Advance Software Engineering

Technical Report on comparative Analysis of developing MVC Based Ecommerce DSL Models using Oboe and MPS

Final Project Report

Submitted to: ***Dr. Farooq Azam***

Submitted By:

Wajeeha Ajmal (363881)

Degree: ***MS CSE 21***

ABSTRACT

Our lives have been transformed by the Internet, which has changed the way we communicate with friends and conduct business.I created an e-commerce store on the oboe and MPS for the proposed study paper. All classes of the store were built after the node was established, followed by relation-based edges in oboe. The final palette was also created so that users could just drag and drop edges onto it. In MPS, concepts were generated and properties were set, followed by the MVC eCommerce store's child construction and editor. Sirus required a long time to implement and had several faults along the way, whereas MPS is simple to learn, understand, and use. Sirus took a long time to develop and contained numerous faults, but MPS is simple to learn, comprehend, and implement. Both platforms have a number of benefits that have already been mentioned. I've covered everything there is to know about MPS and eclipse designers. A comparison of used software, as well as breakthroughs and limitations, is shown in a competitive study. MPS can also be used to extend or include currently utilised languages.

1. INTRODUCTION

Electronic commerce is a business strategy that allows companies and individuals to buy and sell items or services over the Internet (e-commerce). Computers, tablets, smartphones, and other smart devices can all be used to conduct e-commerce in its different forms, which span four major market sectors. eCommerce transactions can be used to buy anything from books to music to plane tickets to financial services like stock trading and online banking. With the usage of e-commerce sites, people confront a variety of challenges. These include cyber security, online identity verification, price and delivery, competition and competitor analysis, enticing the ideal consumer, and so on. The biggest issue was determining which platform is best for setting up an e-commerce business. I wanted to look into the behaviour of e-commerce stores that cater to the oboe-designer community and MPS through this study. I created an e-commerce store on both platforms to investigate how e-commerce stores work and to determine which tool is best for implementation [1], [17], [18].

Ecommerce, as previously stated, is the practise of purchasing and selling tangible goods and services through the internet. A transaction, such as the exchange of data or cash, must be performed by more than one individual. It's part of the larger field of electronic business (e-business), which includes all of the procedures required to run a business online [2], [19].

By utilising e-commerce to provide more affordable and effective distribution channels for their products and services to a bigger audience, small businesses have been able to expand their market reach. Target (TGT) has an online store where customers can purchase everything from clothes to coffeemakers to toothpaste and action figures without ever leaving their homes [3], [20]. Sirius was used to construct all of the model instances, such as classes and ideas, which were then implemented on the oboe, designer community. The next steps were to create nodes, relation-based edges, and ultimately the Sirus graphical editor. It was also supposed to allow users to create new model elements graphically using a drag-and-drop palette. On the other side, MPS was used to create all of the concepts. Property and kid classes have now been added to the list of ideas. All of the composition classes are now available in the superclass editor.Finally, we created a tree view in which we added values to the attributes of concepts [4].

1.1 eCommerce is present in each of the four key market segments listed below:

The following are examples:

ü A business to user (B2C) transaction encompasses both businesses and their customers.

ü In a consumer-to-business (C2B) model, individuals can sell to businesses, including when an artist sells or licences their work to a business. Two examples are [5] and [6].

ü Consumer to business: A consumer-to-business (C2B) model lets individuals to sell to businesses, such as an artist selling or authority their work to a company. [5] and [6].

It is not as straightforward as it appears to provide goods and services. The products and services you wish to sell, as well as the market, your target audience, your competitors, and your expected company expenditures, all require significant research. Following that, you'll need to choose a name and form a legal body, such as a company.For your e-commerce site, set up a payment gateway. Dress stores, for example, can use online payment processing services like PayPal or credit card processing services like these to sell their apparel and other related things by setting up an online storefront [7], [21], [22].

1.2 E-commerce offers consumers the following advantages:

1.2.1 Convenience: You can shop anytime online.

1.2.2 Increased selection: There are a lot more things available online than there are in-store at many businesses. In addition, several online-only retailers may provide customers with access to goods that are not available in any other location. However, there are certain negatives to e-commerce sites as well.

1.3 There are a number of drawbacks, including:

1.3.1 Limited customer service:

You won't be able to ask a salesman to demonstrate you the features in person if you buy a computer online. While some websites allow you to speak with a member of the staff via live chat, this is not a typical occurrence.

