>
  <orientation>
    <value>90</value>
  </orientation>
  <rate>
    <value>1.0</value>
  </rate>
  <timed>
    <value>>false</value>
  </timed>
  <infiniteServer>
    <value>>false</value>
  </infiniteServer>
  <priority>
    <value>1</value>
  </priority>
</transition>
<transition id="Join2">
  <graphics>
    <position x="120.0" y="390.0"/>

```

```

</graphics>
<name>
  <value>Join2</value>
  <graphics>
    <offset x="61.0" y="9.0"/>
  </graphics>
</name>
<orientation>
  <value>90</value>
</orientation>
<rate>
  <value>1.0</value>
</rate>
<timed>
  <value>>false</value>
</timed>
<infiniteServer>
  <value>>false</value>
</infiniteServer>
<priority>
  <value>1</value>
</priority>
</transition>
<transition id="Join3">
  <graphics>
    <position x="255.0" y="630.0"/>
  </graphics>
  <name>
    <value>Join3</value>
    <graphics>
      <offset x="28.0" y="0.0"/>
    </graphics>
  </name>
  <orientation>
    <value>90</value>
  </orientation>
  <rate>
    <value>1.0</value>
  </rate>
  <timed>
    <value>>false</value>
  </timed>
  <infiniteServer>
    <value>>false</value>
  </infiniteServer>

```

```

    <priority>
      <value>1</value>
    </priority>
  </transition>
  <transition id="T00_Poscustomer">
    <graphics>
      <position x="195.0" y="60.0"/>
    </graphics>
    <name>
      <value>T00_Poscustomer</value>
      <graphics>
        <offset x="62.0" y="-15.0"/>
      </graphics>
    </name>
    <orientation>
      <value>0</value>
    </orientation>
    <rate>
      <value>1.0</value>
    </rate>
    <timed>
      <value>>false</value>
    </timed>
    <infiniteServer>
      <value>>false</value>
    </infiniteServer>
    <priority>
      <value>1</value>
    </priority>
  </transition>
  <transition id="T01_Turnclerk">
    <graphics>
      <position x="390.0" y="60.0"/>
    </graphics>
    <name>
      <value>T01_Turnclerk</value>
      <graphics>
        <offset x="57.0" y="-15.0"/>
      </graphics>
    </name>
    <orientation>
      <value>0</value>
    </orientation>
    <rate>
      <value>1.0</value>
    </rate>
  </transition>

```

```

</rate>
<timed>
  <value>false</value>
</timed>
<infiniteServer>
  <value>false</value>
</infiniteServer>
<priority>
  <value>1</value>
</priority>
</transition>
<transition id="T02_Enterfitting">
  <graphics>
    <position x="615.0" y="60.0"/>
  </graphics>
  <name>
    <value>T02_Enterfitting</value>
    <graphics>
      <offset x="52.0" y="-14.0"/>
    </graphics>
  </name>
  <orientation>
    <value>0</value>
  </orientation>
  <rate>
    <value>1.0</value>
  </rate>
  <timed>
    <value>false</value>
  </timed>
  <infiniteServer>
    <value>false</value>
  </infiniteServer>
  <priority>
    <value>1</value>
  </priority>
</transition>
<transition id="T03_Talkclerk">
  <graphics>
    <position x="705.0" y="120.0"/>
  </graphics>
  <name>
    <value>T03_Talkclerk</value>
    <graphics>
      <offset x="95.0" y="-1.0"/>
    </graphics>
  </name>
  <orientation>
    <value>0</value>
  </orientation>
  <rate>
    <value>1.0</value>
  </rate>
  <timed>
    <value>false</value>
  </timed>
  <infiniteServer>
    <value>false</value>
  </infiniteServer>
  <priority>
    <value>1</value>
  </priority>
</transition>

```

```

    </graphics>
  </name>
  <orientation>
    <value>90</value>
  </orientation>
  <rate>
    <value>1.0</value>
  </rate>
  <timed>
    <value>>false</value>
  </timed>
  <infiniteServer>
    <value>>false</value>
  </infiniteServer>
  <priority>
    <value>1</value>
  </priority>
</transition>
<transition id="T04_Enterroom">
  <graphics>
    <position x="600.0" y="330.0"/>
  </graphics>
  <name>
    <value>T04_Enterroom</value>
  <graphics>
    <offset x="-7.0" y="13.0"/>
  </graphics>
  </name>
  <orientation>
    <value>90</value>
  </orientation>
  <rate>
    <value>1.0</value>
  </rate>
  <timed>
    <value>>false</value>
  </timed>
  <infiniteServer>
    <value>>false</value>
  </infiniteServer>
  <priority>
    <value>1</value>
  </priority>
</transition>
<transition id="T05_Enterstore">

```

```

<graphics>
  <position x="720.0" y="315.0"/>
</graphics>
<name>
  <value>T05_Enterstore</value>
  <graphics>
    <offset x="120.0" y="16.0"/>
  </graphics>
</name>
<orientation>
  <value>90</value>
</orientation>
<rate>
  <value>1.0</value>
</rate>
<timed>
  <value>>false</value>
</timed>
<infiniteServer>
  <value>>false</value>
</infiniteServer>
<priority>
  <value>1</value>
</priority>
</transition>
<transition id="T06_Replyclerk">
  <graphics>
    <position x="480.0" y="450.0"/>
  </graphics>
  <name>
    <value>T06_Replyclerk</value>
    <graphics>
      <offset x="63.0" y="-6.0"/>
    </graphics>
  </name>
  <orientation>
    <value>0</value>
  </orientation>
  <rate>
    <value>1.0</value>
  </rate>
  <timed>
    <value>>false</value>
  </timed>
  <infiniteServer>

```

