FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Framework for Multi-Agent Simulation of User Behaviour in E-Commerce Sites

Duarte Duarte

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Hugo Sereno Ferreira, PhD

Second Supervisor: João Azevedo, MSc

June 28, 2016

Framework for Multi-Agent Simulation of User Behaviour in E-Commerce Sites

Duarte Duarte

Mestrado Integrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

June 28, 2016

Abstract

Customers interact with e-commerce websites in multiple ways and the companies operating them rely on optimizing success metrics for profit. Changing what, how and when content such as product recommendations and ads are displayed can influence customers' actions.

Multiple algorithms and techniques in data mining and machine learning have been applied in this context. Summarizing and analysing user behaviour can be expensive and tricky since it's hard to extrapolate patterns that never occurred before and the causality aspects of the system are not usually taken into consideration. Commonly used online techniques have the downside of having a high operational cost. However, there has been studies about characterizing user behaviour and interactions in e-commerce websites that could be used to improve this process.

The goal of this dissertation is to create a framework capable of running a multi-agent simulation, by regarding users in an e-commerce website that react to stimuli that influence their actions. By taking input from web mining, which includes both static and dynamic content of websites as well as user personas, the simulation should collect success metrics so that the experimentation being run can be evaluated.

Resumo

Consumidores interagem com websites de comércio eletrónico de várias formas e as empresas que os operam dependem da otimização de métricas de sucesso tais como *CTR* (*Click through Rate*), *CPC* (*Cost per Conversion*), *Basket* e *Lifetime Value* e *User Engagement* para lucro. Alterar como, onde e quando o conteúdo de páginas web como por exemplo recomendação de produtos e publicidade é mostrado pode influenciar as ações dos consumidores.

Vários algoritmos e técnicas em *data mining* e *machine learning* têm sido aplicados neste contexto. Sumarizar e analisar comportamento de utilizadores pode ser custoso e complicado porque é difícil extrapolar padrões que nunca ocorreram antes e os aspetos causais do sistema geralmente não são tidos em consideração. Técnicas *online* geralmente usadas têm o problema de ter um custo operacional elevado. Porém, existem estudos sobre caracterizar comportamento e interações de utilizadores em sites de comércio eletrónico que podem ser usados para melhorar este processo.

O objetivo desta dissertação é criar uma *framework* capaz de correr uma simulação multiagente, tendo em conta os utilizadores de um site de comércio eletrónico que reagem a estímulos que influenciam as suas ações. Extraindo dados de web mining, que inclui tanto conteúdo estático como dinâmico de websites assim como de perfis de utilizadores, a simulação deve reportar métricas de sucesso para que a experiência possa ser avaliada.

"In recent years, hundreds of the brightest minds of modern civilization have been hard at work not curing cancer. Instead, they have been refining techniques for getting you and me to click on banner ads."

Steve Hanov

Contents

List of Figures

LIST OF FIGURES

List of Tables

LIST OF TABLES

Abbreviations

ABS	Agent Based Simulation
API	Application Programming Interface
CBMG	Customer behavior model graph
CPC	Cost per Conversion
CTR	Click through Rate
DAG	Directed acyclic graph
DES	Discrete Event Simulation
HMM	Hidden Markov model
JPD	Joint probability distribution
JVM	Java Virtual Machine
LTV	Lifetime Value
MOOC	Massive Open Online Course
PDES	Parallel Discrete Event Simulation
PGM	Probabilistic Graphical Model
SBT	Simple (or Scala) Build Tool
SD	System Dynamics
SMAPE	Symmetric mean absolute percentage error
SWOT	Strengths, Weaknesses, Opportunities and Threats
WCM	Web content mining
WSM	Web structure mining
WUM	Web usage mining
WWW	*World Wide Web*

Chapter 1

Introduction

In this chapter we intend to introduce the report, starting by describing its context, motivations and objectives that will drive the dissertation. It ends with a description of the report structure.

1.1 Context

Customers interact with e-commerce websites in multiple ways and the companies operating them rely on optimizing success metrics such as CTR (Click through Rate), CPC (Cost per Conversion), Basket and Lifetime Value and User Engagement for profit. Changing what, how and when content such as product recommendations and ads are displayed can influence customers' actions.

Multiple algorithms and techniques in data mining and machine learning have been applied in this context.

1.2 Motivation

Modelling user behaviour on the web is not a new problem. It has been applied with different objectives, from improving the performance of cache servers, to the improvement of search engine, influencing purchase patterns or recommending related pages or products [DK01, J S00]. However, all these approaches were done with a machine learning mindset – *predicting* which page the user or customer will browse next. This requires extensive use of existing and historical training datasets which might not expose all the causality aspects of the system. What if the data (or the time needed to gather it) is simply not available?

1.3 Problem and goals

Let's imagine that we developed a new recommendation engine algorithm. One of the most common ways to evaluate it is by testing the engine with *A/B testing*[1], which is a randomized experiment where a group of users are presented with one version of the engine (control) and the other group is shown the improved version of the engine. By analysing how the two groups behave differently, it's possible to assess the quality of the two versions, comparatively. However, this approach may not be feasible in all situations. For the experiment to be statistically significant, the number of users shown the two versions of the product must be enough. The experiment also takes time to run and the metrics used to compare both versions might not be easy to choose. [Ama15].

The goal of this dissertation is to create a framework capable of running a multi-agent simulation (chapter 3), by regarding users in an e-commerce website and react to stimuli that influence their actions (chapter 2). Furthermore, some statistical constructs such as Baysian networks, Markov chains or probability distributions (chapter 4) can be used to guide how these agents interact with the system. By taking input from web mining (Web structure mining (WSM), Web usage mining (WUM) and Web content mining (WCM)), which includes both static and dynamic content of websites as well as user personas, the simulation should collect success metrics so that the experimentation being run can be evaluated.

This dissertation is focused on the framework for the simulation and not on the required input of the simulation, however that is a very important aspect. Luckily, web mining has been well studied. The works of [Dia16] proposes a methodology to extract and combine the sheer amount of data related to an e-commerce website, including structure, content and modelling user behaviour.

1.4 Report Structure

Besides this introduction, this report has 5 more chapters.

In chapters 2, 3 and 4, we describe the literature review and state of the art with regard to e-commerce, simulation systems and probabilistic models, respectively. The chapter 2 focuses on e-commerce background, what metrics can be used on e-commerce websites and the customer life cycle, an important part of the simulation. The chapter 3 describes three main topics regarding simulating systems: agent based, discrete event simulation and hybrid approaches. Finally, the chapter 4 deals with describing some probabilistic models, with emphasis on graphical models and Bayesian statistics.

Chapter 5 is concerned with describing the methodology followed during the realization of the framework, it presents the requirements of the framework, its architecture, a scalability study and the technology used.

The validation, chapter 6 was done by both fabricating scenarios with clear expected results and with real data.

[1] formally, two-sample hypothesis testing

The final chapter, chapter 7, concludes the work realized, presents the main contributions of this dissertation and proposes future work.

Chapter 2

Literature Review: E-commerce background

In this chapter we discuss some key concepts related to e-commerce, for the purpose of giving context to the dissertation. We discuss the typical customer life cycle in an e-commerce website, some metrics that might be used and some ways on how the customer interaction with the website might be influenced and improved.

2.1 Introduction

E-commerce, or electronic commerce, can be described by the trading of products or services over the Internet (or other computer networks). The type of e-commerce businesses we are interested are those who sell their goods directly to the customer, e.g online shopping, using an online store or catalog of products. Some popular online stores [AI16] are Amazon[1], Ebay[2] and Alibaba[3].

2.2 Customer life cycle

An important concept to understand the customer is by describing its life cycle, as presented by [SC00, Section 6] in figure 2.1.

It starts by reaching the target audience or market up to an established customer base, not forgetting about those that drop mid way, due to abandonment or attrition.

- **Reach** happens outside of the website and refers to the number of potential customers. For example, if the online store is advertised on a social network, the reach is the number of users who were served the ad in that other website, they may or may not ignore it.

[1] http://www.amazon.com/
[2] http://www.ebay.com/
[3] http://www.alibaba.com/

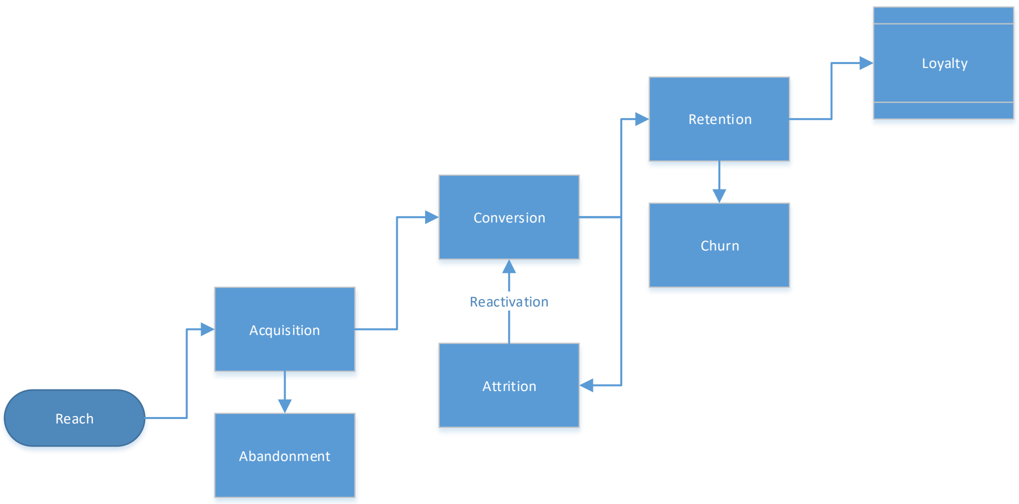


Figure 2.1: Customer lifecycle [SC00]

- **Acquisition** is the next stage, where the user decides to act on and visits the website (or some other action like subscribing to a newsletter).
- **Conversion** is the stage where a visitor stops being a user and starts being a customer. It usually means that the user made a purchase but some companies might consider a sign up or registration in the website as a conversion.
- **Retention** focuses on making existing customers, that made at least one purchase before, repeat purchases.
- **Loyalty** is a stronger form of retention, which represents a greater trust level of the customer in the store.
- **Abandonment** is defined by the customers that started the buying process but do not finish it. For example, a customer may add items to the online shopping cart but instead of moving to the next step, e.g. enter credit card details, they exit the website or go elsewhere. This may happen in any store with a multi-step buying process, which is very common.
- **Attrition** happens when a retained customer ceases buying from the store and starts using a competitor store.
- **Churn** is defined by the number of customers that attrited during a certain period divided by the total number of customers at the end of that period. It measures how much of the customer base "rolls over" in a certain time period.

2.3 Customer Behaviour Model Graph (CBMG)

A state transition graph named Customer Behaviour Model Graph (CBMG) can be used to describe the behaviour of customers browsing a website. The nodes represent the possible states or pages, e.g home page, product page, search, and a probability is associated with each transition. An example of such a CBMG is shown in figure 2.2.

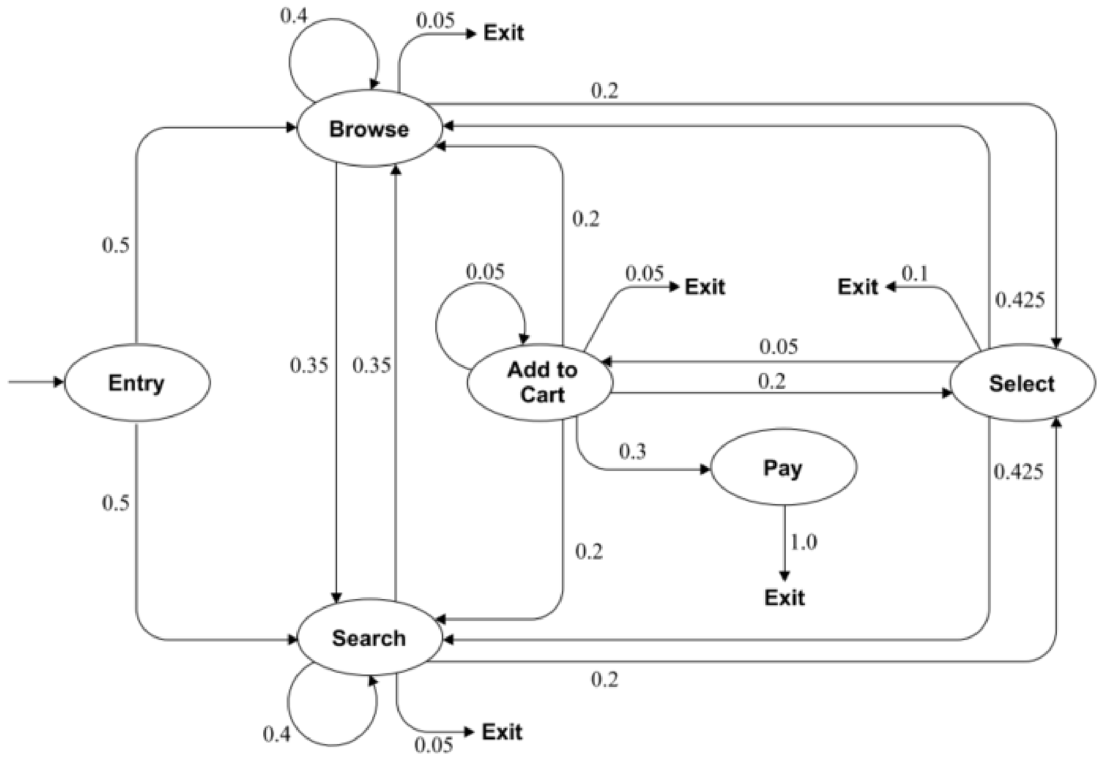


Figure 2.2: Example of a customer behaviour model graph [MAFM99]

[MAFM99] describes how CBMGs can be used to analyse the workload of an e-commerce store server and how metrics can be derived directly from the CBMG alone.