1.3.2 Lack of immediate gratification:

Online purchases must be delivered to your house or place of business before they can be used. Some merchants, such as Amazon, provide same-day delivery as an upgrade, alleviating some of the frustrations of having to wait.

1.3.3 Impossibility of moving goods:

Because product photos don't often tell the whole storey about what you're getting, e-commerce purchases that don't meet customer expectations might be disappointing.

1.3.1 Limited customer service:
You won't be able to ask a salesman to demonstrate you the features in person if you buy a computer online. While some websites allow you to speak with a member of the staff via live chat, this is not a typical occurrence.

1.3.2 Lack of immediate gratification:

Online purchases must be delivered to your house or place of business before they can be used. Some merchants, such as Amazon, provide same-day delivery as an upgrade, alleviating some of the frustrations of having to wait.

1.3.3 Inability to handle goods:

Because product photographs do not often tell the whole storey about what you are getting, e-commerce purchases that do not fulfil customer expectations can be upsetting. A piece of clothing that appears to be made of a higher quality fabric than it is [8], [9], and [22] is an example of this.

1.4 E-commerce Case Study

The e-commerce market is dominated by Amazon. It is the world's largest online store, and it is just becoming bigger.Amazon dominates the e-commerce market. It is the largest online store in the world, and it is only going to get bigger. As a result, it is a major disruptor in the retail industry, prompting several major retailers to rethink their strategies and refocus. To get the business off the ground, the founders leveraged an e-commerce idea for online sales and product delivery Jeff Bezos grew it from an online bookshop to a wide range of products, including apparel, housewares, power tools, food and drink, and electronics, until closing it down in 2013. Revenues increased by 38% year over year to $386.1 billion, up from $280.5 billion the prior year. Amazon's operating income climbed as well, from $14.5 billion in 2019 to $22.9 billion in 2020. Net income increased from $11.6 billion to $21.3 billion at the end of 2020. 5 In addition to shopping, the company now offers cloud storage, movie and music streaming services, and a variety of electronic devices (such as Alexa, the personal assistant, and its Fire TV digital media player) [10], [11], and [23] are three examples.

2. BACKGROUND STUDY AND LITERATURE REVIEW

2.1 Introduction to domain-specific language(DSL)

A programming language is designed to solve a certain problem or address a specific topic.. Its goal is to make application development easier for developers by adding domain-specific terminology and concepts. A grammar expressing domain concepts is frequently used to represent DSL. The concrete syntax that corresponds to it can be written, graphical, or a combination of the two. For example, the user creates a graphical state machine diagram and fills up the text form with the actions for each state. The associated generator then generates classes, listeners, and other necessary components, which the user would otherwise have to build in a programming language. The DSL editor's programmer is usually not turned directly into binary code; instead, the generator develops a solution in a general programming language, which is then run via an appropriate compiler. As a result, there is no requirement for the DSL designer to develop a compiler. We chose Jet Brains MPS as our IDE for creating our DSL. A fully integrated development environment meets today's standards. It allows for the quick construction of all DSL components, including structure concepts, projections, behavior and interactions, the flow of data, and code generation. We chose Jet Brains MPS as our IDE for creating our DSL. A fully integrated development environment meets today's standards. It allows for the quick construction of all DSL components, including structure concepts, projections, behavior and interactions, data flow, and code generation.

Users can use projection editing to get around language parser constraints and create DSL editors with tables and diagrams.Tools like MPS are referred regarded as language workbenches. MPS allows you to use both tools and design languages. Tools and design languages are both possible with MPS. Similar to how IDEs help programmers, this tool is designed to make their lives easier. Code completion, context-aware code completion, and connection with the source code management system are just some of the features that make this a useful tool for developers of all levels of experience [24], [25].

Domain-specific languages (DSLs) are now widely used, despite the fact that domain-specific modelling is not a new concept. A programming language is known to solve a specific problem or address a certain subject. Its goal is to make creation of apps easier for programmers by adding domain-specific vocabulary and concepts. DSL is frequently depicted by a grammar that expresses domain notions. The concrete syntax that correlates to it can be written, graphical, or a combination of the two. Create a graphical state machine diagram and then fill in the actions for each stage in a text form, for example. The associated generator then creates classes, listeners, and other essential components that would otherwise have to be created in a programming language by the user.The programmer in the DSL editor is usually not converted immediately to binary code; instead, the generator creates a solution in a general programming language, which is then run via an appropriate compiler. As a result, the DSL designer is relieved of the need to create a compiler. We chose Jet Brains MPS as our IDE for designing our DSL. The requirements of today's standards necessitate a fully integrated development environment.We chose Jet Brains MPS as our IDE for designing our DSL. The requirements of today's standards necessitate a fully integrated development environment. It allows you to quickly create all DSL components, such as structure concepts, projections, behaviour, and interactions.