```

    <value>>false</value>
  </infiniteServer>
  <priority>
    <value>1</value>
  </priority>
</transition>
<transition id="T07_Checkout">
  <graphics>
    <position x="360.0" y="285.0"/>
  </graphics>
  <name>
    <value>T07_Checkout</value>
  <graphics>
    <offset x="14.0" y="4.0"/>
  </graphics>
</name>
  <orientation>
    <value>90</value>
  </orientation>
  <rate>
    <value>1.0</value>
  </rate>
  <timed>
    <value>>false</value>
  </timed>
  <infiniteServer>
    <value>>false</value>
  </infiniteServer>
  <priority>
    <value>1</value>
  </priority>
</transition>
<transition id="T08_Sitdown">
  <graphics>
    <position x="450.0" y="285.0"/>
  </graphics>
  <name>
    <value>T08_Sitdown</value>
  <graphics>
    <offset x="62.0" y="0.0"/>
  </graphics>
</name>
  <orientation>
    <value>90</value>
  </orientation>

```

```

<rate>
  <value>1.0</value>
</rate>
<timed>
  <value>>false</value>
</timed>
<infiniteServer>
  <value>>false</value>
</infiniteServer>
<priority>
  <value>1</value>
</priority>
</transition>
<transition id="T09_Roomavailable">
  <graphics>
    <position x="285.0" y="150.0"/>
  </graphics>
  <name>
    <value>T09_Roomavailable</value>
    <graphics>
      <offset x="72.0" y="39.0"/>
    </graphics>
  </name>
  <orientation>
    <value>0</value>
  </orientation>
  <rate>
    <value>1.0</value>
  </rate>
  <timed>
    <value>>false</value>
  </timed>
  <infiniteServer>
    <value>>false</value>
  </infiniteServer>
  <priority>
    <value>1</value>
  </priority>
</transition>
<transition id="T10_Walk">
  <graphics>
    <position x="150.0" y="300.0"/>
  </graphics>
  <name>
    <value>T10_Walk</value>
  </name>

```

```

    <graphics>
      <offset x="67.0" y="6.0"/>
    </graphics>
  </name>
  <orientation>
    <value>90</value>
  </orientation>
  <rate>
    <value>1.0</value>
  </rate>
  <timed>
    <value>>false</value>
  </timed>
  <infiniteServer>
    <value>>false</value>
  </infiniteServer>
  <priority>
    <value>1</value>
  </priority>
</transition>
<transition id="T11_Stand">
  <graphics>
    <position x="75.0" y="300.0"/>
  </graphics>
  <name>
    <value>T11_Stand</value>
  <graphics>
    <offset x="7.0" y="9.0"/>
  </graphics>
  </name>
  <orientation>
    <value>90</value>
  </orientation>
  <rate>
    <value>1.0</value>
  </rate>
  <timed>
    <value>>false</value>
  </timed>
  <infiniteServer>
    <value>>false</value>
  </infiniteServer>
  <priority>
    <value>1</value>
  </priority>

```

```

</transition>
<transition id="T12_Actioncheckout">
  <graphics>
    <position x="180.0" y="420.0"/>
  </graphics>
  <name>
    <value>T12_Actioncheckout</value>
    <graphics>
      <offset x="106.0" y="-8.0"/>
    </graphics>
  </name>
  <orientation>
    <value>0</value>
  </orientation>
  <rate>
    <value>1.0</value>
  </rate>
  <timed>
    <value>>false</value>
  </timed>
  <infiniteServer>
    <value>>false</value>
  </infiniteServer>
  <priority>
    <value>1</value>
  </priority>
</transition>
<transition id="T13_Done">
  <graphics>
    <position x="210.0" y="555.0"/>
  </graphics>
  <name>
    <value>T13_Done</value>
    <graphics>
      <offset x="-2.0" y="17.0"/>
    </graphics>
  </name>
  <orientation>
    <value>90</value>
  </orientation>
  <rate>
    <value>1.0</value>
  </rate>
  <timed>
    <value>>false</value>

```

```

</timed>
<infiniteServer>
  <value>>false</value>
</infiniteServer>
<priority>
  <value>1</value>
</priority>
</transition>
<transition id="T14_Finishcheck">
  <graphics>
    <position x="285.0" y="555.0"/>
  </graphics>
  <name>
    <value>T14_Finishcheck</value>
    <graphics>
      <offset x="117.0" y="14.0"/>
    </graphics>
  </name>
  <orientation>
    <value>90</value>
  </orientation>
  <rate>
    <value>1.0</value>
  </rate>
  <timed>
    <value>>false</value>
  </timed>
  <infiniteServer>
    <value>>false</value>
  </infiniteServer>
  <priority>
    <value>1</value>
  </priority>
</transition>
<transition id="T15_Doneout">
  <graphics>
    <position x="375.0" y="630.0"/>
  </graphics>
  <name>
    <value>T15_Doneout</value>
    <graphics>
      <offset x="64.0" y="-7.0"/>
    </graphics>
  </name>
  <orientation>

```

```

    <value>0</value>
  </orientation>
  <rate>
    <value>1.0</value>
  </rate>
  <timed>
    <value>>false</value>
  </timed>
  <infiniteServer>
    <value>>false</value>
  </infiniteServer>
  <priority>
    <value>1</value>
  </priority>
</transition>
<transition id="T16_Pay">
  <graphics>
    <position x="465.0" y="630.0"/>
  </graphics>
  <name>
    <value>T16_Pay</value>
  <graphics>
    <offset x="40.0" y="-5.0"/>
  </graphics>
  </name>
  <orientation>
    <value>0</value>
  </orientation>
  <rate>
    <value>1.0</value>
  </rate>
  <timed>
    <value>>false</value>
  </timed>
  <infiniteServer>
    <value>>false</value>
  </infiniteServer>
  <priority>
    <value>1</value>
  </priority>
</transition>
<arc id="Accept to Branch" source="Accept" target="Branch">
  <graphics/>
  <inscription>
    <value>Default,1</value>

```