2.4 E-commerce metrics

Metrics are a common way to quantify, measure, benchmark or evaluate some process. In an e-commerce setting, businesses are interested in optimizing, mostly, for profit. Different businesses prioritize metrics in different ways, adapted to each use case. Here we present some commonly used metrics, but this list is by no means exhaustive. [SC00, MAFM99]

- *Conversion Rate* (CR) is the percentage of visitors that buy a product or a service;
- *Shopping Cart Abandonment* is the percentage of visitors that added a product to the online cart but did not complete the process;

- *Average Order Value* (AOV) is the average cost of all orders;
- *Customer Lifetime Value* (LTV) is the projected value that a customer will spend on the store;
- *Clicks to Buy* (CTB) is the average number of clicks a visitor has to do to complete a buy order;
- *Churn Rate* the percentage of customers that do not make a repeated purchase;
- *Bounce Rate* is the percentage of visitors that arrive at the homepage of the online store but leave immediately, without clicking anything or visiting a different page.

There are other common metrics such as *Acquisition Cost*, *Cost Per Conversion*, *Net Yield* or *Connection Rate* however they are associated with promotion campaigns that happen outside of the store website, therefore they are not interesting in the context of our work.

2.5 Influencing user behaviour

[Con04] describes functionality, psychological and content factors that can influence the visitor experience, represented in the table 2.1.

Table 2.1: Main building blocks of Web experience and their sub-categories [Con04]

Functionality factors		
Usability	**Interactivity**	
Convenience	Customer	
Site navigation	Interaction with company personnel	
Information architecture	Customization	
Ordering/payment process	Network effects	
Search facilities and process		
Site speed		
Findability/accessibility		
Psychological factors	**Content factors**	
Trust	**Aesthetics**	**Marketing mix**
Transaction security	Design	Communication
Customer data misuse	Presentation quality	Product
Customer data safety	Design elements	Fulfillment
Uncertainty reducing elements	Style/atmosphere	Price
Guarantees/return policies		Promotion
		Characteristics

Regarding usability of the online store, providing a personalized experience to each customer can be very beneficial for both the customer and the business. A common way to do this is by recommending products that the customer might be interested in [AT05]. For example, if we know that a customer buys mostly football related products, recommending her more products in the same category might increase sales.

2.6 Summary

In this chapter we covered a brief overview of e-commerce, starting with the customer lifecycle, how to measure it using metrics and presenting a common way to model the users' behaviour, the CBMG.

Chapter 3

Literature Review: System Simulation

This chapter intends to introduce some approaches to computational simulation systems and engines, namely agent based and discrete event simulation. To finish the chapter, we show some novel approaches to simulation.

3.1 Introduction

Simulations are used to reproduce the behaviour of a system. They have been applied to different areas like physics, weather, biology, economics and many others. There are many types of simulations: stochastic or deterministic, steady-state or dynamic, continuous or discrete and local or distributed [Wik15]. These categories are not exhaustive nor exclusive.

In this literature review, we are particularly interested in studying simulations which can model stochastic processes and not dynamic (dynamic systems are usually described by differential equations and are continuous by definition).

3.2 Agent Based Simulation (ABS)

In agent based simulation (ABS), sometimes described as agent based computing [Woo98, Jen99], the individual entities in the model are represented discretely and maintain a set of behaviours, beliefs or rules that determine how their state is updated. [Nia11] lists three different approaches to agent based modelling and simulation:

- *Agent-oriented programming* which puts emphasis on developing complex individual agents rather than a large set of agents;
- *Multi-agent oriented programming* focus on adding *some* intelligence to agents and observe their interactions;

- *Agent-based or massively multi-agent modelling* where the main idea is to build simple models for the agents which interact with a large population of other agents to observe the global behaviour.

[SA10] describes ABS as “well suited to modelling systems with heterogeneous, autonomous and pro-active actors, such as human-centred systems.”, which make them a good candidate to be used in the development of this dissertation. However, existing literature is quite confusing and broad, using different terms to refer to the same concepts, without clear distinctions between different agent based approaches and without consensus [Nia11, Bra14].

Many platforms and frameworks were developed to support agent-based modelling and simulation. Some notable examples include Repast [Col03], NetLogo [WE99], StarLogo [Res96] or MASON [PL05]. An updated list is maintained at OpenABM [Ope16].

Agents have been applied to e-commerce context mostly in two distinct areas: recommendation systems [XB07, WBS08] and negotiation [RKP02, MGM99]. No relevant literature was found regarding simulating user behaviour in websites with agents.

3.3 Discrete Event Simulation (DES)

A discrete event simulation (DES) models a process as a series of discrete events, where the state of the system changes only at well defined points in time [SA10]. It was originally proposed by Kiviat in 1969 [Kiv69] and there is extensive research in this simulation technique. Banks et al. [BCNN04] provides a comprehensive description and analysis of DES. The algorithm 1 is a possible implementation of a very simple and single-threaded DES.

Algorithm 1 Basic DES algorithm

```
$EndCondition \leftarrow false$
$Clock \leftarrow 0$
$EventList \leftarrow initialEvent$
while $EndCondition = false$ do
    $CurrentEvent \leftarrow$ POP($EventList$)
    $Clock \leftarrow$ TIME($CurrentEvent$)
    EXECUTE($CurrentEvent$)                ▷ might put new events in $EventList$
    UPDATESTATISTICS()
end while
GENERATEREPORT()
```

The major concepts in DES are [BCNN04]:

- Entity, objects explicitly represented in the model (e.g a customer);
- Event, an occurrence that changes the state of the system (e.g a customer enters the website);
- Event list (or future event list or pending event set), a list of future events, ordered by time of occurrence;

- Clock, used to keep track of the current simulation time.

Event list is one of the fundamental parts of the system and it has been widely researched [HOP$^+$86, Jon86, TT00, DGW13].

Pidd [Pid98] proposes a three-phased approach that consists of: jump to the next chronological event, executing all the unconditional events (or type B) that happen that moment and then executing all the conditional events (or type C). This approach has advantages in terms of less usage of resources compared to other simplistic approaches. Also, there has been studies on how to scale DES to distributed and parallel (PDES) executions [Mis86, Fuj90].

[SMG$^+$10] states that "DES is useful for problems (...) in which the processes can be well defined and their emphasis is on representing uncertainty through stochastic distributions", which makes DES a good candidate to model the problem at hand.

3.4 Hybrid and novel approaches

In recent years, there has been research which proposes a marriage between agent based model and simulation with discrete event simulation, however, this concept is not widely recognized [Bra14]. Brailsford states that the line that divides agent based models (and simulation) and DES is spurious and that common distinctions between the two approaches are artificial. Casas et al. [FRJ11] describe a method where multi agent system components have been added to an existing discrete event simulation implemented in OMNeT++[1] [Var01]. Onggo [Ong07] shows how agent based models can be ran on top of a DES engine. Kurve et al. [KKK13] describes an agent based performance model of a PDES kernel. Regarding existing software, AnyLogic claims to be "the only simulation tool that supports Discrete Event, Agent Based, and System Dynamics Simulation" [Any00]. AnyLogic was first shown in 2000 at the Winter Simulation Conference.

3.5 Summary

This literature reviews shows that there is vast research regarding simulation, either agent based or DES, however not everyone is speaking the same language. The extensions to DES seen above are particularly interesting since they can be used to scale the simulation to a greater number of entities as well as modelling real world processes with more fidelity.

[1]C++ based discrete event simulation toolkit

Chapter 4

Literature Review: Probabilistic Models

4.1 Introduction

Probabilistic or statistical models represent explicit assumptions about a problem domain, in the form of a model. This model usually encompasses random variables[1], in the form of probability distributions, and the relation and dependence between the variables. [WB13]

In the following sections we describe a common way to represent probabilistic models, probabilistic graphical models (PGM) or, simply, graphical models.

4.2 Probabilistic Graphical Models

A PGM is a graph based model where the nodes represent random variables and the (directed or undirected) edges represent a conditional dependence between variables. An example is shown in figure 4.1.

PGMs and their extensions, where we show some examples of them in the following sections, are exceptionally well suited for reasoning and to reach conclusions based on available information (both domain expert and data), even in the presence of uncertainty. PGMs provide a general framework that allows representation, inference and learning on these models. [KF09]

There is extensive research and available literature in this area. Some notable examples include, but are not limited to, the books *"Probabilistic Graphical Models: Principles and Techniques"* by Daphne Koller and Nir Friedman [KF09] and *"Pattern Recognition and Machine Learning"* (Chapter 8: Graphical Models) by Christopher Bishop [Bis06]. It is also worth mentioning that there is a MOOC [2] named *"Probabilistic Graphical Models"*, also by Daphne Koller (Stanford), freely available on Coursera [3].

[1]Variable whose value is given by a probability distribution, commonly represented by Θ.
[2]Massive Open Online Course
[3]`https://www.coursera.org/course/pgm`

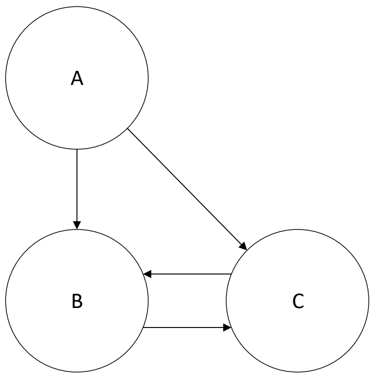


Figure 4.1: Example of a PGM: B and C depend on A, B depends on C and C depends on B.

In the following sections, we describe three important categories of graphical models: Bayesian networks, Markov random fields and its extension to hidden Markov models. There are plenty of other graphical models however they were deemed not relevant enough to be included in this literature review.

4.3 Bayesian Networks

Bayesian networks, also named directed graphical models, is a type of PGM where the edges in the graph representation are directed and represent causal relationships between random variables or group of random variables (see figure 4.1). This concept was first introduced by Pearl in 1985 [Pea85], which uses Bayes' conditioning [Bay63] as the basis for updating information.

Bayesian networks follow the Bayesian approach to statistics and probabilities. In contrast to classical or physical probability, Bayesian probability (of an event) is a person's *degree of belief* in that event [Hec96]. While it may seen that a degree of belief is somewhat arbitrary or may lack precision and accuracy, multiple authors [Ram31, TK74, Sha88] argue that small variations in probability do not have a big influence in the decision making process and that measuring beliefs lead to the same rules of probability (which can be summarized with the product rule 4.1 and the sum rule 4.2 [Mac05]).

$$P(x,y \mid \mathscr{H}) = P(y \mid x,\mathscr{H})P(x \mid \mathscr{H})^{4} \tag{4.1}$$

$$P(x,\mathscr{H}) = \sum_{y} P(x \mid y,\mathscr{H})P(y \mid \mathscr{H}) \tag{4.2}$$

Formally [Pea88], a Bayesian network B represents a joint probability distribution (JPD) over a set of variables $\mathbf{U}$ and can be defined by a pair $B = \langle G, \Theta \rangle$. B is a DAG (directed acyclic graph) where the vertices represent the random variables $X_1, \ldots, X_n$. Θ represents the set of parameters

[4] $\mathscr{H}$: hypotesis or assumptions the probabilities are based

that quantify the network. For each possible value x_i of X_i, and $\prod_{x_i}$ of $\prod_{X_i}$ (set of parents of X_i in G), it contains a parameter $\theta_{x_i|\prod_{x_i}} = P_B(x_i \mid \prod_{x_i})$. Therefore, the JPD can be defined as

$$P_B(X_1, \ldots, X_n) = \prod_{i=1}^{n} P_B(X_i \mid \prod_{X_i}) = \prod_{i=1}^{n} \theta_{X_i|\prod_{X_i}} \tag{4.3}$$

which expresses the factorization properties of the JPD. [Bis06, section 8.1.] goes in detail on how to apply the equation 4.3.

These properties of Bayesian networks make it an excellent tool for expressing causal relationships. Heckerman [Hec96] lists multiple advantages of Bayesian networks on modelling and data analysis: “readily handles situations where some data entries are missing”, “gain understanding about a problem domain and to predict the consequences of intervention”, “ideal representation for combining prior knowledge and data” and “efficient and principled approach for avoiding the overfitting of data”.

