UNIVERSITÀ DEGLI STUDI DELL' AQUILA

TESI DI LAUREA

"Enhancing e-Commerce Applications through Internet of Things and Real Usage Data Mining"

Corso di Laurea Specialistica in Informatica

Candidato:
Fabio Righi

Relatore:
Dott.ssa Romina Eramo

Anno Accademico 2016/2017

Abstract

Marketing personalisation is becoming a key focus area in communication studies and development, and will draw more and more attention of marketers in the next few years. For being able to effectively personalise the experiences of their customers, brands must overcome challenges that range from learning customer behaviour as they interact with the brand itself, to creating customer experiences that are tailored according to brand knowledge of individual preferences.

If in the past brands had not enough data to understand their customers at a personal level, nowadays things are moving towards an entirely different scenario, in which big data plays a fundamental role. For the first time, companies are able to distinguish customers beyond the mere demographic dimensions, leveraging the information collected in both the virtual and the physical worlds, and detecting and analysing patterns in their behaviour.

In this work, we describe how marketing personalisation can be achieved by coupling the customer information resulting from web data mining with the device tracking allowed by the Internet of Things. We will apply Model Driven Engineering techniques to classify and analyse the gathered data and, from that, to deliver unique and more effective online interactions between a brand and its consumers.

Acknowledgements

I would like to express my genuine gratitude to my advisor Dott. Romina Eramo not only for her patience, motivation, and enthusiasm, but also for continuously supportting my studies. Her supervision and vast knowledge were crucial throughout the development of this research and writing of this thesis.

Contents

Introduction

To learn and discover user preferences by analysing actual interactions and to cleverly take advantage of that learning: How can that be achieved? In the first chapter of this thesis, we answer this question by introducing the reader to two possible sources of real usage data: the Internet of Things and the Web Mining process. In addition, we present the software development techniques used for modeling the data gathered from those sources.

Following that general introduction, we proceed to Chapter 2, in which we examine the current state-of-the art of those information sources as well as their related pattern-recognition techniques.

In Chapter 3, we discuss an approach that blends together the discussed data acquisition channels. We also offer different use-case scenarios in which an actual eCommerce platform dialogues with a physical retail store for creating a singular brand experience.

Starting from Chapter 4, we begin to model the data presented previously by using Model Driven Engineering notions and by describing metamodels and models for the domain and the data occurrences.

We proceed then with Chapter 5, illustrating a possible model transformation for the eCommerce platform based on the real usage dynamic instance, to offer users a more customised experience.

After the modeling and processing phase, we continue with discussing a case study. Chapter 6 introduces the eCommerce Magento platform, and suggests a possible transformed model serialisation that would result in compatible code within the Magento ecosystem.

The last chapter covers related works, conclusions and a possible evolution for the presented approach.

Chapter 1

Background

1.1 The Internet of Things and the Web

During the last few years, and in an increasing manner, we have been standing at the forefront of a world where virtually everything can be connected directly to the Internet. This has resulted in an increasingly connected world where emerging technologies enable objects to be linked among themselves through the use of new devices and sensors.

This new era of the Internet has become visible in our everyday lives: from souped-up gadgets tracking our every move to environments capable of predicting our actions and emotions, the list keeps growing every day.

The Internet, consisting of things rather than just computers, is becoming more central to society than the web as we once knew it. This does not necessarily mean that the traditional web is expected to die, but rather that its role will be reduced to that of a language used for displaying content on devices which are supposed to be more ubiquitous but are not as necessary.

The first “pioneer species” within this ecosystem of “invisible buttons” has certainly been the smartphone. Its increasing usage among the masses has made it the perfect catalyst for such a revolutionary change. One of the many uses that has helped us better understand this pioneering technology is the way that it beams information about our location and speed when we take it with us in a car resulting in a real-time traffic information accessible by everyone. In such a scenario, the actual gathering of real traffic data happens without the user ever knowing of the data transmission and without a need for interaction such as a click on a button or navigating to a particular web page.

This sort of awareness, especially as it relates to the physical world, leads to areas in space that are listening to the environment and triggering events depending on certain conditions in an entirely automatic way, like a smartphone getting into the range. There are currently applications in which the smartphone can be placed as close as two centimetres away from the top of a credit card reader to enable touch payments, or where the smartphone is detected in a large space,

such as a room, triggering an event indicating that a user has entered or exited so that the lights can bed switched on or off.

It is important to differentiate these interconnected objects from being simple on-off switches; they would not be very useful if this were the case. However, because the possible actions they can trigger can be affected by an endless list of other variables such as the time of the day, our personal preferences or the actions of others, they can quickly be scaled to build a better program that creates more efficient interactions in our physical world.

Leveraging the use of a smartphone acting as a proximity sensor is just an artefact of the current state of technology. The same result could be accomplished with any number of sensors directly connected to the Internet so long as those sensors are capable of dealing with motion, sound, light temperature, humidity, and other variables.

Companies like Apple have been embracing the idea of invisible buttons since the beginning of this new era of sensors and devices.

While the company has been embracing the technology by foreseeing its potential from the beginning, it recently rolled out a new line of devices called iBeacon.

In a nutshell, the iBeacon allows any newer iPhone or Android phone to know its position in space with centimetre precision. Similarly to a more precise Global Positioning System (GPS) that works indoors, an iBeacon allows the developers to take advantage of the technology to define “invisible buttons” of just about any dimension.

Figure 1.1: Distance categories for iBeacon

In fact, nothing is stopping this technology from being squeezed into something as small as a typical credit card, or from being embedded in any clothing or other discrete wearable de-

vices such fitness sensors, wristwatches or even temporary tattoos. The opportunities are indeed countless. Whether we wear those sensors or use them in our homes and businesses (see smart thermostats, lighting and security systems for example), they can all be prepared and trained to cooperate in sophisticated and unexpected ways once the Internet knows that we are present nearby and what our intentions might be. Imagine a smart home capable of knowing when we wake up based on the activity monitor on our wrist and begin warming up the house, brewing a pot of coffee and switching off the security system. It is evident that with such a capacity for sensing and responding to our needs, the Internet of Things is slowly shaping a brand new world capable of being alive in ways that it has never been before.

Figure 1.2: A smart home ecosystem

1.2 Data and web mining

Very often, company management requires selecting the most adequate Business Intelligence solutions that will fit its needs in order to perform crucial strategic decisions.

One of the tools used for this particular goal is a technique called data mining, which is the result of a continuous evolution that has been occurring during the last thirty years of data review. Up until the late 1970s, Business Intelligence decisions carried out their role through the use of standardised reports which contained simple summarised data and analysis.

In the early 1980s, companies began to query data in more detailed and complex ways. This made it easier to detect patterns based, for example, on an individual product or geographic area.

Currently, the advanced software available on the market for data mining is capable of perform-

ing real time pattern detection on a vast quantity of data thrown at it. This expedites a company's decision making processes and the creation of robust long-term strategies.

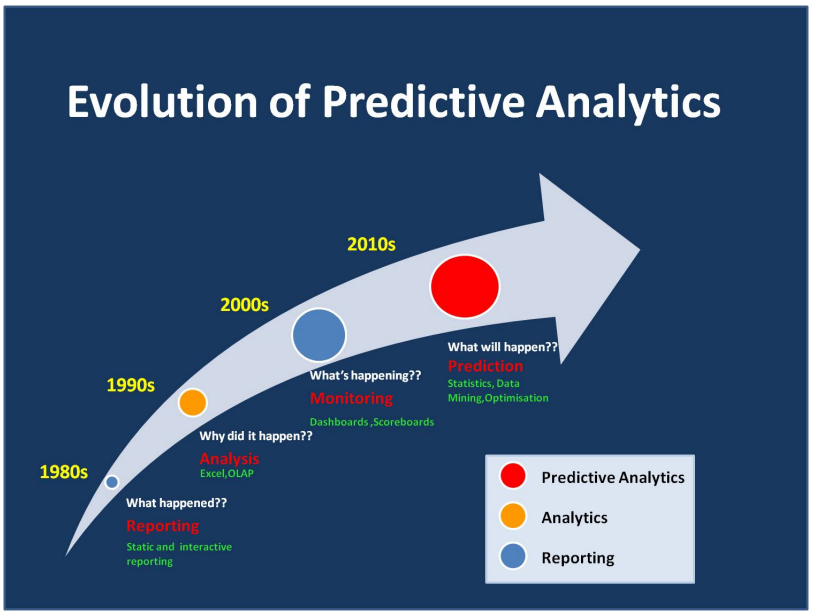


Figure 1.3: Evolution of Predictive Analysis over the last 40 years

Interpretation and analysis of the available data allows data mining process to create a better overall understanding, and helps in making better decisions.

In fact, thanks to advanced examination techniques, it is possible to find hidden information, create analytical models and data groups, and identify relationships among activities while also correcting errors.

All of this certainly leads to real advantages for a company leveraging these processes, both in terms of revenue and cost. For example, on the side of income, data mining allows companies to identify and classify the best, real and potential customers, discover additional sales opportunities, increase economic productivity, and find new ways and new solutions to grow. In parallel, regarding the aspect of cost, the process could maintain clientele by identifying customer loyalty elements, reducing exposure to non-payment risks and distributing resources more efficiently.

For an organisation, the reasons behind using data mining may be different. The unifying point, however, is the need to derive insights from the data that will guide the transformation, reorganisation or innovation of business processes. It is evident that decisions based on accurate and reliable knowledge are always the best. Data mining, in fact, provides exactly this type of information. While Enterprise Resource Planning (ERP) systems improve operational efficiency, they do not provide the strategic drive for business to either growth or change. Warehousing systems can efficiently store data, but they lack the tools to transform those figures into valuable information focused on reporting and answering mostly static questions such as areas where the

company has sold the most. On the other side, data mining tools try to present a solution to a wider range of more interesting problems, such as why sales are not taking off as expected, why customers prefer competitors, or which previous marketing campaigns had the best outcomes.

Understanding the answers to these questions means taking the right measures to improve the business's performance.

Besides data visualization techniques, one of the most popular data mining processes on the market is based on the simplified transposition of the neural networks and the neural process of the human brain: when presented with models, the brain understands that some patterns are associated with other desired results. Similarly, artificial neural networks are capable, by learning about sets of historical data called learning sets, to generate patterns and validate them on other subsets of data called test sets. They operate iteratively by modifying patterns from time to time to reach an optimal solution, and they have the ability to evaluate and provide feedback on unknown data, thus making them very useful for forecasting and classifying knowledge. They are very often presented as a "black box" approach to data mining, and they turn out to be very useful when creating parametric models that are difficult to construct and when the emphasis is on forecasting rather than explaining complex patterns.

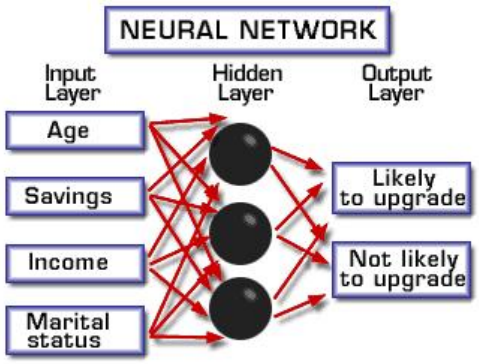


Figure 1.4: Neural networks learn to predict outputs after proper training and weighting.

After discovering what data mining means, who uses it the most, and how it is usually implemented, it is important to focus our attention on the utilisation of this tool on the web and its characteristics in such an environment.

In fact, web data mining can detect behavioural models of website visitors, generate reports and implement actions based on those identified patterns. This process is possible because website visitors — often unknowingly — provide information about themselves and how they respond to the content presented. Monitoring what links they click, where most of their time is spent, what search terms are entered, and when the visitors leave the website are just a few appetisers of the endless stream of possible data inputs.

Some visitors also provide information about their lifestyle or personal information such as

names and addresses.

Because of all of that, a thorough and adaptive analysis of this considerable amount of stored information is fundamental. This is exactly where web data mining kicks in by helping to design the web shopper behavioural model and making valuable predictions.

One of the features that unquestionably contributes to the strength of data mining is the ability to combine emerging traffic data to the site with those related to the transactions and the profile of the buyers.

Through these combinations and by highlighting the patterns that are uncovered, it is possible to both gather complex and strategic considerations and generate predictions that may be vital for managing a website that wants to improve its business.

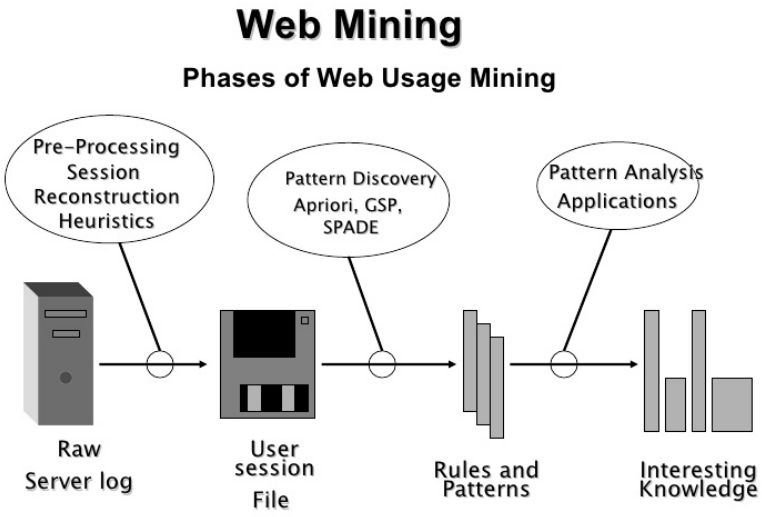


Figure 1.5: Phases of Web Usage Mining

Due to the heterogeneity nature of the source data, Web Mining is far from simply being an application of traditional data mining techniques. In fact, it can be categorised into three main types:

- **Web structure mining**: This focuses on analysing and discovering useful knowledge from hyperlinks representing the structure of the Web. For example, these links allow us to detect relevant Web pages in a way similar to what search engines are already doing. Alternatively, it is possible to determine shared common interests among users and so on.

- **Web content mining**: The main goal of this technique is to extract valuable information or data from the content of the web pages. After doing so, it is possible to automatically classify and group this information according to topic area. While these tasks are apparently similar

to those in traditional data mining, we still can discover relevant behavioural models using product descriptions, forum posts, customer reviews and much more.

- **Web usage mining**: This technique usually refers to the identification of user access patterns from Web usage logs once the sanitisation and preprocessing of the clickstream data has occurred.

Although the web mining process is similar to traditional data mining techniques, the data gets collected in an entirely different way. In traditional data mining, the data is often already available and stored in a data warehouse, whereas for the web counterpart, the effort for the data acquisition can be a cumbersome task, especially for web structure and content mining. This is due to the potentially high number of links and large quantity of pages to crawl.

1.3 Model-driven techniques

Software development techniques are continuously evolving while also trying to solve the main problems that still affect the building and maintenance of software: time, costs, and susceptability to errors.

On this topic, one of the latest research trends in software engineering is the Model-Driven Engineering (MDE) technique, which was born as an extension of more specific approaches such as the Model-Driven Architecture (MDA) of the Object Management Group (OMG).[1]

MDE's primary goal is to define the methodologies and techniques to support the process related to the entire lifecycle of software development through the manipulation of models.

Before proceeding further, it is beneficial to explain the difference between MDA, Model-Driven Development (MDD), MDE and Model-Based Engineering (MBE).

The first is, by all means, an OMG standard focused on software development and using a set of defined languages used for a specific purpose (e.g. UML[2]). On the other hand, the focus of MDD is still on software development, but is independent of mandatory language constraints to perform its tasks. MDE, as a superset of MDD, does not only drop the software-related restrictions, but it unties itself from a particular development process, therefore expediting the definition of model driven processes to facilitate a complete software engineering process. Finally, we use MBE to refer to a softer version and a superset of MDE where models still play an important role, but are not the central artefacts of the development process. (e.g. blueprints or sketches of the system handed out to programmers directly without automatic code generation) (Figure 1.6).

[1]The Object Management Group (OMG) is an international, open membership, not-for-profit technology standards consortium, founded in 1989. OMG standards are driven by vendors, end-users, academic institutions and government agencies.

[2]The Unified Modeling Language (UML) is a general-purpose, developmental, modeling language in the field of software engineering, that is intended to provide a standard way to visualise the design of a system.

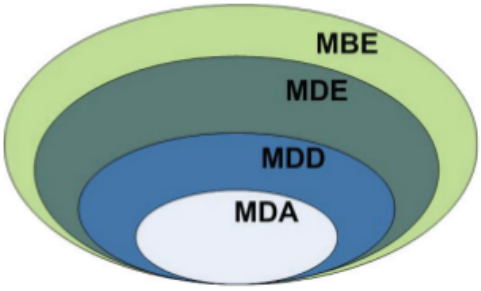


Figure 1.6: Relationship between the different MD* acronyms.

The model is at the centre of any MDE process and, to be considered as such, the modeling language that generated it needs to have well-formalised syntax and semantics. This is a fundamental condition for the automatic transformation of models. In fact, the model represents the system and constitutes an abstract and conforming formalisation to a particular language. Such a language is often tailored to a certain domain and is often called Domain-Specific Language (DSL).

The advantages of using the model driven approach rely on the consideration of the generating language as a type of scheme that is also fully modelable through a formal and abstract definition known as the meta-model. In fact, the model must represent an abstraction of the concepts of the system that needs to be built, while also respecting a meta-model likewise determined by the application domain.

This reasoning can be further iterated up to the determination of a model for the representation of languages used to formalise other languages (the meta-meta model). Therefore, model-driven approaches are often defined by models on a stack basis as shown on Figure 1.7.

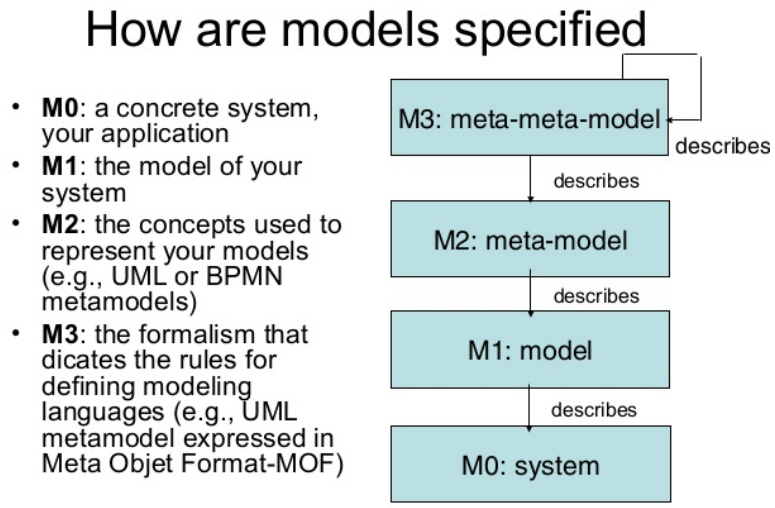


Figure 1.7: Models stack

One of the most commonly used techniques in MDE is the automatic model generation based on the definition of a set of automatic transformations defined by the starting and target languages. In a nutshell, the MDE approach consists of:

- **Models Generation**, based on modeling stacks definition supporting meta-modelling methods where lower-level models comply with what is specified by higher-level models.
- **Models Manipulation** performed through transformations expected to generate destination models which conform to destination meta-models when provided with source models that conform to source meta-models.

In summary, MDE techniques allow, on the one hand, for the management of the complexity of a system by ensuring increased productivity, and for quality improvements by promoting a higher level of abstraction and reuse, on the other. This result is achieved through a process of engineering, both for the languages and the transformations involved: they are, in fact, the basis of a system's development process because they represent the transition from higher-level models into lower-level ones until they can be made executable using either automatic code generation or model interpretation.

Chapter 2

State of the art

2.1 Iot Devices : RFID, NFC, Beacons and Sensors

As introduced in 1.1, The Internet of Things (IoT) describes an ecosystem or network of Internet-connected objects able to collect and transfer data using embedded sensors. This ecosystem is vast but can be organised into a number of major dimensions that are discussed below:

- **Radio Frequency Identification (RFID)** is a technology predominantly applied to one-way inventory tracking and supply chain problems. Packages are affixed with passive RFID tags containing important product information that are detectable to a distance of 100 meters by stationary readers. RFID tags are comprised of a small antenna and a silicon chip capable of storing the product information to facilitate its identification. The tags are remarkably small, and can be effectively integrated with adhesive labels attached to cases and pallets, incorporated into security cards, or even implanted in pets, to name just a few examples.

  The primary difference between RFIDs and the most common electronic barcode mechanism is that barcode readers require “line-of-site”. In other words, barcode readers must directly see the barcode lines in order to properly read the data, and this reading can be done only one barcode at a time. The RFID, on the other hand, does not require a direct reading as tags can be scanned through a variety of materials. From an IoT perspective, RFID readers are powerful mechanisms not only to read and write data across networks of RFID tags, but also to transfer automatically large amounts of data to any type of data cloud or backend system.

- **Near Field Communication (NFC)** comprises a set of close-range wireless communication standards offering functions similar to the most common Bluetooth and RFID technologies. Much like RFID, NFC can detect and access data from special tags but have the additional advantage of being suitable for virtually unlimited applications because information can be easily retrieved from any conventional NFC-enabled device. Similar to Bluetooth technology, NFC supports two-way secure data exchange with a simple tap or wave among devices.

Currently, NFC is already incorporated into over 1 billion devices globally, including an increasing number of tablets, PCs, household appliances, electronic devices, gaming consoles and smartphones. For enhanced security and control, NFC operates only when devices are in close proximity (approximately 10 centimetres), thus making this technology optimal for more protected applications like financial transactions and secure login access at a particular location.

- **Beacons** collectively refer to small wireless devices that are capable of transmitting simple radio signals embedded within a unique identification number. At any time, a nearby device such as a smartphone using Bluetooth Low Energy technology can detect the transmitted signals, reading the beacon's ID, calculating the distance to the beacon and, depending on the result, triggering an action in a beacon-compatible mobile app.

  Despite the simplistic mechanism, beacons represent a substantial technological advancement and have opened new opportunities to allow more precise tracking for indoor positioning and behaviour compared with standard GPS technology.

  The simple communication associated with beacons, which importantly is not reliant on significant power consumption, unfolds a wide range of new seamless interactions with application in numerous areas. The retail sector, for example, is currently one of the most popular fields in which beacon technology is employed because it offers an unprecedented way to track in-store interactions between customers and product displays, ultimately resulting in more personalised offers and a better retail experience based on the acquired data. Additional applications of the beacon technology range from supporting and improving physical navigation in large spaces such as museums, airports and stadiums, to assisting impaired people in public transportation.

Figure 2.1: RFID, NFC and Beacon Tags

- **Sensors** are pieces of hardware responsible for monitoring processes, taking measure-

ments and collecting data. There are literally infinite measurements that can be gathered by sensor arrays, ranging from temperature to proximity, from pressure to water quality, and from gas detection to liquid level tracking. A wide variety of devices currently include these sensors, and current trends indicate that we are witnessing a move towards multi-sensors platforms capable of simultaneously incorporating different sensing elements. In the retail sector, for example, it is possible to make the store shelves “smarter” by providing visibility from the product’s arrival to final sale thanks to a combination of shelf sensors, smart displays, digital price tags and high-resolution cameras. In fact, sensor-based technology now enables the retail businesses to precisely determine product volume on store shelves as well as in stock rooms, leading to more efficient product ordering and restocking. A final illustration of sensor application is the next-generation of personalised self-tracking products in the form of wearables and smartwatches. Examples include combinations of accelerometers, GPS, temperature and heart rate sensors that provide a comprehensive data insight on the activity of the user.

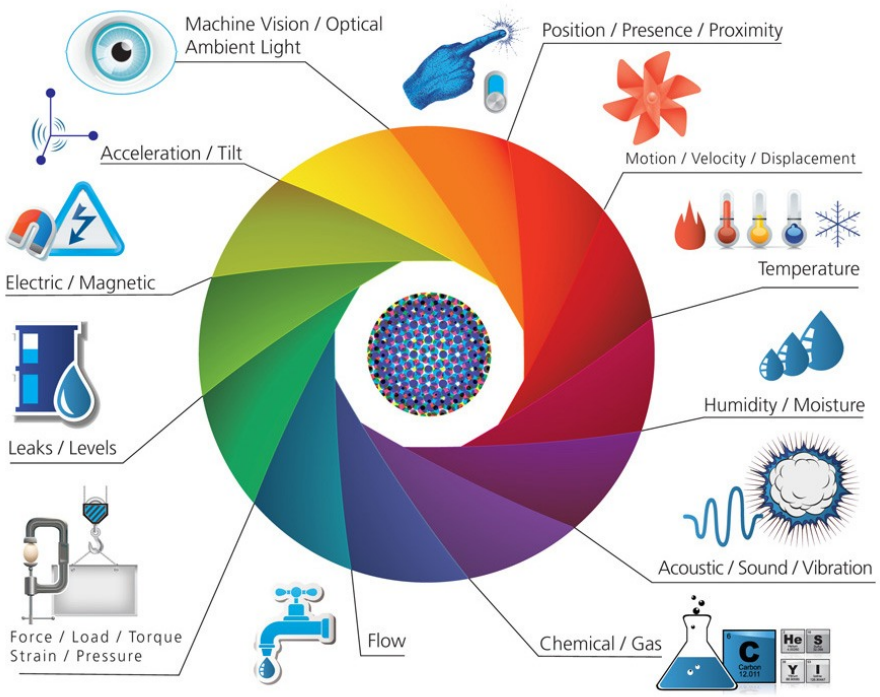


Figure 2.2: Sensors ecosystem

2.2 Big Data and Predictive Analysis applications

The concept of Big Data relates directly to the Internet of Things. IoT devices can collect terabytes of data over very short time frames, thus emphasising the importance of being able to support the efficient processing and interpretation of data as it is collected.

The power of IoT devices in generating profile data at an incredibly high rate can be coupled with contemporary paradigms in digital communication (see chapter 1.2). Social network platforms, for instance, are sources of endless data that describes not only user preferences and behaviours but also their navigation patterns and daily activities, at any time. This vast volume of "Big Data", available in a whole array of formats and growing constantly, provides unparalleled opportunities for addressing any number of socially-important questions.

To obtain economic value from Big Data, companies are investing in advancements in artificial intelligence and machine learning algorithms to efficiently process data, create products, and implement solutions that extend well beyond traditional systems currently used for managing and storing information. This includes novel approaches supporting high calculation inclination that are efficient at sorting data in a structured and meaningful manner to detect relationships and patterns in complex data streams. This new approach to data management differs from what companies used to do when priorities were bound to an IT level governance only, and they were solely accessed by a restricted set of users. Moreover, it also becomes vital for a company to identify new sources of big data as they are available on the market, and quickly incorporating them into the data management platform following a continuous improvement logic.