```

    <graphics/>
  </inscription>
  <tagged>
    <value>>false</value>
  </tagged>
  <arcpath id="000" x="658" y="146" curvePoint="false"/>
  <arcpath id="001" x="659" y="151" curvePoint="false"/>
  <arcpath id="002" x="651" y="148" curvePoint="false"/>
  <arcpath id="003" x="651" y="154" curvePoint="false"/>
  <arcpath id="004" x="656" y="186" curvePoint="false"/>
  <type value="normal"/>
</arc>
<arc id="Action to Branch3" source="Action" target="Branch3">
  <graphics/>
  <inscription>
    <value>Default,1</value>
    <graphics/>
  </inscription>
  <tagged>
    <value>>false</value>
  </tagged>
  <arcpath id="000" x="266" y="446" curvePoint="false"/>
  <arcpath id="001" x="266" y="486" curvePoint="false"/>
  <type value="normal"/>
</arc>
<arc id="Ask to T03_Talkclerk" source="Ask" target="T03_Talkclerk">
  <graphics/>
  <inscription>
    <value>Default,1</value>
    <graphics/>
  </inscription>
  <tagged>
    <value>>false</value>
  </tagged>
  <arcpath id="000" x="722" y="85" curvePoint="false"/>
  <arcpath id="001" x="721" y="82" curvePoint="false"/>
  <arcpath id="002" x="716" y="126" curvePoint="false"/>
  <type value="normal"/>
</arc>
<arc id="Atcheckout to Join2" source="Atcheckout" target="Join2">
  <graphics/>
  <inscription>
    <value>Default,1</value>
    <graphics/>
  </inscription>

```

```

    <tagged>
      <value>>false</value>
    </tagged>
    <arcpath id="000" x="152" y="368" curvePoint="false"/>
    <arcpath id="001" x="136" y="397" curvePoint="false"/>
    <type value="normal"/>
  </arc>
  <arc id="Atcustomer1 to T13_Done" source="Atcustomer1" target="T13_Done">
    <graphics/>
    <inscription>
      <value>Default,1</value>
    </graphics/>
  </inscription>
  <tagged>
    <value>>false</value>
  </tagged>
  <arcpath id="000" x="221" y="536" curvePoint="false"/>
  <arcpath id="001" x="221" y="561" curvePoint="false"/>
  <type value="normal"/>
</arc>
  <arc id="Atcustomer2 to T14_finishcheck" source="Atcustomer2"
target="T14_Finishcheck">
    <graphics/>
    <inscription>
      <value>Default,1</value>
    </graphics/>
  </inscription>
  <tagged>
    <value>>false</value>
  </tagged>
  <arcpath id="000" x="296" y="536" curvePoint="false"/>
  <arcpath id="001" x="296" y="561" curvePoint="false"/>
  <type value="normal"/>
</arc>
  <arc id="Branch1 to Forcustomer1" source="Branch1" target="Forcustomer1">
    <graphics/>
    <inscription>
      <value>Default,1</value>
    </graphics/>
  </inscription>
  <tagged>
    <value>>false</value>
  </tagged>
  <arcpath id="000" x="416" y="402" curvePoint="false"/>
  <arcpath id="001" x="418" y="395" curvePoint="false"/>

```

```

    <arcpath id="002" x="383" y="366" curvePoint="false"/>
    <type value="normal"/>
</arc>
<arc id="Branch1 to Forcustomer2" source="Branch1" target="Forcustomer2">
  <graphics/>
  <inscription>
    <value>Default,1</value>
    <graphics/>
  </inscription>
  <tagged>
    <value>>false</value>
  </tagged>
  <arcpath id="000" x="431" y="396" curvePoint="false"/>
  <arcpath id="001" x="433" y="395" curvePoint="false"/>
  <arcpath id="002" x="452" y="368" curvePoint="false"/>
  <type value="normal"/>
</arc>
<arc id="Branch2 to clerk" source="Branch2" target="clerk">
  <graphics/>
  <inscription>
    <value>Default,1</value>
    <graphics/>
  </inscription>
  <tagged>
    <value>>false</value>
  </tagged>
  <arcpath id="000" x="106" y="212" curvePoint="false"/>
  <arcpath id="001" x="150" y="256" curvePoint="false"/>
  <type value="normal"/>
</arc>
<arc id="Branch2 to Thecustomer" source="Branch2" target="Thecustomer">
  <graphics/>
  <inscription>
    <value>Default,1</value>
    <graphics/>
  </inscription>
  <tagged>
    <value>>false</value>
  </tagged>
  <arcpath id="000" x="96" y="211" curvePoint="false"/>
  <arcpath id="001" x="89" y="252" curvePoint="false"/>
  <type value="normal"/>
</arc>
<arc id="Branch3 to Atcustomer1" source="Branch3" target="Atcustomer1">
  <graphics/>

```

```

<inscription>
  <value>Default,1</value>
  <graphics/>
</inscription>
<tagged>
  <value>>false</value>
</tagged>
<arcpath id="000" x="251" y="492" curvePoint="false"/>
<arcpath id="001" x="232" y="511" curvePoint="false"/>
<type value="normal"/>
</arc>
<arc id="Branch3 to Atcustomer2" source="Branch3" target="Atcustomer2">
  <graphics/>
  <inscription>
    <value>Default,1</value>
    <graphics/>
  </inscription>
  <tagged>
    <value>>false</value>
  </tagged>
  <arcpath id="000" x="266" y="496" curvePoint="false"/>
  <arcpath id="001" x="285" y="512" curvePoint="false"/>
  <type value="normal"/>
</arc>
<arc id="Branch to Customer1" source="Branch" target="Customer1">
  <graphics/>
  <inscription>
    <value>Default,1</value>
    <graphics/>
  </inscription>
  <tagged>
    <value>>false</value>
  </tagged>
  <arcpath id="000" x="651" y="196" curvePoint="false"/>
  <arcpath id="001" x="649" y="204" curvePoint="false"/>
  <arcpath id="002" x="619" y="254" curvePoint="false"/>
  <type value="normal"/>
</arc>
<arc id="Branch to Customer2" source="Branch" target="Customer2">
  <graphics/>
  <inscription>
    <value>Default,1</value>
    <graphics/>
  </inscription>
  <tagged>

```