MPS is a programme that enables the creation of domain-specific languages (DSL). Users can work around parser constraints by creating DSL editors that include tables and diagrams, for

example, utilising projection editing. MPS is a workbench and an integrated development environment that creates new programming languages utilizing language-oriented programming (IDE). As its title suggests, MPS is a Metamodeling facilitation tool. Language workbenches, such as MPS, are referred to as such. With MPS, you may use both tools and design languages. This tool, like IDEs, is intended to make the life of programmers easier.Just a few of the features that make this a helpful tool for developers of all skill levels are code completion, context-aware code completion, and interaction with the source code management system [24], [25].

For a long time, raising the level of abstraction and reusability in software development has been a popular issue [1]. Model-Driven Engineering (MDE) [2] enabling technologies may be a means to deal with platform complexity and third-generation languages' inability to alleviate this complexity and accurately define domain concepts. When designing MDE software, the most important thing to remember is that the system is a model that follows its meta-model [four, 5]. Modeling languages, which are separated into two types, constitute the foundation of the MDE technique. GPLs (General-Purpose Modeling Languages) are multifunctional programming languages that can be used for a wide range of tasks. They can be used in a range of situations and domains.DSLs (Domain-Specific Languages) are programming languages that are intended to do a single purpose. They're most commonly utilised when a complete description of a domain is necessary. When DSL is employed, the problem domain and solution are greatly simplified. As a result, the number of people who can utilise it grows, as does productivity. DSL programmers are also easier to employ for analysis, verification, optimization, parallelization, and transformation because they are smaller than GPL programmers.DSLs have the disadvantages of being more difficult to design and deploy, as well as having greater initial costs. In this situation, MDE comes into play: DSLs and MDE are an excellent match. The MDE approach makes the task easier.

The MDE technique simplifies the process of defining a DSL. Domain-Specific Modeling is the term used in the MDE community to describe the process of creating a DSL (DSM). In contrast to DSL, DSM uses models to explain the domain system's many components. Graphic design tools are widely used to aid models that are based on a graphical representation. Users can use the DSM tool to construct domain models and, in most situations, generate particular code from them. It takes a long time to create a graphical tool for DSM. In this case, MDE comes into play: DSLs and MDE are a fantastic match. Because it's nearly hard to design an editor from the ground up these days, it takes a long time and usually relies on reliable frameworks.

A Sirius allows users to create views that are relevant to their business domain, irrespective of what that domain is. Despite the fact that domain definition is outside its scope, Sirius offers a graphical modeller for constructing a DSM that expresses concepts and their relationships in the abstract. After you've generated DSM models, Sirius makes it straightforward to construct precise physical representations. Diagrams, tables, matrices (cross-tables), and hierarchies are among some of the ways graphs can be displayed (trees). Users may build, revise, and assess their concepts in a modelling environment thanks to dynamic representations. It can be logically organised into views that end-users can enable or disable in order to give a different, logically coherent representation of the same model. In short, a Sirius creates a domain-specific graphical

editor while reducing the product, decreasing design time, and greatly increasing productivity. The following are some of the key features of the Sirius platform: • EMF, an open and widely adopted industry standard;

• the ability to adapt to any EMF-compatible DSM;

• a powerful separation between syntactic and representation models;

• support for a variety of domain model representations;

• ease of use and rapid development; and a high level of extensibility.

The five fundamental concepts of a VSM seem to be: - perspective – a key element comprised of a logical set of representation specifications and representation extension specifications that characterises a model's structure, look, and behaviour; - representation – a set of graphical representations that describe domain data; - perspective – a key element comprised of a logical set of representation specifications and representation extension specifications that characterises a model's structure, look, and behaviour. The four representations available are diagrams, tables, matrices, and trees (dialects). - Mapping is dialect-dependent, and it determines which elements from the semantic model can be included in the representation, as well as how they should be represented. - style – is used to change the appearance of an item; - tool – is used to define behaviour mapping. The Viewpoint element is the most basic feature of the VSM model, as it contains all of the model's data. includes all of the information regarding the components of the construction parts , which are characterized by their quality and features in general Type names are given to some of the attributes that are frequently types from the provided semantic domain model.