Figure 2.3: The five fundamental Vs of big data

It is challenging to forecast the added value brought by Big Data analysis to the various dis-

ciplines in which they can be applied. However, we can already affirm that it has been improving the quality of the forecasts that contribute to more prudent decisions supported by more robust empirical evidence. In fact, Big Data analysis constitutes a fantastic instrument in the field of decision-making by minimising the risks and reducing the consequences of inadvertent human errors. For example, the marketing department of an organisation could potentially leverage big data for increasing marketing intelligence by predicting customer interests. At the same time, Big Data could be employed to provide better forecasts of product stock replenishment, and to optimise production requests.

In summary, the potential applications mentioned above represent the core of the predictive analysis notion: the practice of extracting information from Big Data with the goal of determining patterns and predicting future outcomes and possible trends.

It is also important to remark that this kind of analysis does not offer a precise assessment of what will occur in the future, but rather they forecast the likelihood of particular outcomes with an acceptable level of reliability. This often includes what-if scenarios and risk assessments.

Focusing on the Marketing intelligence example mentioned above, it is important to fully understand the benefits of using Predictive Analysis. Digital companies have been increasingly embracing that idea because it offers a better and more constructive customer experience on all points of contact between the business itself and its clients. This can increase customer loyalty and, by extension, economic returns.

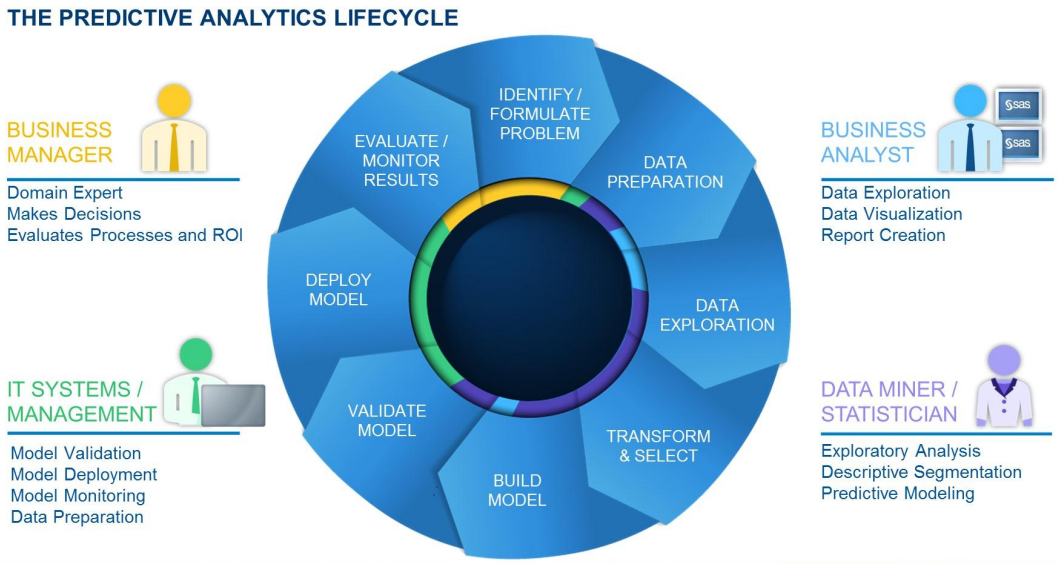


Figure 2.4: The predictive analysis lifecycle

More specifically, this outcome can be achieved by using sophisticated algorithms and mathematical models on top of the datasets of customers activity, accessible from Big Data sources. Eventually, this data gets sanitised, structured, filtered, and finally grouped in a meaningful way.

The behaviours and patterns of interaction detected by this analysis process can indicate, for example, a more appealing product offer with a major chance of conversion for particular segments of customer profile.

Different typologies and techniques are used to perform enhanced analysis for behavioural prediction. A few of the major approaches are discussed in the following:

- **Clustering or Unsupervised learning**: This methodology seeks to group similar individuals, identifying behaviour patterns on a given customer base with high precision. These algorithms can process hundreds of attributes, aiming at identifying the characteristics that best discriminate individuals according to their actions. The underlying notion is that, statistically, distinct groups of individuals behave in similar ways.

  Group clusters obtained with this process are similar to groups determined through a priori classification; the only fundamental difference between the two methodologies is that clustering or unsupervised learning allows individuals to be categorised into different groups based solely on data, and not on pre-conceived attribute differences.

  Clusters types can differ depending mostly on the criteria by which customers are grouped. Product-based clusters, for example, collate all customers who tend to buy products or combinations of several products in the same category. By contrast, brand-based clusters focus on grouping customers who prefer certain brands instead of others, while behaviour-based clusters combine consumers with similar buying behaviours. Creating specific clusters allows marketing managers to better identify the most effective way to address each customer type.

- **Propensity models or Supervised learning**: This approach is based on probabilistic models using advanced machine learning techniques such as neural networks, logistic regression, random forest, and genetic algorithms. The main purpose of these procedures is to predict future customer behaviour based on past examples. Over time, these algorithms become more efficient, improving predictions as more data is collected.

- **Reinforcement learning and Collaborative filtering**: This increasingly popular technique has been identified in a number of major applications, one of the most famous being the ability of companies to recommend products to specific customers, driving specific purchases. The recommendations are targeted and tailor-made for the client and they are defined by considering the entire relationship between customer and brand. For this reason, they can upsell products for higher value, cross-sell them to same category items, or dynamically link them to other products based on the modelled associations.

Chapter 3

Data acquisition

The evolution of the Internet and the amount of data available from web usage mining and IoT devices have led to an enormous proliferation of the accessible information as per described in the previous sections. In this chapter, we focus on describing the possible channels of acquisition of customer behavioural data in both the virtual and the physical world. This valuable information represents the milestone of the journey towards the enhancement of the eCommerce experience for its users, and aims at addressing some of the weaknesses of the standard approaches for web personalisation.

Before we can efficiently represent content visualised on user interfaces, navigation paths and user-triggered events on a web application, we need to take a little detour briefly introducing a standard modeling language able to shape such interactions. This language will be analysed in detail in the following sections.

3.1 The IFML language

The Interaction Flow Modeling Language (IFML)[1, 2] is designed for describing and controlling the behaviour of front-end software applications. IFML brings several advantages to the development process, such as promoting the separation of concerns between roles and increasing the overall understanding of the product for non-technical stakeholders. That can be achieved because IFML supports formal specification for interface composition, user interaction, and event management independently of the implementation platform. This language was adopted as a standard by the Object Management Group (OMG) in March 2013.

IFML supports the following concepts:

- **The view structure** describes *ViewContainers*, their nesting relationships, their visibility, and their reachability.
- **The view content** manages *ViewComponents*, i.e., content and data entry elements contained within ViewContainers.

- **The events** defines the *Events* that may affect the state of the user interface. *Events* can be produced by the user interaction, by the application, and by an external system.
- **The actions** triggered by user events. The effect of an *Event* is represented by an *InteractionFlow* connection, which connects the event to the *ViewContainer* or *ViewComponent* affected by the *Event*. The *InteractionFlow* expresses a change of state of the user interface: the occurrence of the event triggers a change in the state that produces a transition in the user interface.
- **The navigation flow** indicates the effect of an Event on the user interface.
- **The data flow** indicates the data passed between *ViewComponents* and *Actions*.
- **The parameter binding** illustrates the input-output dependencies between *ViewComponents* and between *ViewComponents* and *Actions*.

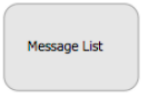
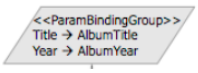
Concept	Meaning	IFML Notation	PSM Example
View Container	An element of the interface that comprises elements displaying content and supporting interaction and/or other ViewContainers.		Web page Window Pane.
View Component	An element of the interface that displays content or accepts input		An HTML list. A JavaScript image gallery. An input form.
Event	An occurrence that affects the state of the application		
Action	A piece of business logic triggered by an event		A database update. The sending of an email.
Navigation Flow	An input-output dependency. The source of the link has some output that is associated with the input of the target of the link		Sending and receiving of parameters in the HTTP request
Data Flow	Data passing between ViewComponents or Action as consequence of a previous user interaction.		
Parameter Binding Group	Set of ParameterBindings associated to an InteractionFlow (being it navigation or data flow)		

Figure 3.1: Main IFML concepts and notations.

3.2 Web navigation profiling

The front-end of eCommerce websites is usually built using shared and reusable components (forms, list views, detail views, etc.), which have a specific and expected behaviour. For example, product lists and grids show users record details and create possibilities for interaction. Similarly, “add to cart” buttons are placed on strategic points within product pages to trigger a specific user action. These interactions and any other user action withing the website can be represented using the IFML notation.

To demonstrate the versatility and adaptability of this modeling language, we introduce a real-life example which will serve as reference from this chapter forward: an online boutique website called *“Madison Island”* and specialised in fashion items running on an eCommerce platform.

Madison Island presents all the features of a typical digital store including catalogue navigation, product searching, customer account, shopping cart, and order processing.

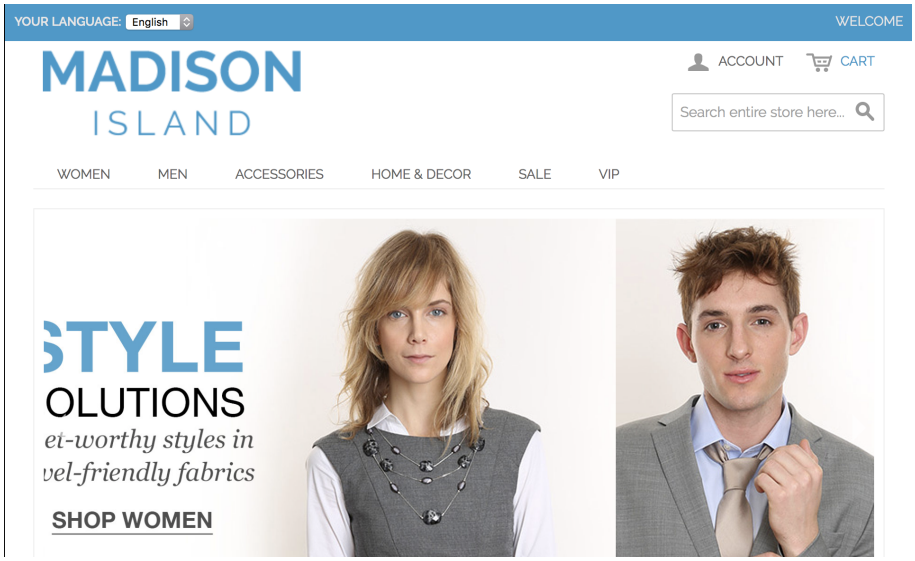


Figure 3.2: Madison Island digital store homepage

Figure 3.2 shows the home page of the website. In this section, the user can select one of the product categories, access his customer area, switch the language of the website, search for an item or go directly to the shopping cart.

In the following subsections, we analyse some scenarios related to users navigational behaviours, exposing both their representation in IFML notation as well as the log entries in the application server associated with those representations.

3.2.1 The product page journey

Starting from the homepage, the user can interact with the navigation menu by choosing from a set of available categories. (Figure 3.3)

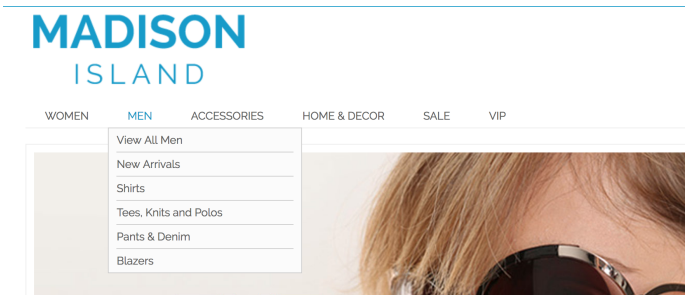
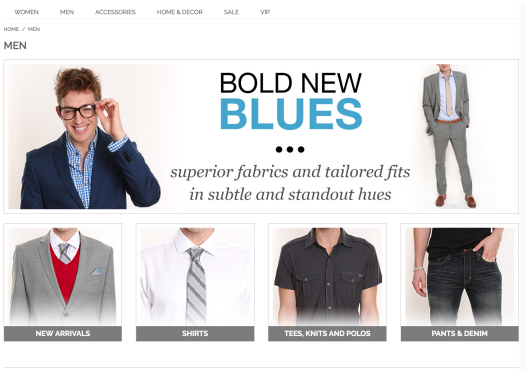
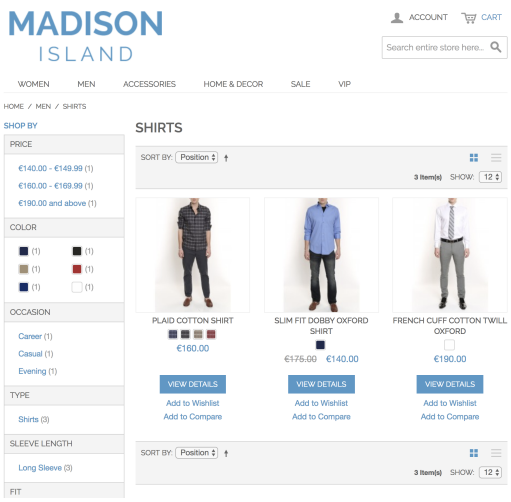


Figure 3.3: Navigation menu

Depending on the category display mode, a category page can show users either a list of links to children categories or a set of products. In the former case, children categories work as transitional pages that help further filtering products (Figure 3.4).



(a) Category listing



(b) Product listing

Figure 3.4: Different category page view modes

Finally, from the product listing screen, the user can access any of the product detail pages by clicking on the *"View Details"* button placed below the selected product thumbnail. The following image illustrates the product page seen when the user clicks on *"View Details"* of *"Plaid Cotton Shirt"* (Figure 3.5).

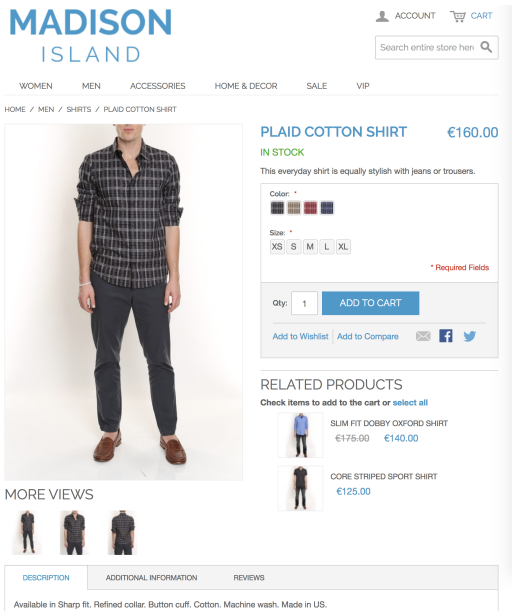


Figure 3.5: Product detail page

The end-to-end interaction from the homepage to the product detail page can be represented by a model using the following IFML notation:

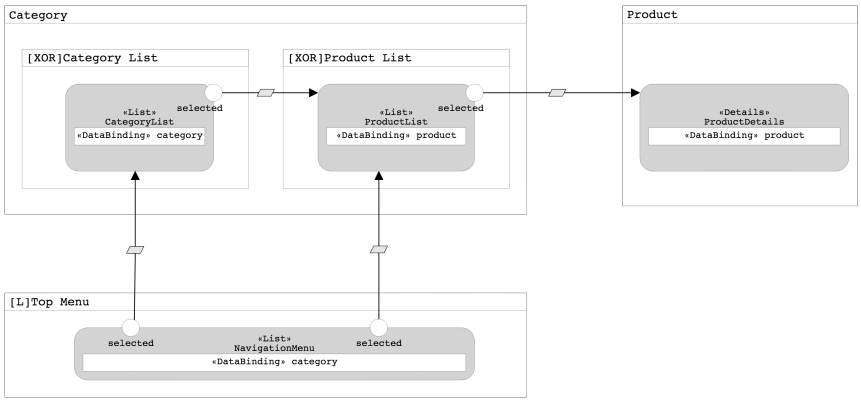


Figure 3.6: IFML representation of the Product Detail interaction

The same sequence of actions can be expressed as a stream of records in the access logs on the application server. Such logs track all the requests processed by the web platform. The following table presents how the above mentioned user journey is logged on the server:

Application Server Access Log		
ID	**Page**	**Log Entry**
1	Home Page	*[29/Nov/2017:06:30:45 +0000] "GET /" 200 0 - 29505*
2	*"View All Men"* Category Page	*[29/Nov/2017:06:49:38 +0000] "GET /men.html /" 200 0 - 29505*
3	*"Shirts"* Category Page	*[29/Nov/2017:07:04:15 +0000] "GET /men/shirts.html" 200 0 - 29505*
4	*"Plaid Cotton Shirt"* Product Page	*[29/Nov/2017:07:08:40 +0000] "GET /men/shirts/plaid-cotton-shirt-476.html" 200 0 - 29505*

It is important to notice that both entries 2 and 3 record a *"Category Page"* pageview action by logging their related URLs. The entry 4, on its turn, tracks a *"Product Page"* visit that happened after visiting a category page, logging a specific URL that concatenates the product URL key and the category path.

3.2.2 Products association and correlation

In this section, we analyse the set of actions that can generate pageviews among different product pages on the Madison Island website.

For doing so, we consider a *"Related Products"* widget presented to customers just below the *"Add to Cart"* button on the *"Plaid Cotton Shirt"* page discussed on 3.2.1.

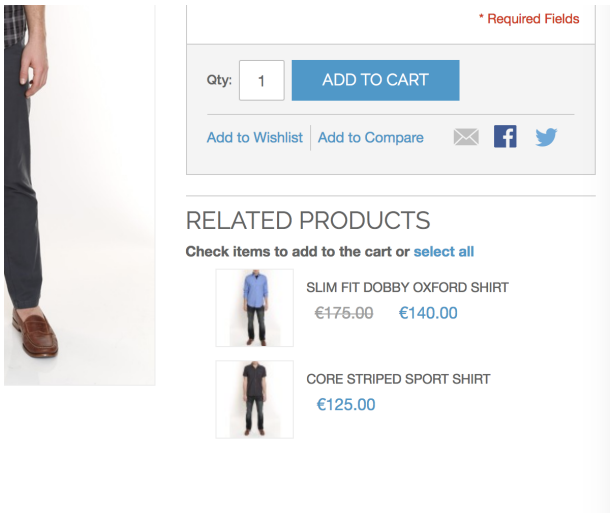


Figure 3.7: Related products section

By clicking either on the product name or on its thumbnail, the user is directed to the related product page. In this case, we simulate an interest on the listed *"Core Striped Sport Shirt"* (Figure 3.8).

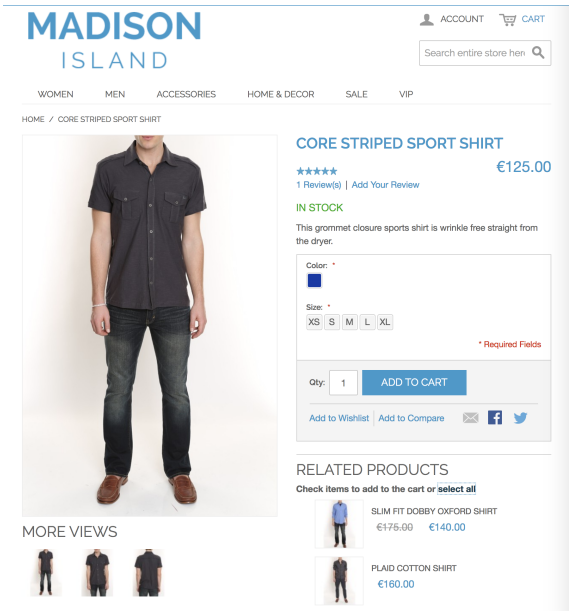


Figure 3.8: A related product page

Besides using the simple navigational shortcut available on the related products section, the user can reach a product page in several other ways. For instance, users can click on the “back” button in their browser to quickly go to the previous category page and to choose a different item. They can also reach the navigation menu to browse another category, or use the search function on the top bar to perform a global product search. Consequently, the tracking of all these possible user interactions can help in establishing a correlation pattern among different products on the website. Figure 3.9 illustrates the result of the search by the term “blazer” on the search bar within a given product page.

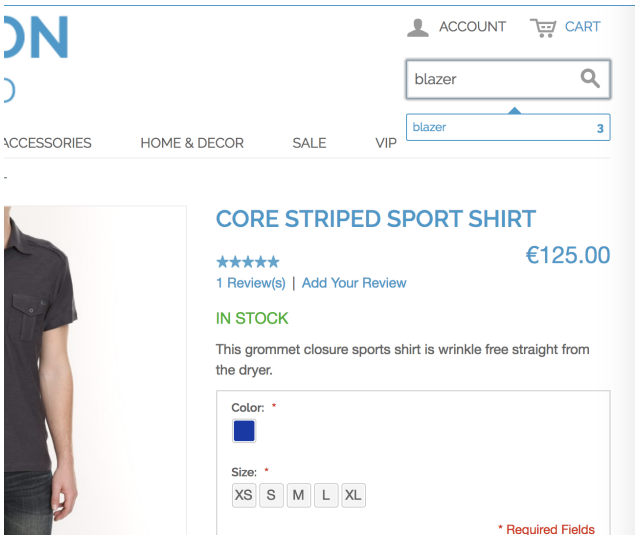


Figure 3.9: Product search bar

When the search is performed, the user is taken to the search result page, which lists the products matching the searched term. From there, the user can freely browse to any of the available product pages, similarly to the navigation process described for the category listing page (Figure 3.10).

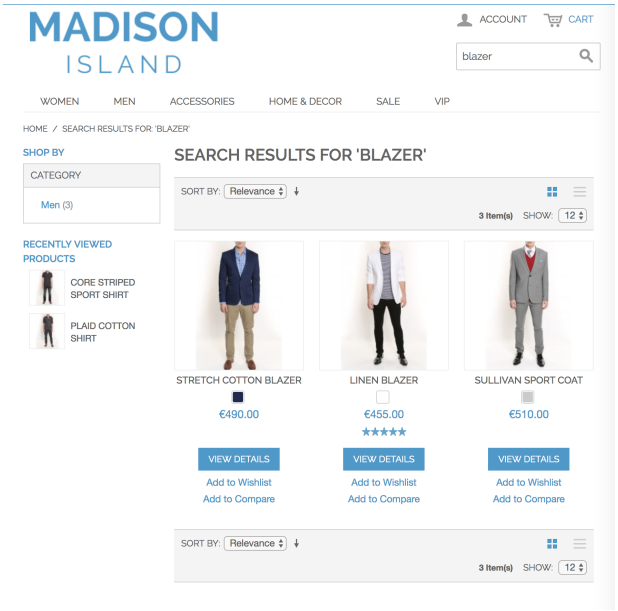


Figure 3.10: Search results page

At this point, we can update and extend the IFML model shown in Figure 3.6 according to the new notions and interactions that have been described so far.

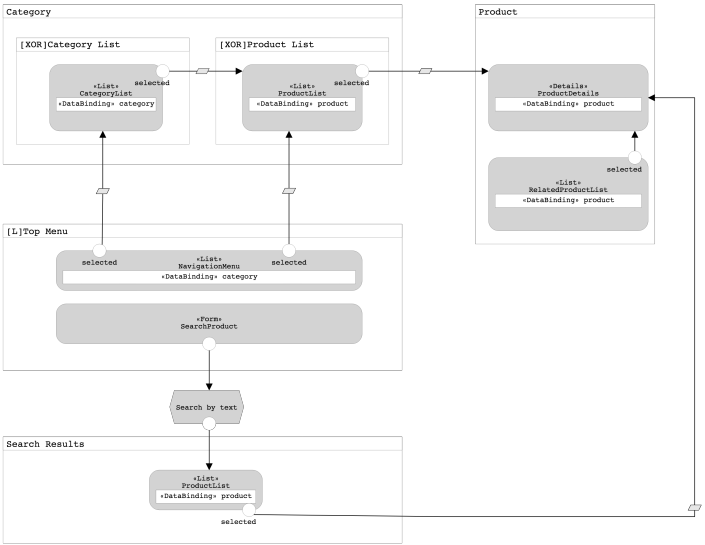


Figure 3.11: Updated IFML representation of the navigational behaviours

These new navigational paths are represented in the application server access log in the following manner:

Application Server Access Log		
ID	**Page**	**Log Entry**
1	*"Plaid Cotton Shirt"* Product Page	*[04/Dec/2017:06:37:06 +0000] "GET /men/shirts/plaid-cotton-shirt-476.html" 200 0 - 29505*
2	*"Core Striped Sport Shirt"* Product Page	*[04/Dec/2017:06:37:15 +0000] "GET /core-striped-sport-shirt-551.html" 200 0 - 29505*
3	*"Plaid Cotton Shirt"* Product Page	*[04/Dec/2017:06:37:21 +0000] "GET /men/shirts/plaid-cotton-shirt-476.html" 200 0 - 29505*
5	*"Tees Knits And Polos"* Category Page	*[04/Dec/2017:06:38:06 +0000] "GET /men/tees-knits-and-polos.html" 200 0 - 29505*
6	*"Blazer"* Search By Term	*[04/Dec/2017:06:38:20 +0000] "GET /catalogsearch/result/?q=blazer" 200 0 - 29505*
7	*"Stretch Cotton Blazer"* Product Page	*[04/Dec/2017:06:38:43 +0000] "GET /stretch-cotton-blazer-587.html" 200 0 - 29505*

The sequence of actions as seen in this log reveals that the user browsed from one product to another, taking advantage of the related product links (ID 2). In fact, the target URL does not include any category path beside the URL key related to the product, which indicates a direct access. The entries 3 and 4 illustrate the journey of an user who has clicked on the browser back button and has performed the same actions again. The last three recorded actions show, respectively, a direct access to a category page through the navigational menu, a search by the *"blazer"* term as per the previous example, and the related redirection to the product page.

3.3 IoT behaviour profiling

As mobile devices surpass desktop computers in driving purchases and in enriching behaviour data, location and proximity tracking becomes a valuable tool for brands and stores. In the following subsections, we analyse two possible scenarios of customer interaction in the physical world based on the recording and reporting capabilities of IoT devices. Such data would then be collected together with other web-based information (2.1) to form a comprehensive behavioural data stream that could leverage for generating tailored customisations of the Madison eCommerce portal.

3.3.1 Apple iBeacon technology and Estimote Beacons overview

The IoT device chosen to illustrate the scenarios related to the behavioural modeling would be the Estimote Beacons, which use Apple iBeacon technology and are compatible with iBeacon-enabled Apple products and applications.

The company from Cupertino jumped first on the beacon bandwagon by publishing in 2013 a detailed specification (IDs, transmission intervals, etc.) for developing the iBeacon protocol, which in turn allowed vendors, such as Estimote, to ship iBeacon-compatible hardware transmitters worldwide.