```

    <value>>false</value>
  </tagged>
  <arcpath id="000" x="661" y="197" curvePoint="false"/>
  <arcpath id="001" x="661" y="202" curvePoint="false"/>
  <arcpath id="002" x="661" y="202" curvePoint="false"/>
  <arcpath id="003" x="720" y="256" curvePoint="false"/>
  <type value="normal"/>
</arc>
<arc id="Checkout to Join3" source="Checkout" target="Join3">
  <graphics/>
  <inscription>
    <value>Default,1</value>
    <graphics/>
  </inscription>
  <tagged>
    <value>>false</value>
  </tagged>
  <arcpath id="000" x="285" y="621" curvePoint="false"/>
  <arcpath id="001" x="271" y="637" curvePoint="false"/>
  <type value="normal"/>
</arc>
<arc id="clerk(1)(1) to T11_Walk(1)(1)" source="clerk" target="T10_Walk">
  <graphics/>
  <inscription>
    <value>Default,1</value>
    <graphics/>
  </inscription>
  <tagged>
    <value>>false</value>
  </tagged>
  <arcpath id="000" x="147" y="268" curvePoint="false"/>
  <arcpath id="001" x="148" y="268" curvePoint="false"/>
  <arcpath id="002" x="161" y="306" curvePoint="false"/>
  <type value="normal"/>
</arc>
<arc id="Customer1 to T04_Enterroom" source="Customer1"
  target="T04_Enterroom">
  <graphics/>
  <inscription>
    <value>Default,1</value>
    <graphics/>
  </inscription>
  <tagged>
    <value>>false</value>
  </tagged>

```

```

    <arcpath id="000" x="611" y="281" curvePoint="false"/>
    <arcpath id="001" x="611" y="336" curvePoint="false"/>
    <type value="normal"/>
  </arc>
  <arc id="Customer2 to T05_EnterStore" source="Customer2"
    target="T05_Enterstore">
    <graphics/>
    <inscription>
      <value>Default,1</value>
      <graphics/>
    </inscription>
    <tagged>
      <value>>false</value>
    </tagged>
    <arcpath id="000" x="732" y="282" curvePoint="false"/>
    <arcpath id="001" x="732" y="286" curvePoint="false"/>
    <arcpath id="002" x="731" y="321" curvePoint="false"/>
    <type value="normal"/>
  </arc>
  <arc id="CustomerIN to T01_Turnclerk" source="CustomerIN"
    target="T01_Turnclerk">
    <graphics/>
    <inscription>
      <value>Default,1</value>
      <graphics/>
    </inscription>
    <tagged>
      <value>>false</value>
    </tagged>
    <arcpath id="000" x="311" y="72" curvePoint="false"/>
    <arcpath id="001" x="396" y="72" curvePoint="false"/>
    <type value="normal"/>
  </arc>
  <arc id="Done1 to T16_Pay" source="Done1" target="T16_Pay">
    <graphics/>
    <inscription>
      <value>Default,1</value>
      <graphics/>
    </inscription>
    <tagged>
      <value>>false</value>
    </tagged>
    <arcpath id="000" x="446" y="642" curvePoint="false"/>
    <arcpath id="001" x="471" y="642" curvePoint="false"/>
    <type value="normal"/>
  </arc>

```

```

</arc>
<arc id="Done to Join3" source="Done" target="Join3">
  <graphics/>
  <inscription>
    <value>Default,1</value>
  </graphics/>
</inscription>
<tagged>
  <value>>false</value>
</tagged>
<arcpath id="000" x="234" y="619" curvePoint="false"/>
<arcpath id="001" x="261" y="636" curvePoint="false"/>
<type value="normal"/>
</arc>
<arc id="Enteredroom to Join" source="Enteredroom" target="Join">
  <graphics/>
  <inscription>
    <value>Default,1</value>
  </graphics/>
</inscription>
<tagged>
  <value>>false</value>
</tagged>
<arcpath id="000" x="624" y="410" curvePoint="false"/>
<arcpath id="001" x="681" y="456" curvePoint="false"/>
<type value="normal"/>
</arc>
<arc id="EnteredStore to Join" source="EnteredStore" target="Join">
  <graphics/>
  <inscription>
    <value>Default,1</value>
  </graphics/>
</inscription>
<tagged>
  <value>>false</value>
</tagged>
<arcpath id="000" x="733" y="409" curvePoint="false"/>
<arcpath id="001" x="733" y="410" curvePoint="false"/>
<arcpath id="002" x="733" y="410" curvePoint="false"/>
<arcpath id="003" x="691" y="457" curvePoint="false"/>
<type value="normal"/>
</arc>
<arc id="Enteringfitting to Branch2" source="Enteringfitting" target="Branch2">
  <graphics/>
  <inscription>

```

```

    <value>Default,1</value>
    <graphics/>
  </inscription>
  <tagged>
    <value>>false</value>
  </tagged>
  <arcpath id="000" x="101" y="176" curvePoint="false"/>
  <arcpath id="001" x="101" y="201" curvePoint="false"/>
  <type value="normal"/>
</arc>
<arc id="Forcustomer1 to T07_Checkout" source="Forcustomer1"
  target="T07_Checkout">
  <graphics/>
  <inscription>
    <value>Default,1</value>
    <graphics/>
  </inscription>
  <tagged>
    <value>>false</value>
  </tagged>
  <arcpath id="000" x="372" y="342" curvePoint="false"/>
  <arcpath id="001" x="373" y="335" curvePoint="false"/>
  <arcpath id="002" x="371" y="301" curvePoint="false"/>
  <type value="normal"/>
</arc>
<arc id="Forcustomer2 to T08_Sitdown" source="Forcustomer2"
  target="T08_Sitdown">
  <graphics/>
  <inscription>
    <value>Default,1</value>
    <graphics/>
  </inscription>
  <tagged>
    <value>>false</value>
  </tagged>
  <arcpath id="000" x="457" y="342" curvePoint="false"/>
  <arcpath id="001" x="455" y="335" curvePoint="false"/>
  <arcpath id="002" x="461" y="301" curvePoint="false"/>
  <type value="normal"/>
</arc>
<arc id="Infitting to Join2" source="Infitting" target="Join2">
  <graphics/>
  <inscription>
    <value>Default,1</value>
    <graphics/>

```