Regarding the area of e-commerce specifically, some research has been done where Bayesian networks are applied. [NMK14] is an attempt at predicting sales in e-commerce using social media data. [MCGM02] also proposes a Bayesian based model to predict online purchasing behaviour using navigational clickstream data.

4.4 Markov Random Fields

Markov random fields (MRF) or Markov networks are undirected graphical models [Kin80] (in contrast to Bayesian networks which are directed and acyclic). The nodes still represent variables or group of variables however the links do not carry arrows. The concept was originally proposed as the general setting for the Ising model[5] [Kin80]. Again, Bishop [Bis06] provides a very good overview of this topic.

MRFs factorize as

$$p(x_1, \ldots, x_n) = \frac{1}{Z} \prod_{C \subset \mathfrak{C}} \psi_C(x_C) \tag{4.4}$$

where C is a clique[6] of the graph and x_C is the set of variables in that clique, Z is a constant used to normalize the distribution (might be defined for each x), ψ_C is a compatibility or potential function [WJ08, section 2.1.2] [Bis06, section 8.3]. The equation 4.4 highlights an important property of MRFs: the Markov property or memoryless property. That is, the conditional probability distribution of future states depends only on the present state.

Markov models were shown to be well suited for modelling and predicting e-commerce purchasing and user’s browsing behaviour [DK01].

[5]Ising model: mathematical model of ferromagnetism in statistical mechanics

[6]clique: fully connected subset of vertices

4.5 Hidden Markov Models

Hidden Markov models (HMMs) are a PGM with unobserved or hidden states. They are considered a dynamic Bayesian network[7]. They have been originally defined in the 60s by Baum and colleagues [BP66]. [Rab89] defines HMMs as "the resulting model (...) is a doubly embeded stochastic process that is not observable, but can only be observed though another set of stochastic processes that produce the sequence of observations.".

A common example found in literature is the Coin Toss Model [Rab89]: imagine someone on one side of a curtain performing a coin (or multiple coin) tossing experiment. The other person will not tell us about what she is doing, only the outcome of each coin flip (heads or tails). Multiple HMMs can be built to explain the coin toss outcomes, i.e, assuming that one, two or more biased coins are being used in the experiment. The figure 4.2 is a possible model that can account to 3 coins being tossed.

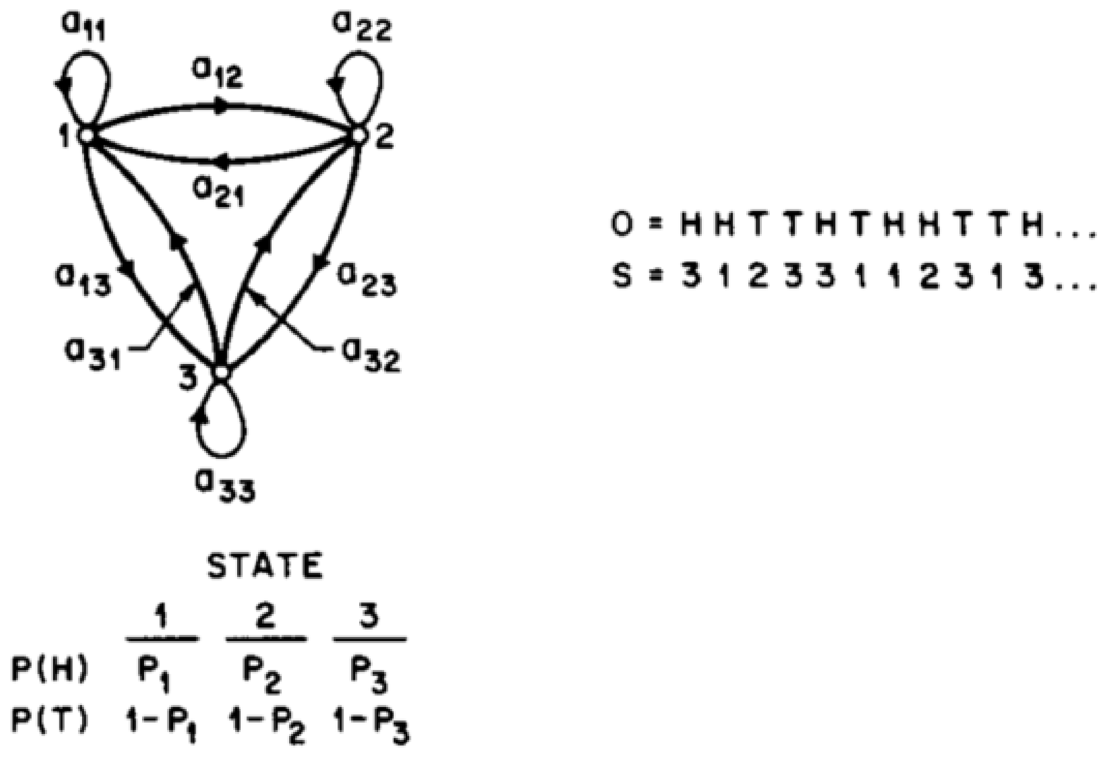


Figure 4.2: Example of a 3-coin model [Rab89]

A HMM is characterized by the following:

- N which is the number of states in the model where individual states are represented by $S = \{S_1, ..., S_N\}$ and the state at time t is q_t;

[7]dynamic Bayesian network: Bayesian networks adapted with time steps

- M which is the number of distinct observation symbols per state (individual symbols are represented by $V = \{V_1, ..., V_M\}$);
- $A = \{a_{i,j}\}$, the state transition probability distribution where

$$a_{ij} = p(q_{t+1} = S_j \mid q_t = S_i), 1 \leq i, j \leq N \tag{4.5}$$

- $B = \{b_j(k)\}$, the observation symbol probability distribution in state j:

$$b_j(k) = p(v_k \ at \mid q_t = S_j), 1 \leq j \leq N, 1 \leq k \leq M \tag{4.6}$$

- Finally, $\pi = \{\pi_i\}$, the initial state distribution:

$$\pi_i = p(q_1 = S_i), 1 \leq i \leq N \tag{4.7}$$

The formal model can be summarized as $\lambda = (A, B, \pi)$ [Rab89].

Multiple algorithms have been studied and applied to HMMs: for inference, the forward algorithm, forward-backward algorithm [BE+67] or the Viterbi algorithm [FJ05, MJ00]. Regarding learning, the algorithm Baum-Welch [BP66, BE+67] can be used.

Regarding e-commerce and web user behaviour there is some research done. [XY09] explains how to use a hidden semi-Markov model to detect anomalies on user browsing behaviour. [ADW02] describes very briefly a relational hidden Markov model for the behaviour of web site users, in order to improve predictions and personalization of websites.

4.6 Summary

In this section we reviewed the literature for graphical models. They provide a tool of excellence to model real world phenomena, enabling decision making under uncertainty and noisy observations.

There are multiple categories of graphical models however we focused on Bayesian and Markov networks and hidden Markov models, due to their applicability in the work at hand (in chapter 5 we will define how PGMs can be applied).

Chapter 5

Implementation

This chapter presents the implementation of the framework. It starts by describing the methodology followed in the realization of the dissertation. It is followed by the identification of requirements, the architecture of the framework, a study on the scalability of the solution and it is finalized by the description of the technology used in the implementation.

5.1 Methodology

Like any software development project, a simulation project also has a life cycle. In this section we describe the steps to apply in the simulation methodology, based on Ulgen et al. [UBJK94] and Banks et al. [BCNN04, section 1.11], which can be summarized as follows:

1. *Problem formulation*: Clear statement of the problem by the analyst and stakeholders;
2. *Setting of objectives and overall project plan*: Questions to be answered by the simulation, plans for the study, cost and number of days for each phase, with the results expected at each stage;
3. *Model conceptualization*: Select, modify and iterate over the assumptions that characterize the system;
4. *Data collection*: Collect the necessary data to run and validate the model, assuming that required data will change with the increasing complexity of the system;
5. *Model translation*: Materialization of the system in a program;
6. *Verification*: Making sure that the program behaves correctly accordingly to its inputs;
7. *Validation*: Calibration of the model, comparing the model against an actual system;
8. *Experimental design*: Tweak the experiments, comparing alternative designs;

9. *Production runs and analysis*: Estimate measures of performance for the systems that are being simulated;

10. *Documentation and reporting*: Document both the program and the progress of the study;

11. *Implementation*: End result of the study, including the entire simulation process.

This process can be visualized in figure 5.1.

5.2 Requirements

We have looked at several e-commerce websites, both national and worldwide, like Amazon[1], eBay[2], PCDIGA[3], Clickfiel[4], KuantoKusta[5], and analysed features and characteristics common to all of them, in order to better assess what the framework should be able to represent and model. To keep things simple and realistically implementable in the given time frame, some limitations had to be done. This section lists the requirements and assumptions of the framework.

5.2.1 Website Representation

- A website is a collection of web pages;
- The common entry point is named homepage but it is possible to enter the website directly from a different page;
- Structure and navigation between pages is done with links;
- Pages have a purpose like displaying information about a product, listing multiple products, informing about warranty and payment of products and services, etc.. We categorize the pages by using tags;
- Product pages have, at least, the product name, its description and price;
- A virtual shopping cart is used as a staging area for the products that are going to be bought;
- Checkout is the act of taking all the products in the shopping cart and effectively buying and paying them;
- Usually, a customer has to create an account and login in the website in order to buy something or access restricted pages.

[1] https://www.amazon.com/
[2] http://www.ebay.com/
[3] https://www.pcdiga.com
[4] http://clickfiel.pt/
[5] http://www.kuantokusta.pt/

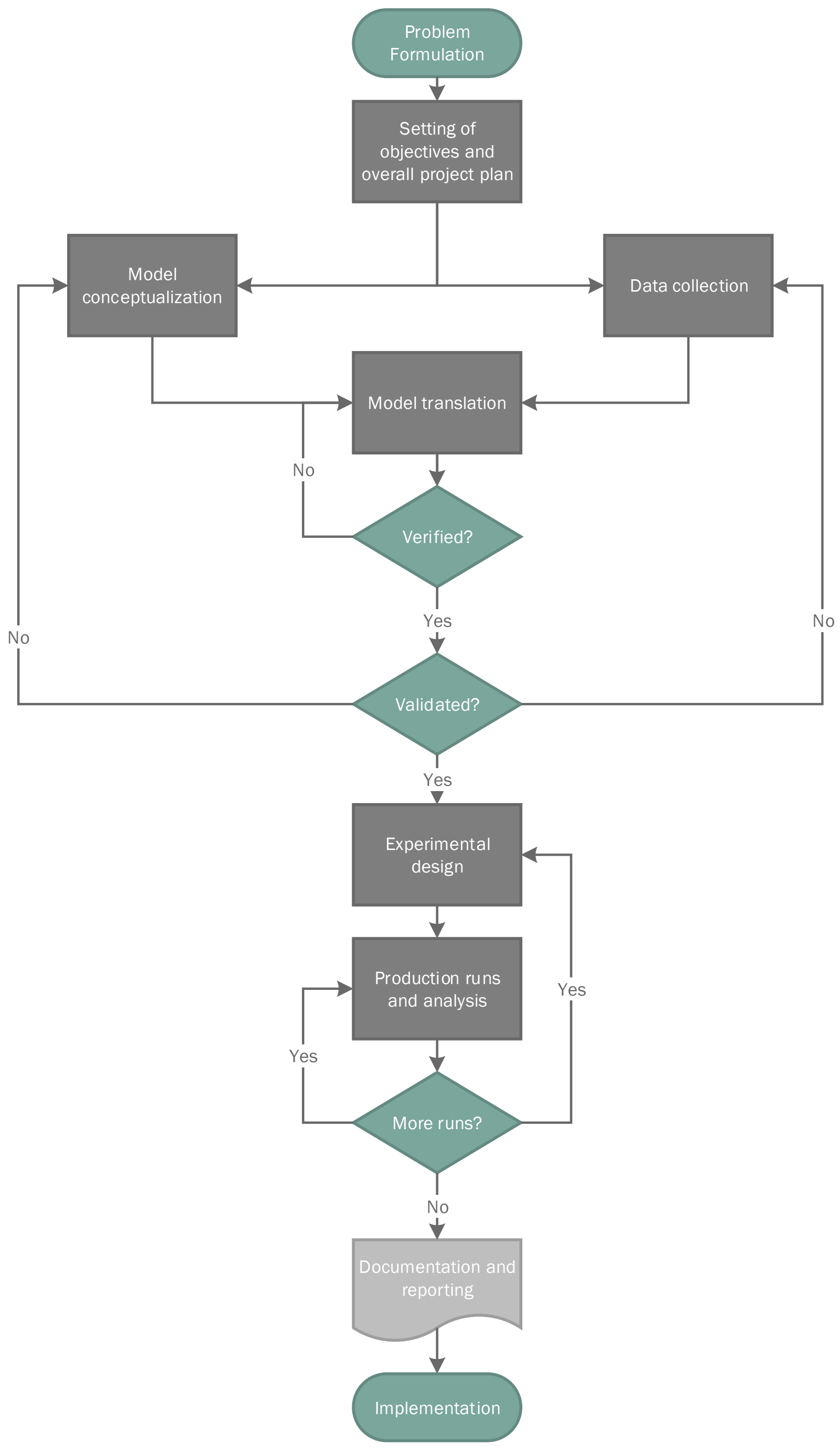


Figure 5.1: Steps in a simulation study [BCNN04]

5.2.2 Navigation Agents

- Navigation agents represent users or customers interacting with a website;
- Some common interactions are:
  - jumping from page to page (or browsing);
  - exiting the website;
  - adding a product to the shopping cart;
  - checking out;
  - rating a product;
  - writing a review or comment;
  - bidding on a product;
  - filling out forms (login, addresses, bank information, etc.);
  - comparing two products side by side.