Figure 3.12: An Estimote iBeacon compatible device

As described in Section 2.1, a beacon can simply be seen as a lighthouse that broadcasts information in certain intervals and at a defined power, leveraging Low Energy Bluetooth connection. In the case of iBeacons, the information sent to listening mobile devices would contain:

- The Universal Unique ID (UUID), which is globally unique. Example: de2b45ae-ed98-11e4-3432-78616d6f6f6d
- The Major ID, which uniquely identifies our customer's system: e.g. 51314
- The Minor ID, which reports the exact location or object (in our case, the spot): e.g. 23369

Unlike QR and NFC communication, the customer needs to have an app installed on their phone to receive this one-way data stream. Technically, the app obeys only to iBeacons with UUID, Major and Minor IDs matching to predefined values. When that happens, the app can react accordingly. Such mechanism ensures that only the installed app can track users as they walk around the transmitters.

In other words, the phone OS will keep listening for beacons even if the app is not running, and even if the phone is locked or rebooted. Once either an "enter" or and "exit" event happens, the OS will launch the app into the background (if needed) and let it execute, for a few seconds, some code to handle the event.

3.3.2 Proximity Marketing

Proximity Marketing is an efficient tool not only for discovering and engaging new customers, but also for better targeting existing ones. This marketing technique operates in a given physical location by leveraging the IoT ecosystem to promote products and services. This communication channel acts on a clear target: all the customers close to, or within, an area covered by the diffusion devices.

Such innovative form of relationship marketing aims to activate and involve users by drawing their attention at the right time and in the right place, creating more intense and stimulating shopping experiences.

In this work, we will focus on a basic scenario where the recognition of proximity in the physical world does not directly trigger an immediate action to grab customer attention, but instead is limited to the silent tracking of the event as well as the automatic recording of data on a specific web server.

We start by defining a possible allocation of items for a "Madison Island" retail store, which resembles the catalogue presented on its website.

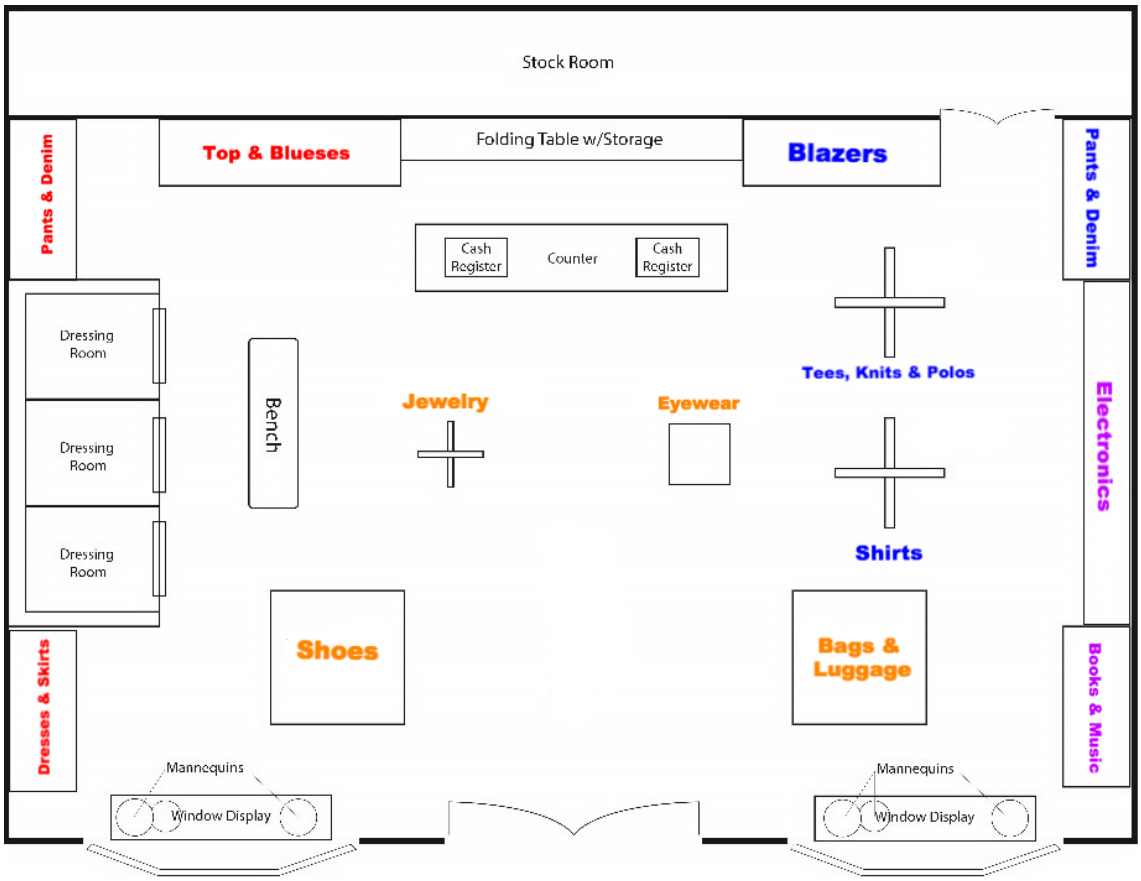


Figure 3.13: Madison Island brick-and-mortar store map

Each label described in Figure 3.13 represents a specific category of the website, whereas the color of the label indicates the parent category to which the products belong. More specifically:

- Red represents the **Women** category, which maps the content available at /**women.html**.
- Blue represents the **Men** category, which maps the content available at /**men.html**.
- Purple represents the **Home & Decor** category, which maps the content available at /**home-decor.html**.
- Orange represents the **Accessories** category, which maps the content available at /**accessories.html**.

As a customer walks around the physical shop, the Madison Island mobile app looks for a predefined set of beacon regions. Whenever the device either enters or exits each region, the app registers proximity data and tracks the aisles (regions) visited or not by the customer.

In this scenario, the following image illustrates what would be a suitable Estimote beacon allocation for the store:

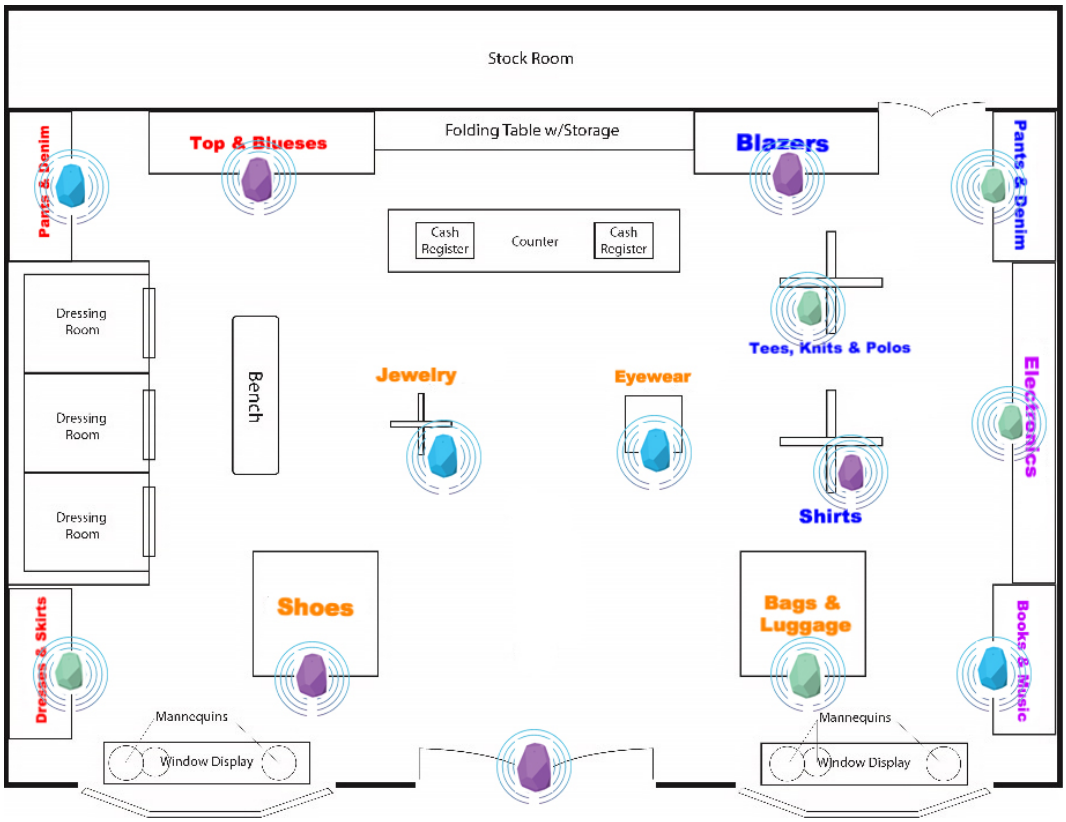


Figure 3.14: Madison Island brick-and-mortar store with Estimote beacons allocation

Depending on the business use case, the mobile app can either send an event to the listening server whenever the user crosses a region boundary, or submit a single event with a full list of regions and their minimum proximities detected during the customer visit.

Due to the behavioural profiling nature of this activity, we assume that the app does the latter, detecting all the entering and leaving of each virtual fence and activating a *ranging* procedure that detects proximity information based on the strength of the Bluetooth signal.[3].

Once the shoppers leaves the physical store, the app performs a single data push to a REST API endpoint with all the tracked data.

From a technical perspective, all these operations are accomplished leveraging the Estimote iOS-SDK[4], which allows the developer to quickly define regions and ranges for each beacon and facilitates the tracking of the actual proximity to beacon through the ranging process.

As an example, the JSON payload sent from the app to the REST endpoint for an hypothetical customer 3045678 (e.g. **POST /users/3045678/sessions**) would have the following structure:

```
1  {
2    “data”:{
3        “customerId”:3045678,
4        “storeId”:8784,
```

```
5       “storeLabel” :“Madison1”,
6       “sessionId” :“89376f84−065b−11e8−ba89−0ed5f89f718b”
7
8       “sessionRegions”:[
9           {
10              ”regionId: :156,
11              “regionLabel”:“store−entrance”,
12              “detectionCount”:2,
13              “maxSecondsInRegion”: 5,
14              “maxProximity”:“unknown”,
15              “firstDetectionTimestamp”:”2018−02−21T18:09:07Z”,
16              “lastDetectionTimestamp”:”2018−02−21T18:16:02Z”,
17              “beaconData” :{
18                  “uuid”:“0686a88e−fed6−11e7−8be5−0ed5f89f718b”,
19                  “majorId”:2553,
20                  “minorId”:79
21              }
22          },
23          {
24              ”regionId: :645,
25              “regionLabel”:“shoes”,
26              “detectionCount”:1,
27              “maxSecondsInRegion”: 24,
28              “maxProximity”:“near”,
29              “firstDetectionTimestamp”:”2018−02−21T18:09:20Z”,
30              “lastDetectionTimestamp”:”2018−02−21T18:09:20Z”,
31              “beaconData” :{
32                  “uuid”:“0686a88e−fed6−11e7−8be5−0ed5f89f718b”,
33                  “majorId”:19029,
34                  “minorId”:49
35              }
36          },
37          {
38              “regionId” :6875,
39              “regionLabel”:“jewelry”,
40              “detectionCount”:1,
41              “maxSecondsInRegion”: 15,
42              “maxProximity”:“far”,
43              “firstDetectionTimestamp”:”2018−02−21T18:10:15Z”,
44              “lastDetectionTimestamp”:”2018−02−21T18:10:15Z”,
45              “beaconData” :{
46                  “uuid”:“0686a88e−fed6−11e7−8be5−0ed5f89f718b”,
```

```
                    ‘‘majorId”:38415,
                    ‘‘minorId”:59
                }
            },
            {
                ‘‘regionId” :2563,
                ‘‘regionLabel”:‘‘blazers”,
                ‘‘detectionCount”:1,
                ‘‘maxSecondsInRegion”: 195,
                ‘‘maxProximity”:‘‘immediate”,
                ‘‘firstDetectionTimestamp”:”2018−02−21T18:11:01Z”,
                ‘‘lastDetectionTimestamp”:”2018−02−21T18:11:01Z”,
                ‘‘beaconData” :{
                    ‘‘uuid”:‘‘0686a88e−fed6−11e7−8be5−0ed5f89f718b”,
                    ‘‘majorId”:25911,
                    ‘‘minorId”:27
                }
            },
            {
                ‘‘regionId” :456,
                ‘‘regionLabel”:‘‘tees−knits−polos”,
                ‘‘detectionCount”:1,
                ‘‘maxSecondsInRegion”: 10,
                ‘‘maxProximity”:‘‘far”,
                ‘‘firstDetectionTimestamp”:”2018−02−21T18:14:56Z”,
                ‘‘lastDetectionTimestamp”:”2018−02−21T18:14:56Z”,
                ‘‘beaconData” :{
                    ‘‘uuid”:‘‘0686a88e−fed6−11e7−8be5−0ed5f89f718b”,
                    ‘‘majorId”:42037,
                    ‘‘minorId”:36
                }
            },
            {
                ‘‘regionId” :998,
                ‘‘regionLabel”:‘‘bags−and−luggage”,
                ‘‘detectionCount”:1,
                ‘‘maxSecondsInRegion”: 7,
                ‘‘maxProximity”:‘‘far”,
                ‘‘firstDetectionTimestamp”:”2018−02−21T18:15:12Z”,
                ‘‘lastDetectionTimestamp”:”2018−02−21T18:15:12Z”,
                ‘‘beaconData” :{
                    ‘‘uuid”:‘‘0686a88e−fed6−11e7−8be5−0ed5f89f718b”,
```

```
                    "majorId":37931,
                    "minorId": 85
                }
            }
        ]
    }
}
```

The above example session shows an evident preference for "Blazer" items, and a slight interest in "Shoes" items. More precisely, the "Blazer" region registered a session that lasted over 3 minutes and that was the closest to a beacon.

3.3.3 Customer Rewards

Besides allowing proximity based marketing, beacon technology can also be used to reward customers for particular actions based on geolocalisation data. Such rewards could increase brand engagement, enhancing customer loyalty and fostering lasting customer relationships.

Achieving such results is possible by extending the set of actions that enable customers to earn bonuses and discounts on the website (newsletter subscriptions, minimum order amount, etc..) to activities performed in the physical world, including the simple act of visiting and walking around the brick-and-mortar store.

For example, the brand can rank customers by the amount of time spent at each Madison Retail shop, rewarding them with monthly offers tailored according to their purchases. The brand can also focus on offering special offers on the website to the customers that visited a retail store in a specific time span, such as Christmas.

For our Madison Island example, we are considering a scenario where customers are rewarded with a fixed amount of points on their online account if they scan three QR codes from in-store products. Specifically, when the beacon detects their entrance into the store, the mobile app pushes a CheckPoints notification to the customer, inviting them to scan codes of items available in the shop. Once the scans are correctly performed within a session, the mobile app pushes the information to the same REST API presented in the previous chapter, which then stores the data.

Figure 3.15: Madison Island mobile app push notification example

The JSON payload sent to the server (e.g. **POST /users/3045678/scans**) after each successful scan would then have a structure similar to the following one:

```
{
  “data”:{
     “customerId”:3045678,
     “storeId”:8784,
     “storeLabel” :“Madison1”,
     “sessionId” :“89376f84−065b−11e8−ba89−0ed5f89f718b”
     “sessionDuration” :456,
     “sessionActions”: [{
        “userAgent” :“Iphone 6s”,
        “scannedItems”:
          [{
            “barcode”:“042100005264”,
            “name” :“Elizabeth Knit Top−Red−S”
```

```
14              "sku": "wbk012c-Red-S"
15            },
16            {
17              "barcode":"042100005931",
18              "name" :"Plaid Cotton Shirt-Khaki-L"
19              "sku": "msj006c-Khaki-L"
20            },
21            {
22              "barcode":"042100007717",
23              "name" :"Broad St Saddle Shoes"
24              "sku": "shm00110"
25            }
26           ]
27      }]
28    }
29 }
```

Similarly to the process outlined in 3.3.2, the product data collected when customers scan codes during their visits to the physical store will potentially be converted into reward points to be used by the same customer on the website.

Chapter 4

Our approach to modeling

While in the last chapter we introduced three different streams of real usage data obtained both from the physical and the virtual world, we now focus on expanding those representations in a more detailed way with the help of the Model-Driven Engineering techniques briefly described in 1.3. Concretely, the goal of the first two sections is to illustrate the defining languages (metamodels) for both the real usage data and the eCommerce platform interactions described in the previous examples. Besides, we will generate actual models based upon those metamodels, which represent the very same information. Finally, in the last section, we will use the same model generation for updating the models instantiated previously, leveraging model transformation techniques based upon usage pattern detection resulting from the Big Data analysis.

4.1 Real usage data modeling

4.1.1 Metamodel

The representation of the real usage data starts from the definition of the metamodel that defines the languages and processes from which to form a model without making statements about its content. In fact, a metamodel is itself a model that is used to describe another model by using a modeling language at a different level of abstraction.

Figure in 4.1 describes the processed metamodel as a UML Class Diagram according to the data retrieved for our real usage data analysis.

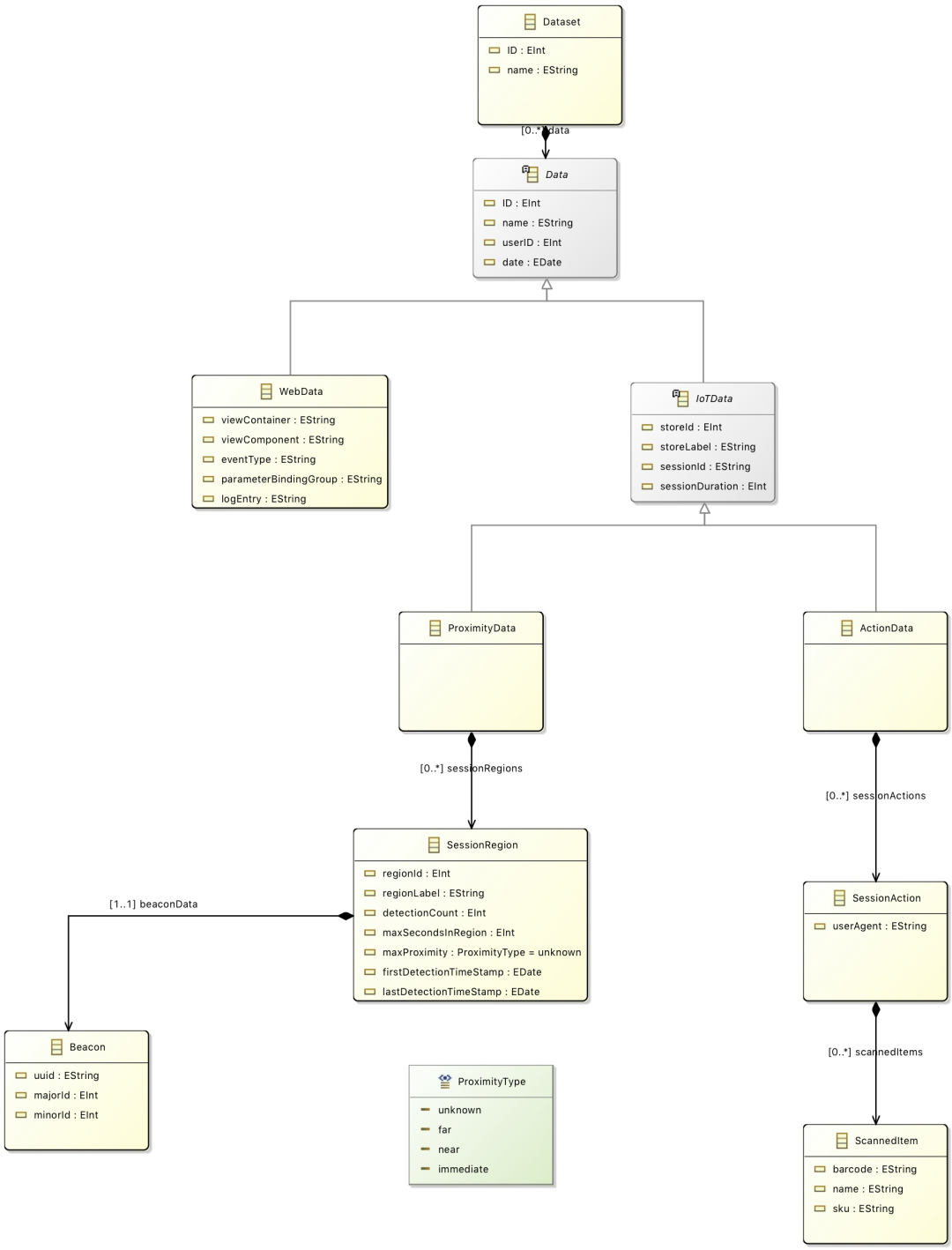


Figure 4.1: Real Usage Data Metamodel Diagram Class

4.1.2 Model

The RealUsageData metamodel defined above allows us to create dynamic instances that precisely map the real usage data collected both from web mining process and from IoT devices tracking. Figure 4.2 illustrates this processed model in its ecore representation form in Eclipse Modeling Tools (EMF). The figure is followed by the corresponding model content.

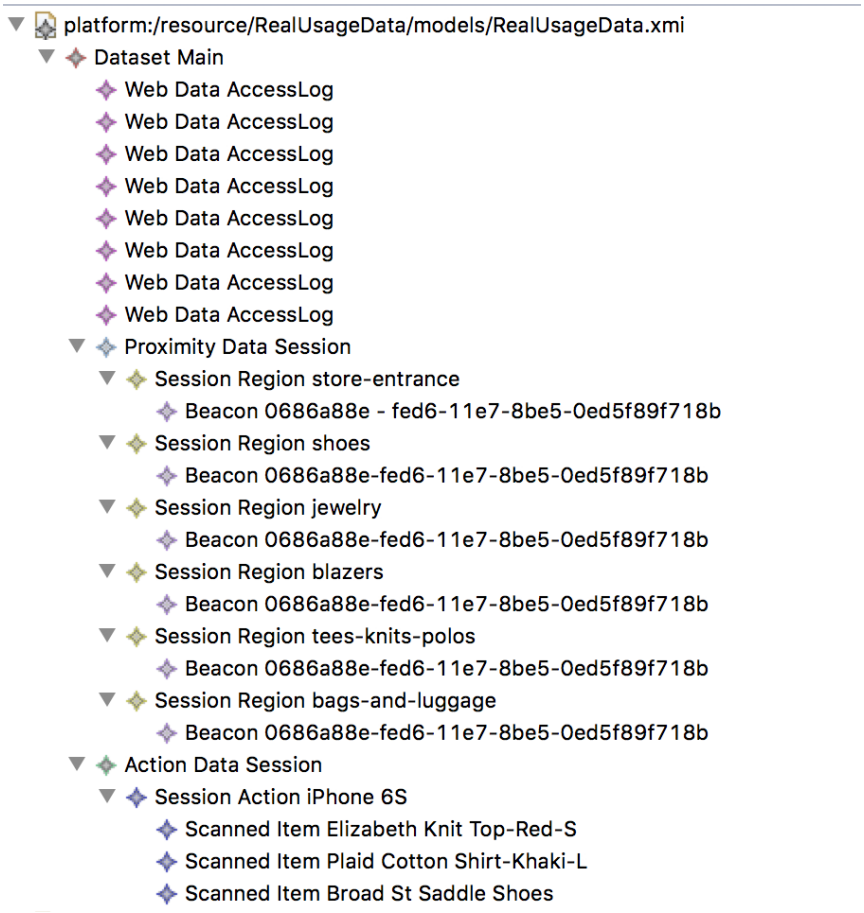


Figure 4.2: Real Usage Data Model in EMF

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <RealUsageData:Dataset
3      xmi:version="2.0"
4      xmlns:xmi="http://www.omg.org/XMI"
5      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
6      xmlns:RealUsageData="RealUsageData"
7      xsi:schemaLocation="RealUsageData ../metamodels/RealUsageData.ecore"
8      ID="1" name="Main">
9    <data xsi:type="RealUsageData:WebData"
10       ID="1"
```

```
11      name="AccessLog"
12      userID="3045678"
13      date="2017-11-29T17:06:49.000+0100"
14      viewContainer="Homepage"
15      viewComponent="TopMenu"
16      eventType="click"
17      parameterBindingGroup="Category/5"
18      logEntry="GET /men.html / 200 0 - 29505"/>
19  <data xsi:type="RealUsageData:WebData"
20      ID="2"
21      name="AccessLog"
22      userID="3045678"
23      date="2017-11-29T17:07:04.000+0100"
24      viewContainer="Category #5"
25      viewComponent="CategoryList"
26      eventType="click"
27      parameterBindingGroup="Category/15"
28      logEntry="GET /men/shirts.html 200 0 - 29505"/>
29  <data xsi:type="RealUsageData:WebData"
30      ID="3"
31      name="AccessLog"
32      userID="3045678"
33      date="2017-11-29T07:08:40.000+0100"
34      viewContainer="Category #15"
35      viewComponent="ProductList"
36      eventType="click"
37      parameterBindingGroup="Product/404"
38      logEntry="GET /men/shirts/plaid-cotton-shirt-476.html 200 0 - 29505"/>
39  <data xsi:type="RealUsageData:WebData"
40      ID="4"
41      name="AccessLog"
42      userID="3045678"
43      date="2017-12-04T06:37:15.000+0100"
44      viewContainer="Product #404"
45      viewComponent="RelatedProductList"
46      eventType="click"
47      parameterBindingGroup="Product/413"
48      logEntry="GET /core-striped-sport-shirt-551.html 200 0 - 29505"/>
49  <data xsi:type="RealUsageData:WebData"
50      ID="5"
51      name="AccessLog"
52      userID="3045678"
```

```
53      date="2017-12-04T06:37:21.000+0100"
54      viewContainer=""
55      viewComponent=""
56      eventType="backButton"
57      parameterBindingGroup=""
58      logEntry="GET /men/shirts/plaid-cotton-shirt-476.html 200 0 - 29505"/>
59  <data xsi:type="RealUsageData:WebData"
60      ID="6"
61      name="AccessLog"
62      userID="3045678"
63      date="2017-12-04T06:38:06.000+0100"
64      viewContainer="Product #404"
65      viewComponent="TopMenu"
66      eventType="click"
67      parameterBindingGroup="Category/16"
68      logEntry="GET /men/tees-knits-and-polos.html 200 0 - 29505"/>
69  <data xsi:type="RealUsageData:WebData"
70      ID="7"
71      name="AccessLog"
72      userID="3045678"
73      date="2017-12-04T06:38:20.000+0100"
74      viewContainer="Category #16"
75      viewComponent="TopSearch"
76      eventType="submit"
77      parameterBindingGroup="SearchText/blazer"
78      logEntry="GET /catalogsearch/result/?q=blazer 200 0 - 29505"/>
79  <data xsi:type="RealUsageData:WebData"
80      ID="8"
81      name="AccessLog"
82      userID="3045678"
83      date="2017-12-04T06:38:20.000+0100"
84      viewContainer="Search Results"
85      viewComponent="ProductList"
86      eventType="click"
87      parameterBindingGroup="Product/407"
88      logEntry="GET /stretch-cotton-blazer-587.html 200 0 - 29505"/>
89  <data xsi:type="RealUsageData:ProximityData"
90      ID="9"
91      name="Session"
92      userID="3045678"
93      date="2018-02-21T18:16:07.000+0100"
94      storeId="8784"
```