A Viewpoint can also define Model File Extensions, which restrict the viewpoint's modelling capabilities to dataset includes object semantic models. Representation Descriptions for diagrams, flowcharts, tables, bridges, and trees can all be included in a Viewpoint element. It is possible to validate a single element or the whole model. VSM also supports *.design files. All of the aforementioned capabilities are present in these files. Multiple VSM elements, as well as multiple views, can be created. MPS and Java have a long and complementary connection. The language definitions are based on a Java dialect called Base Language, which was the first to be fully implemented in MPS. MPS is written in Java and runs on the JVM. MPS allows you to easily import Java source code and libraries and use them in MPS projects. MPS languages have Java libraries that can be used in Java IDEs. As a result of all of this, Java developers will feel more at ease with MPS. MPS contains a variety of Base Language extensions, such as collections, closures, time/date manipulation, regular expressions, and constructors, that greatly expand its capabilities beyond plain Java.

2.2 How does MPS works:

The Real-World Implementation of MPS Jet Brains developed You Track 5, a web-based bug-tracking application, as a real-world example of MPS. In less than three years, You Tracks became

a major component of Jet Brains' product range, validating both the LOP concept and its implementation in MPS. MPS provides more information about how You Track benefits from LOP. Several universities worked with the MPS team to help them implement MPS as a language design research tool. A student developed a real-time Java programming environment as part of his master's thesis at Charles University in Prague's Faculty of Mathematics and Physics in 2012. Despite the fact that MPS is most closely associated with Java and the JVM, it has a wide range of applications. its principles apply to many computer languages.The editor for the concepts is defined in the second phase. This illustrates MPS's projection nature. MPS languages are never parsed since code is never shown as plaintext (neither on the screen nor on the disc), hence no grammar is necessary. Editors can be modularized to allow the reusability of visual syntax elements, and a notion can be assigned to multiple editors, allowing the programmer to choose which one to use. The default MPS editor requires some getting accustomed to for programmers who are used to text editing. It's a very different experience to develop software using established language principles rather than individual characters. MPS, on the other hand, significantly improves text editors' editing experience. You may utilise node factories, side transformations, and replacement rules, among other things, to update many of the editor's dynamic properties.Subject-specific dialects and languages allow people to communicate more precisely and efficiently about their fields of knowledge. In terms of programming languages, MPS offers the same level of flexibility. Unlike traditional programming languages, MPS allows users to build, alter, and expand a language from the bottom up. Parsers, which analyse strings to generate data structures, pose a hurdle when it comes to extending languages. Parsers limit the code-persistent representation to a single notation, making it impossible to combine them with other parsers and obstructing language modularization. The most common solution to this challenge is to implement a non-textual display of computer code. This approach does not require parsing, which is a huge advantage. MPS, on the other hand, favours a different strategy. As a result, we constructed an Abstract Syntax Tree (AST) data structure that stores all of the characteristics, children, and references needed to understand the code. The MPS editor's job is to make the AST understandable and give users the tools they need to change it successfully. For example, a conventional text editor experience should be mimicked in the editor for a textual language that mimics regular textual languages. When building a language using graphical notations, the editor should provide a comparable experience as a diagramming editor. Create a language in MPS and determine the editing rules as well as the appearance of the editing user interface. You can also create a set of rules and constraints for the type system in your language. They make MPS possible. They enable MPS to evaluate programmer code in real time, making the creation of new languages easier and less prone to errors. MPS is based on a generative model. Using generators for your language, end-user input can be turned into a more conventional, generally general-purpose vernacular.However, MPS is currently best recognised for its ability to generate Java code.

You can also build languages like C#, XML, PDF, Latex, JavaScript, and others. [13], [26] are two examples.