5.2.3 Website Agents

- Website agents can modify any page before it is *served* to a user/customer;
- Example use cases:
  - Recommend products to the user based on its preferences or browsing behaviour;
  - Targeted flash sales or promotions;
  - A/B testing analysis.

5.2.4 Simulation Engine

- Given a website, the type of navigation and website agents and pretended simulation time, the simulation can be started, stopped and store its state and calculated metrics in a database.
- A simulation run can have thousands of navigation agents entering the simulation at each step;
- A simulation run can have one or more website agents.

5.2.5 Reporting

- Once a simulation run ends, it can be analysed by taking a look at its results, metrics and other previously stored characteristics;
- At least two simulation runs can be put side and by side so they can be quickly compared;
- The calculated metrics should be relevant to the business, some examples are [Wat15]:

  - Bounce rate
  - Conversion rate
  - Total/average order value
  - Average order value
  - Items per order
  - New visitor conversion rate
  - Shopping cart sessions
  - Shopping cart conversion rate
  - Shopping cart abandonment rate
  - Average session length
  - Number of browsing sessions
  - Page views per session
  - Product views per session

5.2.6 Limitations

Some requirements did not make it to the actual implementation:

- Adding a product to the cart and the checkout are a single step;
- There are no customer accounts, logins, registration, sign ups or sign ins;
- Visual information about the pages and products is not represented, e.g., a customer cannot pick a product to buy because its associated picture is *appealing*;
- It is not possible to remove an item from the cart;
- Interactions with the website are limited and "hard coded" (listed in sub-section 5.3.1), not extensible;
- The metrics gathered during the simulation are limited, we have implemented some of the metrics listed above, non exhaustively.

5.3 Architecture

5.3.1 Multi-agent Architecture

The simulation framework encompasses two different kinds of agents, navigation agents and website agents, as shown in figure 5.2.

Navigation agents represent users interacting with the website. They have a limited view of the system: they have access to the website (pages and links between them) and they know the

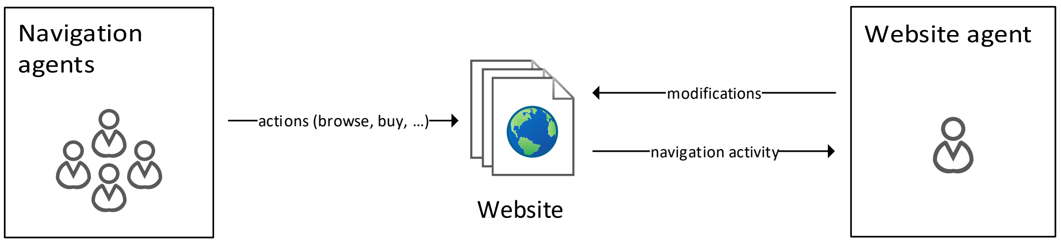


Figure 5.2: Agent interaction with the environment

current page they are visiting. Each simulation step, the framework asks each navigation agent which action will they pick. The action may be to visit another page (`BrowseToAction`), exit the website (`ExitAction`), add a product to the cart (`AddToCartAction`), finish the purchase (`CheckoutAction`) or simply do nothing (`IdleAction`). Also related to the navigation agents subsystem, an implementation of `NavigationAgentFactory` is used to decide how many navigation agents are added to the system in each step. For example, a simplistic implementation might create a fixed number of navigation agents or a different one closer to reality could follow a Poisson distribution model [GÖ03].

Website agents are able to modify the pages before they are served to the users. They have a broader view of the system than navigation agents. They are notified of all the actions that navigation agents do. The most common use case of the website agents is to recommend products to the users: before the page is served to a user, a website agent can modify a section of the page to display a custom list of products, based on the previous activity of the other users or preferences of the current user. However they are not limited to only recommendations, a website agent might replace a page's content entirely, increase or decrease the price of products (e.g promotions, sales), do nothing, etc.

The framework does not assume how these agents behave however the interactions between them are limited. The agents do not send messages between each other and may only interact indirectly, through the framework (e.g a website agent modifies a page before it is "seen" by a navigation agent). While a simulation run might have hundreds or thousands of navigation agents, to simplify, each run only has one website agent instance (this does not impose a limit on the solution, the agent can still be modelled after a composite agent[6]).

It is out of the scope of the framework to provide concrete implementations of the agents but we provide 2 implementations of navigation agents and 3 implementations of website agents, as a way to validate and verify the simulation runs. This will be further discussed in chapter 6.

[6]An agent that represents multiple composite or virtual agents (our name)

5.3.2 Simulation Engine

The simulation engine follows a fairly standard and simple discrete event simulation architecture, as described in 3.3. The domain model we are dealing with allows certain simplifications of the simulation:

- the event list only contains events scheduled for the next step;
- there are no conditional events (type C [Pid98]);
- all the events happen instantaneously;
- the events do not depend on other events, they do not require synchronization and may be implemented in a single-threaded engine.

The process that the simulation engine follows is described next. In each simulation loop, the engine starts by calling NEWNAVIGATIONAGENTS() which adds new navigation agents to the simulation. The number and type of these agents are decided by the `NavigationAgentFactory`. After that, each navigation agent currently active (i.e did not leave the website) chooses an action to do (buy, browse, etc.). Depending on the action that was picked, the engine updates its internal state. The simulation state is represented by `WebsiteState` and contains statistics and other performance metrics. Whenever the picked action implies presenting the navigation agent a page from the website, the website agent can modify that page before it is presented, by calling MODIFYPAGE(NAVAGENT, PAGE). The website agent is also notified about all actions that the navigation agents do (NOTIFY(NAVAGENT, ACTION)). The simulation is configured to end after a fixed number of steps, otherwise it could run forever.

This process is illustrated in figure 5.3.

5.3.3 Class Model

In this sub-section we describe all the classes used to represent all the entities in the simulation engine (figure 5.4).

`Website` represents a website, it contains a set of `pages` and a reference to its `homepage`, the entry point of the website. A `Page` has a set of `links`, which are all the outbound hyperlinks that a page contains, it has a set of `tags`, which is used to categorize a page (e.g electronics category, clothing category, cart page, product search page, etc.) and the page may also contain a `Product`, if the page is a product page. A `Product` has a `name`, a `description` and a `price`.

The `Simulation` is an abstract class that contains an `agenda` which stores all the `Actions` (an arbitrary function) to be executed in the next steps. It provides a way to enqueue work in the simulation using SCHEDULE(DELAY, ACTION) and a RUN() method that consumes the `agenda` until there's no more work to do. A subclass of `Simulation`, `WebsiteSimulation` represents a simulation happening over websites. It contains the `Website` itself, a `WebsiteState`, a `NavigationUserFactory` and a `WebsiteAgent`. The `WebsiteState` is used to keep track

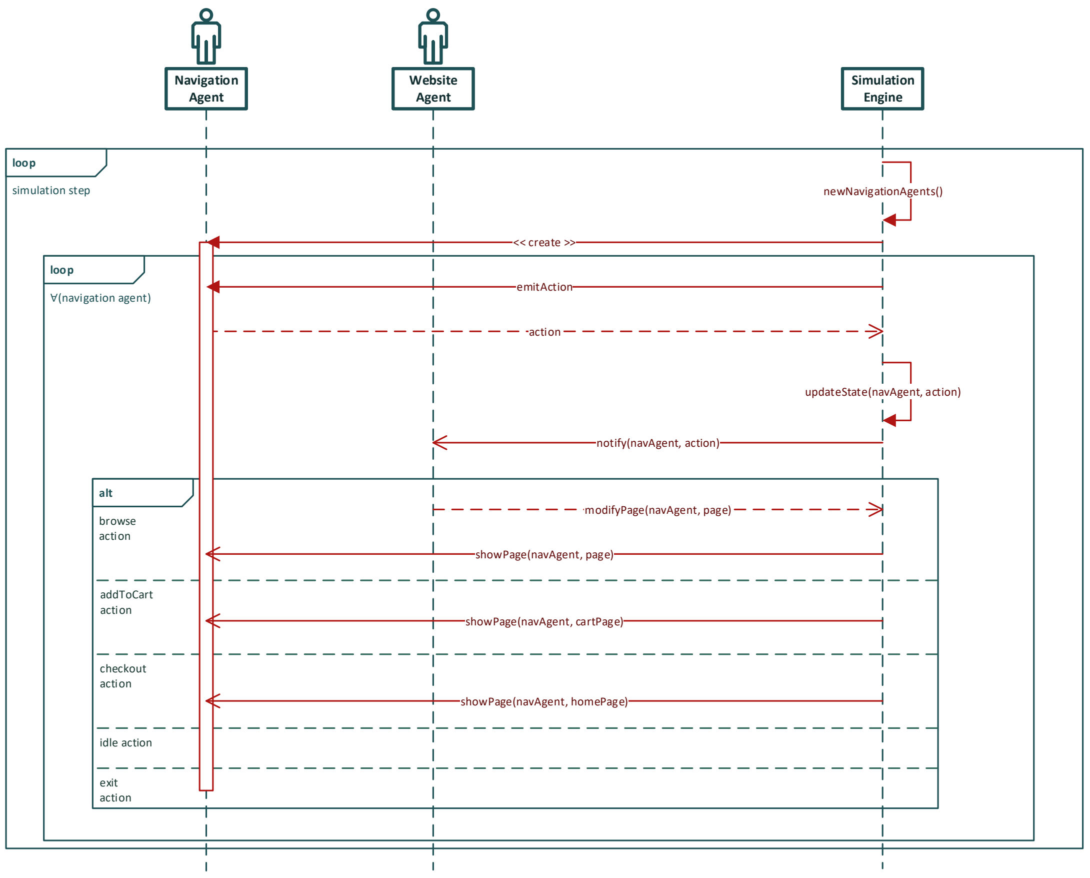


Figure 5.3: Sequence diagram for the simulation engine

of all the statistics and metrics that the simulation produces. This state can be stored in a database to analyse the results once the simulation is finished.

`NavigationAgent` is an interface that represents users interacting with the website. Implementations of it have to implement EMITACTION, which returns the `Action` the agents wants to do based on their internal state and their current page. These agents are added to the simulation by an implementation of `NavigationAgentFactory`. `WebsiteAgent` is an interface that represents the agents that may modify the website and that are notified about all the navigation agent activity. The code for these three interfaces is displayed in listing 5.1.

The mutability of the system is contained to the `Simulation` (due to its `agenda`) and `WebsiteState` which is updated every simulation step.

The points of extensibility of the framework are the agents interfaces (`NavigationAgent`, `NavigationAgentFactory` and `WebsiteAgent`) and `WebsiteState` (e.g provide additional tracking metrics or visualizations).

```scala
trait NavigationAgentFactory[+T] {
def users: Iterator[List[T]]
}

trait NavigationAgent {
def emitAction(currentPage: Page, website: Website): Action
}

trait WebsiteAgent {
def modifyPage(page: Page, user: NavigationAgent): Page
def notifyUserAction(user: NavigationAgent, currentPage: Option[Page],
action: Action)
}
```

Listing 5.1: Definition of the agents interfaces

5.3.4 Graphical User Interface

A frontend website has been developed to aid in displaying and visualizing the results of each simulation run. The data is loaded asynchronously from a database which stores `WebsiteState` snapshots. All the data is rendered to the user server-site except the data required to display charts (e.g Visits per Category chart).

The interface has three distinct views: a simulation list, details about a simulation run and comparison between two simulation runs:

- The simulation list view (`GET /simulations`) (figure A.1) displays a table with all the simulation runs stored in the database. It shows the identifier, name, agent types and timestamp of each run.
- The detail view (`GET /simulations/<id>`) (figure A.2) display information regarding a single simulation run. This info describes the simulation and it contains data regarding the types of the agents used, start and finish time of the simulation, collected metrics (e.g bounce rate, conversion rate, total order value, etc.), visits per page, visits per page category, purchases per product and others. This information is displayed using mostly tables and charts.
- The last view, the comparison page (`GET /simulations/compare/<idA>/<idB>`) (figure A.3) displays information regarding two simulation runs (A and B) side by side, so they can be compared and analysed. The planned use case of this view is to quickly spot differences between two runs and see how different agent configurations affect the results.