```

</inscription>
<tagged>
  <value>>false</value>
</tagged>
<arcpath id="000" x="97" y="367" curvePoint="false"/>
<arcpath id="001" x="126" y="396" curvePoint="false"/>
<type value="normal"/>
</arc>
<arc id="In Progress to T06_Replyclerk" source="In Progress"
  target="T06_Replyclerk">
  <graphics/>
  <inscription>
    <value>Default,1</value>
    <graphics/>
  </inscription>
  <tagged>
    <value>>false</value>
  </tagged>
  <arcpath id="000" x="567" y="462" curvePoint="false"/>
  <arcpath id="001" x="496" y="462" curvePoint="false"/>
  <type value="normal"/>
</arc>
<arc id="Join1 to Progress" source="Join1" target="Progress">
  <graphics/>
  <inscription>
    <value>Default,1</value>
    <graphics/>
  </inscription>
  <tagged>
    <value>>false</value>
  </tagged>
  <arcpath id="000" x="401" y="177" curvePoint="false"/>
  <arcpath id="001" x="371" y="166" curvePoint="false"/>
  <type value="normal"/>
</arc>
<arc id="Join2 to Progress2" source="Join2" target="Progress2">
  <graphics/>
  <inscription>
    <value>Default,1</value>
    <graphics/>
  </inscription>
  <tagged>
    <value>>false</value>
  </tagged>
  <arcpath id="000" x="131" y="406" curvePoint="false"/>

```

```

    <arcpath id="001" x="131" y="417" curvePoint="false"/>
    <type value="normal"/>
  </arc>
  <arc id="Join3 to Progress3" source="Join3" target="Progress3">
    <graphics/>
    <inscription>
      <value>Default,1</value>
      <graphics/>
    </inscription>
    <tagged>
      <value>>false</value>
    </tagged>
    <arcpath id="000" x="281" y="642" curvePoint="false"/>
    <arcpath id="001" x="312" y="642" curvePoint="false"/>
    <type value="normal"/>
  </arc>
  <arc id="Join to In Progress" source="Join" target="In Progress">
    <graphics/>
    <inscription>
      <value>Default,1</value>
      <graphics/>
    </inscription>
    <tagged>
      <value>>false</value>
    </tagged>
    <arcpath id="000" x="671" y="462" curvePoint="false"/>
    <arcpath id="001" x="597" y="462" curvePoint="false"/>
    <type value="normal"/>
  </arc>
  <arc id="Progress2 to T12_Actioncheckout" source="Progress2"
    target="T12_Actioncheckout">
    <graphics/>
    <inscription>
      <value>Default,1</value>
      <graphics/>
    </inscription>
    <tagged>
      <value>>false</value>
    </tagged>
    <arcpath id="000" x="146" y="432" curvePoint="false"/>
    <arcpath id="001" x="186" y="432" curvePoint="false"/>
    <type value="normal"/>
  </arc>
  <arc id="Progress3 to T15_Doneout" source="Progress3" target="T15_Doneout">
    <graphics/>

```

```

<inscription>
  <value>Default,1</value>
  <graphics/>
</inscription>
<tagged>
  <value>>false</value>
</tagged>
<arcpath id="000" x="341" y="642" curvePoint="false"/>
<arcpath id="001" x="381" y="642" curvePoint="false"/>
<type value="normal"/>
</arc>
<arc id="Progress to T09_Roomavailable" source="Progress"
  target="T09_Roomavailable">
  <graphics/>
  <inscription>
    <value>Default,1</value>
    <graphics/>
  </inscription>
  <tagged>
    <value>>false</value>
  </tagged>
  <arcpath id="000" x="342" y="162" curvePoint="false"/>
  <arcpath id="001" x="301" y="162" curvePoint="false"/>
  <type value="normal"/>
</arc>
<arc id="Replied to Branch1" source="Replied" target="Branch1">
  <graphics/>
  <inscription>
    <value>Default,1</value>
    <graphics/>
  </inscription>
  <tagged>
    <value>>false</value>
  </tagged>
  <arcpath id="000" x="427" y="447" curvePoint="false"/>
  <arcpath id="001" x="425" y="440" curvePoint="false"/>
  <arcpath id="002" x="425" y="440" curvePoint="false"/>
  <arcpath id="003" x="431" y="406" curvePoint="false"/>
  <type value="normal"/>
</arc>
<arc id="Sit to Join1" source="Sit" target="Join1">
  <graphics/>
  <inscription>
    <value>Default,1</value>
    <graphics/>

```

```

</inscription>
<tagged>
  <value>>false</value>
</tagged>
<arcpath id="000" x="452" y="225" curvePoint="false"/>
<arcpath id="001" x="421" y="182" curvePoint="false"/>
<type value="normal"/>
</arc>
<arc id="Start workflow to T00_Poscustomer" source="Start workflow"
  target="T00_Poscustomer">
  <graphics/>
  <inscription>
    <value>Default,1</value>
    <graphics/>
  </inscription>
  <tagged>
    <value>>false</value>
  </tagged>
  <arcpath id="000" x="101" y="72" curvePoint="false"/>
  <arcpath id="001" x="201" y="72" curvePoint="false"/>
  <type value="normal"/>
</arc>
<arc id="T00_Poscustomer to CustomerIN" source="T00_Poscustomer"
  target="CustomerIN">
  <graphics/>
  <inscription>
    <value>Default,1</value>
    <graphics/>
  </inscription>
  <tagged>
    <value>>false</value>
  </tagged>
  <arcpath id="000" x="207" y="87" curvePoint="false"/>
  <arcpath id="001" x="207" y="74" curvePoint="false"/>
  <arcpath id="002" x="212" y="74" curvePoint="false"/>
  <arcpath id="003" x="212" y="74" curvePoint="false"/>
  <arcpath id="004" x="212" y="74" curvePoint="false"/>
  <arcpath id="005" x="212" y="74" curvePoint="false"/>
  <arcpath id="006" x="282" y="72" curvePoint="false"/>
  <type value="normal"/>
</arc>
<arc id="T01_Turnclerk to Turned" source="T01_Turnclerk" target="Turned">
  <graphics/>
  <inscription>
    <value>Default,1</value>

```