2.3 Why did jet beans invent MPS?

Jet Beans planned to construct MPs in order to maximize the effectiveness of their programme.. MPs can effectively collaborate and communicate with software engineers and experts. The development of a Domain Specific Language (DSL) is a remarkable milestone in software development in and of itself [14]. Jet Brains is a software firm specializing in integrated development environments (IDEs) (IDE). Jet Brains MPS makes it easy to design language constructs (Meta Programming System). MPS is a programming language that can be used to both create & utilize DSLs that have already been created.Jet Brains MPS began as a research project in 2003. Since 2006, Jet Brains used this IDE to develop their products. MPS is an Apache 2.0-licensed free and open-source project.Other DSL development environments aren't like MPS. It is not a grammar-based tool like Flex1 or Bison2, where establishing rules for specific tokens and lexemes is required when constructing a DSL.These tools allow language-oriented programming in a persistent abstract form using a projection editor. The purpose of MPS'score is to make a base language or a DSL written in the language better. Concepts form the foundation of the language. The notion is a linguistic element in the MPS environment that explains how elements emerge, behave, and are formed, among other things. Models are used to define the numerous properties of a notion. Each model is in charge of a particular feature of the DSL statement. In The default base language in MPS covers the full Java language and transforms its statements, expressions, assignments, and other features into portions that may be used by the user and the IDE.

2.4 What is the purpose of MPS?

MPS departs from other IDEs since its code is strictly tree-like, opposing Eclipse or Microsoft Visual Studio. A tree-like structure can be examined and updated even in the editor. Although it has a similar appearance to other IDEs, the user will notice significant changes while editing. Let's pretend the user wishes to use a basic assignment as the body of an if-statement. The MPS is divided into two parts:

- A language editor that allows the user to customise the structure, behaviour, generator, and other aspects of a DSL;

- A solution editor that allows anyone to design their own real-time systems using the given DSLs. The output of the editor is referred to as a solution.
  • You can create your own domain-specific language with MPS (DSL). Keep in mind that a DSL is designed to handle a specific set of problems when building a programming language. A DSL is a programming language that utilizes concepts and rules from a specific subject or domain. The complexity of a domain-specific language is generally smaller than that of a general-purpose language like Java or Ruby.
  Non-programmers with authority over the domain that the DSL targets are more likely to use DSL than software developers. To connect DSL to programming language code, there are two methods.

  The first step is to segregate the DSL and programming language code, then use an automated code generator or software to load and run the code. DSL and non-DSL

programmers are combined in a single programmer file in a second way. The DSL makes use of the GPL grammar and parser, as well as the host language's extension options. In the vast majority of cases, IDEs were unfamiliar with the DSL and thus did not support MPS. We can use the MPS framework with its specialised DSLs because it is used for language development. Jet Brains' MPS (Metaprogramming System) is a workbench for languages such As java and C++. MPS is a program that allows the production of domain-specific languages. By bypassing language parser restrictions through projectional editing, it enables the design of DSL editors, such as those with tables and diagrams [15], [16], [28-30].

Lexers and parsers are used in traditional compiler technology to read programmes written in text files [Lam+06] and convert them into data structures called Abstract Syntax Trees (AST). The top panel of Figure 1 depicts this process.

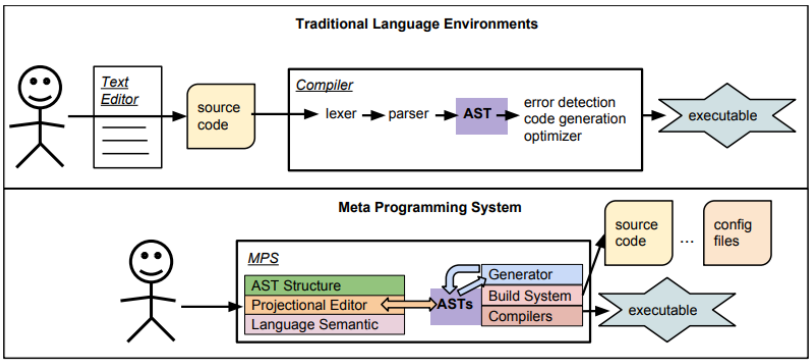


Figure 1 Abstract Syntax Trees (AST)

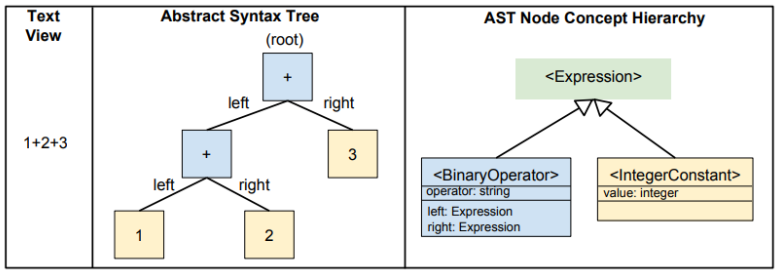


Figure 2

The MPS programming paradigm is compared to a more traditional programming method in this diagram. The top panel depicts how programmers write code in a typical language environment that includes a text editor and a compiler. The bottom panel depicts a programmer's direct

interaction with the MPS system. The data structures that a compiler uses to build executable code are known as ASTs. In the Meta Programming System's paradigm, on the other hand, the user interacts directly with one or more ASTs using a projection editor. The relationships between the text representation of an arithmetic statement and the matching AST are shown in Figure 2. Without the requirement to express programmes as text, the MPS technique works directly with the AST. This has a number of advantages:

1. Language extensions are straightforward to create. Extending a lexer and parser for a complex language necessitates specialised knowledge, whereas extending a language in MPS entails specifying additional AST concepts, along with associated editors and semantics. Extending a language entails specifying its structure and developing an editor to accommodate the additional notions. This technique is significantly easier than extending a compiler for an existing language because developing a text syntax that eliminates ambiguities gets more difficult as the language grows larger.

2. Different languages can be efficiently concatenated without the risk of causing ambiguities in the concrete syntax.

The MPS Language Workbench:

Figure 3 shows a screenshot of the MPS Language Workbench. This illustration should assist you get a feel of how the MPS user interface operates. In the following section, we'll take a closer look at the Project Tab. The MPS Workbench's user interface is shown in this screenshot. When first getting started with MPS, it's a good idea to concentrate on the Editor and Project Tabs. To the right is the Editor, and to the left is the Project Tab. To see this view, you must first open an existing project or create a new one.

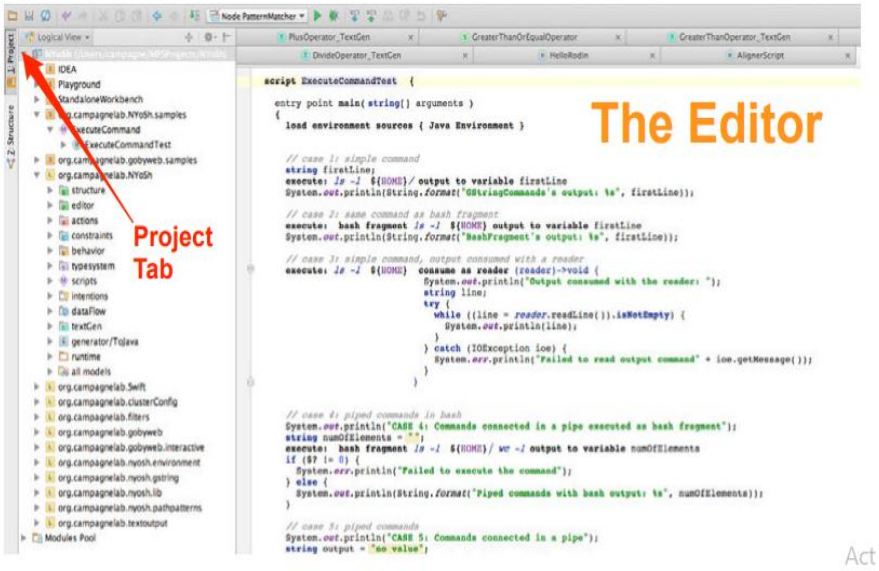


Figure 3

The Projects Section

The Project Tab is a critical element of MPS's user interface. This page offers a logical structure of the languages that have been included or imported in a project. When dealing with MPS, the Project Tab is explained in figure-4. Solutions and models Orange icons with a S inside them represent solutions. The name of the Solution comes after the orange square. Solutions, in return, are made up of ASTs, which contain models.

Language and Aspects of

Language Languages are represented by the yellow square symbols with a L in them. The language's name appears to the right of the emblem. A little triangle precedes the solution and language icons. To open the solution or language, click the triangle. To collapse the solution or language, click it once more. When you open a solution, the models in it are displayed (Execute Command is a model in the org.Campagne lab.NYoSh.sample solution). In contrast to solutions, opening a language exposes a collection of language features. Several elements will be discussed in further detail in the following chapters. Table 2.1 provides a brief overview of the functionality of these features.

Pool of Modules

The Modules Pool is shown in green at the bottom. The additional Languages and Solutions shown in the Modules Pool are not part of the project, but rather are given by the platform or a running MPS plugin. It's an excellent site to look for languages you might like to employ in your solutions

or languages. The Project Tab is frequently used to change the attributes of languages or solutions that make up an MPS project.