5.4 Scalability

To assess the scalability and performance of the simulation engine, some benchmarks were made and they are described next. The tests were ran in a Windows 10 laptop with a Intel® Core$^{\mathrm{TM}}$ i7-

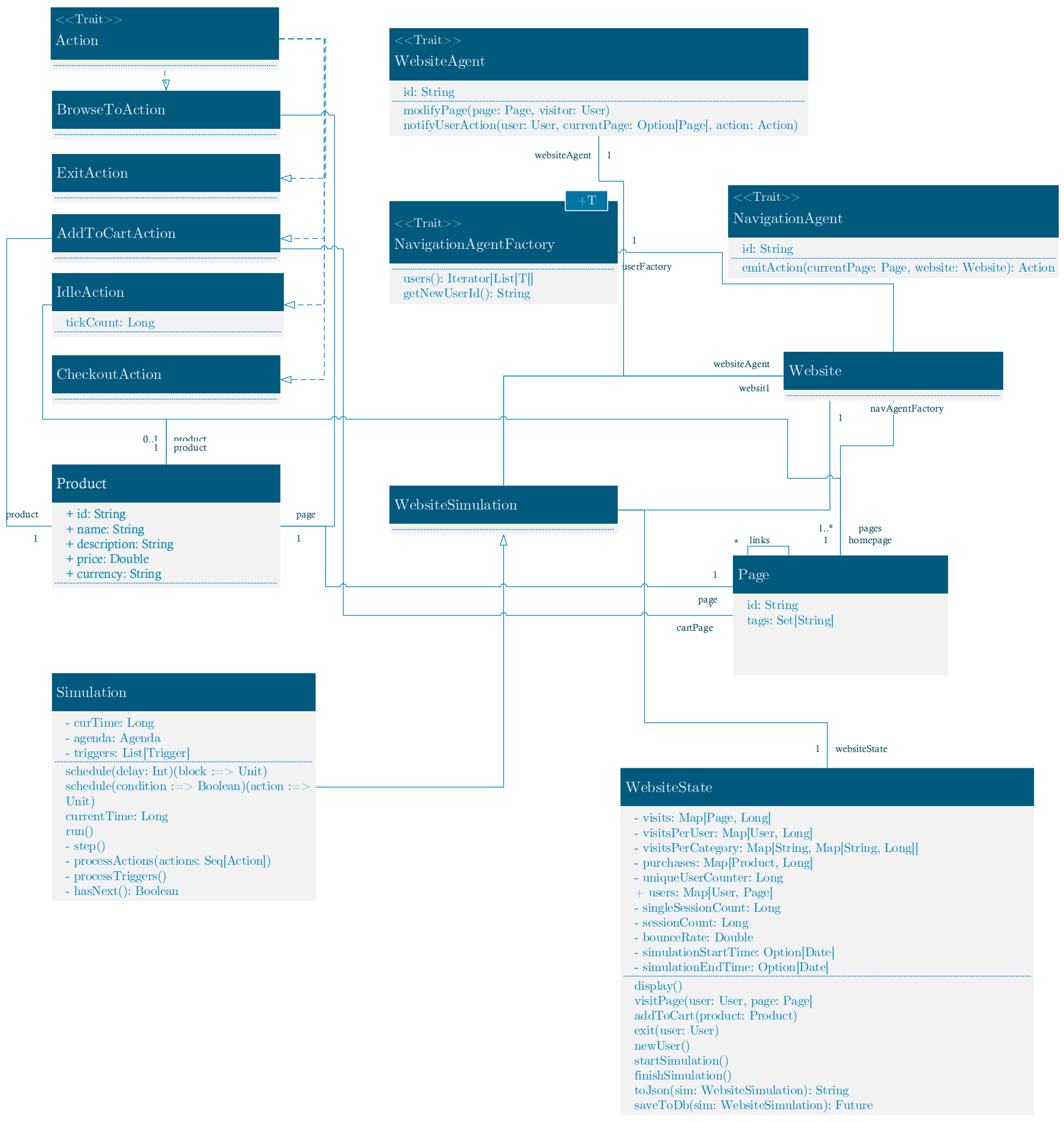


Figure 5.4: Class diagram

4710HQ CPU @ 2.50GHz (8 CPUs) processor. A modified[7] version of the library Benchmark.scala[8] was used, which is based on Ruby's Benchmark module[9]. The focus is not necessarily in the raw speed of the engine but rather in the variation of the simulation time when the number of agents in the system or the number of steps of the simulation are increased.

The test performed consists of running the same simulation with an increasing number of navigation agents and number of simulation steps, set up in the following way:

[7]Changed each measurement to run the same block of code 10 times, drop the first 2 runs and take the average of the 8 runs instead of running it only once.

[8]https://github.com/balagez/Benchmark.scala

[9]http://ruby-doc.org/stdlib-1.9.2/libdoc/benchmark/rdoc/Benchmark.html

- **Website**: Toy sample website with 9 pages and 32 total links between pages (1 homepage, 1 cart page, 3 product list pages and 4 product pages);
- **Website agent**: Dummy agent, does not modify any page;
- **Navigation agent**: Sample agent implementation which picks the next action randomly. Configured with a chance of exiting the website of $\frac{1}{3}$ and a change of adding a product to the cart of $\frac{1}{20}$;
- **Number of navigation agents**: From 1000 to 10000 with increments of 1000;
- **Number of simulation steps**: From 100 to 1000 with increments of 100.

The result of the 100 simulation runs is shown in figure 5.5 (whose data is in table 5.1). A quick analysis shows that the simulation time scales linearly ($\bar{R^2} = 0.99149, \sigma = 0.00648$) with both the number of agents and the number of simulation steps. For instance, a simulation with 1000 steps and 10000 navigation agents (entering the system each step) took 41.95 seconds. These initial results are very satisfactory however they should be improved, especially when the number of steps is increased, so that simulations that span a longer period of time can be evaluated (e.g simulate the effect of seasonal customers over an entire year).

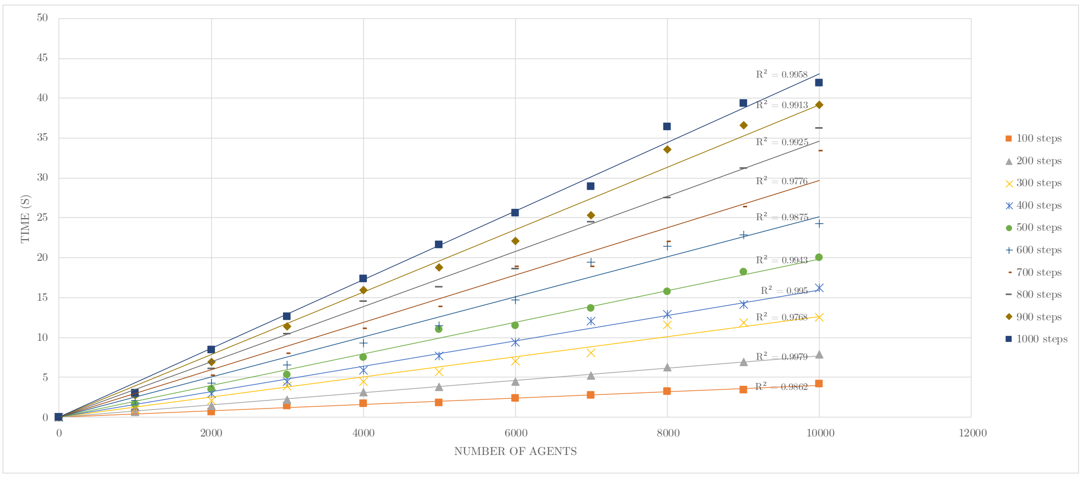


Figure 5.5: Simulation running time for different number of navigation agents and simulation steps

5.5 Technology

Scala[10] was the language of choice to implement the framework and accompanying projects. Scala is a statically typed, general purpose programming language that leverages both object oriented

[10] http://www.scala-lang.org/

Table 5.1: Simulation running time (in seconds) for different number of navigation agents and simulation steps

Steps \ Agents	1000	2000	3000	4000	5000	6000	7000	8000	9000	10000
100	0.62	0.69	1.45	1.72	1.81	2.34	2.76	3.23	3.42	4.16
200	0.67	1.43	2.20	3.17	3.84	4.49	5.22	6.30	6.93	7.89
300	0.95	2.07	3.90	4.44	5.72	7.01	8.05	11.62	11.89	12.53
400	1.32	2.91	4.42	5.88	7.69	9.39	12.01	12.93	14.15	16.18
500	1.58	3.48	5.34	7.49	11.03	11.44	13.65	15.77	18.20	20.00
600	2.10	4.31	6.59	9.26	11.52	14.71	19.44	21.47	22.83	24.29
700	2.55	5.24	7.97	11.14	13.89	18.91	18.89	22.04	26.33	33.35
800	2.56	6.10	10.46	14.48	16.31	18.56	24.49	27.52	31.19	36.26
900	2.77	6.94	11.37	15.97	18.77	22.11	25.36	33.56	36.57	39.21
1000	3.07	8.47	12.64	17.40	21.59	25.62	28.94	36.46	39.40	41.95

and functional programming paradigms, while being fully interoperable with the JVM (and Java).

The version of Scala used was 2.11.8 with sbt 0.13.11[11], the *de facto* build tool for Scala projects.

The library Breeze (version 0.12)[12] of the ScalaNPL[13] package was used for numerical processing and statistics.

Apache Spark™ 1.5[14] was used to train recommendation models in one of the implementations of website agents.

GraphStream 1.3[15] was used to visualize websites as a dynamic graph.

To store simulation run results, the database MongoDB 3.2.3[16] was used due to its practicality and rapid development.

The frontend was built with the Play Framework 2.5[17].

[11]http://www.scala-sbt.org/
[12]https://github.com/scalanlp/breeze
[13]http://www.scalanlp.org/
[14]http://spark.apache.org/
[15]http://graphstream-project.org/
[16]https://www.mongodb.com/
[17]https://www.playframework.com/

Chapter 6

Validation

To validate the framework 2 testbeds were prepared. The first is a collection of small fabricated test cases where we compare the output of multiple simulation runs to the expected results. The second case deals with a real use of the framework, applied to an online store.

6.1 Sanity checks

6.1.1 Expected number of agents in the simulation

This test compares the number of navigation agents expected to be *alive* at each simulation step with the actual number of them.

At each simulation step, k navigation agents enter the system and p_{exit} of them leaves, which leads to the recurrence equation 6.1.

$$\begin{cases} a_1 = (1-p_{exit})\,k \\ a_n = (1-p_{exit})\,(k+a_{n-1}) \end{cases} \Leftrightarrow a_n = \frac{k(p_{exit}-1)((1-p_{exit})^n-1)}{p_{exit}} \tag{6.1}$$

The simulation run was configured in the following way:

- **Website**: Sample website with 9 pages and 32 total links between pages
- **Website agent**: Dummy agent, does not modify any page;
- **Navigation agent**: Sample agent implementation with a chance of exiting the website of $\frac{1}{3}$ (p_{exit});
- **Number of new navigation agents each step**: 100 (k)
- **Number of simulation steps**: 1000

Replacing the values in equation 6.1: $a_n = -200\left(\frac{2}{3}^n - 1\right)$. After a few simulation runs, the expected number of agents in the system stabilizes: $\lim_{n\to\infty} -200\left(\frac{2}{3}^n - 1\right) = 200$.

The results of a simulation run were gathered and plotted in figure 6.1. Triangles (△) represent the actual value (A_t) and circles (●) represent the expected value (E_t) according to the equations above. SMAPE (symmetric mean absolute percentage error)[Mak93] is used to measure the accuracy of the results: $SMAPE = \frac{1}{n}\sum_{t=1}^{n}\frac{|E_t - A_t|}{|A_t|+|E_t|} = 3.07\%$, which is a reasonable low *error* rate given the randomness of the system.

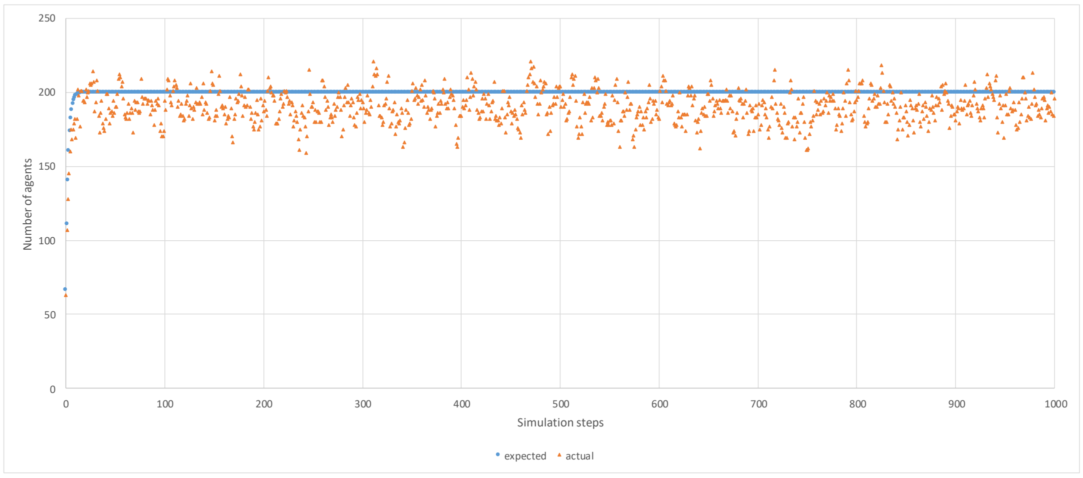


Figure 6.1: Expected number of agents in the simulation

6.1.2 Expected number of visits

This test compares the number of visits (page hits) for a given website and agents setup. The simulation run was configured in the following way:

- **Website**: Website configured as displayed in figure 6.2. The homepage links to 5 product pages and the product page link to the cart page.
- **Website agent**: Dummy agent, does not modify any page;
- **Navigation agent**: The agent picks one linked page randomly, however, if current page is for a product, it always buys it. If the current page is the cart page, it leaves the website;
- **Number of new navigation agents each step**: 100
- **Number of simulation steps**: 1000

The table 6.1 displays the expected and observed number of visits for a simulation run as described above. The percent error is calculated and the obtained results are very close to the predicted values.