```
95      storeLabel="Madison1"
96      sessionId="89376f84-065b-11e8-ba89-0ed5f89f718b"
97      sessionDuration="345">
98    <sessionRegions
99        regionId="156"
100       regionLabel="store-entrance"
101       detectionCount="2"
102       maxSecondsInRegion="5"
103       firstDetectionTimeStamp="2018-02-21T18:09:07.000+0100"
104       lastDetectionTimeStamp="2018-02-21T18:16:02.000+0100">
105     <beaconData
106         uuid="0686a88e-fed6-11e7-8be5-0ed5f89f718b"
107         majorId="2553"
108         minorId="79"/>
109   </sessionRegions>
110   <sessionRegions
111       regionId="645"
112       regionLabel="shoes"
113       detectionCount="1"
114       maxSecondsInRegion="24"
115       maxProximity="near"
116       firstDetectionTimeStamp="2018-02-21T18:09:20.000+0100"
117       lastDetectionTimeStamp="2018-02-21T18:09:20.000+0100">
118     <beaconData
119         uuid="0686a88e-fed6-11e7-8be5-0ed5f89f718b"
120         majorId="19029"
121         minorId="49"/>
122   </sessionRegions>
123   <sessionRegions
124       regionId="6875"
125       regionLabel="jewelry"
126       detectionCount="1"
127       maxSecondsInRegion="15"
128       maxProximity="far"
129       firstDetectionTimeStamp="2018-02-21T18:10:15.000+0100"
130       lastDetectionTimeStamp="2018-02-21T18:10:15.000+0100">
131     <beaconData
132         uuid="0686a88e-fed6-11e7-8be5-0ed5f89f718b"
133         majorId="38415"
134         minorId="59"/>
135   </sessionRegions>
136   <sessionRegions
```

```
137         regionId="2563"
138         regionLabel="blazers"
139         detectionCount="1"
140         maxSecondsInRegion="195"
141         maxProximity="immediate"
142         firstDetectionTimeStamp="2018-02-21T18:11:01.000+0100"
143         lastDetectionTimeStamp="2018-02-21T18:11:01.000+0100">
144       <beaconData
145           uuid="0686a88e-fed6-11e7-8be5-0ed5f89f718b"
146           majorId="25911"
147           minorId="27"/>
148     </sessionRegions>
149     <sessionRegions
150         regionId="456"
151         regionLabel="tees-knits-polos"
152         detectionCount="1"
153         maxSecondsInRegion="10"
154         maxProximity="immediate"
155         firstDetectionTimeStamp="2018-02-21T18:14:56.000+0100"
156         lastDetectionTimeStamp="2018-02-21T18:14:56.000+0100">
157       <beaconData
158           uuid="0686a88e-fed6-11e7-8be5-0ed5f89f718b"
159           majorId="42037"
160           minorId="36"/>
161     </sessionRegions>
162     <sessionRegions
163         regionId="998"
164         regionLabel="bags-and-luggage"
165         detectionCount="1"
166         maxSecondsInRegion="7"
167         maxProximity="far"
168         firstDetectionTimeStamp="2018-02-21T18:15:12.000+0100"
169         lastDetectionTimeStamp="2018-02-21T18:15:12.000+0100">
170       <beaconData
171           uuid="0686a88e-fed6-11e7-8be5-0ed5f89f718b"
172           majorId="37931"
173           minorId="85"/>
174     </sessionRegions>
175   </data>
176   <data xsi:type="RealUsageData:ActionData"
177       ID="10"
178       name="Session"
```

```
179       userID="3045678"
180       date="2018-02-22T15:27:09.000+0100"
181       storeId="8784"
182       storeLabel="Madison1"
183       sessionId="89376f84-065b-11e8-ba89-0ed5f89f718b"
184       sessionDuration="456">
185     <sessionActions
186         userAgent="iPhone 6S">
187       <scannedItems
188           barcode="042100005264"
189           name="Elizabeth Knit Top-Red-S"
190           sku="wbk012c-Red-S"/>
191       <scannedItems
192           barcode="042100005931"
193           name="Plaid Cotton Shirt-Khaki-L"
194           sku="msj006c-Khaki-L"/>
195       <scannedItems
196           barcode="042100007717"
197           name="Broad St Saddle Shoes"
198           sku="shm00110"/>
199     </sessionActions>
200   </data>
201 </RealUsageData:Dataset>
```

4.2 eCommerce application modeling

As we have briefly introduced in 3.1, IFML is used to design platform independent-level models that can be used to define the interactions between the users of an application and the application itself. At its core, IFML is meant to be flexible and straightforward, but perhaps more importantly, the language is intended to be abstract enough to allow the definition of the main traits of an application front-end making as few visual commitments as possible. Furthermore, its extendibility allows modelers and designers to specialise specific components, enriching the semantics of their models and making the diagrams more readable.

In fact, the models generated using IFML describe the user-interface components required at the front-end of the application, without specifying layout details of these elements. Such approach enhances the separation of concerns among developers and UX designers, in which the latter builds the user interface according to an interaction flow model. Besides defining components of the User Interface, these models explain how data flows among different sections of the application upon triggering events, and introduces the business logic carried out using that data.

4.2.1 IFML Metamodel

The IFML metamodel is organised into three packages: the Core, the Extension, and the DataTypes. The Core package contains the concepts that build up the interaction infrastructure of the language concerning InteractionFlowElements, InteractionFlows and Parameters. The Extension package extends the Core package components with more complex behaviours. The DataTypes package, in its turn, contains the custom data types defined by IFML.

Figure 4.3: IFML metamodel in EMF

By using the primitive data types from the UML metamodel and a UML representation for the IFML Domain Model, the IFML metamodel specifies a set of UML metaclasses as the foundation for the IFML metaclasses.

The following structure is a high-level representation of the IFML metamodel and its areas of concern:

- IFML Model
- Interaction Flow Model
- Interaction Flow Elements
- View Elements
- Events
- Specific Events and View Components
- Parameters
- Expressions
- ContentBinding

Figure 4.4 shows an excerpt of the IFML metamodel. The IFMLModel is the top-level container of all the model elements and represents an *IFMLModel*. It contains an *InteractionFlowModel* that is the user view of an application, a *DomainModel* represented in UML, and optional ViewPoints. The concepts extending *ViewContainer*, *ViewComponents*, *ViewComponentPart*, and *ViewElementEvent* represent the visual elements of an *IFMLModel*.

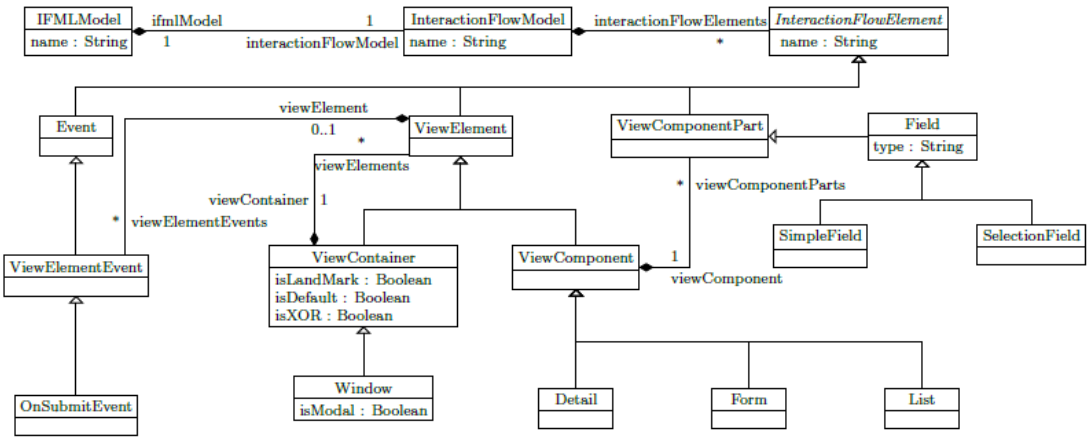


Figure 4.4: Class diagram of an IFML subset.

4.2.2 Model

As mentioned previously, interaction flow models are described using the Interaction Flow Modelling Language and, together with the domain model and optionally viewpoints, they form the core of the *IFMLModel*.

Essentially, the goal of the domain model is to offer to the interaction flow references about the available content. An example of a domain model for an e-commerce website is given in Figure 4.5.

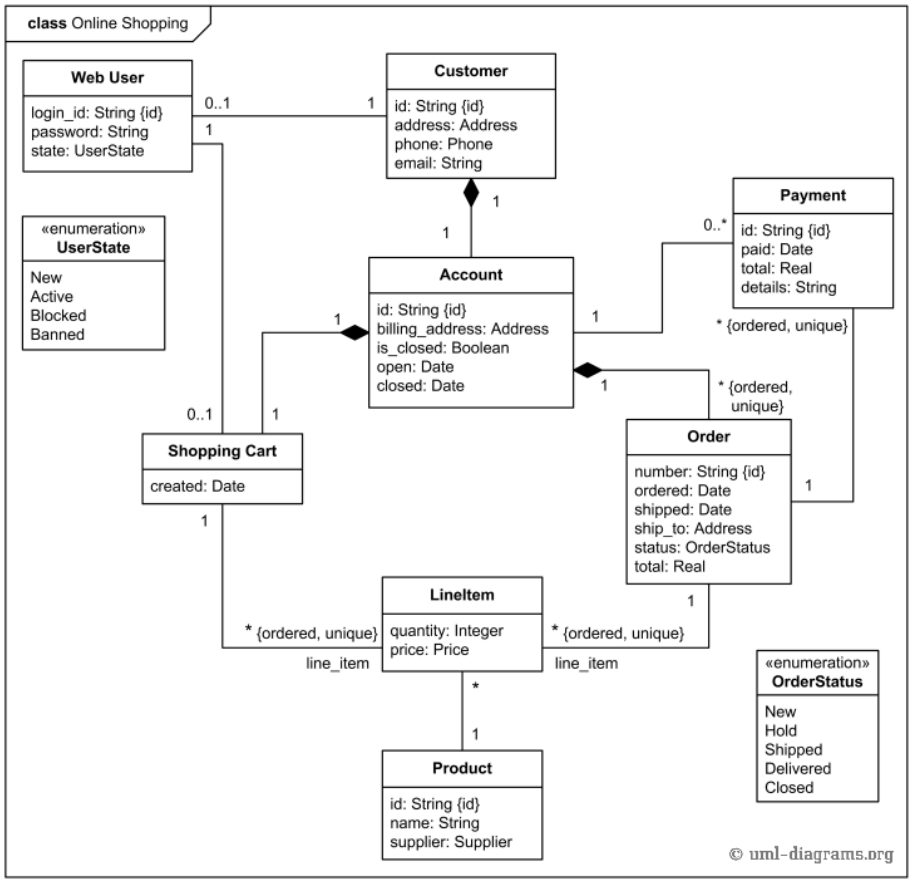


Figure 4.5: Example of a Domain Model Class Diagram for an eCommerce website

Although some partial *IFMLModel* representations for the Madison Island eCommerce platform have already been introduced in 3.2, in this subsection we explore the most important ones in more detail and with a broader approach that is not strictly related to the navigational modeling. Our final goal is to build an *IFMLModel* which would represent the main pages and website interactions, taking advantage of the IFML metamodel described just above.

Global overview

The Madison Island Interaction Model is formed by a different combination of *IFMLWindow* elements connected through *IFMLAction*s reacting to distinct *Events* with different *ParameterBindings*. The detail of the modeling is contextual to the purpose of this work. Consequently, we have limited the inclusion of the possible interactions and elements presented on the website to the IFMLModel. In so doing, we keep the model design lightweight and reduce its complexity. Overall, the main *IFMLWindow* standalone nodes have been described with more detail when compared to other shared *ViewContainer* elements, such as the Header and the Footer. In Figure 4.6, a visual representation of the IFMLDiagram corresponding to the main *IFMLModel* for the Madison Island eCommerce platform is presented.

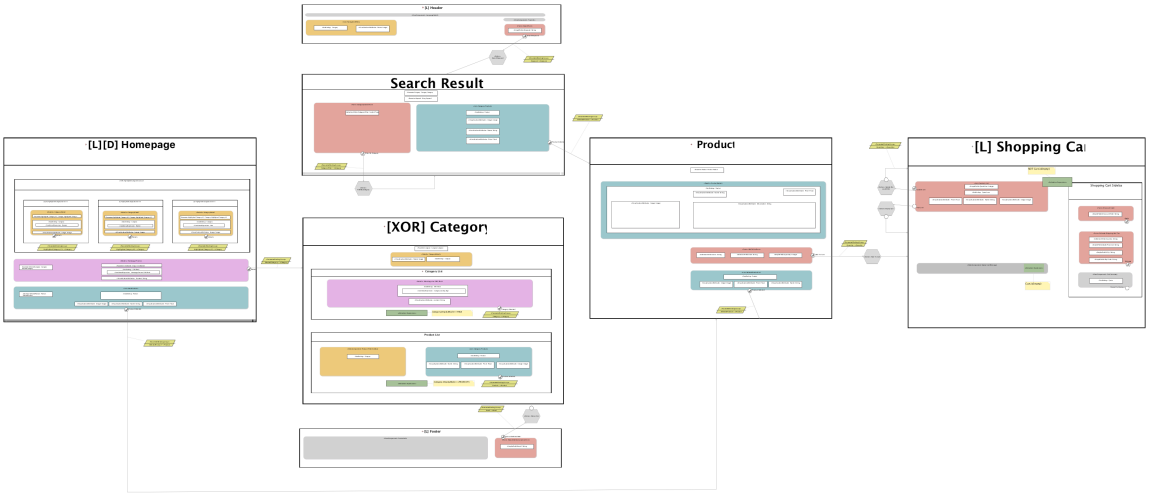


Figure 4.6: Main Madison Island IFML Diagram overview

Homepage

The Madison Island Interaction Model for the Homepage (Figure 4.7 and 4.8) is composed by a parent *IFMLWindow* element, which contains three children elements: another *IFMLWindow* for the Highlighted Categories Carousel, a *Detail View Component* for the Homepage promos CMS Block, and a *List View Component* for the New Products sections that is bound to the Product Entity of the *Domain Model* . The parent HighlightedCategoriesCarousel *IFMLWindow* has the *isXOR* property set to true, representing three exclusive main configurations for the initial screen within the carousel mechanism. Each *Data Binding* within all these banner components is limited by a *Conditional Expressions* defining the instance of the content to show.

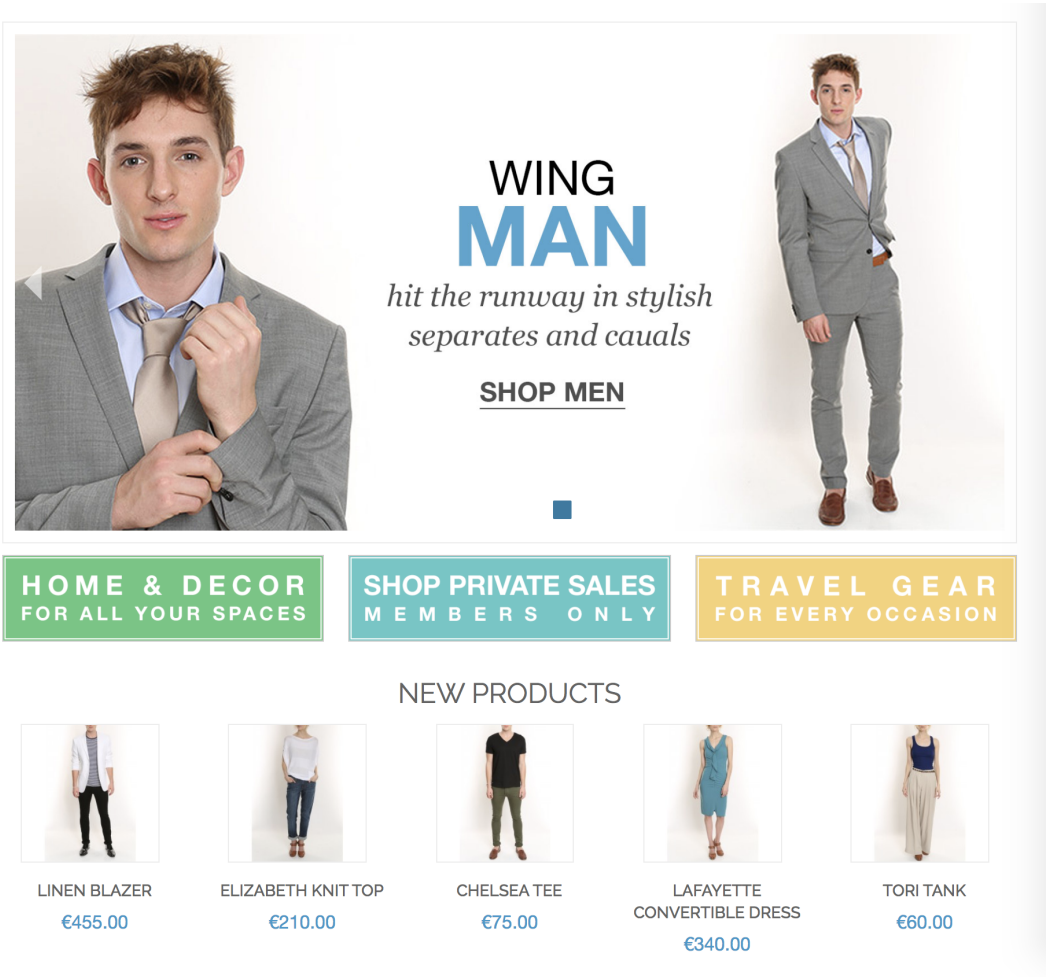


Figure 4.7: Homepage Desktop Version

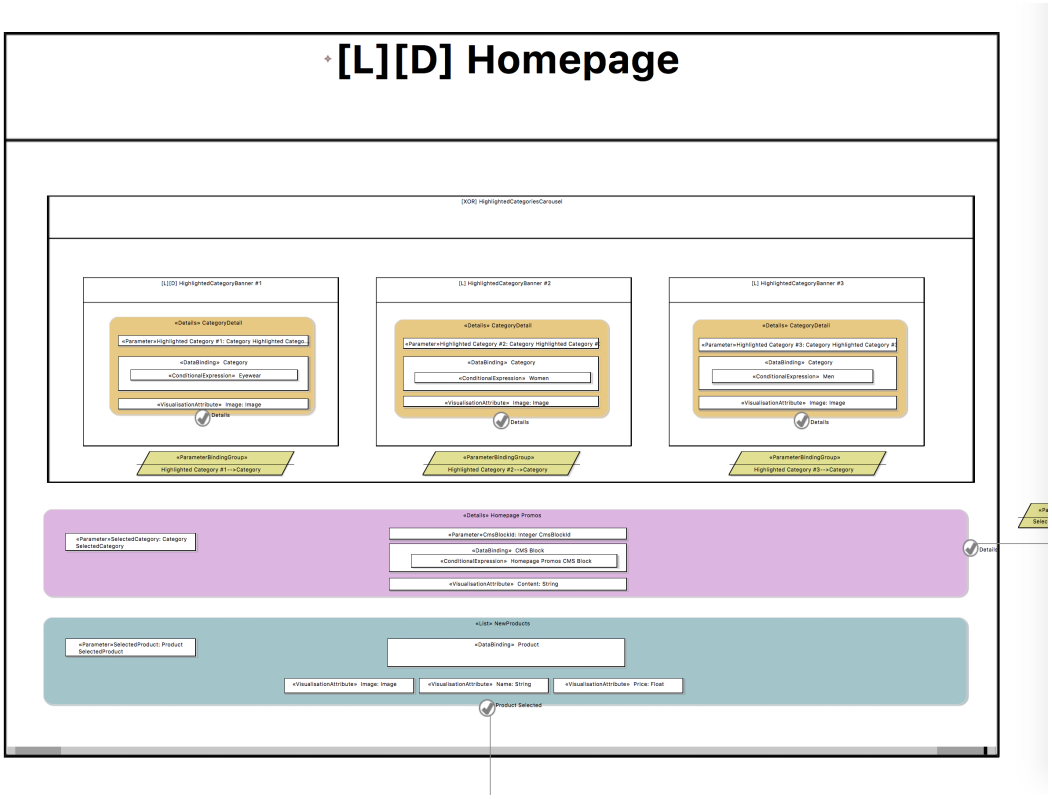


Figure 4.8: Homepage IFML Diagram

The following snippet of code is an extract from the *IFMLModel* for the first HighlighedCategoryBanner *View Container* element:

```
1   <viewElements xsi:type=”ext:Details” name=“CategoryDetail”>
2     <parameters name=”Highlighted Category #1” direction=“inout”>
3       <constraints language=“SQL” body=”Category.ID=18”/>
4     </parameters>
5     <viewElementEvents xsi:type=”ext:OnSelectEvent” name=“Details” viewElement=”//
          @interactionFlowModel/@interactionFlowModelElements.0/@viewElements.0/
          @viewElements.0”>
6       <outInteractionFlows xsi:type=”core:NavigationFlow” targetInteractionFlowElement=”//
            @interactionFlowModel/@interactionFlowModelElements.6”>
7         <parameterBindingGroup >
8           <parameterBindings sourceParameter=”//@interactionFlowModel/
                @interactionFlowModelElements.0/@viewElements.0/@viewElements.0/
                @viewElements.0/@parameters.0” targetParameter=”//@interactionFlowModel/
                @interactionFlowModelElements.6/@parameters.0”/>
9         </parameterBindingGroup>
10      </outInteractionFlows>
11    </viewElementEvents>
```

```
12      <viewComponentParts xsi:type="core:DataBinding" name="Category" uniformResourceIdentifier
            ="">
13        <subViewComponentParts xsi:type="core:ConditionalExpression" language="SQL" body="
              Category.ID=18" name="Eyewear"/>
14      </viewComponentParts>
15      <viewComponentParts xsi:type="core:VisualizationAttribute" name="Image" featureConcept="//
            @domainModel/@domainElements.4"/>
16    </viewElements>
17  </viewElements>
```

The above snippet belongs to a more complex *IFMLModel* hierarchy, as shown in 4.9.

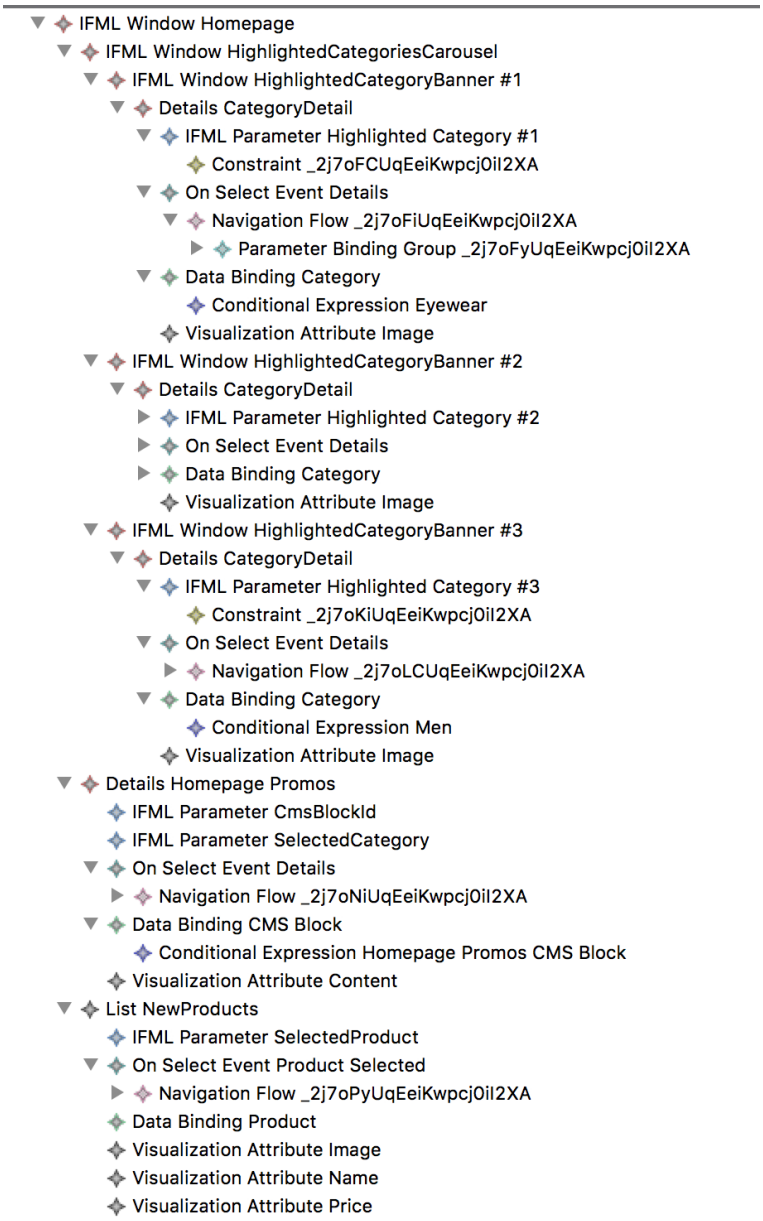
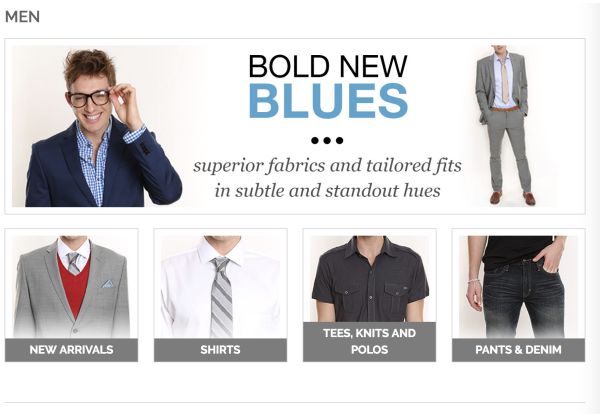


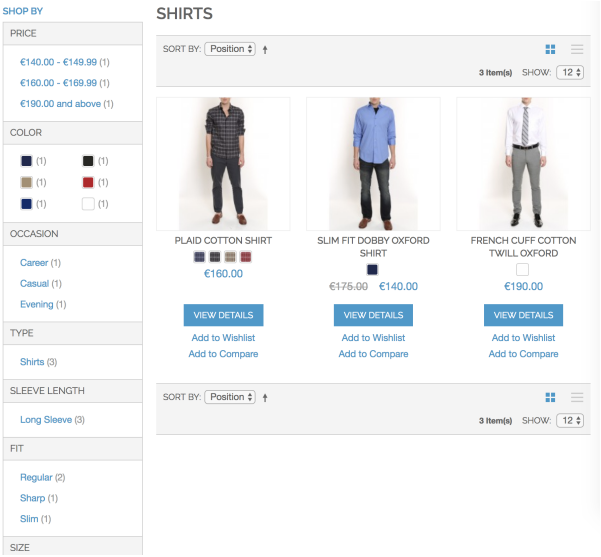
Figure 4.9: Interaction Flow Homepage Model in EMF

Category Page

The Madison Island Interaction Model for the Category Page (Figure 4.10 and 4.11) is composed by a parent *IFMLWindow* element with a true-ish *isXOR* property, which presents information about the current category on the top of the page. Depending on the display mode property for the Category Entity, the user can be presented with two different *View Containers* that are respectively activated using different *Activation Expressions* based on the value of the property itself. Whilst the first scenario presents a *Detail View Component* attached to a linked CMS Block, the second option shows two children view components representing both the filter sidebar and the products listing section with this last one having multiple *Visualization Attribute* children nodes. These nodes indicate that the user is presented with an image used as thumbnail, a name and a price for each product belonging to the category shown.



(a) Category Page with Display Mode PAGE



(b) Category Page with Display Mode PRODUCTS

Figure 4.10: Category Desktop Versions

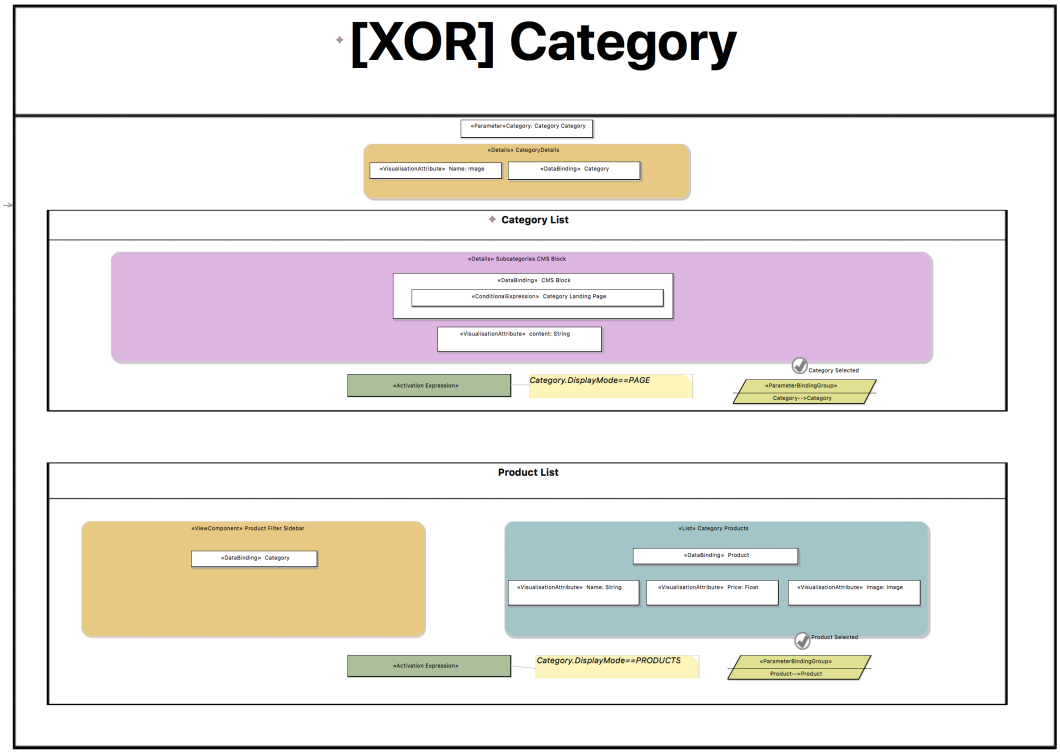


Figure 4.11: Category IFML Diagram

The *IFMLModel* code for the first Category Products List element we have just described takes this form:

```
1  <viewElements xsi:type="ext:List" name="Category Products">
2  <viewElementEvents xsi:type="ext:OnSelectEvent" name="Product Selected" viewElement="//
       @interactionFlowModel/@interactionFlowModelElements.6/@viewElements.1/@viewElements.0"
       >
3    <outInteractionFlows xsi:type="core:NavigationFlow" targetInteractionFlowElement="//
         @interactionFlowModel/@interactionFlowModelElements.1">
4      <parameterBindingGroup >
5        <parameterBindings sourceParameter="//@interactionFlowModel/
             @interactionFlowModelElements.1/@parameters.0" targetParameter="//
             @interactionFlowModel/@interactionFlowModelElements.1/@parameters.0"/>
6      </parameterBindingGroup>
7    </outInteractionFlows>
8  </viewElementEvents>
9  <viewComponentParts xsi:type="core:DataBinding" name="Product" domainConcept="//
       @domainModel/@domainElements.3">
10   <conditionalExpression language="SQL" body="Category IN Product.Categories" name="Category
         Products"/>
```

```
11      </viewComponentParts>
12      <viewComponentParts xsi:type="core:VisualizationAttribute" name="Image" featureConcept="//
            @domainModel/@domainElements.7"/>
13      <viewComponentParts xsi:type="core:VisualizationAttribute" name="Name" featureConcept="//
            @domainModel/@domainElements.8"/>
14      <viewComponentParts xsi:type="core:VisualizationAttribute" name="Price" featureConcept="//
            @domainModel/@domainElements.9"/>
15    </viewElements>
16    <viewElements xsi:type="core:ViewComponent" name="Product Filter Sidebar">
17      <viewComponentParts xsi:type="core:DataBinding" name="Category"/>
18    </viewElements>
19  </viewElements>
```

The full expanded model hierarchy for the *IFMLWindow* Category element is shown in Figure 4.12.

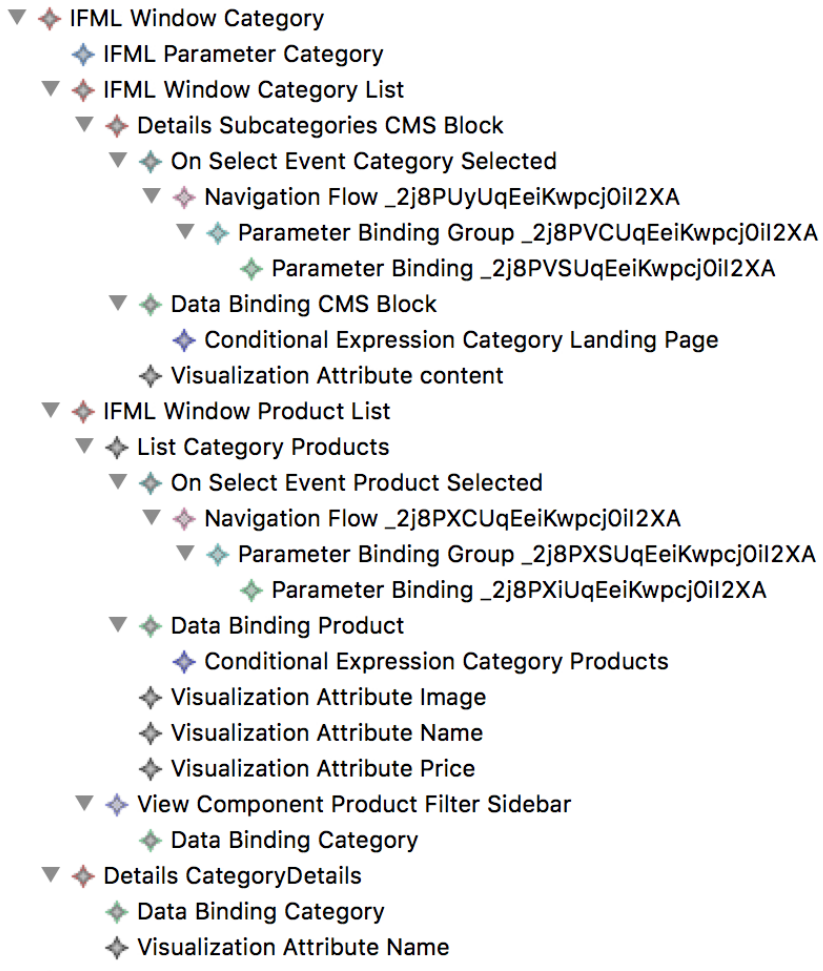


Figure 4.12: Interaction Flow Category Model in EMF

Product Page

The Madison Island Interaction Model for the Product page (Figure 4.13 and 4.14) is mainly built with a single *IFMLWindow* element that contains different types of *View Component* nodes, the main one being a *Detail View Component* instance bound to the current product data entity. The other two elements are the single *Form View Component* describing the Add to Cart section and its possible interactions, and the *List View Component* holding the information for the Related Product widget.

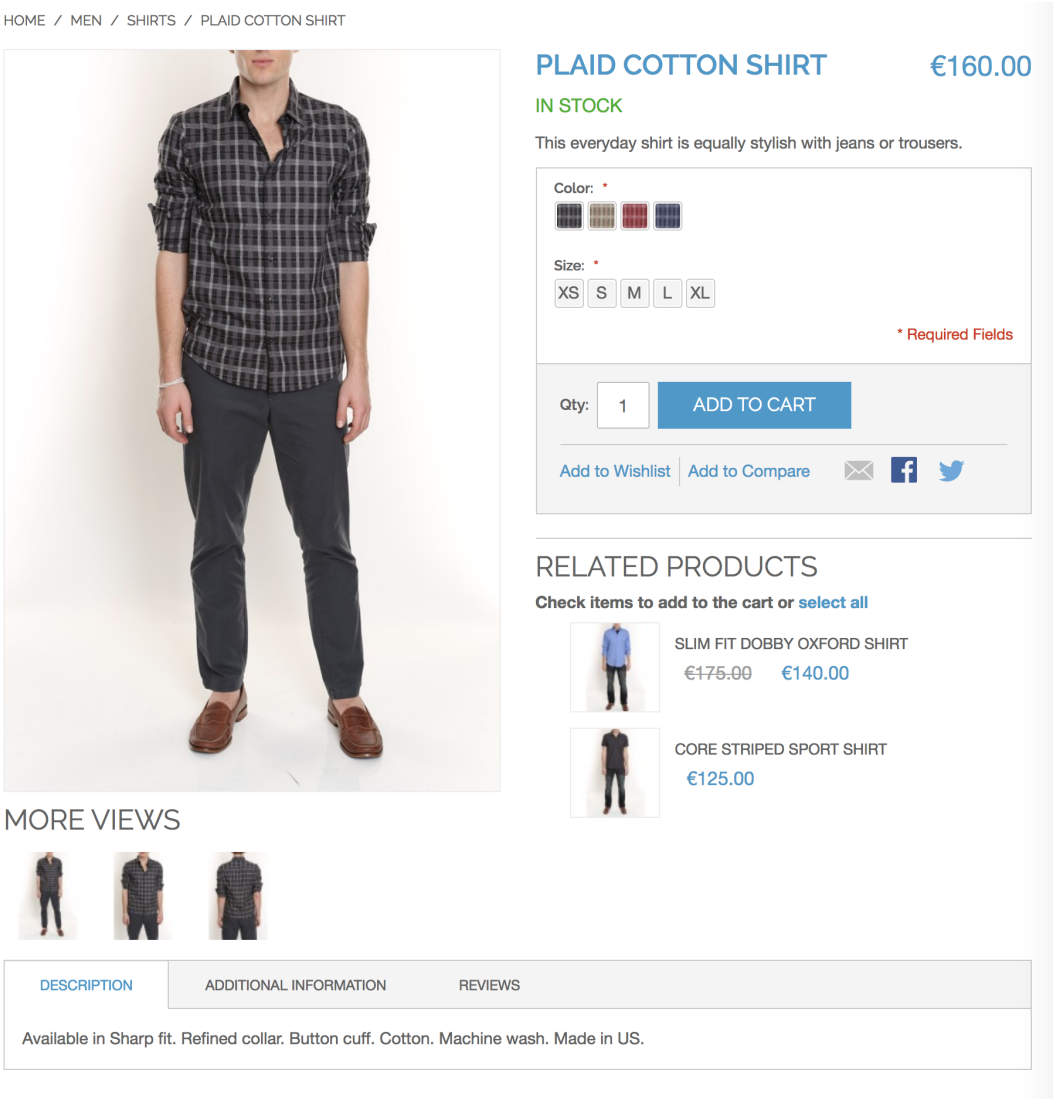


Figure 4.13: Product Page Desktop Version

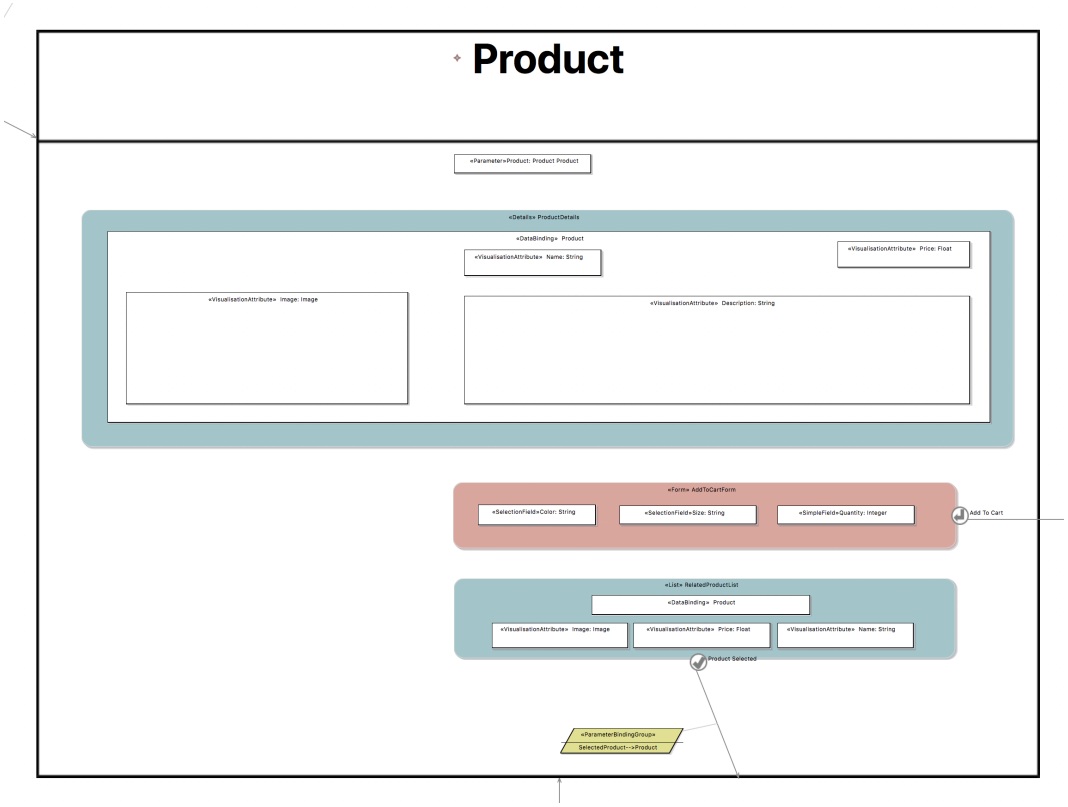


Figure 4.14: Product Page IFML Diagram

The structure of the model just outlined produces the following *IFMLModel* code:

```
1  <interactionFlowModelElements xsi:type="ext:IFMLWindow" name="Product" inInteractionFlows="//
       @interactionFlowModel/@interactionFlowModelElements.1/@viewElements.2/
       @viewElementEvents.0/@outInteractionFlows.0 //@interactionFlowModel/
       @interactionFlowModelElements.0/@viewElements.2/@viewElementEvents.0/
       @outInteractionFlows.0 //@interactionFlowModel/@interactionFlowModelElements.10/
       @viewElements.0/@viewElementEvents.0/@outInteractionFlows.0 //@interactionFlowModel/
       @interactionFlowModelElements.6/@viewElements.1/@viewElements.0/@viewElementEvents.0/
       @outInteractionFlows.0">
2    <parameters name="Product" />
3    <viewElements xsi:type="ext:Details" name="ProductDetails">
4      <viewComponentParts xsi:type="core:DataBinding" name="Product" uniformResourceIdentifier=""
         >
5        <subViewComponentParts xsi:type="core:VisualizationAttribute" name="Price" featureConcept=
           "//@domainModel/@domainElements.9"/>
6        <subViewComponentParts xsi:type="core:VisualizationAttribute" name="Image" featureConcept
           ="//@domainModel/@domainElements.7"/>
7        <subViewComponentParts xsi:type="core:VisualizationAttribute" name="Name" featureConcept
           ="//@domainModel/@domainElements.8"/>
```

```
8           <subViewComponentParts xsi:type="core:VisualizationAttribute" name="Description"
                featureConcept="//@domainModel/@domainElements.10"/>
9         </viewComponentParts>
10      </viewElements>
11      <viewElements xsi:type="ext:Form" name="AddToCartForm">
12        <viewElementEvents xsi:type="ext:OnSubmitEvent" name="Add To Cart" viewElement="//
              @interactionFlowModel/@interactionFlowModelElements.1/@viewElements.1">
13          <outInteractionFlows xsi:type="core:NavigationFlow" targetInteractionFlowElement="//
                @interactionFlowModel/@interactionFlowModelElements.9">
14            <parameterBindingGroup >
15              <parameterBindings sourceParameter="//@interactionFlowModel/
                    @interactionFlowModelElements.1/@viewElements.1/@viewComponentParts.2"
                    targetParameter="//@interactionFlowModel/@interactionFlowModelElements.1/
                    @viewElements.1/@viewComponentParts.2"/>
16            </parameterBindingGroup>
17          </outInteractionFlows>
18        </viewElementEvents>
19        <viewComponentParts xsi:type="ext:SelectionField" name="Color" />
20        <viewComponentParts xsi:type="ext:SelectionField" name="Size" />
21        <viewComponentParts xsi:type="ext:SimpleField" name="Quantity" />
22      </viewElements>
23      <viewElements xsi:type="ext:List" name="RelatedProductList">
24        <viewElementEvents xsi:type="ext:OnSelectEvent" name="Product Selected" viewElement="//
              @interactionFlowModel/@interactionFlowModelElements.1/@viewElements.2">
25          <outInteractionFlows xsi:type="core:NavigationFlow" targetInteractionFlowElement="//
                @interactionFlowModel/@interactionFlowModelElements.1">
26            <parameterBindingGroup >
27              <parameterBindings sourceParameter="//@interactionFlowModel/
                    @interactionFlowModelElements.0/@viewElements.2/@parameters.0" targetParameter
                    ="//@interactionFlowModel/@interactionFlowModelElements.1/@parameters.0"/>
28            </parameterBindingGroup>
29          </outInteractionFlows>
30        </viewElementEvents>
31        <viewComponentParts xsi:type="core:DataBinding" name="Product"/>
32        <viewComponentParts xsi:type="core:VisualizationAttribute" name="Image" featureConcept="//
              @domainModel/@domainElements.7"/>
33        <viewComponentParts xsi:type="core:VisualizationAttribute" name="Name" featureConcept="//
              @domainModel/@domainElements.8"/>
34        <viewComponentParts xsi:type="core:VisualizationAttribute" name="Price" featureConcept="//
              @domainModel/@domainElements.9"/>
35      </viewElements>
36    </interactionFlowModelElements>
```

The model representation of this product page structure is shown in Figure 4.15

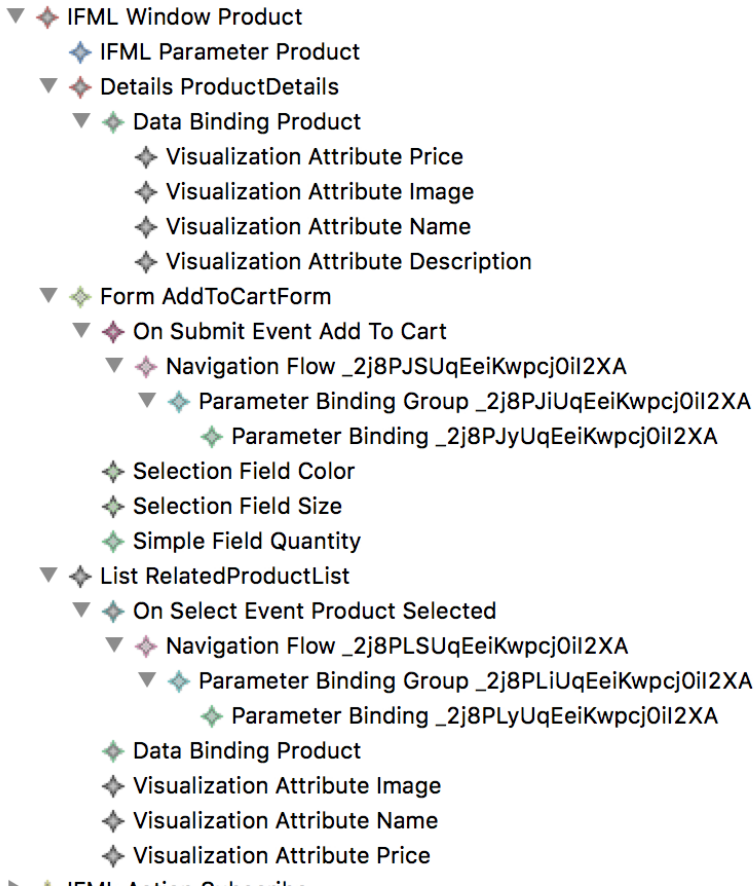


Figure 4.15: Interaction Flow Product Model in EMF

Shopping Cart Page

The Madison Island Interaction Model for the Shopping Cart page (Figure 4.16 and 4.17) is built with a single *IFMLWindow* container with a true-ish *isLandmark* property (flagging the window as accessible from everywhere) including multiple *Form View Component* instances representing the sidebar interactions with the discount codes and shipping estimation widgets. Besides the sidebar, the area holding the cart status and the items in the cart information is displayed with an additional *Form View Component* controlled by the *Activation Condition* responsible for showing items only when the cart is not empty. The section is modelled with a *Form View Component* because of the Qty *SimpleInput* text fields, which allow the user to update the related item quantities or empty the whole cart at any time. Both interactions are controlled with specific *IFMLAction* elements triggered on these *Events*.

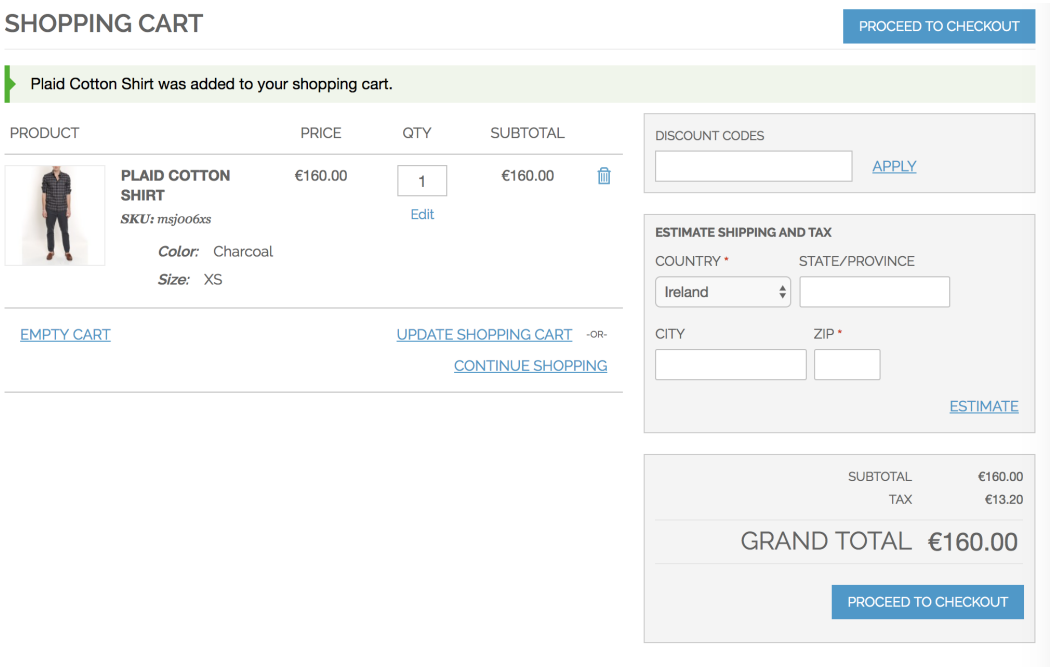


Figure 4.16: Shopping Cart Page Desktop Version

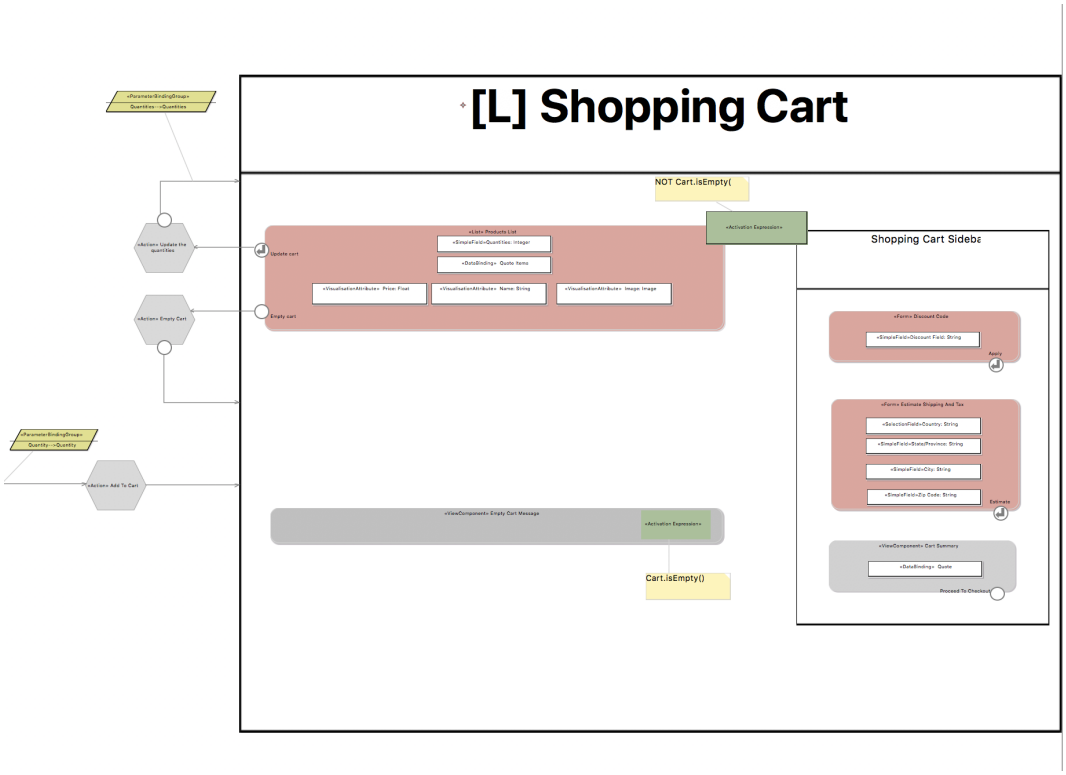


Figure 4.17: Shopping Cart Page IFML Diagram

As shown in Figure 4.18, the Interaction Flow model representing the shopping page has the following form:

```
1  <interactionFlowModelElements xsi:type="ext:IFMLWindow" name="Product" inInteractionFlows="//
       @interactionFlowModel/@interactionFlowModelElements.1/@viewElements.2/
       @viewElementEvents.0/@outInteractionFlows.0 //@interactionFlowModel/
       @interactionFlowModelElements.0/@viewElements.2/@viewElementEvents.0/
       @outInteractionFlows.0 //@interactionFlowModel/@interactionFlowModelElements.10/
       @viewElements.0/@viewElementEvents.0/@outInteractionFlows.0 //@interactionFlowModel/
       @interactionFlowModelElements.6/@viewElements.1/@viewElements.0/@viewElementEvents.0/
       @outInteractionFlows.0">
2    <parameters name="Product"/>
3    <viewElements xsi:type="ext:Details" name="ProductDetails">
4      <viewComponentParts xsi:type="core:DataBinding" name="Product" uniformResourceIdentifier=""
           >
5        <subViewComponentParts xsi:type="core:VisualizationAttribute" name="Price" featureConcept=
             "//@domainModel/@domainElements.9"/>
6        <subViewComponentParts xsi:type="core:VisualizationAttribute" name="Image" featureConcept
             ="//@domainModel/@domainElements.7"/>
7        <subViewComponentParts xsi:type="core:VisualizationAttribute" name="Name" featureConcept
             ="//@domainModel/@domainElements.8"/>
```

```
8           <subViewComponentParts xsi:type="core:VisualizationAttribute" name="Description"
                featureConcept="//@domainModel/@domainElements.10"/>
9         </viewComponentParts>
10      </viewElements>
11      <viewElements xsi:type="ext:Form" name="AddToCartForm">
12        <viewElementEvents xsi:type="ext:OnSubmitEvent" name="Add To Cart" viewElement="//
              @interactionFlowModel/@interactionFlowModelElements.1/@viewElements.1">
13          <outInteractionFlows xsi:type="core:NavigationFlow" targetInteractionFlowElement="//
                @interactionFlowModel/@interactionFlowModelElements.9">
14            <parameterBindingGroup >
15              <parameterBindings sourceParameter="//@interactionFlowModel/
                    @interactionFlowModelElements.1/@viewElements.1/@viewComponentParts.2"
                    targetParameter="//@interactionFlowModel/@interactionFlowModelElements.1/
                    @viewElements.1/@viewComponentParts.2"/>
16            </parameterBindingGroup>
17          </outInteractionFlows>
18        </viewElementEvents>
19        <viewComponentParts xsi:type="ext:SelectionField" name="Color" />
20        <viewComponentParts xsi:type="ext:SelectionField" name="Size" />
21        <viewComponentParts xsi:type="ext:SimpleField" name="Quantity" />
22      </viewElements>
23      <viewElements xsi:type="ext:List" name="RelatedProductList">
24        <viewElementEvents xsi:type="ext:OnSelectEvent" name="Product Selected" viewElement="//
              @interactionFlowModel/@interactionFlowModelElements.1/@viewElements.2">
25          <outInteractionFlows xsi:type="core:NavigationFlow" targetInteractionFlowElement="//
                @interactionFlowModel/@interactionFlowModelElements.1">
26            <parameterBindingGroup >
27              <parameterBindings sourceParameter="//@interactionFlowModel/
                    @interactionFlowModelElements.0/@viewElements.2/@parameters.0" targetParameter
                    ="//@interactionFlowModel/@interactionFlowModelElements.1/@parameters.0"/>
28            </parameterBindingGroup>
29          </outInteractionFlows>
30        </viewElementEvents>
31        <viewComponentParts xsi:type="core:DataBinding" name="Product"/>
32        <viewComponentParts xsi:type="core:VisualizationAttribute" name="Image" featureConcept="//
              @domainModel/@domainElements.7"/>
33        <viewComponentParts xsi:type="core:VisualizationAttribute" name="Name" featureConcept="//
              @domainModel/@domainElements.8"/>
34        <viewComponentParts xsi:type="core:VisualizationAttribute" name="Price" featureConcept="//
              @domainModel/@domainElements.9"/>
35      </viewElements>
36    </interactionFlowModelElements>
```

The *IFMLModel* representation of the Shopping Cart page structure is presented in Figure 4.18

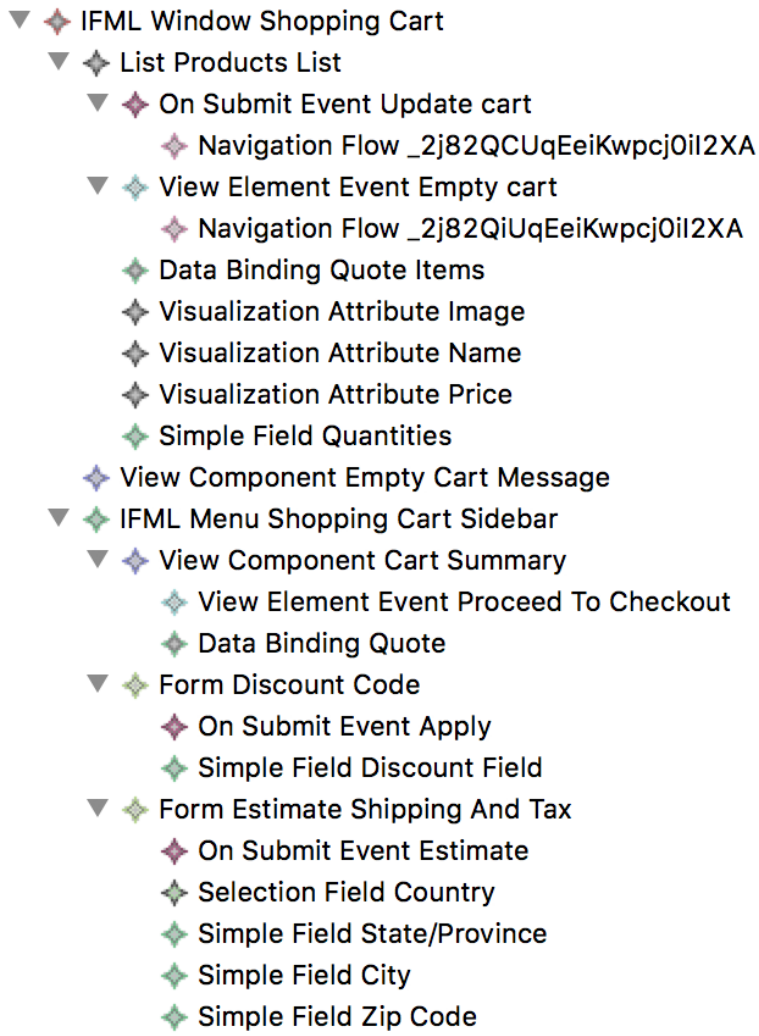


Figure 4.18: Interaction Flow Shopping Cart Model in EMF

Shared Elements and Interactions

As mentioned previously, shared sections of the platform such as the Header and the Footer have been modelled as *IFMLWindow* nodes reachable from any other area of the website, thus their *isLandmark* attribute set to true. In more detail, the Header section contains a Navigation Menu in the form of a *List View Component* with a children *DataBinding* component correlated to the Category *Domain Model* and a simple *Form View Component* for the search mechanism. When the user performs a search submitting a keyword, the *SearchKeyword* is triggered and the

Search Keyword *IFMLAction* is executed based on the parameters held in the associated *ParameterBinding* element. The resulting Search Results page, which presents a sidebar for filtering and a list of items matching the search, is another *IFMLWindow*, whose structure resembles the "Product List Mode" for a Category Page described earlier (Figure 4.19).

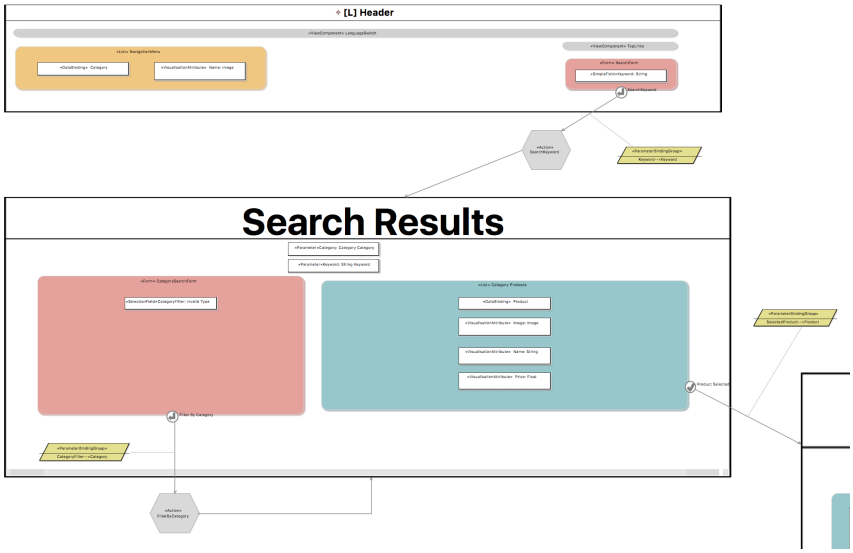


Figure 4.19: Header and Search sections IFML Diagrams

The Footer *IFMLWindow* model is quite straightforward. It retains the information about the footer links presented on the bottom of all the pages of the website in the form of a simple *ViewComponent* and a Newsletter subscription *Form View Component* reacting to SubscribeNewsletter submit *Events*. Similarly to the header case we have just described, the email passed as an argument from the *Event* is carried to the *IFMLAction* which, in this specific case, performs the actual Customer subscription and redirects them to the current page.(Figure 4.20).

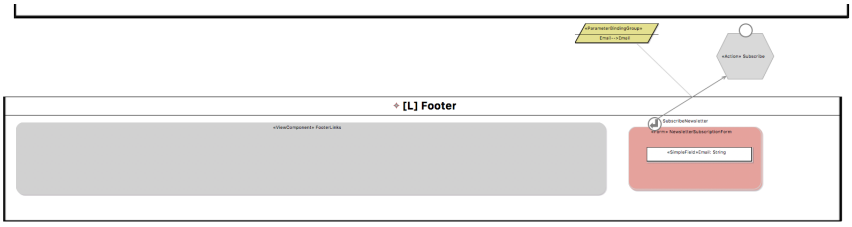


Figure 4.20: Footer section IFML Diagram

We conclude this chapter with both a snippet of the Header *IFMLWindow* element code coming from the *IFMLModel*, and an excerpt of the ecore model diagram in EMF for the areas of the website we described in this subsection (Figure 4.21).

```
1  <interactionFlowModelElements xsi:type="ext:IFMLWindow" id="_LvuL0BPREei3TrqMA9Bdvw" name="
       Header" isLandmark="true">
2  <viewElements xsi:type="ext:List" id="_v0p5YA9BEeiZ14TmPBeBNA" name="NavigationMenu">
3    <viewComponentParts xsi:type="core:DataBinding" id="__mcXIA9BEeiZ14TmPBeBNA" name="
         Category"/>
4    <viewComponentParts xsi:type="core:VisualizationAttribute" id="_Kbzt3xIZEeijSslFDgCgZA" name="
         Name" featureConcept="//@domainModel/@domainElements.5"/>
5  </viewElements>
6  <viewElements xsi:type="ext:Form" id="_YWvrMA9GEeiZ14TmPBeBNA" name="SearchForm">
7    <viewElementEvents xsi:type="ext:OnSubmitEvent" id="_ULm7WRIWEeijSslFDgCgZA" name="
         SearchKeyword" viewElement="//@interactionFlowModel/@interactionFlowModelElements.4/
         @viewElements.1">
8      <outInteractionFlows xsi:type="core:NavigationFlow" id="_Y6uV1xIWEeijSslFDgCgZA"
           targetInteractionFlowElement="//@interactionFlowModel/@interactionFlowModelElements.3">
9        <parameterBindingGroup id="_yCY84hPOEei3TrqMA9Bdvw">
10         <parameterBindings id="_y5vbohPOEei3TrqMA9Bdvw" sourceParameter="//
               @interactionFlowModel/@interactionFlowModelElements.4/@viewElements.1/
               @viewComponentParts.0" targetParameter="//@interactionFlowModel/
               @interactionFlowModelElements.3/@parameters.0"/>
11       </parameterBindingGroup>
12     </outInteractionFlows>
13   </viewElementEvents>
14   <viewComponentParts xsi:type="ext:SimpleField" id="_lWoCAA9GEeiZ14TmPBeBNA" name="
         Keyword" direction="out" />
15 </viewElements>
16 <viewElements xsi:type="core:ViewComponent" id="_aQFfjQ9TEeiZ14TmPBeBNA" name="TopLinks"/>
17 <viewElements xsi:type="core:ViewComponent" id="_ezXdfQ9TEeiZ14TmPBeBNA" name="
       LanguageSwitch"/>
18 </interactionFlowModelElements>
```

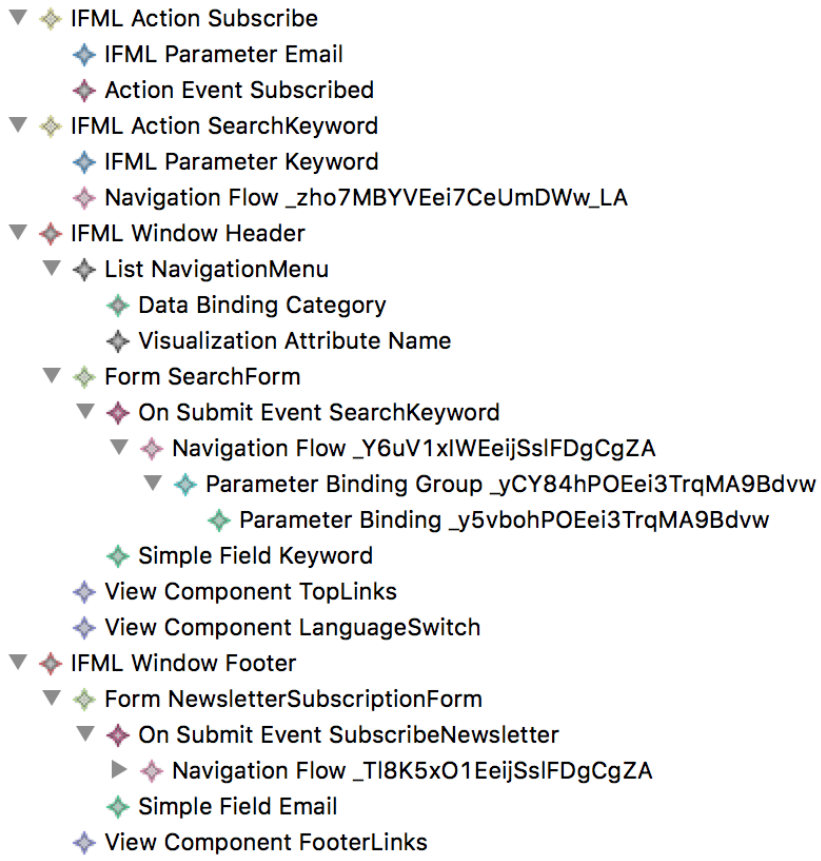


Figure 4.21: Interaction Flow for the shared elements in EMF

Chapter 5

Enhancing web models via model transformations

5.1 Transformation

In the previous chapter, after describing both metamodels for the Real Usage Data and the Interaction Flow Modeling Language, we proceeded with creating dynamic instances for both models, respectively describing the actual input data available for the customisation process and the initial IFMLModel of the eCommerce Madison Island website. In this chapter, we illustrate a possible transformation for this model, based on the Real Usage instance data. Our goal is to upgrade the model version, taking into account customers preferences and their actions.

5.1.1 Homepage

Considering the data coming from the Proximity sensors within the actual Madison Island retail store, we are aware of a set of regions corresponding to the customer preferred categories. That information will be used to change the content and, therefore, the IFMLModel for the Homepage carousel.

For instance, we know that the **Blazers**, **Tees, Knits and Polos** and **Shoes** categories correspond to the *sessionRegion* elements within the store in which the user spent more time after sorting the entries by *maxSecondsInRegion* and *maxProximity* attributes. In the case of the Blazers category for example, the data belonging to the *RealUsageData:ProximityData* parent node takes this form:

```
1 <sessionRegions
2 regionId="40"
3 regionLabel="blazers"
4 detectionCount="1"
```

```
  maxSecondsInRegion="195"
  maxProximity="immediate"
  firstDetectionTimeStamp="2018-02-21T18:11:01.000+0100"
  lastDetectionTimeStamp="2018-02-21T18:11:01.000+0100">
  <beaconData
    uuid="0686a88e-fed6-11e7-8be5-0ed5f89f718b"
    majorId="25911"
    minorId="27"/>
</sessionRegions>
```

By applying this knowledge, we can say that the *subViewComponentParts* and the *parameter* nodes of the first HightlightedCategoriesCarousel *IFMLWindow* elements from the initial *IFMLModel* can be modified from this:

```
<parameters name="Highlighted Category #1" direction="inout">
  <constraints language="SQL" body="Category.ID=18"/>
</parameters>

...

<subViewComponentParts xsi:type="core:ConditionalExpression" language="SQL" body="Category.ID=18
    " name="Eyewear"/>
```

to:

```
  <parameters name="Highlighted Category #1" direction="inout">
  <constraints language="SQL" body="Category.ID=40"/>
</parameters>

...

<subViewComponentParts xsi:type="core:ConditionalExpression" language="SQL" body="Category.ID=40
    " name="Blazers"/>
```

Besides updating the content by prioritising some categories, the *IFMLModel* can also be changed to use different IFML elements under certain conditions. For example, the *List View Component*, which is responsible for displaying the most recent products on the homepage, can be replaced by another *List View Component*, responsible for presenting the latest products with which customers interacted in the physical store. Information about these products is available in

the RealUsageData model under the *RealUsageData:ActionData* section. In our specific example, *RealUsageData:ActionData* contains the data about the product SKUs that the customer had scanned for collecting reward points.

```
<sessionActions userAgent="iPhone 6S">
  <scannedItems
  barcode="042100005264" name="Elizabeth Knit Top−Red−S" sku="wbk012c−Red−S"/>
  <scannedItems
  barcode="042100005931" name="Plaid Cotton Shirt−Khaki−L" sku="msj006c−Khaki−L"/>
  <scannedItems
  barcode="042100007717" name="Broad St Saddle Shoes" sku="shm00110"/>
</sessionActions>
```

The model transition in this case would occur not only replacing the IFML *View Component* with a different type, but also adding a *ConditionalExpression* that filters the items by their SKUs, as shown in Figure 5.1.

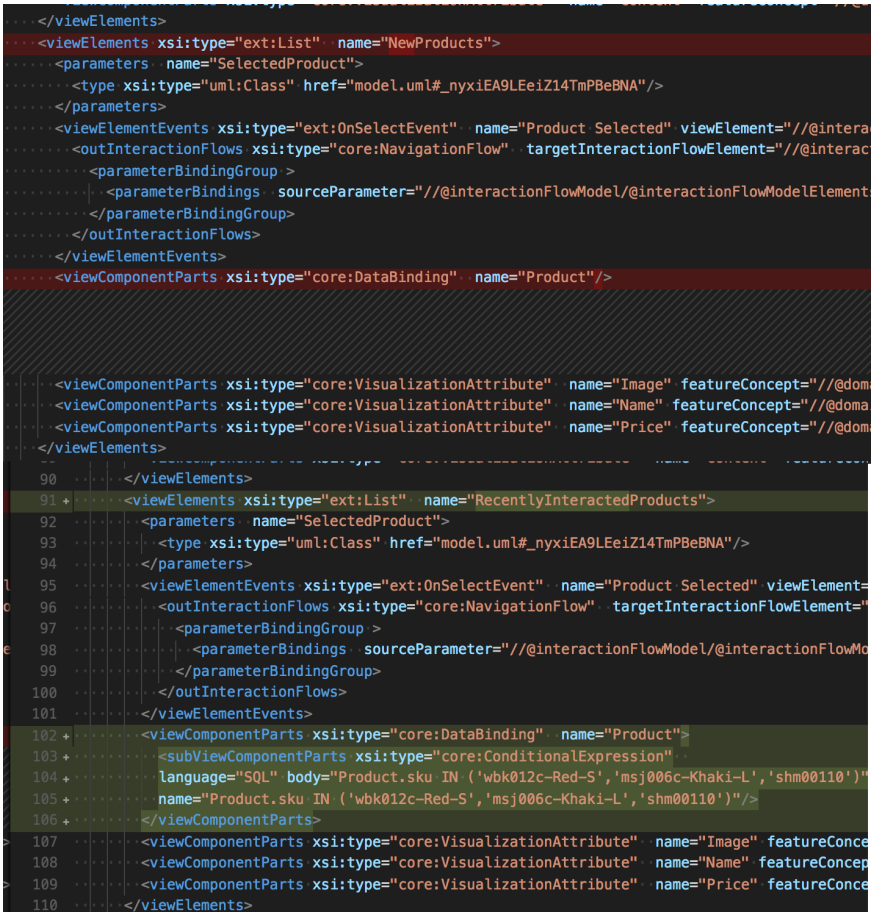


Figure 5.1: IFML Homepage transformation example

5.1.2 Header

Leveraging once more the *RealUsageData:ActionData*, the Header node of the *IFMLModel* describing the top area in the Madison Island website could also be enhanced to show a list of recent actions performed by customers in the physical stores that generated reward points. This can be achieved by appending an instance of *List View Component* to the *interactionFlowModelElements* parent node, called *RecentRewardActions*. This instance is then linked to the Rewards *Data Binding* entity, which allows the visualization of the *Action Name* and *Points Attribution* properties to the front-end, as shown in the following snippet of *IFMLModel* code:

```
1  <interactionFlowModelElements xsi:type="ext:IFMLWindow" name="Header" isLandmark="true">
2  <viewElements xsi:type="ext:List" name="NavigationMenu">
3   ...
4  </viewElements>
5  <viewElements xsi:type="ext:Form" name="SearchForm">
6   ...
7  </viewElements>
8  <viewElements xsi:type="core:ViewComponent" name="TopLinks"/>
9  <viewElements xsi:type="core:ViewComponent" name="LanguageSwitch"/>
10
11 <!-- ADDED NODE -->
12 <viewElements xsi:type="ext:List" name="RecentRewardActions">
13   <viewComponentParts xsi:type="core:DataBinding" name="Rewards" domainConcept="//
         @domainModel/@domainElements.13"/>
14   <viewComponentParts xsi:type="core:VisualizationAttribute" name="Action Name" featureConcept="//
         @domainModel/@domainElements.15"/>
15   <viewComponentParts xsi:type="core:VisualizationAttribute" name="Points Attribution" featureConcept
         ="//@domainModel/@domainElements.14"/>
16 </viewElements>
17 <!-- ADDED NODE -->
18
19 </interactionFlowModelElements>
```

5.1.3 Category Page

In light of the information gathered in the Real Usage Data model about the recently viewed products available in the *RealUsageData:WebData* node, the Category page can be enhanced by appending an additional visual element to the user interface for tracking that information.

To do so, we filter out the RealUsageData model, fetching the nodes containing a *parameterBindingGroup* property and including a Product reference like the following entries:

```
<data xsi:type="RealUsageData:WebData"
ID="3"
name="AccessLog"
userID="3045678"
date="2017-11-29T07:08:40.000+0100"
viewContainer="Category #15"
viewComponent="ProductList"
eventType="click"
parameterBindingGroup="Product/404"
logEntry="GET /men/shirts/plaid-cotton-shirt-476.html 200 0 - 29505"/>

<data xsi:type="RealUsageData:WebData"
ID="4"
name="AccessLog"
userID="3045678"
date="2017-12-04T06:37:15.000+0100"
viewContainer="Product #404"
viewComponent="RelatedProductList"
eventType="click"
parameterBindingGroup="Product/413"
logEntry="GET /core-striped-sport-shirt-551.html 200 0 - 29505"/>

<data xsi:type="RealUsageData:WebData"
ID="8"
name="AccessLog"
userID="3045678"
date="2017-12-04T06:38:20.000+0100"
viewContainer="Search Results"
viewComponent="ProductList"
eventType="click"
parameterBindingGroup="Product/407"
logEntry="GET /stretch-cotton-blazer-587.html 200 0 - 29505"/>
```

With the data described above, we can then proceed with transforming the initial *IFMLModel* for the Category page, appending to it a *RecentlyViewedProducts* section in the form of a *List View Component* bound to the Product *Data Binding* entity.

This new IFML item would be responsible for presenting customers with the thumbnails, names, and prices of these products, separating them through a distinct *ConditionalExpression*. That expression maps the children node of the *RealUsageData:WebData* dataset shown previously, before finally returning the items in the order they were browsed, and regardless of the

display mode property of the Category itself (Figure 5.2).