Figure 4

3. EXPERIMENTAL SETTINGS

3.1 What is Sirus?

This technology can be used to develop user-created graphical workbenches. Workbenches are created using Eclipse editors. For defining workbenches, Sirus supports editors like as diagrams and trees. Using this technology, existing environments can be customised by specialisation and extension. Sirus has a number of modelling editors for customers' convenience, all of which can be synchronised with one another. Sirus is compatible with eclipse 2020-09, oxygen, and neon. Specifications for Viewpoints The met model and its representation can be conceptually separated using models. The mapping model in the design format explains the structure, appearance, and behaviour of an editor that follows the domain model. Users can additionally offer appropriate extensions to their custom representations, such as the representation extensions that allow for customization of [17], [27].

Sirius-based workbench

Thales developed Capella, a one-of-a-kind graphical MBSE workstation with powerful visual editing features:

• customised diagrams depending on the type of representations used for the Arcadia approach (dataflow, scenario, functional chain, breakdown, modes & states, data model, component wiring,

allocation) (dataflow, scenario, workable chain, breakdown, modes & states, model, component wiring, assignment, and etc)

• an uniform colour scheme (all function-related elements are green, while all component-related elements are blue) (all elements connected to functions are green, while all elements related to components are blue)

• Taking good care of complexity (display of calculated cross exchanges, automated contextual diagrams, filters that automatically hide or reveal specific parts, etc.) (display of computed function exchanges, automated contextual diagrams, filters to hide or reveal specific aspects, and so on.) Modeling and quick fixes are divided into so many areas (Integrity, design, completeness, etc.). (Integrity, design, and completeness, for example.)

- ✓ etc.). (Integrity, design, completeness, etc.).

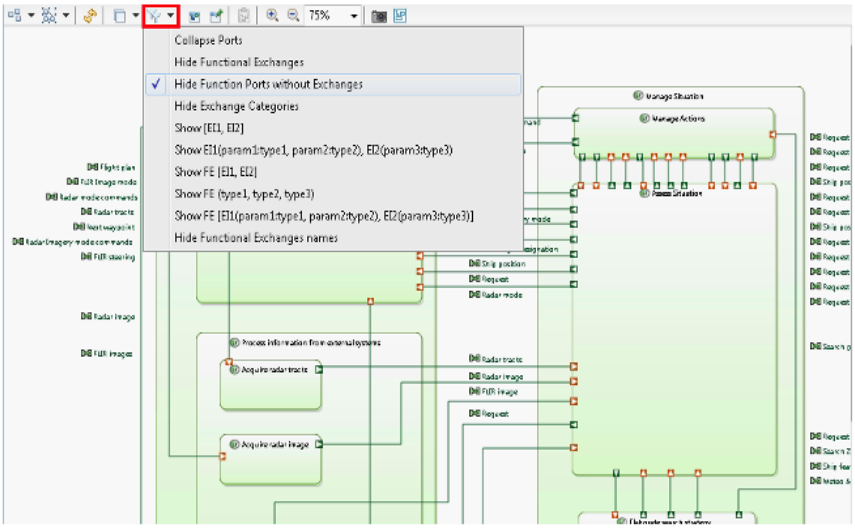


Figure 5 Filters

- ✓ Sirius could also be used to create new graphical representations targeted to specific engineering challenges utilising the Capella Studio environment, called as viewpoint:
- ✓ electrical power systems
- ✓ redundancy rules, failure scenarios and propagation
- ✓ reconfiguration issues

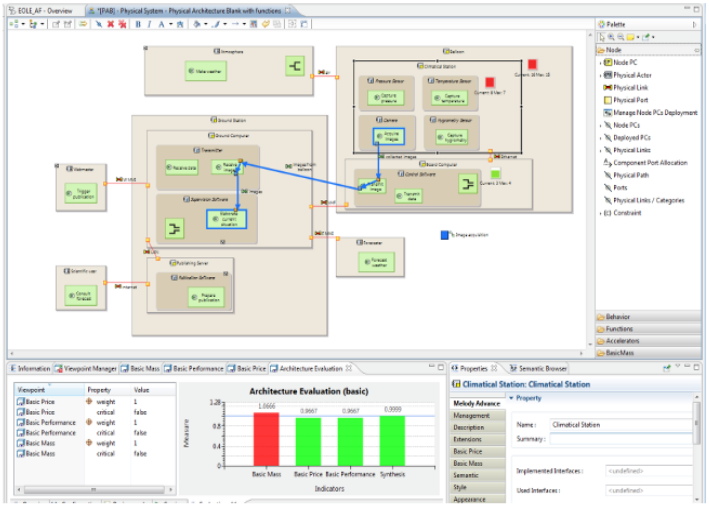


Figure 6 Specific Viewpoint

How does it work?