```

    <graphics/>
  </inscription>
  <tagged>
    <value>>false</value>
  </tagged>
  <arcpath id="000" x="406" y="72" curvePoint="false"/>
  <arcpath id="001" x="403" y="73" curvePoint="false"/>
  <arcpath id="002" x="492" y="72" curvePoint="false"/>
  <type value="normal"/>
</arc>
<arc id="T02_Enterfitting to Ask" source="T02_Enterfitting" target="Ask">
  <graphics/>
  <inscription>
    <value>Default,1</value>
    <graphics/>
  </inscription>
  <tagged>
    <value>>false</value>
  </tagged>
  <arcpath id="000" x="627" y="57" curvePoint="false"/>
  <arcpath id="001" x="624" y="66" curvePoint="false"/>
  <arcpath id="002" x="624" y="66" curvePoint="false"/>
  <arcpath id="003" x="702" y="71" curvePoint="false"/>
  <type value="normal"/>
</arc>
<arc id="T03_Talkclerk to Accept" source="T03_Talkclerk" target="Accept">
  <graphics/>
  <inscription>
    <value>Default,1</value>
    <graphics/>
  </inscription>
  <tagged>
    <value>>false</value>
  </tagged>
  <arcpath id="000" x="701" y="132" curvePoint="false"/>
  <arcpath id="001" x="708" y="136" curvePoint="false"/>
  <arcpath id="002" x="671" y="133" curvePoint="false"/>
  <type value="normal"/>
</arc>
<arc id="T04_Enterroom to Enteredroom" source="T04_Enterroom"
  target="Enteredroom">
  <graphics/>
  <inscription>
    <value>Default,1</value>
    <graphics/>

```

```

</inscription>
<tagged>
  <value>>false</value>
</tagged>
<arcpath id="000" x="611" y="346" curvePoint="false"/>
<arcpath id="001" x="613" y="343" curvePoint="false"/>
<arcpath id="002" x="613" y="343" curvePoint="false"/>
<arcpath id="003" x="612" y="387" curvePoint="false"/>
<type value="normal"/>
</arc>
<arc id="T05_EnterStore to EnteredStore" source="T05_Enterstore"
  target="EnteredStore">
  <graphics/>
  <inscription>
    <value>Default,1</value>
    <graphics/>
  </inscription>
  <tagged>
    <value>>false</value>
  </tagged>
  <arcpath id="000" x="731" y="331" curvePoint="false"/>
  <arcpath id="001" x="743" y="387" curvePoint="false"/>
  <type value="normal"/>
</arc>
<arc id="T06_Replyclerk to Replied" source="T06_Replyclerk" target="Replied">
  <graphics/>
  <inscription>
    <value>Default,1</value>
    <graphics/>
  </inscription>
  <tagged>
    <value>>false</value>
  </tagged>
  <arcpath id="000" x="486" y="462" curvePoint="false"/>
  <arcpath id="001" x="446" y="462" curvePoint="false"/>
  <type value="normal"/>
</arc>
<arc id="T07_Checkout to Wait" source="T07_Checkout" target="Wait">
  <graphics/>
  <inscription>
    <value>Default,1</value>
    <graphics/>
  </inscription>
  <tagged>
    <value>>false</value>
  </tagged>

```

```

</tagged>
<arcpath id="000" x="371" y="291" curvePoint="false"/>
<arcpath id="001" x="371" y="251" curvePoint="false"/>
<type value="normal"/>
</arc>
<arc id="T08_Sitdown to Sit" source="T08_Sitdown" target="Sit">
  <graphics/>
  <inscription>
    <value>Default,1</value>
    <graphics/>
  </inscription>
  <tagged>
    <value>>false</value>
  </tagged>
  <arcpath id="000" x="461" y="291" curvePoint="false"/>
  <arcpath id="001" x="461" y="251" curvePoint="false"/>
  <type value="normal"/>
</arc>
<arc id="T09_Roomavailable to Enteringfitting" source="T09_Roomavailable"
  target="Enteringfitting">
  <graphics/>
  <inscription>
    <value>Default,1</value>
    <graphics/>
  </inscription>
  <tagged>
    <value>>false</value>
  </tagged>
  <arcpath id="000" x="297" y="177" curvePoint="false"/>
  <arcpath id="001" x="297" y="162" curvePoint="false"/>
  <arcpath id="002" x="297" y="162" curvePoint="false"/>
  <arcpath id="003" x="117" y="162" curvePoint="false"/>
  <type value="normal"/>
</arc>
<arc id="T10_Walk to Atcheckout" source="T10_Walk" target="Atcheckout">
  <graphics/>
  <inscription>
    <value>Default,1</value>
    <graphics/>
  </inscription>
  <tagged>
    <value>>false</value>
  </tagged>
  <arcpath id="000" x="161" y="316" curvePoint="false"/>
  <arcpath id="001" x="161" y="342" curvePoint="false"/>

```