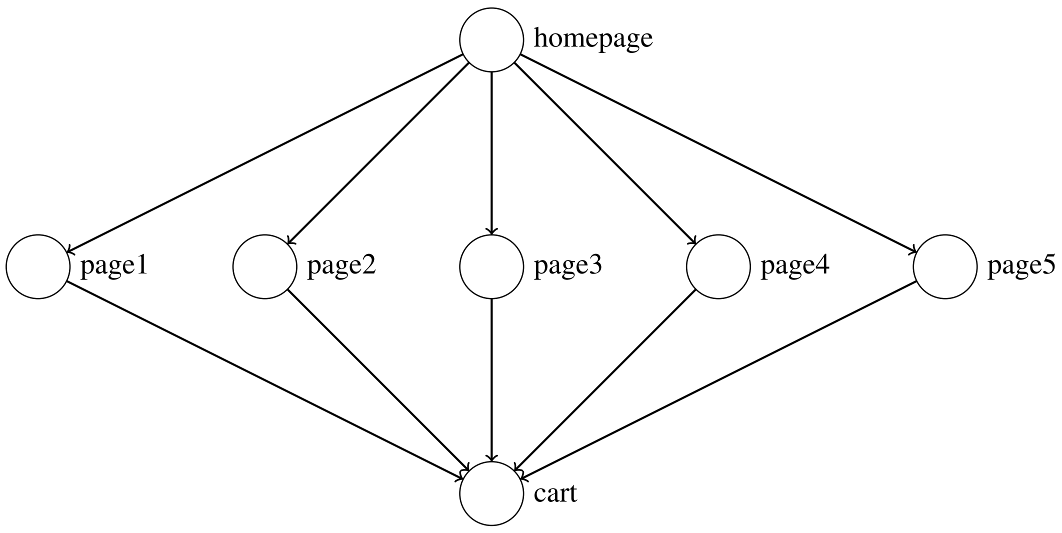


Figure 6.2: Website graph for the expected visits test

Table 6.1: Expected and observed number of visits

Page	Observed	Expected		Error
homepage	100000	100×1000	$= 100000$	0.00%
page1	19864	$\frac{1}{5} \times 100 \times 1000$	$= 20000$	0.68%
page2	20100	$\frac{1}{5} \times 100 \times 1000$	$= 20000$	0.50%
page3	19696	$\frac{1}{5} \times 100 \times 1000$	$= 20000$	1.52%
page4	20096	$\frac{1}{5} \times 100 \times 1000$	$= 20000$	0.48%
page5	20244	$\frac{1}{5} \times 100 \times 1000$	$= 20000$	1.22%
cart	99900	20000×5	$= 100000$	0.10%

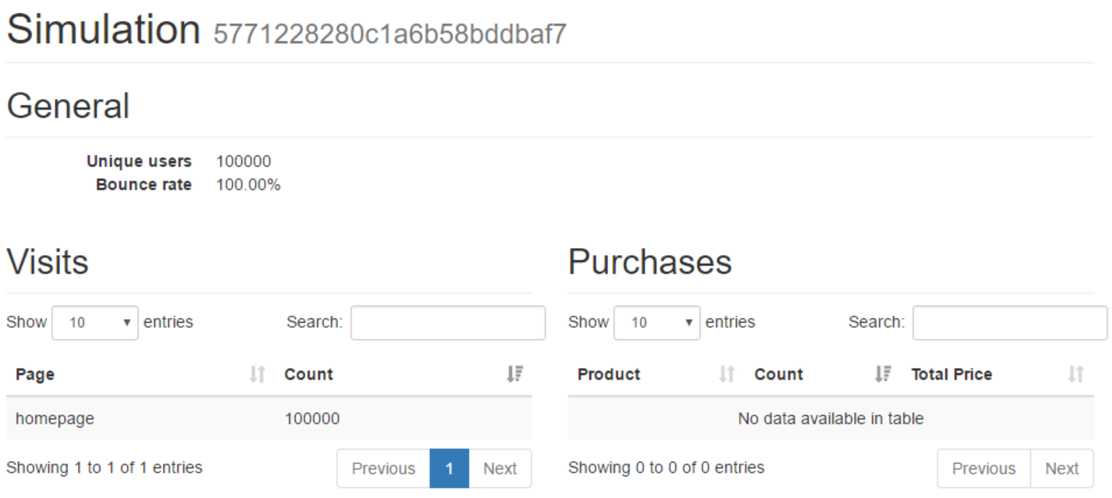
6.1.3 Expected bounce rate

This case compares the bounce rate for a website that only has one page. We define the bounce rate as the percentage of navigation agent sessions that only view a single page before existing the website.

The simulation run was configured in the following way:

- **Website**: One page only, the homepage;
- **Website agent**: Dummy agent, does not modify any page;
- **Navigation agent**: Agent that picks its actions randomly;
- **Number of new navigation agents each step**: 100
- **Number of simulation steps**: 1000

As expected, the simulation results yield 100% bounce rate, all of the visits were to the homepage, 10000 unique users (100×1000) and no purchases, as it can be seen on figure 6.3.


Simulation 5771228280c1a6b58bddbaf7

General

Unique users	100000
Bounce rate	100.00%

Visits

Show 10 entries Search:

Page	Count
homepage	100000

Showing 1 to 1 of 1 entries

Purchases

Show 10 entries Search:

Product	Count	Total Price
No data available in table		

Showing 0 to 0 of 0 entries

Figure 6.3: Screenshot of the frontend results for this test run

6.2 Online store

This test case uses data from a real online store that sells electronics and computers products. This website presents a fairly standard online store, mostly consisting of product listing and product pages. There are 3 places where it is possible to recommend products: the homepage has two sections, one with product highlights and another with product promotions and each product page has a tab to show related products.

6.2.1 Input data and configuration

The website consists of 2540 pages with 343201 links between pages, spanning 25 base product categories and 103 sub-categories. There are 750 product list pages, 1748 product pages, 1 cart page and 41 uncategorised/generic pages, visualised in figure 6.4.

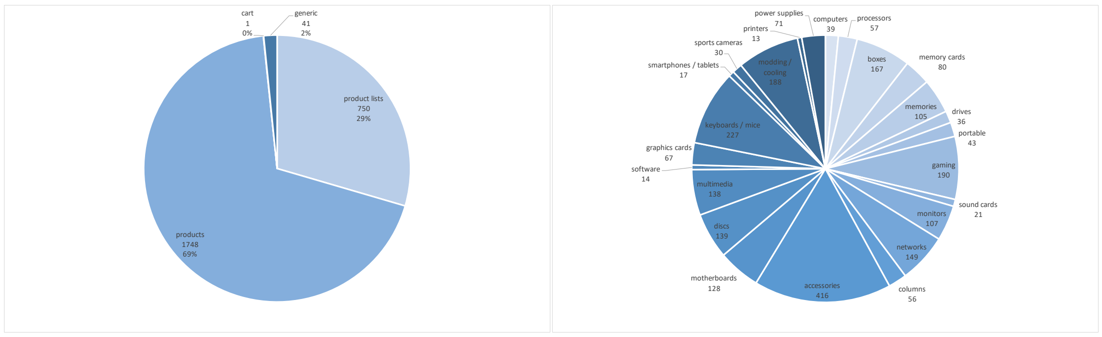


Figure 6.4: Distribution of the type of pages in the website (left) and distribution of the categories of the products (right)

To simulate users and customers (the `NavigationAgents`) interacting with this particular website, a model based on affinities was built. This model is composed by the *affinities* themselves (a mapping between product categories and the likelihood of the user liking or having interest on products of that category), the probability of buying a product, the probability of exiting the website and the arrival rate.

Because real usage website data is not available for this website, a sample profile was created with the following properties: the affinities were set up as displayed in table 6.2, probability of buying set to 5%, probability of leaving the website of 15% and a rate of arrival to the website following a *Poisson* distribution with $\lambda = 500$.

Table 6.2: Affinities for a sample user

Category	**Weight**
Computadores	14.29%
MSI	14.29%
Pen Drives	7.14%
Portáteis	14.29%
Intel 2011	14.29%
Cartões de Memória	7.14%
Brand	14.29%
Processadores	14.29%

6.2.2 Simulation

The simulation was configured as described in the subsection above. All the navigation agents use the same profile. The "thought" process for each agent is fairly simple: at each step, they try to buy a product and exit the website in accordance to the probabilities defined *a priori* or navigate to a different page based on their categories, with preference as stated by the affinity table. The simulation was run for 30 steps.

6.2.3 Results

The results of a sample simulation run are summarized in the tables 6.3 and 6.4. They are expected: the number of unique users is 14894 and the expected value is 15000 (500×25); the bounce rate is 14.58% and the prior leaving rate is 15%; and the conversion rate is 4.77% and the prior buy rate is 5%.

Table 6.3: Visits per category for a sample simulation run

Category	**Sub-category**	**Count**
Cartões de Memória	Pen Drives	6492
	SD/MiniSD/MicroSD	1203
	Leitor de Cartões	1199
	Compact Flash	1158
	-	37
Portáteis	MSI	14326
	HP	2623
	Asus	2584
	-	263
Processadores	Intel 2011	7097
	Intel 1151	2752
	Intel 1150	2379
	AMD	2234
	-	240
Computadores	Brand	19802
	-	188
Motherboards	Intel 2011	4917

Table 6.4: Metrics/info regarding a sample simulation run

Field	**Value**
Unique users	14894
Bounce rate	14.58%
Conversion rate	4.77%
Purchases	676
NavAgentFactory	AffinityFactory
NavAgent	AffinityUser
WebsiteAgent	DummyWebsiteAgent
Start time	Thu Jul 07 14:14:36 BST 2016
End time	Thu Jul 07 14:14:39 BST 2016

Chapter 7

Conclusion and Future Work

This chapter concludes the work realized for the dissertation, it presents the main contributions and proposes future work on the framework.

7.1 Overview & Main Contributions

The main results obtained in the realization of this dissertation are summarized as follows:

- Implementation of a framework capable of running a multi-agent simulation applied to the problem of modelling users interacting with an e-commerce website;
- A novel way to exploit multi-agent interaction, in this context, by using two representation of agents, the navigation agents (i.e users) and website agents (i.e recommendation engines), which interact with each other indirectly and form a feedback loop;
- A tool which appeals to both the academic community and the industry. The framework can be used, for example, to validate and test recommendation engines and algorithms or be used by an e-commerce company to optimize their own platform (e.g A/B testing);
- Implementation available to the community, licensed under MIT and hosted on GitHub[1].

7.2 Future Work

The implementation of the framework should be seen as a foundation for further development. There are certain limitations and assumptions in the developed model that should be resolved in order for the tool be even more useful and usable than it currently is. In no particular order, we list some ideas for the future:

[1]https://github.com/DDuarte/Manchester

- **Parallel simulator**. The implemented discrete event simulator is single threaded and it consumes the events sequentially, which limits the size of the simulation as seen in section 5.4. The implementation could change to a parallel simulator which hopefully processes more events in the same time frame, taking advantage of multi-core setups.
- **More metrics**. We have implemented only a handful of metrics to be calculated after each simulation run, however, there is a myriad of other metrics and statistics that we have not looked at. Further development could increase the pool of available metrics.
- **Metrics extensibility**. Related to the point above, current implementation *hardcodes* the calculation of certain metrics in the framework itself and it is not very practical to extend and add new metrics to the system. The framework could be modified to ease the process of adding new metrics, the visitor design pattern [Gam95] seems particularly well suited for this task.
- **Metrics for website agents**. While there are plenty of metrics for the navigation agents (i.e users/consumers), metrics for website agents (i.e recommendation engines) were overlooked and are not present in the current implementation. It might be useful to gather metrics and statistics regarding the behaviour of website agents.
- **Hypothesis testing**. Especially relevant when comparing two simulation runs, simply comparing single numeric metrics side by side might not be the best approach. In the field of statistical hypothesis testing (e.g A/B testing) there has been plenty of research in which standard tests to use for each case. For example, to compare conversion rates Fisher's exact test could be used however to compare the number of products bought a χ^2 test would be more appropriate [Wik16].
- **Limited number of actions**. The actions emitted by the navigation agents are finite and not exhaustive. The framework does not currently support extending the number of actions.
- **Visual aspects**. Pages are currently represented by their name/URL, tags and links, leaving no space to represent visual information and other meta-data. A navigation agent cannot use visual properties of the pages or products (usability, aesthetics) to decide on which action to do. If the intention is to model human behaviour with fidelity, this might be a major hindrance. A future version of the framework should take into account these aspects however it requires further research, since it is not obvious how to model and represent these concepts.