```
<viewElements xsi:type="ext:List" name="RecentlyViewedProducts">
  <viewElementEvents xsi:type="ext:OnSelectEvent" name="Product
  Selected"
  viewElement="//@interactionFlowModel/@interactionFlowModelElements.
  6/@viewElements.3">
    <outInteractionFlows xsi:type="core:NavigationFlow"
    targetInteractionFlowElement="//@interactionFlowModel/@interactionFlow
    ModelElements.1">
      <parameterBindingGroup >
        <parameterBindings
        sourceParameter="//@interactionFlowModel/@interactionFlowModelElem
        ents.0/@viewElements.2/@parameters.0"
        targetParameter="//@interactionFlowModel/@interactionFlowModelElem
        ents.1/@parameters.0"/>
      </parameterBindingGroup>
    </outInteractionFlows>
  </viewElementEvents>
  <viewComponentParts xsi:type="core:DataBinding" name="Product">
    <subViewComponentParts xsi:type="core:ConditionalExpression"
    language="SQL" body="Product.ID IN (407,413,404)" name="Product.ID IN
    (407,413,404)"/>
  </viewComponentParts>
  <viewComponentParts xsi:type="core:VisualizationAttribute"
  name="Image" featureConcept="//@domainModel/@domainElements.7"/>
  <viewComponentParts xsi:type="core:VisualizationAttribute" name="Name"
  featureConcept="//@domainModel/@domainElements.8"/>
  <viewComponentParts xsi:type="core:VisualizationAttribute"
  name="Price" featureConcept="//@domainModel/@domainElements.9"/>
</viewElements>
```

Figure 5.2: Recently Viewed IFML definition snippet

5.1.4 Product Page

As far as the possible improvements on the product page of the Madison Island website are concerned, we concentrated on the Related Products widget shown below the Add To Cart section. In the first *IFMLModel*, the list of products related to the currently shown item has no particular assigned logic. Our potential transformation is to extend the *viewElements* node of the *IFMLModel*, taking into account users browsing patterns that have been identified by examining the RealUsageData instances entries.

For example, we could have discovered that, during a same recorded sessions, the **Core Striped Sport Shirt** and the **Plaid Cotton Shirt** product pages are frequently browsed together. That could be done by analysing the values set in the *viewContainer*, *viewComponent*, and *parameterBindingGroup* properties for the nodes belonging to the *RealUsageData:WebData* dataset. Such change allows us to take advantage of this correlation, pushing specific products on the

Related Products section and consequently offering a more tailored user experience.

The change on the starting *IFMLModel* that would occur in that case restricts the *Data Binding* scope, appending a *subViewComponentParts* to it. That change limits the products collection to the computed set of related products SKUs. The additional *ConditionalExpression* would also be bound to a specific *ActivationExpression*, which would enable the above mentioned filtering only when a customised collection is available for the current product.

From a *IFMLModel* code perspective, we would transition from the following node configuration for the *List View Component* that controls the Related Product widget:

```
1  <viewElements xsi:type="ext:List" name="RelatedProductList">
2  <viewElementEvents xsi:type="ext:OnSelectEvent" name="Product Selected" viewElement="//
       @interactionFlowModel/@interactionFlowModelElements.1/@viewElements.2">
3    ...
4  </viewElementEvents>
5  <viewComponentParts xsi:type="core:DataBinding" name="Product"/>
6  <viewComponentParts xsi:type="core:VisualizationAttribute" name="Image" featureConcept="//
       @domainModel/@domainElements.7"/>
7  <viewComponentParts xsi:type="core:VisualizationAttribute" name="Name" featureConcept="//
       @domainModel/@domainElements.8"/>
8  <viewComponentParts xsi:type="core:VisualizationAttribute" name="Price" featureConcept="//
       @domainModel/@domainElements.9"/>
9  </viewElements>
```

to this one:

```
1  <viewElements xsi:type="ext:List" name="RelatedProductList">
2  <viewElementEvents xsi:type="ext:OnSelectEvent" name="Product Selected" viewElement="//
       @interactionFlowModel/@interactionFlowModelElements.1/@viewElements.2">
3    ...
4  </viewElementEvents>
5  <viewComponentParts xsi:type="core:DataBinding" name="Product">
6    <subViewComponentParts xsi:type="core:ConditionalExpression" language="SQL" body="Product.ID
         IN getCustomizedRelatedProducts()" name="Product.ID IN getCustomizedRelatedProducts"
         activationExpression="//@interactionFlowModel/@interactionFlowModelElements.17"/>
7  </viewComponentParts>
8  <viewComponentParts xsi:type="core:VisualizationAttribute" name="Image" featureConcept="//
       @domainModel/@domainElements.7"/>
9  <viewComponentParts xsi:type="core:VisualizationAttribute" name="Name" featureConcept="//
       @domainModel/@domainElements.8"/>
10 <viewComponentParts xsi:type="core:VisualizationAttribute" name="Price" featureConcept="//
```

```
        @domainModel/@domainElements.9"/>
</viewElements>
```

Moreover, the *ActivationExpression* defined by the *interactionFlowModelElements* XPath expression presented above links to a node responsible for controlling the feature with the following content:

```
<interactionFlowModelElements
  xsi:type="core:ActivationExpression" id="CurrentProduct.hasCustomizedRelatedProduct" language="
      SQL" body="CurrentProduct.hasCustomizedRelatedProduct()">
</interactionFlowModelElements>
```

5.1.5 Shopping Cart

As described in 4.2.2, the Shopping Cart page model includes two main sections. The first is a *List Product List* component, responsible for the interactions with the cart items. The second is the *IFMLWindow*, used for grouping the actions triggered by customers when they apply discounts or estimate shipping and taxes for their current quote.

Figure 5.3: Shopping Cart Sidebar IFML definition snippet

Along with the cases analysed previously and by applying the data from the Real Data Usage model, we can modify and extend the interaction flow of this page by adding a new *View Component*. That component shows the Reward points accumulated by the user in the Shopping Cart Sidebar (Figure 5.3). The component not only displays the information by the *Data Binding* with the Reward entity, but also controls the *Event* associated with the **"Apply Discount"** button and

the *Action* connected to it. The structure of this new node, which gets appended to its parent *IFMLMenu*, is as follows:

```
1  <viewElements xsi:type="core:ViewComponent" name="‘Applicable Reward Points">
2    <viewElementEvents xsi:type="ext:OnSelectEvent" name="‘Apply Reward Points" viewElement="//
         @interactionFlowModel/@interactionFlowModelElements.11/@viewElements.2/@viewElements.3
         ">
3      <outInteractionFlows xsi:type="core:NavigationFlow" targetInteractionFlowElement="//
           @interactionFlowModel/@interactionFlowModelElements.18/@actionEvents.0"/>
4    </viewElementEvents>
5    <viewComponentParts xsi:type="core:DataBinding" name="‘Reward"/>
6    <viewComponentParts xsi:type="core:VisualizationAttribute" name="‘Available Reward Points"
         featureConcept="//@domainModel/@domainElements.14"/>
7  </viewElements>
```

5.2 Updated web models

In this section, we present an overview of the updated IFMLModels for the Madison Island website based on the transformations discussed in the last chapter. We also describe how their IFMLDiagrams and corresponding website front-end screens would look like.

5.2.1 Homepage

According to what we described in 5.1.1, the enhanced homepage presented to the customer would show a custom banner carousel and a product widget, both containing items and categories with which the user interacted in the retail store.

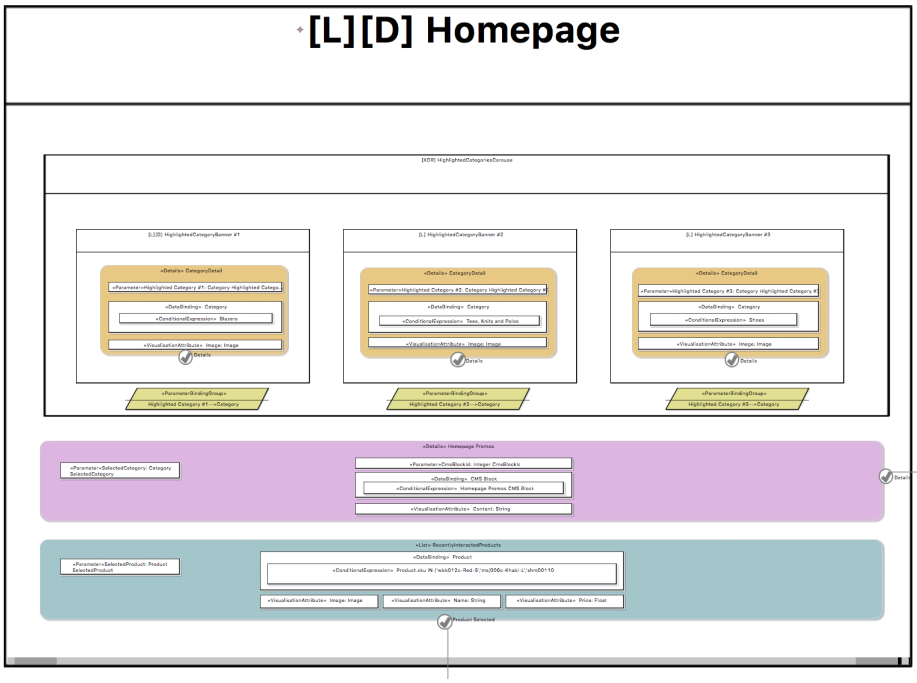


Figure 5.4: Updated Homepage IFML Diagram

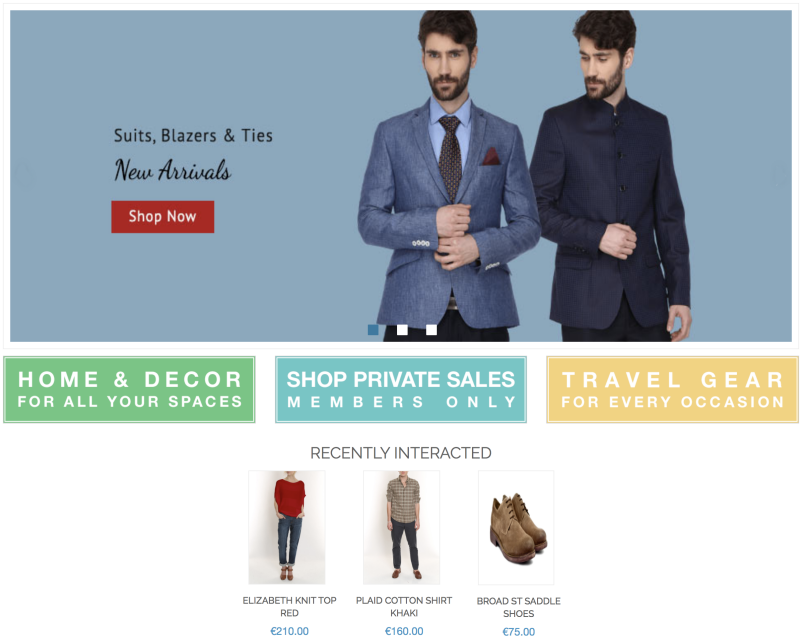


Figure 5.5: Updated Homepage Desktop

5.2.2 Header

After transforming its IFMLModel component (5.1.2), the shared header section on the website reveals new information through a new widget. This widget is aligned on the right side of the area used for advertising the actions taken by the user in the retail store and their respective rewarded points.

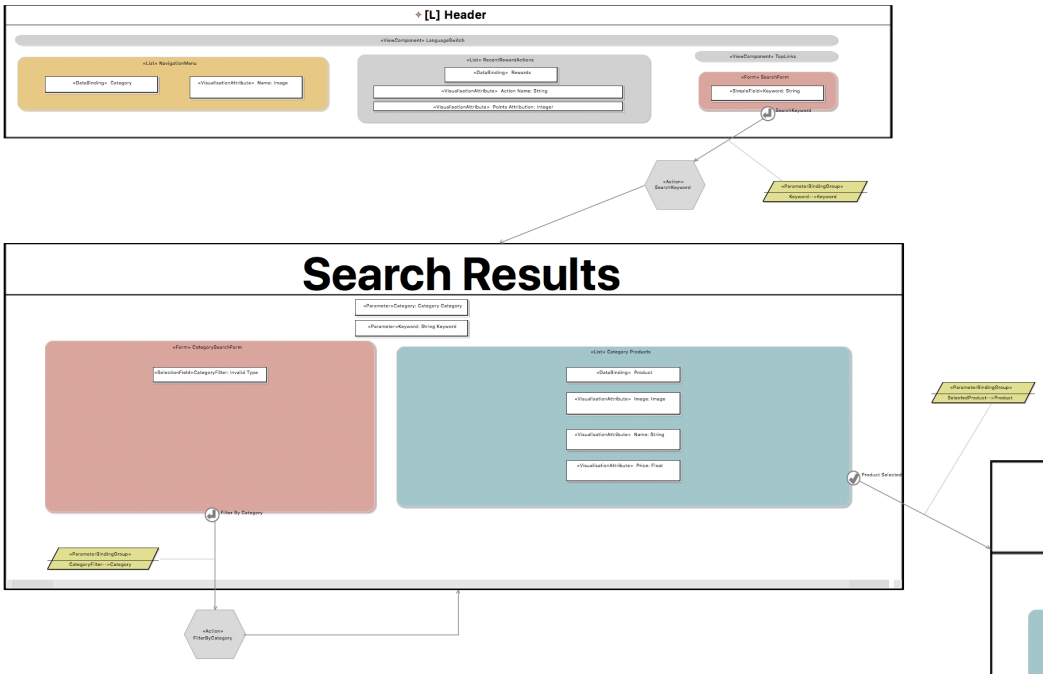


Figure 5.6: Updated Header IFML Diagram

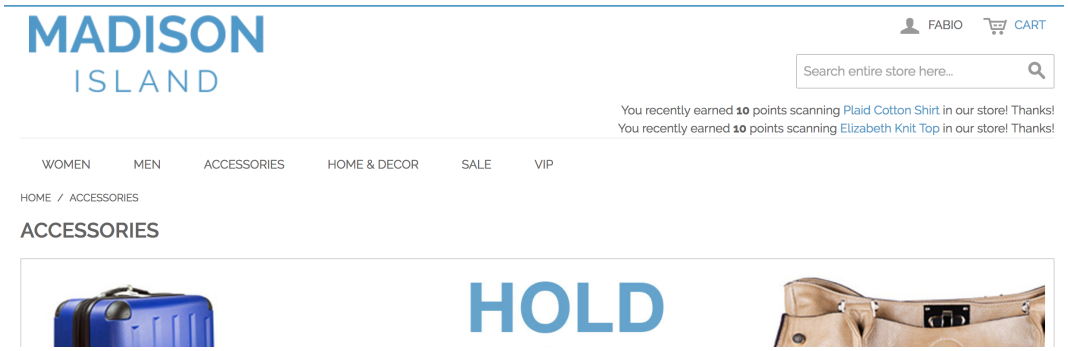


Figure 5.7: Updated Header Desktop Version

5.2.3 Category Page

Regardless of the DisplayMode property set for each Category Page on the eCommerce platform, the updated web model presents a vertical section on the right sidebar. That section maps the URLs browsed recently in a same user session that have been identified during the model transformation phase (5.1.3).

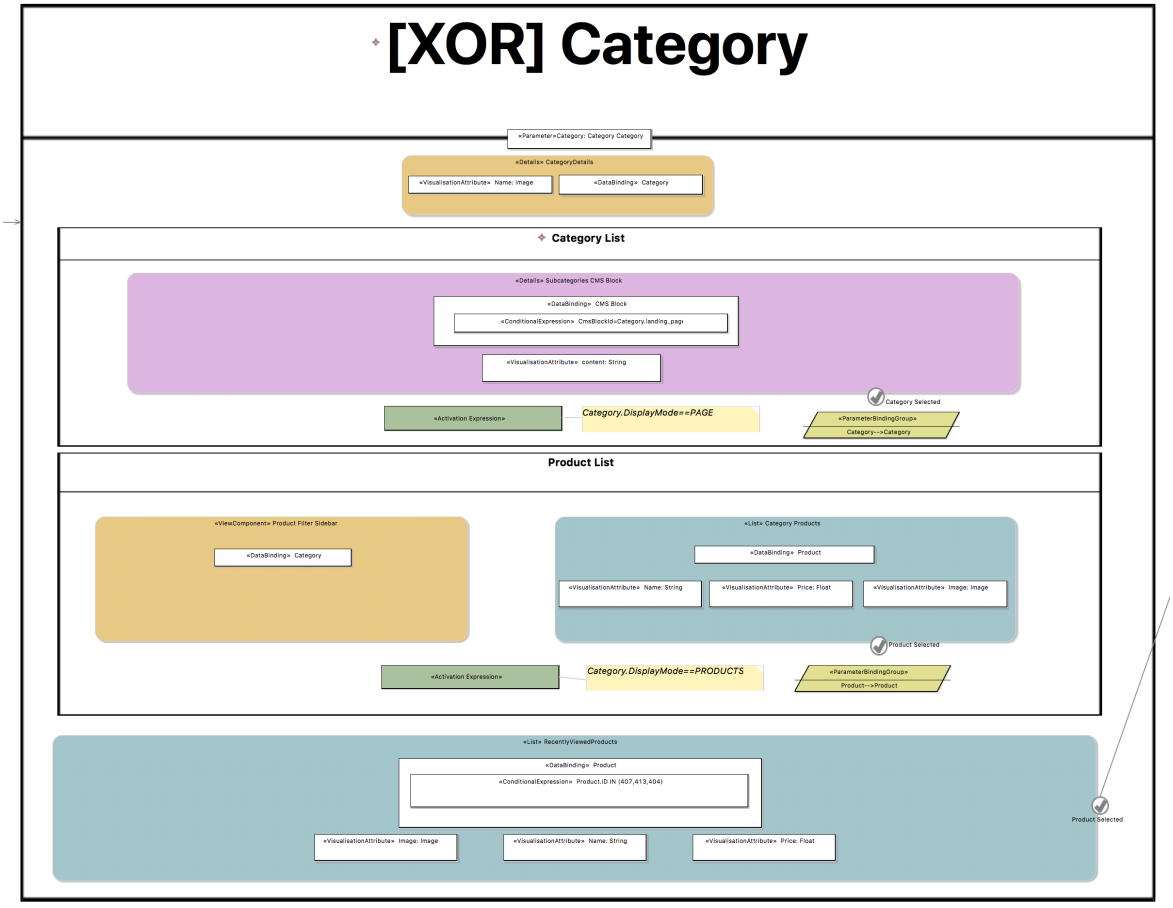


Figure 5.8: Updated Category Page IFML Diagram

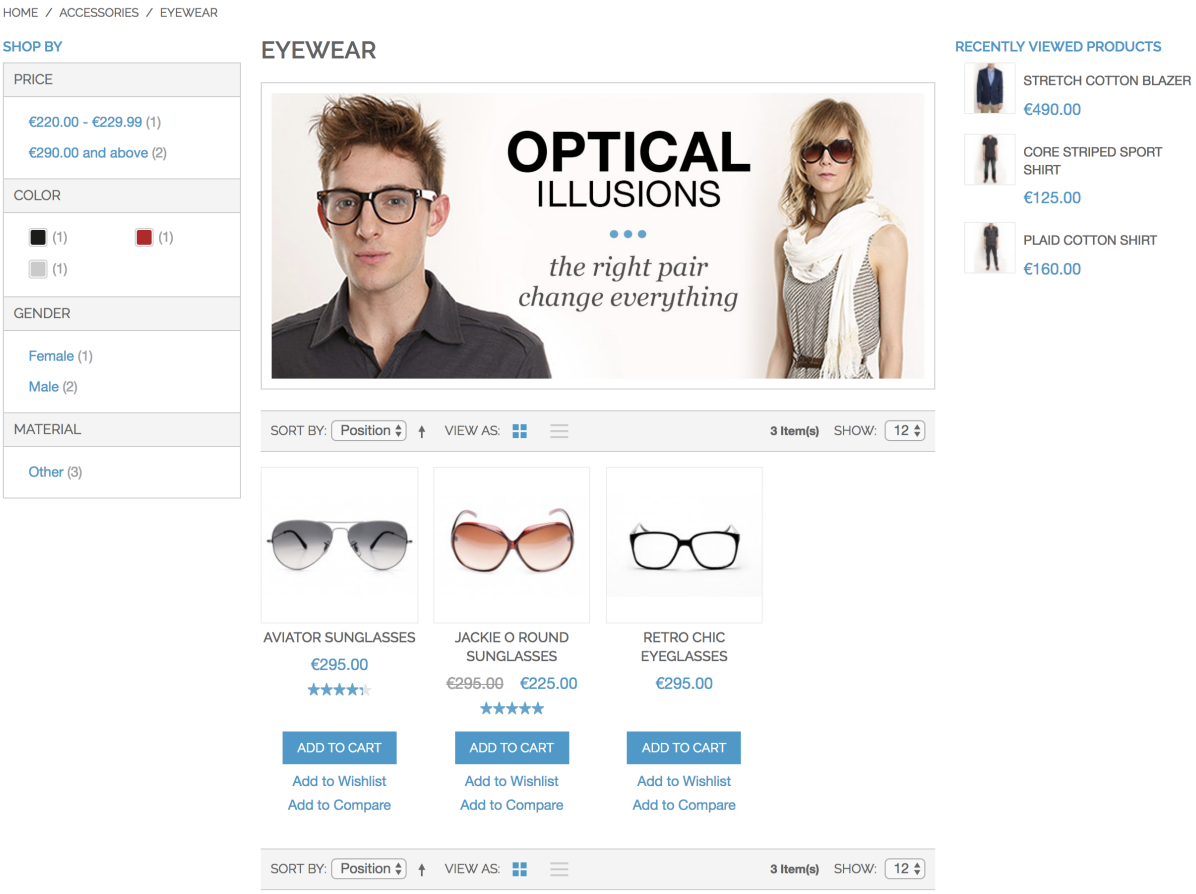


Figure 5.9: Updated Category Page Desktop Version

5.2.4 Product Page

Differently from the other page enhancements presented so far, the modified web model for the Product Page does not offer any additional visual segments to the page when compared to its starting IFMLModel. As discussed in 5.1.4, the change focused on improving the logic behind the related products selection by filtering the items on top of which a browsing correlation was detected. In Figure 5.11, we can see an example of this behaviour, where the **Plaid Cotton Shirt** product is placed as the first one in the Related Products region on the **Core Striped Sport Shirt** product page.

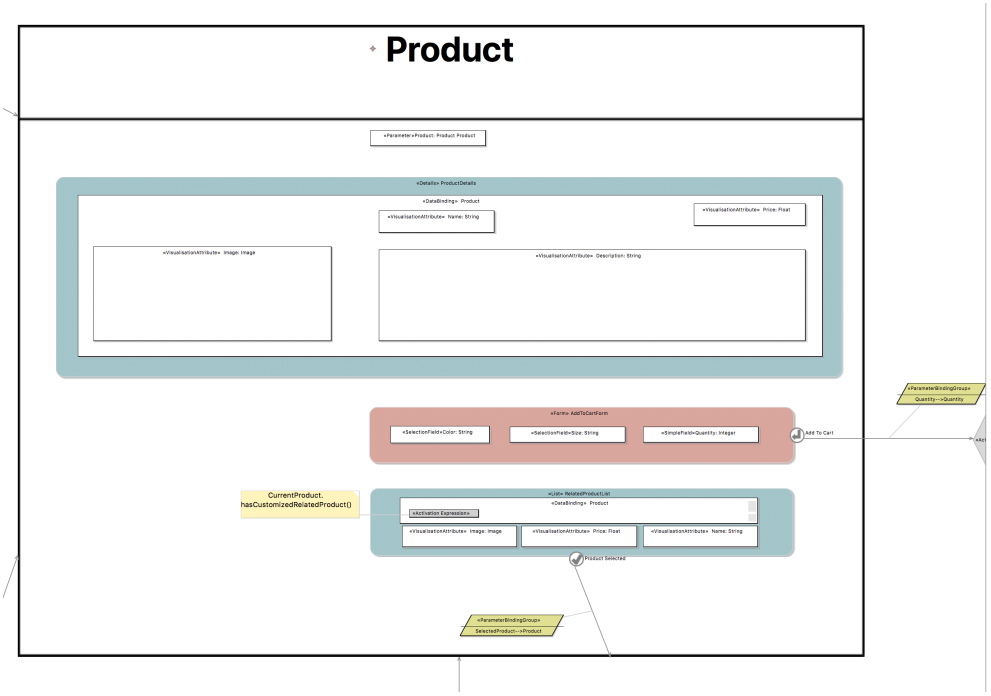


Figure 5.10: Updated Product Page IFML Diagram

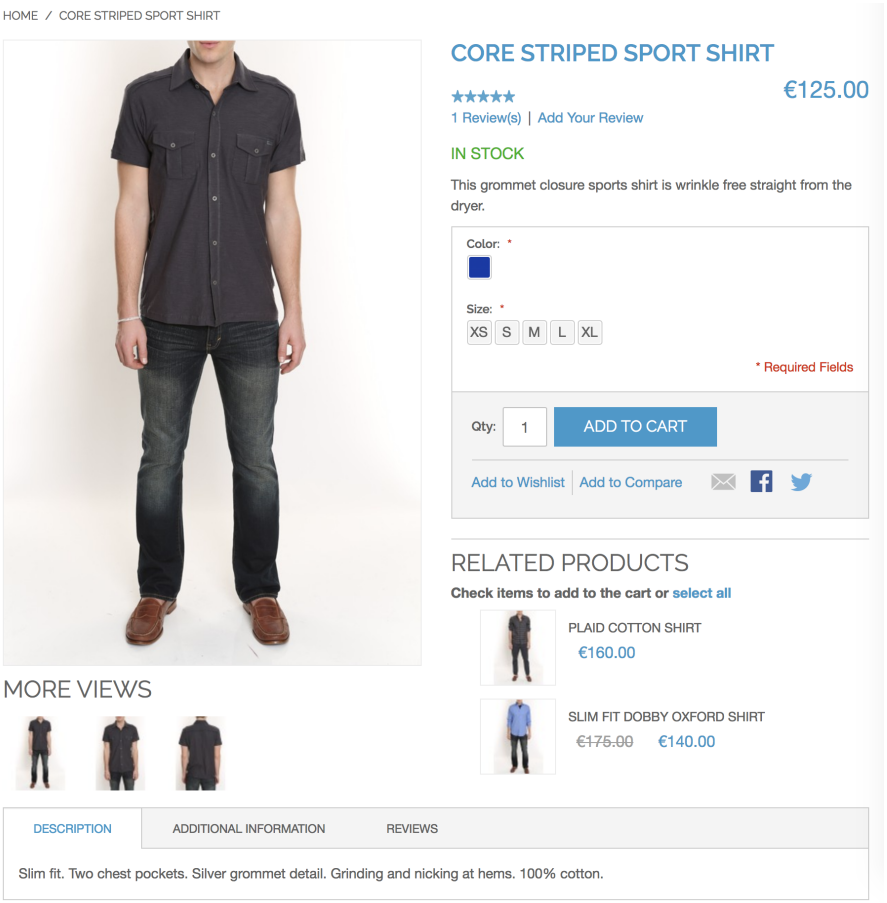


Figure 5.11: Updated Product Page Desktop Version

5.2.5 Shopping Cart

In 5.1.5, we outlined how the visual aspects of the shopping cart sidebar would be transformed through an *IFMLModel* change. This change would result in a new form element within the sidebar, which would be responsible for controlling the application of the reward points available to the customer at that point. Those reward points — generated through actions recorded as part of the real usage data profiling — would be converted into discounts that the customer could apply to their current quote at any time. As shown by the attached action in the IFML Diagram, the customer is redirected to the shopping cart page after the discount is applied.

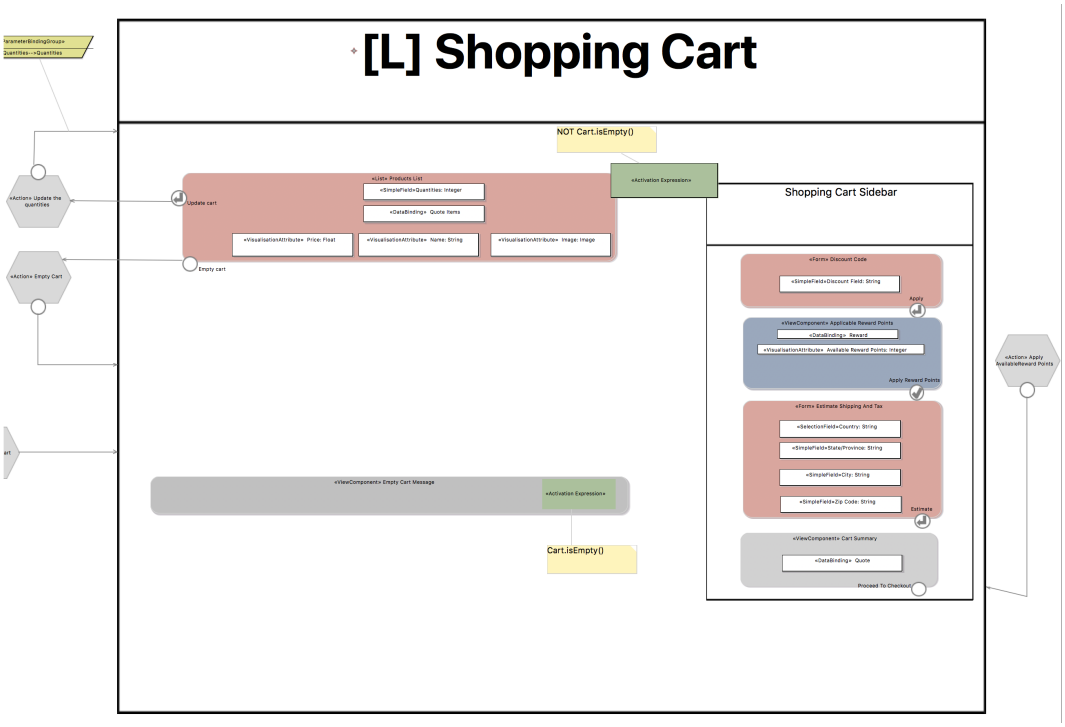


Figure 5.12: Updated Shopping Cart Page IFML Diagram

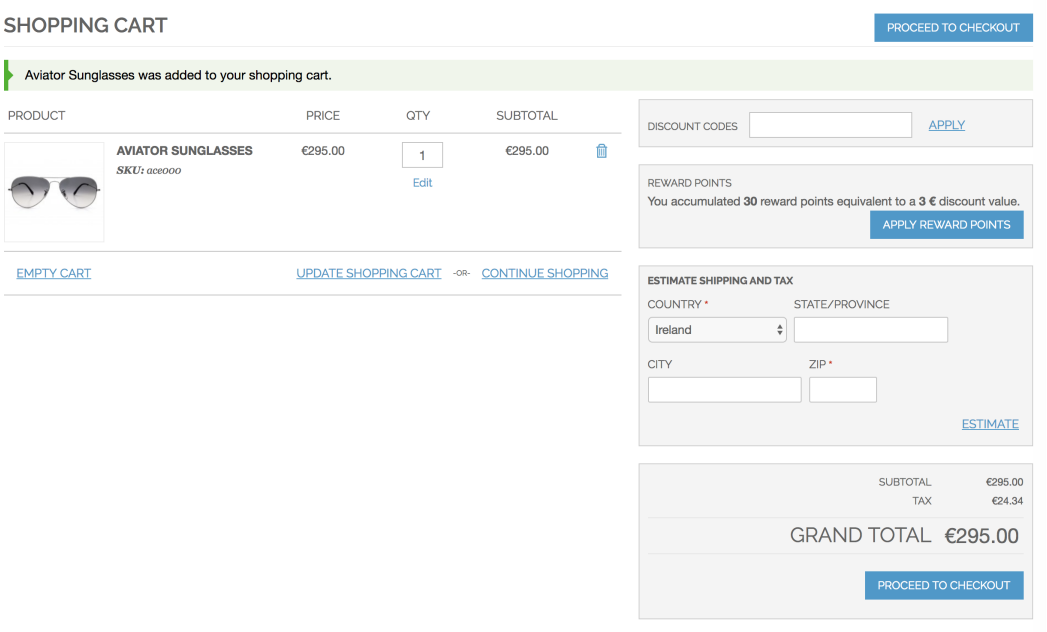


Figure 5.13: Updated Shopping Cart Page Desktop Version

Chapter 6

From models to code, a case study.

This chapter covers a case study based on the applications of the notions previously retrieved describing the Magento platform the Madison Island eCommerce website would run on and proposing a possible automation for code generation compatible with the platform ecosystem and architecture.

6.1 Magento overview

Magento is one of the most powerful and feature-rich online eCommerce platform that was launched on March 31, 2008.

The platform grants merchants complete flexibility and control over the look, content and functionality of their online store. Thanks to its intuitive administration interface for content management and robust marketing and merchandising tools it gives merchants the ability to create sites that are fully fit their unique business needs, putting no constraints on business processes and flow.

From a technical perspective, the platform incorporates the core architectural principles of an object-oriented, PHP-based application that can be easily used to personalize and expand the existing and already ample out of the box set of features. In fact, central to the Magento model of software development is the strategy of replacing or extending core code rather than editing it to support maintainability and flexibility.

To achieve this goal Magento software architecture has been built around the concept of self-contained modules holding discrete code organized by feature, thereby reducing each module's external dependencies.

The Magento frontend is designed to optimize storefront customization, with highly extensible themes being the central customization mechanism so that merchants can easily extend and transform the appearance of their storefronts.

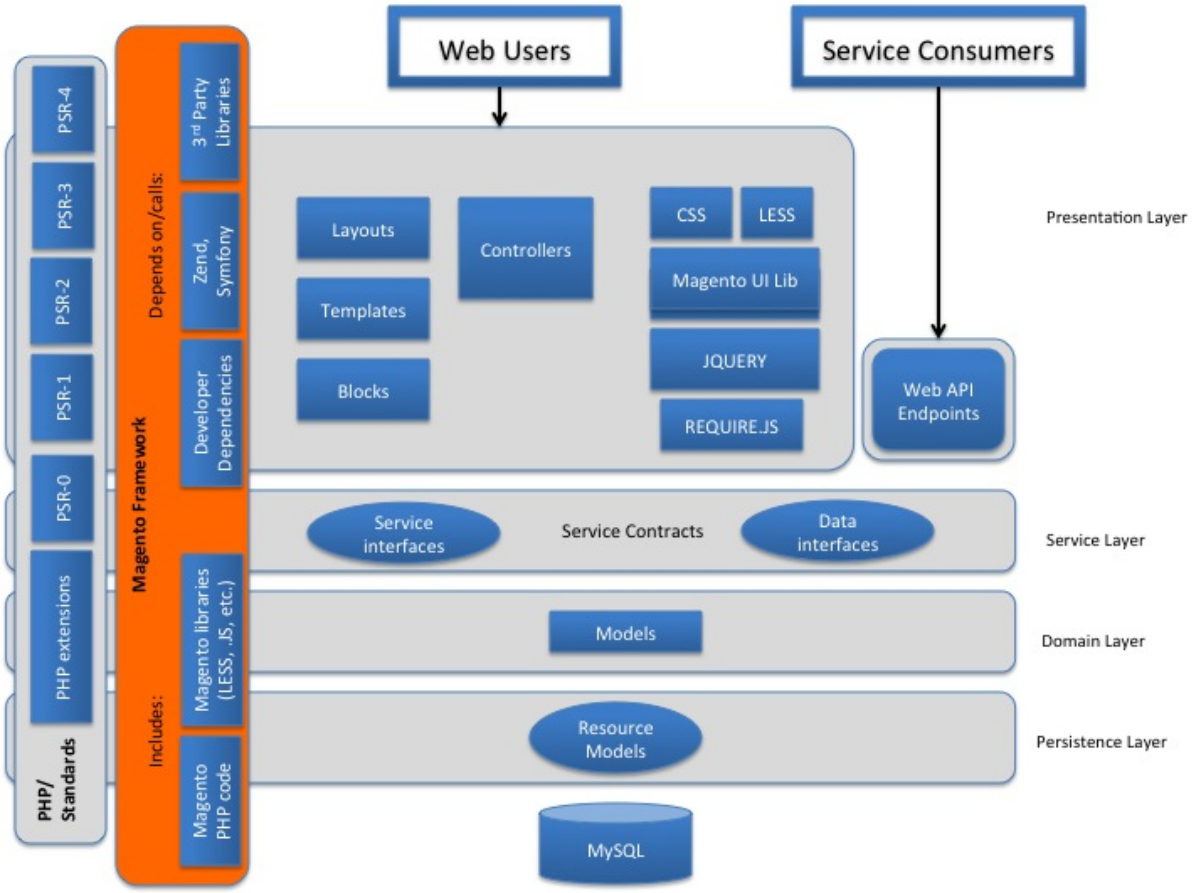


Figure 6.1: Magento Architecture Overview

Interacting with the product and its appearance on a Magento web interface means interacting with presentation layer code composed by both view elements (layouts, blocks, templates) and controllers, which process commands to and from the user interface.

We can extensively customize this user interface by extending Magento themes capabilities personalizing its content modifying this presentational layer accordingly to the requirements because the theme is responsible for organizing both the visual aspect of the interface and the product behavior.

Each Magento theme resides in a specific directory and includes custom page layouts, templates, skins, and language files that work together to create a distinct user experience.

In our case study example, the Madison Island frontend is built around the technology that we just described using a personalized and customized Magento theme for the user experience on the website.

6.1.1 Layout Updates XML

Magento offers great flexibility and re-usability of design by layouts defined in XML files. In fact, each Magento module may define its own layout XML file and participate in the process of building the page output. This result is accomplished through a set of instructions contained in these XML files and grouped in *layout handle*s. These directions play a fundamental role to determine the final look and feel of the page returned back to the user. For example, the handle *catalog_product_view* is used to add content to the product detail page, and *checkout_cart_index* to the shopping cart page. The following is an excerpt of the *catalog_category_default* and *catalog_product_gallery* layout handle instructions contained in the layout XML file liked to the module Magento uses for all the catalog related functionalities (Mage_Catalog):

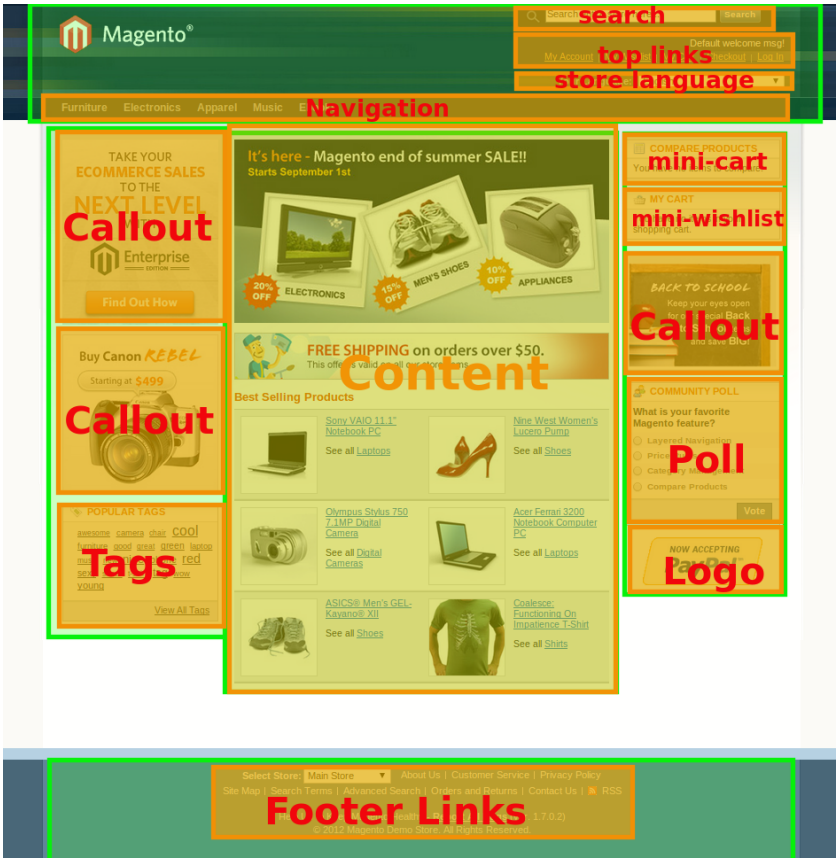
```
1
2  <catalog_category_default translate="label">
3  <label>Catalog Category (Non-Anchor)</label>
4  <reference name="left">
5      <block type="catalog/navigation" name="catalog.leftnav" after="currency" template="catalog/
           navigation/left.phtml"/>
6  </reference>
7  <reference name="content">
8      <block type="catalog/category_view" name="category.products" template="catalog/category/view.
           phtml">
9          <block type="catalog/product_list" name="product_list" template="catalog/product/list.phtml">
10             <block type="catalog/product_list_toolbar" name="product_list_toolbar" template="catalog/
                   product/list/toolbar.phtml">
11                 <block type="page/html_pager" name="product_list_toolbar_pager"/>
12                 <!-- The following code shows how to set your own pager increments -->
13                 <!--
14                     <action method="setDefaultListPerPage"><limit>4</limit></action>
15                     <action method="setDefaultGridPerPage"><limit>9</limit></action>
16                     <action method="addPagerLimit"><mode>list</mode><limit>2</limit></action>
17                     <action method="addPagerLimit"><mode>list</mode><limit>4</limit></action>
18                     <action method="addPagerLimit"><mode>list</mode><limit>6</limit></action>
19                     <action method="addPagerLimit"><mode>list</mode><limit>8</limit></action>
20                     <action method="addPagerLimit" translate="label"><mode>list</mode><limit>all
                           </limit><label>All</label></action>
21                 -->
22             </block>
23             <action method="addColumnCountLayoutDepend"><layout>empty</layout><count>6</
                   count></action>
24             <action method="addColumnCountLayoutDepend"><layout>one_column</layout><count
```