```

    <type value="normal"/>
  </arc>
  <arc id="T11_Stand to Infitting" source="T11_Stand" target="Infitting">
    <graphics/>
    <inscription>
      <value>Default,1</value>
      <graphics/>
    </inscription>
    <tagged>
      <value>>false</value>
    </tagged>
    <arcpath id="000" x="86" y="316" curvePoint="false"/>
    <arcpath id="001" x="86" y="342" curvePoint="false"/>
    <type value="normal"/>
  </arc>
  <arc id="T12_Actioncheckout to Action" source="T12_Actioncheckout"
    target="Action">
    <graphics/>
    <inscription>
      <value>Default,1</value>
      <graphics/>
    </inscription>
    <tagged>
      <value>>false</value>
    </tagged>
    <arcpath id="000" x="196" y="432" curvePoint="false"/>
    <arcpath id="001" x="252" y="432" curvePoint="false"/>
    <type value="normal"/>
  </arc>
  <arc id="T13_Done to Done" source="T13_Done" target="Done">
    <graphics/>
    <inscription>
      <value>Default,1</value>
      <graphics/>
    </inscription>
    <tagged>
      <value>>false</value>
    </tagged>
    <arcpath id="000" x="221" y="571" curvePoint="false"/>
    <arcpath id="001" x="221" y="597" curvePoint="false"/>
    <type value="normal"/>
  </arc>
  <arc id="T14_finishcheck to Checkout" source="T14_Finishcheck" target="Checkout">
    <graphics/>
    <inscription>

```

```

    <value>Default,1</value>
    <graphics/>
  </inscription>
  <tagged>
    <value>>false</value>
  </tagged>
  <arcpath id="000" x="296" y="571" curvePoint="false"/>
  <arcpath id="001" x="296" y="597" curvePoint="false"/>
  <type value="normal"/>
</arc>
<arc id="T15_Doneout to Done1" source="T15_Doneout" target="Done1">
  <graphics/>
  <inscription>
    <value>Default,1</value>
    <graphics/>
  </inscription>
  <tagged>
    <value>>false</value>
  </tagged>
  <arcpath id="000" x="391" y="642" curvePoint="false"/>
  <arcpath id="001" x="417" y="642" curvePoint="false"/>
  <type value="normal"/>
</arc>
<arc id="T16_Pay to EndWF" source="T16_Pay" target="EndWF">
  <graphics/>
  <inscription>
    <value>Default,1</value>
    <graphics/>
  </inscription>
  <tagged>
    <value>>false</value>
  </tagged>
  <arcpath id="000" x="481" y="642" curvePoint="false"/>
  <arcpath id="001" x="522" y="642" curvePoint="false"/>
  <type value="normal"/>
</arc>
<arc id="Thecustomer2(1)(1) to T10_Stand(1)(1)" source="Thecustomer"
target="T11_Stand">
  <graphics/>
  <inscription>
    <value>Default,1</value>
    <graphics/>
  </inscription>
  <tagged>
    <value>>false</value>

```

```

</tagged>
<arcpath id="000" x="86" y="281" curvePoint="false"/>
<arcpath id="001" x="86" y="306" curvePoint="false"/>
<type value="normal"/>
</arc>
<arc id="Turned to T02_Enterfitting" source="Turned" target="T02_Enterfitting">
  <graphics/>
  <inscription>
    <value>Default,1</value>
    <graphics/>
  </inscription>
  <tagged>
    <value>>false</value>
  </tagged>
  <arcpath id="000" x="521" y="71" curvePoint="false"/>
  <arcpath id="001" x="523" y="71" curvePoint="false"/>
  <arcpath id="002" x="524" y="71" curvePoint="false"/>
  <arcpath id="003" x="532" y="71" curvePoint="false"/>
  <arcpath id="004" x="621" y="72" curvePoint="false"/>
  <type value="normal"/>
</arc>
<arc id="Wait to Join1" source="Wait" target="Join1">
  <graphics/>
  <inscription>
    <value>Default,1</value>
    <graphics/>
  </inscription>
  <tagged>
    <value>>false</value>
  </tagged>
  <arcpath id="000" x="380" y="224" curvePoint="false"/>
  <arcpath id="001" x="411" y="181" curvePoint="false"/>
  <type value="normal"/>
</arc>
</net>
</pnm1>

```

APPENDIX C – RETAIL SALES WORKFLOW LUA FILES

Each role has its own Lua script file. The scripts for the customer and sales clerk roles for shopping in the retail sales workflow are listed below. A number of functions are exported from the C# avatarbot control program into the Lua execution environment. This allows Lua to call functions to access the C# AvatarBot object, perform leaf-level commands, pause activity, record log information, and inform the Petri net interpreter when a task is complete. The list of functions includes `getABot`, `doCmd`, `pause`, `logMsg`, and `callBack`.

RetailWF_Customer1.lua

```
-- For Customer1
luanet.load_assembly("AvatarBot.exe")

function init()
    myAvatarBot = getABot()
    myName = myAvatarBot.ToString()
    myRole = myAvatarBot.role
    sleep(3000)
    ret = doCmd("regequip Store")
    logMsg("Lua saw: " .. ret, 1)
    return "T00_Poscustomer, poscust||" ..
        "T02_Enterfitting, enterfit||" ..
        "T07_Checkout, checkout||" ..
        "T12_Actioncheckout, action||" ..
        "T13_Done, done"
end

function poscust(taskname)
    doCmd("pause 8")
    doCmd("movewp <110.65, 61.36, 28.07>
        <110.04, 52.17, 28.11>")
    doCmd("pause 2")
    doCmd("touch <<Store_shirt_7>>")
    doCmd("pause 1")
    callBack(myName, "okay||" .. taskname)
end

function enterfit(taskname)
    doCmd("movewp <107.25, 49.76, 28.11>
        <113.51, 30.19, 28.11>")
    doCmd("pause 1")
    doCmd("play TALK")
end
```