References

[ADW02] Corin R Anderson, Pedro Domingos, and Daniel S Weld. Relational Markov Models and their Application to Adaptive Web Navigation. pages 143–152, 2002.

[AI16] Inc. Alexa Internet. Alexa - top sites by category: Shopping, 2016. [Online; accessed 24-June-2016].

[Ama15] Xavier Amatriain. How do you measure and evaluate the quality of recommendation engines?, 2015. Available on `http://qr.ae/RUNcIK`, accessed last time at 14 of February 2016.

[Any00] AnyLogic. Anylogic simulation software, 2000. Available on `http://www.anylogic.com/`, accessed last time at 14 of February 2016.

[AT05] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

[Bay63] Mr Bayes. An essay towards solving a problem in the doctrine of chances. by the late rev. mr. bayes, frs communicated by mr. price, in a letter to john canton, amfrs philosophical transactions, 53: 370–418. *URL http://rstl. royalsocietypublishing. org/content/53/370. short*, 1763.

[BCNN04] Jerry Banks, John Carson, Barry L Nelson, and David Nicol. Discrete-Event System Simulation. *PrenticeHall international series in industrial and systems engineering*, page 624, 2004.

[BE$^+$67] Leonard E Baum, John Alonzo Eagon, et al. An inequality with applications to statistical estimation for probabilistic functions of markov processes and to a model for ecology. *Bull. Amer. Math. Soc*, 73(3):360–363, 1967.

[Bis06] Christopher M Bishop. Pattern recognition. *Machine Learning*, 2006.

[BP66] Leonard E. Baum and Ted Petrie. Statistical Inference for Probabilistic Functions of Finite State Markov Chains. *The Annals of Mathematical Statistics*, 37(6):1554–1563, 1966.

[Bra14] Sally Brailsford. Modeling Human Behaviour - An (ID)entity Crisis? *Proceedings of the 2014 Winter Simulation Conference*, (Id):1539–1548, 2014.

[Col03] Nick Collier. Repast: An extensible framework for agent simulation. *The University of Chicago's Social Science Research*, 36:371–375, 2003.

[Con04] Efthymios Constantinides. Influencing the online consumer's behavior: the Web experience. *Internet Research*, 14(2):111–126, 2004.

[DGW13] Tom Dickman, Sounak Gupta, and Philip A. Wilsey. Event pool structures for pdes on many-core beowulf clusters. In *Proceedings of the 1st ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, SIGSIM PADS '13, pages 103–114, New York, NY, USA, 2013. ACM.

[Dia16] João Pedro Matos Teixeira Dias. Reverse engineering static content and dynamic behaviour of e-commerce websites for fun and profit, July 2016.

[DK01] M Deshpande and G Karypis. Selective Markov Models for Predicting Web Page Access. *Proc. of First SIAM Intl Conf on Data Mining*, 4(2):163–184, 2001.

[FJ05] G David Forney Jr. The viterbi algorithm: A personal history. *arXiv preprint cs/0504020*, 2005.

[FRJ11] Pau Fonseca i Casas, Miquel Ramo Nñerola, and Angel A. Juan. Using specification and description language to represent users' profiles in OMNET++ simulations. *Proceedings of the 2011 Symposium on Theory of Modeling & Simulation: DEVS Integrative M&S Symposium*, 2011.

[Fuj90] Richard M. Fujimoto. Parallel discrete event simulation. *Commun. ACM*, 33:30–53, 1990.

[Gam95] Erich Gamma. *Design patterns: elements of reusable object-oriented software*. Pearson Education India, 1995.

[GÖ03] Şule Gündüz and M Tamer Özsu. A poisson model for user accesses to web pages. In *Computer and Information Sciences-ISCIS 2003*, pages 332–339. Springer, 2003.

[Hec96] David Heckerman. A Tutorial on Learning With Bayesian Networks. *Innovations in Bayesian Networks*, 1995(November):33–82, 1996.

[HOP$^+$86] James O Henriksen, Robert M O'Keefe, C Dennis Pegden, Robert G Sargent, Brian W Unger, and Douglas W Jones. Implementations of time (panel). *Proceedings of the 18th conference on Winter simulation*, pages 409–416, 1986.

[J S00] M Deshpande J Srivastava, R Cooley. Web Usage Mining : Discovery and Applications of Usage Patterns from Web Data. *ACM SIGKDD Explorations Newsletter 1.2*, 1(2):12–23, 2000.

[Jen99] Nicholas R Jennings. Agent-based computing: Promise and perils. 1999.

[Jon86] Douglas W. Jones. An empirical comparison of priority-queue and event-set implementations. *Commun. ACM*, 29(4):300–311, April 1986.

[KF09] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[Kin80] Markov random fields and their applications, 1980.

[Kiv69] P.J. Kiviat. Simscript ii programming language (automatic computation), 1969.

[KKK13] Aditya Kurve, Khashayar Kotobi, and George Kesidis. An agent-based framework for performance modeling of an optimistic parallel discrete event simulator. *Complex Adaptive Systems Modeling*, 1(1):12, 2013.

[Mac05] David J C MacKay. *Information Theory, Inference, and Learning Algorithms David J.C. MacKay*, volume 100. 2005.

[MAFM99] Daniel A. Menascé, Virgilio A. F. Almeida, Rodrigo Fonseca, and Marco A. Mendes. A Methodology for Workload Characterization of E-commerce Sites. *Proceedings of the 1st ACM conference on Electronic commerce - EC '99*, pages 119–128, 1999.

[Mak93] Spyros Makridakis. Accuracy measures: theoretical and practical concerns. *International Journal of Forecasting*, 9(4):527–529, 1993.

[MCGM02] Wendy M Moe, Hugh Chipman, Edward I George, and Robert E McCulloch. A Bayesian Treed Model of Online Purchasing Behavior Using In-Store Navigational Clickstream. (April), 2002.

[MGM99] Pattie Maes, Robert H Guttman, and Alexandros G Moukas. Agents that buy and sell. *Communications of the ACM*, 42(3):81–ff, 1999.

[Mis86] Jayadev Misra. Distributed discrete-event simulation. *ACM Comput. Surv.*, 18(1):39–65, March 1986.

[MJ00] James H Martin and Daniel Jurafsky. Speech and language processing. *International Edition*, 2000.

[Nia11] Muaz Ahmed Khan Niazi. Towards A Novel Unified Framework for Developing Formal , Network and Validated Agent-Based Simulation Models of Complex Adaptive Systems. page 275, 2011.

[NMK14] Wamukekhe Everlyne Nasambu, Waweru Mwangi, and Michael Kimwele. Predicting Sales In E-commerce Using Bayesian Network Model. 11(6):144–152, 2014.

[Ong07] Bhakti S S Onggo. Running agent-based models on a discrete-event simulator. 2007.

[Ope16] Openabm modeling platforms, 2016. Available on https://www.openabm.org/page/modeling-platforms, accessed last time at 9 of February 2016.

[Pea85] Judea Pearl. Bayesian Networks A Model of Self-Activated Memory for Evidential Reasoning, 1985.

[Pea88] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[Pid98] Michael Pidd. Computer simulation in management science. 1998.

[PL05] Liviu Panait and Sean Luke. Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 11(3):387–434, 2005.

[Rab89] Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition, 1989.

[Ram31] Frank P Ramsey. Truth and Probability. *The Foundations of Mathematics and Other Logical Essays*, Ch. VII(1926):156–198, 1931.

[Res96] Mitchel Resnick. Starlogo: An environment for decentralized modeling and decentralized thinking. In *Conference companion on Human factors in computing systems*, pages 11–12. ACM, 1996.

[RKP02] Iyad Rahwan, Ryszard Kowalczyk, and Ha Hai Pham. Intelligent agents for automated one-to-many e-commerce negotiation. In *Australian Computer Science Communications*, volume 24, pages 197–204. Australian Computer Society, Inc., 2002.

[SA10] Peer-Olaf Siebers and Uwe Aickelin. Introduction to Multi-Agent Simulation. *Ecology*, pages 1–25, 2010.

[SC00] Jim Sterne and Matt Cutler. E-Metrics: Business Metrics for the New Economy. page 61, 2000.

[Sha88] Ross D Shachter. Decision Making Using Probabilistic Inference Methods. 1988.

[SMG$^+$10] P. O. Siebers, Charles M. Macal, J. Garnett, D. Buxton, and M. Pidd. Discrete-event simulation is dead, long live agent-based simulation! *Journal of Simulation*, 4(3):204–210, 2010.

[TK74] Amos; Tversky and Daniel Kahneman. Judgment under Uncertainty: Heuristics and Biases. *Science (New York, N.Y.)*, 185(4157 (Sept. 27, 1974)):1124–31, 1974.

[TT00] Kah Leong Tan and Li-Jin Thng. Snoopy calendar queue. In *Proceedings of the 32Nd Conference on Winter Simulation*, WSC '00, pages 487–495, San Diego, CA, USA, 2000. Society for Computer Simulation International.

[UBJK94] Onur M. Ulgen, John J. Black, Betty Johnsonbaugh, and Roger Klungle. Simulation Methodology - a Practitioner'S Perspective. *International Journal of Industrial Engineering, Applications and Practice*, 1(2):16, 1994.

[Var01] Andras Varga. The OMNeT++ Discrete Event Simulation System. *Proceedings of the European Simulation Multiconference*, pages 319–324, 2001.

[Wat15] IBM Watson. Cyber Monday Report 2015. Technical report, 2015.

[WB13] John Winn and Christopher Bishop. Model-based machine learning, 2013. Available on `http://www.mbmlbook.com/`, accessed last time at 9 of February 2016.

[WBS08] Frank Edward Walter, Stefano Battiston, and Frank Schweitzer. A model of a trust-based recommendation system on a social network. *Autonomous Agents and Multi-Agent Systems*, 16(1):57–74, 2008.

[WE99] Uri Wilensky and I Evanston. Netlogo: Center for connected learning and computer-based modeling. *Northwestern University, Evanston, IL*, pages 49–52, 1999.

[Wik15] Wikipedia. Computer simulation — wikipedia, the free encyclopedia, 2015. [Online; accessed 14-February-2016].

[Wik16] Wikipedia. A/b testing — Wikipedia, the free encyclopedia, 2016. [Online; accessed 26-June-2016].

[WJ08] Martin J. Wainwright and M I Jordan. Graphical Models, Exponential Families, and Variational Inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008.

[Woo98] Michael Wooldridge. Agent-based computing. *Interoperable Communication Networks*, 1:71–98, 1998.

[XB07] Bo Xiao and Izak Benbasat. E-commerce product recommendation agents: Use, characteristics, and impact. *Mis Quarterly*, 31(1):137–209, 2007.