```
                    >5</count></action>
25              <action method="addColumnCountLayoutDepend"><layout>two_columns_left</layout><
                    count>4</count></action>
26              <action method="addColumnCountLayoutDepend"><layout>two_columns_right</layout><
                    count>4</count></action>
27              <action method="addColumnCountLayoutDepend"><layout>three_columns</layout><
                    count>3</count></action>
28              <action method="setToolbarBlockName"><name>product_list_toolbar</name></action>
29          </block>
30      </block>
31  </reference>
32 </catalog_category_default>
33
34 ...
35
36 <catalog_product_gallery translate="label">
37          <label>Catalog Product Image Gallery Popup</label>
38          <!-- Mage_Catalog -->
39          <reference name="root">
40              <action method="setTemplate"><template>page/popup.phtml</template></action>
41          </reference>
42          <reference name="content">
43              <block type="catalog/product_gallery" name="catalog_product_gallery" template="catalog/
                    product/gallery.phtml"/>
44          </reference>
45      </catalog_product_gallery>
```

As shown above, child nodes are created inside these *layout handle*s to determine which content should appear on any distinct page where these layout handles gets added to an XML structure called *page tree*.

These child nodes are called *block*s, which may, in turn, contain child *block*s of their own. Magento blocks can be structural or content blocks depending on whether they have a template associated with them or not. Usually, structural blocks, as shown in Figure 6.2, define the primary structure of the page and they do not explicitly render anything to the screen mostly acting as wrappers for their children blocks. Finally, each non-structural *block* would contain a template associated to it which would be responsible for displaying some specific content.

(a) Structural Blocks



(b) Content Blocks

Figure 6.2: Magento Blocks Page Structure

In summary, each page of the website can be thought as a composition of Magento *block*s representing a specific page tree configuration. As shown in 6.3 for the Madison Island theme example, each one of these blocks holds its associated rendering template that gets assembled and internally rendered before being returned to the user browser in the HTTP response.

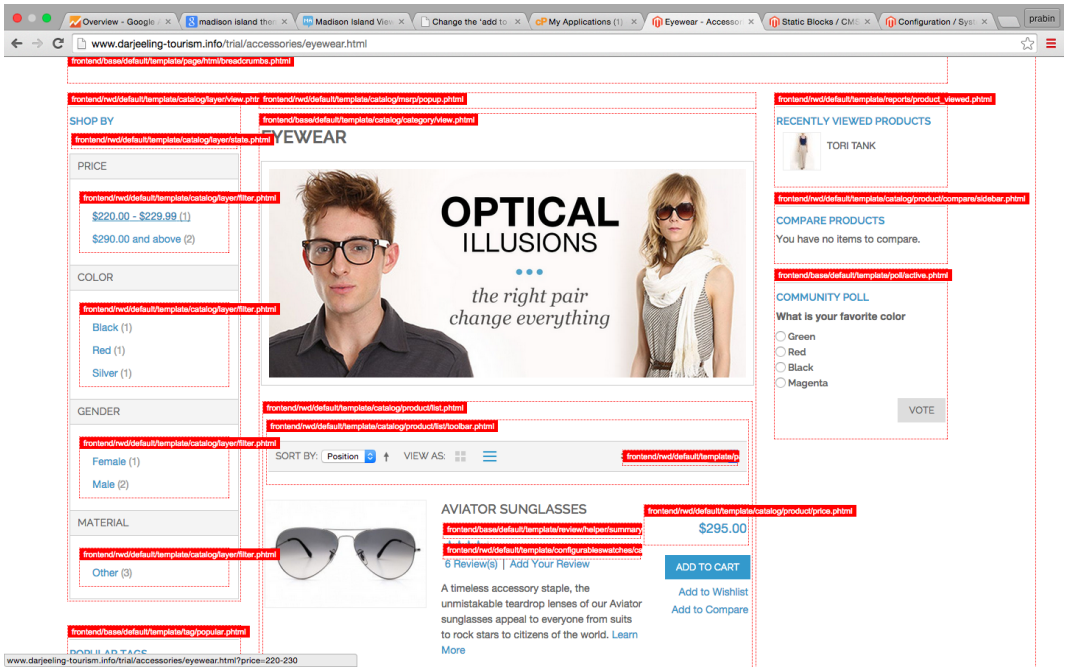


Figure 6.3: Madison Island templates composition

The versatility offered by the Magento rendering engine through the layout update XML mechanism allows us to dynamically append block children under certain conditions or pass parameters to the existing blocks that would affect their rendering behavior. This behavior is beneficial to map the changes we identified through the personalization process integrated within the updated *IFMLModel* describing the enhanced interface offered to the customers on the platform.

In fact, we are in a position now to generate and forward customized XML updates to Magento to reflect those requested changes and efficiently see them implemented on the website.

6.2 An approach for code generation

The main idea behind the code generation method is to use the updated *IFMLModel* processed in the previous stage and use it as the input document of an additional transformation capable of generating code understandable by the target platform, in this case, Magento. This generated code would have the form of the Layout Update XML instructions described in the previous section. Finally, these automatically generated XML update snippets will be interpreted and treated from the Magento rendering engine as any other layout update XML file.

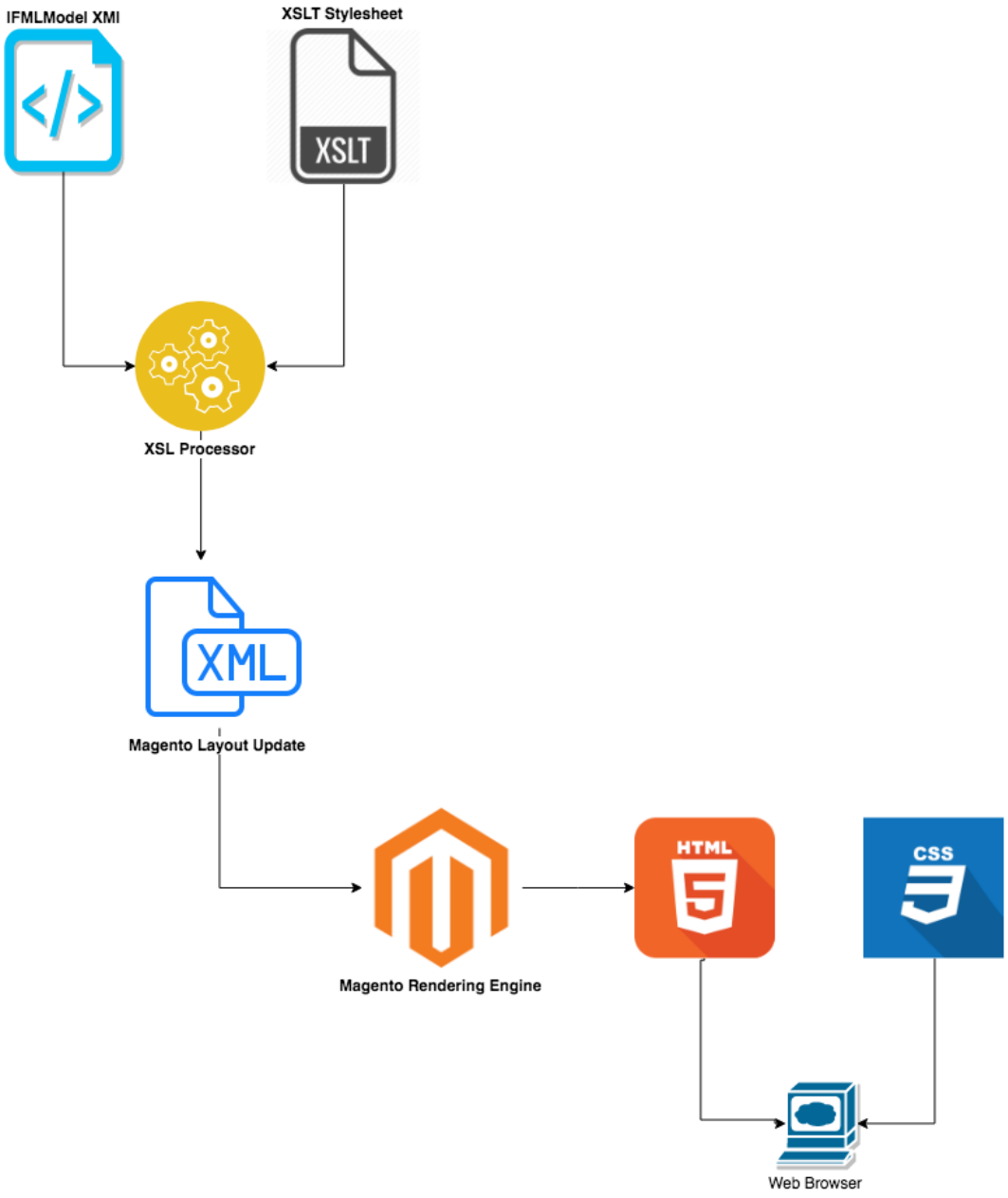


Figure 6.4: Code generation overview

6.2.1 XML Metadata Interchange overview

XMI (XML Metadata Interchange) is a proposed use of the Extensible Markup Language (XML) that is intended to provide a standard way for programmers and other users to exchange information about metadata (essentially, information about what a set of data consists of and how it is organized)[6].

Both the *RealDataUsage* and *IFML* models and metamodels definition presented in the previous chapters have been using and persisted throught the XMI specification.

```
<?xml version="1.0" encoding="ASCII"?>
<core:IFMLModel
    xmi:version="2.0"
    xmlns:xmi="http://www.omg.org/XMI"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:core="http://www.omg.org/spec/20130218/core"
    xmlns:ext="http://www.omg.org/spec/20130218/ext"
    xsi:schemaLocation="http://www.omg.org/spec/20130218/core ../metamodels/IFML.ecore#//Core http://www.omg.org/spec/20130218/ext ../metamodels/IFML.ecore#//Extensions"
    id="MadisonIslandAfter"
    name="MadisonIslandAfter">
```

Figure 6.5: XMI definition for the updated IFMLModel

Since XMI is XML-based, we decided to employ XSLT to transformat the XMI representation of the enhanced *IFMLModel* for code generation of the target documents.

6.2.2 Applying XSL Transformations

XSL Transformations (XSLT 2.0) is a language for transforming XML documents into other XML documents, text documents or HTML documents[5] whose detailed language specifications are beyond the scope of this work.

Still, in our journey towards code generation, we considered XSLT a suitable tool because we are effectively looking for a valid XMI-XML transformation. Specifically, our proposed approach limits itself to this XML documents conversion only, leaving the Magento rendering engine to implement the rest of the operations as usual. In fact, Magento would still be the authority for building up the final HTML response after merging all the layout updates collected and recursively rendering all the associated block templates.

The goal described above is achievable thanks to XSLT functionalities for adding nodes and attributes to or from the target file, rearranging and sorting elements and making decisions about which items to hide and display.In fact, XSLT is capable of efficiently transforming an XML source-tree (our *IFMLModel* XMI in our case) into an XML result-tree (Magento layout update XML in our case).

Because XSLT uses *XPath* to define parts of the source document that should match some predefined templates to be converted into some others (**??**), our objective is to identify those templates from the source XMI document and propose a possible transformation definition resulting in a valid layout update XML instruction Magento will take care of.

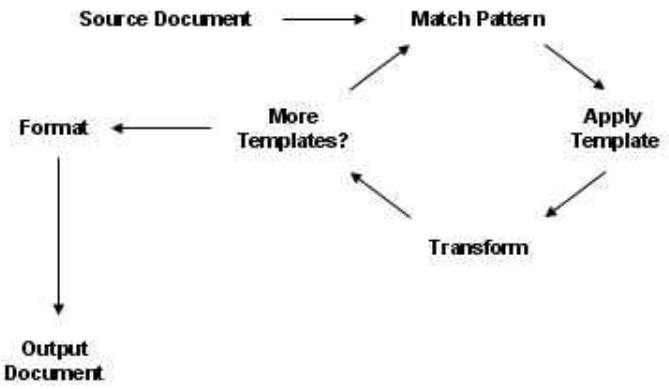


Figure 6.6: XSLT output processing

6.2.3 Practical examples

Conclusions

Loren ipsum

Bibliography

[1] Marco Brambilla, Eric Umuhoza and Roberto Acerbis - Model-driven development of user interfaces for IoT systems via domain-specific components and patterns

https://jisajournal.springeropen.com/articles/10.1186/s13174-017-0064-1

[2] "IFML Specification document". OMG - Object Management Group. Retrieved 9 April 2013.

[3] Region Monitoring and iBeacon

https://developer.apple.com/library/content/documentation/UserExperience/Conceptual/LocationAwarenessPG/RegionMonitoring/RegionMonitoring.html

[4] GitHub - Estimote/iOS-SDK: Estimote SDK for iOS devices https://github.com/Estimote/iOS-SDK

[5] Transformation - W3C https://www.w3.org/standards/xml/transformation

[6] What is XMI (XML Metadata Interchange)? http://searchmicroservices.techtarget.com/definition/XMI-XML-Metadata-Interchange