```

doCmd("say primaudio|voice=2|text=can I enter the fitting room")
doCmd("pause 8")
callBack(myName, "okay||" .. taskname)
end

function enterroom(taskname)
doCmd("say 1313 open")
doCmd("moveto 115.81 26.83 28.13")
doCmd("say 1313 open")
doCmd("pause 5")
callBack(myName, "okay||" .. taskname)
end

function checkout(taskname)
doCmd("say 1313 open")
doCmd("moveto 113.55 29.45 28.11")
doCmd("turnto 180")
doCmd("forward")
doCmd("say 1313 open")
doCmd("moveto 106.62 41.78 28.11")
doCmd("pause 1")
callBack(myName, "okay||" .. taskname)
end

function action(taskname)
doCmd("drop <<Store_shirt_7>>
      <104.782,45.055,29.198>
      <0.28960,-0.21617,-0.73037,0.57963>")
doCmd("pause 1")
doCmd("say primaudio|voice=2|text=Yes,thanks")
doCmd("pause 8")
callBack(myName, "okay||" .. taskname)
end

function done(taskname)
doCmd("touch <<Store_bag_3>>")
doCmd("pause 2")
doCmd("movewp <107.94,50.65,28.11>
      <109.55,58.21,28.10>
      <114.44,63.21,28.07>")
doCmd("pause 1")
doCmd("drop <<Store_bag_3>> ")
doCmd("pause 1")
doCmd("shout 115 !resetbag!")
callBack(myName, "okay||" .. taskname)
end

```

RetailWF_Customer2.lua

```

-- For Customer2
luanet.load_assembly("AvatarBot.exe")

```

```

function init()
  myAvatarBot = getABot()
  myName = myAvatarBot.ToString()
  myRole = myAvatarBot.role
  sleep(3000)
  return "T05_Enterstore,enterstore||" ..
        "T08_Sitdown,sit||" ..
        "T11_Stand,stand||" ..
        "T14_Finishcheck,finish||" ..
        "T16_Pay,paid"
end

function enterstore(taskname)
  doCmd("movewp <103.16,57.53,28.10>
        <109.38,57.61,28.10>
        <110.08,52.67,28.11>")
  doCmd("pause 1")
  doCmd("touch <<Store_shirt_2>>")
  doCmd("movewp <107.43,50.49,28.11>
        <112.25,31.92,28.11>")
  doCmd("turnto 300")
  doCmd("forward")
  doCmd("pause 1")
  doCmd("play TALK")
  doCmd("say primaudio|voice=4|text=Hi,Can I enter the fitting
room?")
  doCmd("pause 6")
  callBack(myName, "okay||" .. taskname)
end

function sit(taskname)
  doCmd("moveto 105.83 26.56 28.13")
  doCmd("sit")
  doCmd("pause 3")
  callBack(myName, "okay||" .. taskname)
end

function stand(taskname)
  doCmd("pause 2")
  doCmd("stand")
  doCmd("say primaudio|voice=4|text=Thank you")
  doCmd("pause 7")
  doCmd("moveto 113.51 30.19 28.11")
  doCmd("say 1313 open")
  doCmd("moveto 115.81 26.83 28.13")
  doCmd("say 1313 open")
  callBack(myName, "okay||" .. taskname)
end

function finish(taskname)
  doCmd("say 1313 open")
  doCmd("moveto 113.55 29.45 28.11")
  doCmd("turnto 200")
  doCmd("say 1313 open")

```

```

doCmd("moveto 106.62 41.78 28.11")
doCmd("pause 4")
doCmd("drop <<Store_shirt_2>>
      <104.799,44.414,29.261>
      <0.28960,-0.21617,-0.73037,0.57963>>")
doCmd("pause 4")
callBack(myName, "okay||" .. taskname)
end

function paid(taskname)
doCmd("pause 1")
doCmd("say primaudio|voice=4|text=Yes,Thanks")
doCmd("pause 8")
doCmd("touch <<Store_bag_5>>")
doCmd("pause 1")
doCmd("movewp <109.39,57.02,28.10><93.84,58.93,28.01>")
doCmd("pause 1")
doCmd("drop <<Store_bag_5>>")
doCmd("pause 2")
doCmd("shout 115 !resetbagg!")
callBack(myName, "okay||" .. taskname)
end

```

RetailWF_Clerk.lua

```

-- For the Clerk
luanet.load_assembly("AvatarBot.exe")

function init()
myAvatarBot = getABot()
myName = myAvatarBot.ToString()
myRole = myAvatarBot.role
return "T01_Turnclerk,turnclerk||" ..
      "T03_Talkclerk,talkclerk||" ..
      "T06_Replyclerk,reply||" ..
      "T09_Roomavailable,available||" ..
      "T10_Walk,walk||" ..
      "T15_Doneout,doneout"
end

function turnclerk(taskname)
doCmd("turnto 180")
doCmd("forward")
doCmd("pause 1")
callBack(myName, "okay||" .. taskname)
end

function talkclerk(taskname)
doCmd("play TALK")
doCmd("say primaudio|voice=3|text=Sure,there is one available")
doCmd("pause 7")
callBack(myName, "okay||" .. taskname)
end

```

```

end

function reply(taskname)
    doCmd("play TALK ")
    doCmd("say primaudio|voice=3|text=It is not available,Have a
seat,I'll let you know when it is available")
    doCmd("pause 12")
    callBack(myName, "okay||" .. taskname)
end

function available(taskname)
    doCmd("moveto 107.98 27.23 28.11")
    doCmd("say primaudio|voice=3|text=The Fitting room is available,
you can enter now.")
    doCmd("pause 10")
    callBack(myName, "okay||" .. taskname)
end

function walk(taskname)
    doCmd("movewp <108.43,36.67,28.11>
        <104.24,39.35,28.11>
        <103.45,42.61,28.11>")
    doCmd("turnto 300")
    doCmd("forward")
    doCmd("pause 1")
    doCmd("say primaudio|voice=3|text=that's all for you today")
    doCmd("pause 6")
    callBack(myName, "okay||" .. taskname)
end

function doneout(taskname)
    doCmd("say primaudio|voice=3|text=did you find everything
alright?")
    doCmd("pause 7")
    callBack(myName, "okay||" .. taskname)
end

```