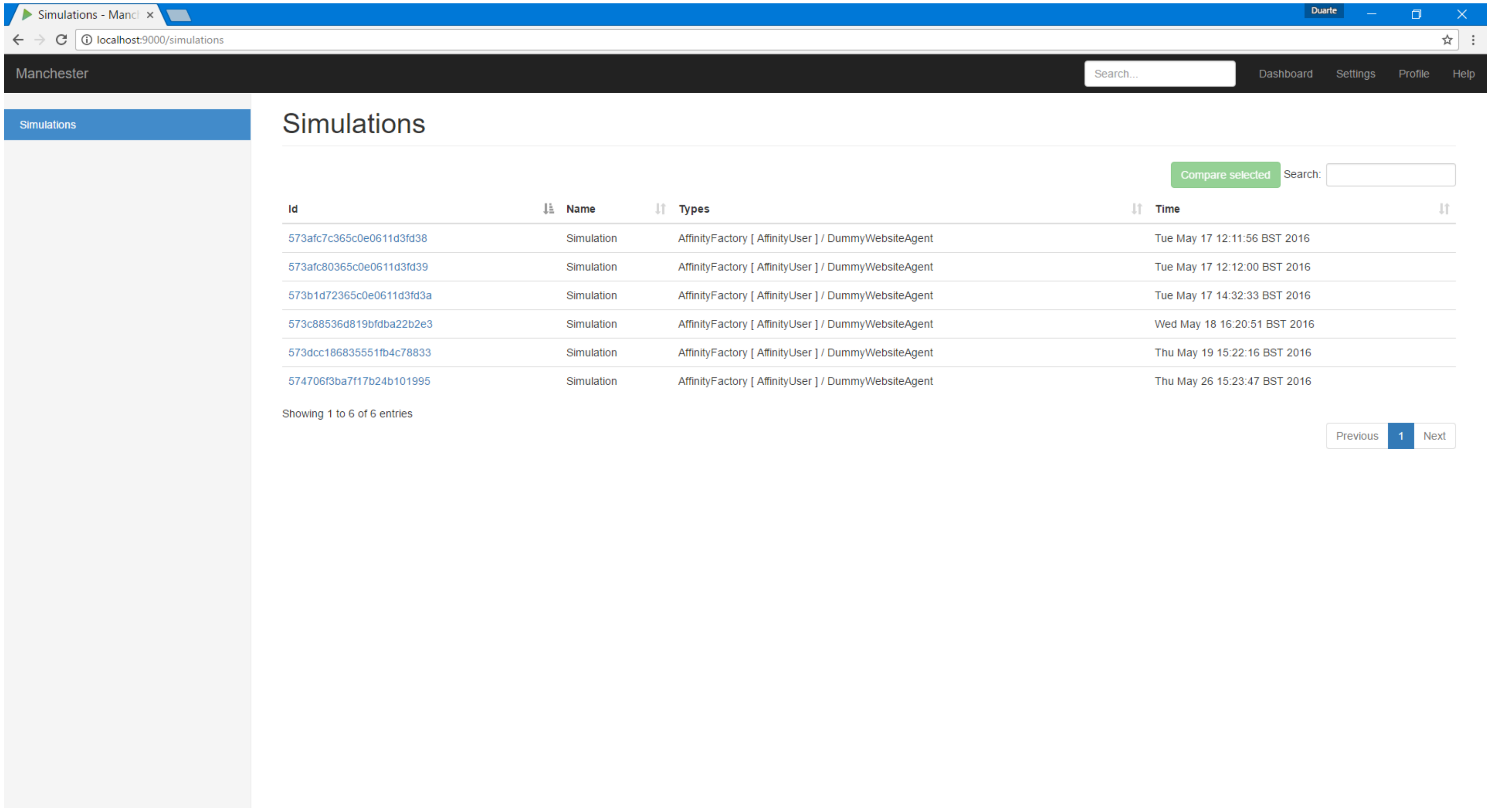
[XY09] Yi Xie and Shun-zheng Yu. A Large-Scale Hidden Semi-Markov Model for Anomaly Detection on User Browsing Behaviors. *IEEE/ACM Transactions on Networking*, 17(1):54–65, 2009.

REFERENCES

Appendix A

Graphical User Interface

The following images are screenshots of the interface developed to visualize simulation runs.

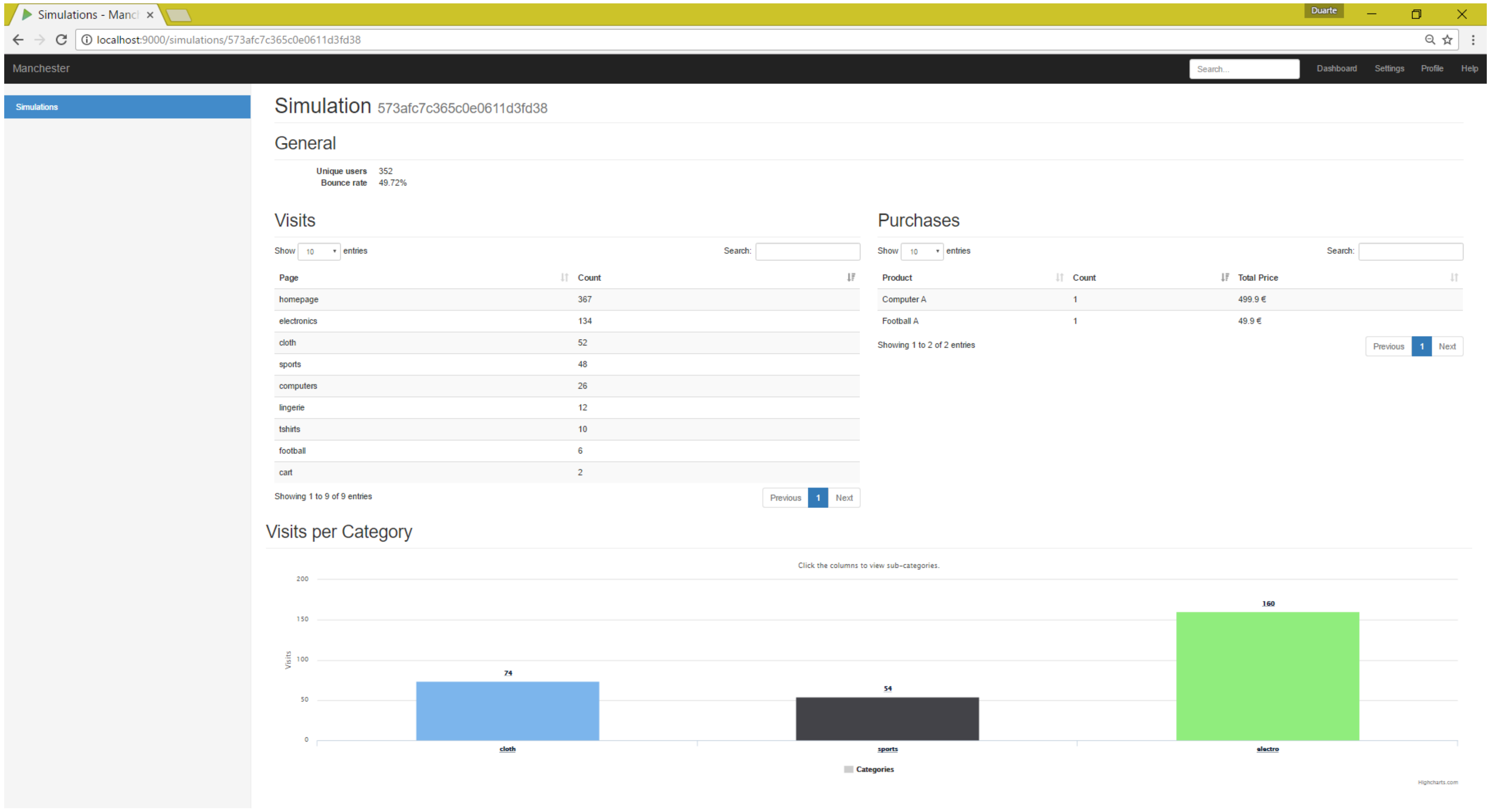


Simulations

Id	Name	Types	Time
573afc7c365c0e0611d3fd38	Simulation	AffinityFactory [AffinityUser] / DummyWebsiteAgent	Tue May 17 12:11:56 BST 2016
573afc80365c0e0611d3fd39	Simulation	AffinityFactory [AffinityUser] / DummyWebsiteAgent	Tue May 17 12:12:00 BST 2016
573b1d72365c0e0611d3fd3a	Simulation	AffinityFactory [AffinityUser] / DummyWebsiteAgent	Tue May 17 14:32:33 BST 2016
573c88536d819bfdba22b2e3	Simulation	AffinityFactory [AffinityUser] / DummyWebsiteAgent	Wed May 18 16:20:51 BST 2016
573dcc186835551fb4c78833	Simulation	AffinityFactory [AffinityUser] / DummyWebsiteAgent	Thu May 19 15:22:16 BST 2016
574706f3ba7f17b24b101995	Simulation	AffinityFactory [AffinityUser] / DummyWebsiteAgent	Thu May 26 15:23:47 BST 2016

Showing 1 to 6 of 6 entries

Figure A.1: Screenshot of the simulations list page



Simulation 573afc7c365c0e0611d3fd38

General

Unique users	352
Bounce rate	49.72%

Visits

Show 10 entries Search:

Page	Count
homepage	367
electronics	134
cloth	52
sports	48
computers	26
lingerie	12
tshirts	10
football	6
cart	2

Showing 1 to 9 of 9 entries

Purchases

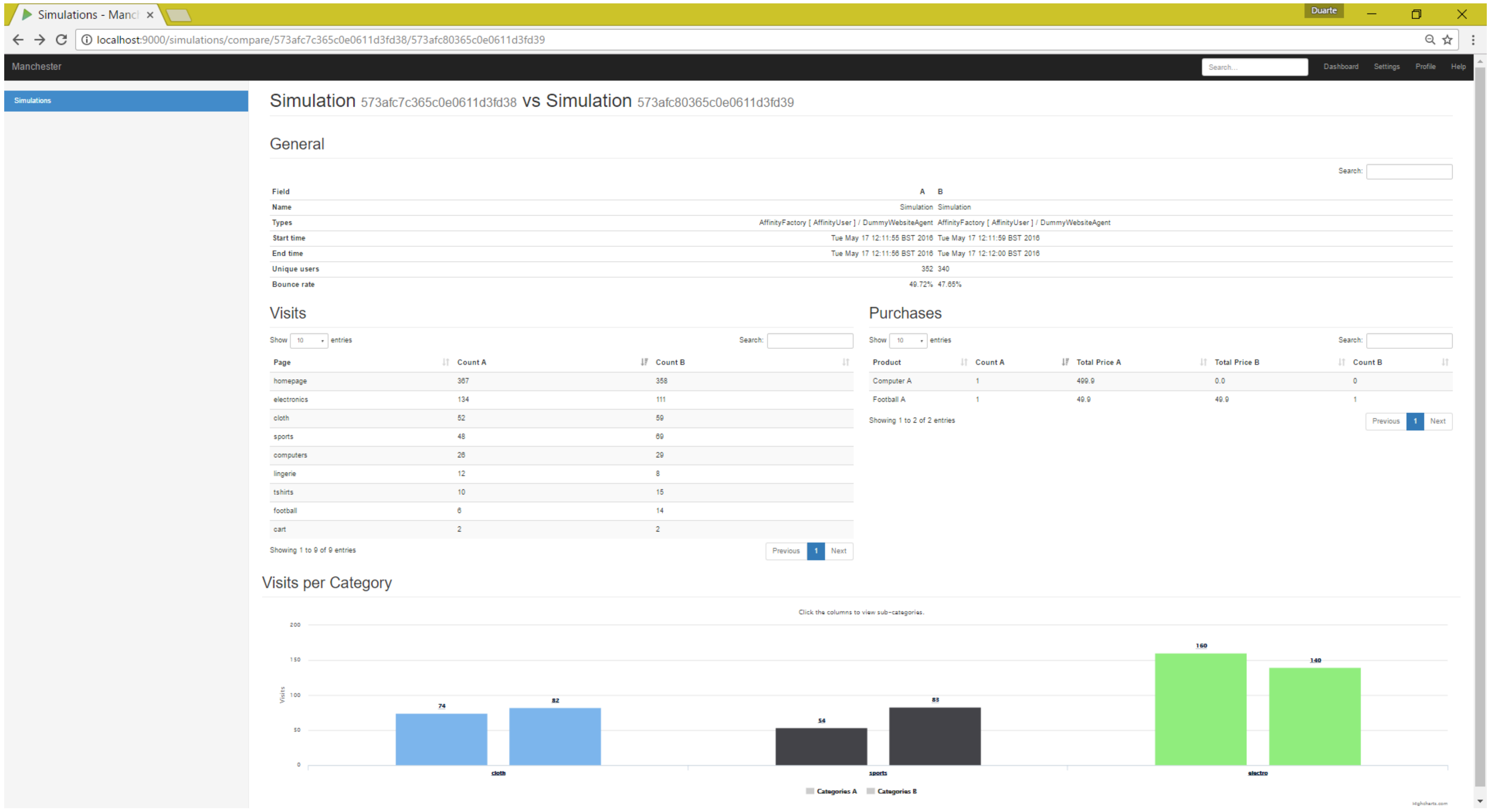
Show 10 entries Search:

Product	Count	Total Price
Computer A	1	499.9 €
Football A	1	49.9 €

Showing 1 to 2 of 2 entries

Visits per Category

Figure A.2: Screenshot of the simulation detail page


Simulations - Manc
localhost:9000/simulations/compare/573afc7c365c0e0611d3fd38/573afc80365c0e0611d3fd39
Duarte
Manchester
Search...
Dashboard
Settings
Profile
Help
Simulations

Simulation 573afc7c365c0e0611d3fd38 vs Simulation 573afc80365c0e0611d3fd39

General

Search:

Field	A	B
Name	Simulation	Simulation
Types	AffinityFactory [AffinityUser] / DummyWebsiteAgent	AffinityFactory [AffinityUser] / DummyWebsiteAgent
Start time	Tue May 17 12:11:55 BST 2016	Tue May 17 12:11:59 BST 2016
End time	Tue May 17 12:11:56 BST 2016	Tue May 17 12:12:00 BST 2016
Unique users	352	340
Bounce rate	49.72%	47.65%

Visits

Show 10 entries Search:

Page	Count A	Count B
homepage	367	358
electronics	134	111
cloth	52	59
sports	48	69
computers	26	29
lingerie	12	8
tshirts	10	15
football	6	14
cart	2	2

Showing 1 to 9 of 9 entries

Previous 1 Next

Purchases

Show 10 entries Search:

Product	Count A	Total Price A	Total Price B	Count B
Computer A	1	499.9	0.0	0
Football A	1	49.9	49.9	1

Showing 1 to 2 of 2 entries

Previous 1 Next

Visits per Category

Click the columns to view sub-categories.

Figure A.3: Screenshot of the simulation comparison page

]