To customize the graphical editors, you just need basic technical expertise.

Sirius to specify the whole structure and behavior of model editors as well as the editing and navigation tools available use a description of the modeling workbench. A mapping between DSL notions (the Ecore model's components) and the visual workbench may be observed in this description (the diagrams, the tables, the tools, etc).

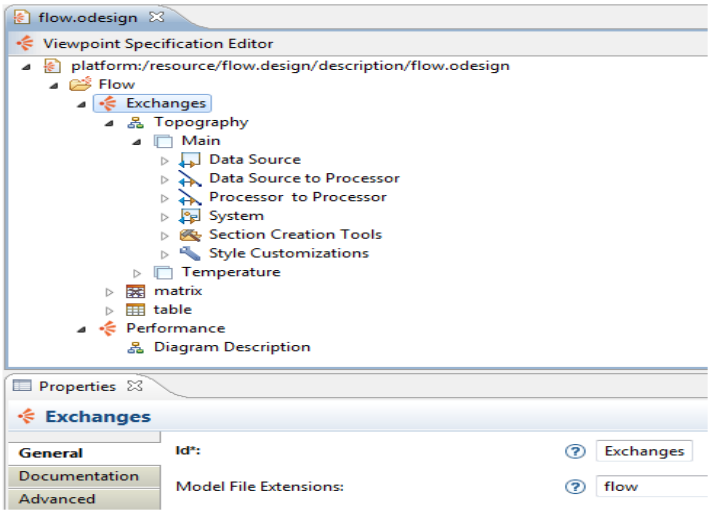


Figure 7 The definition of a Sirius modeling workbench

Additionally, Sirius provides a variety of options for users to focus on intriguing components in order to better understand the possible complexity of their models:

Styles that are conditional: Adapt the visual depiction of items in accordance with their characteristics. Making the backdrop red when an attribute value exceeds a given threshold or changing the shape's dimensions dependent on the attribute's magnitude are two examples.

Filters and layers: Conditions can be applied on the diagram's visibility or concealment.

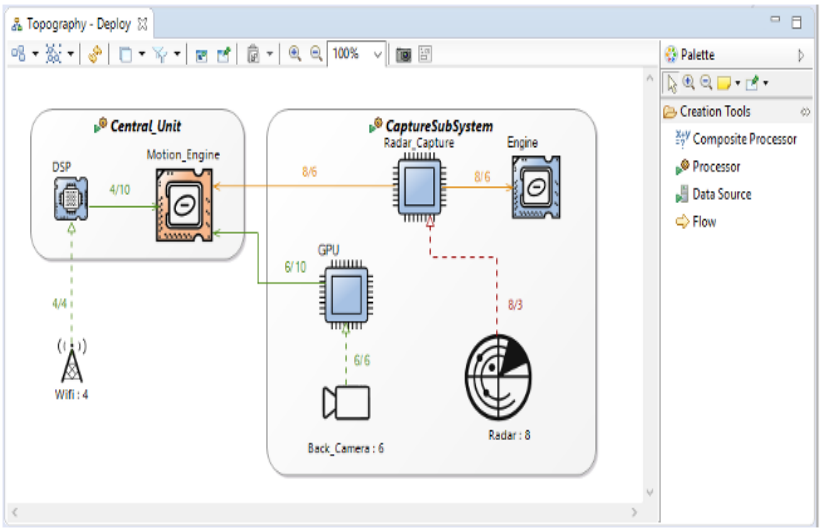
Validation and short-term fixes: The model may be verified using certain rules. A list of issues is displayed in the Problems View, along with a visual depiction of the pertinent components. Some model flaws can be readily fixed by offering short fixes. Sirius also will provide a "Modeling Project" connection to the Project Explorer. It integrates workspace resources and in-memory instances. The application will allow editors to share command stacks and editing zones with others.The Eclipse IDE's modelling workbench dynamically interprets the editors' descriptions.

The editors' description is dynamically reviewed and the workbench specifier receives quick feedback while altering the description to create the modelling workbench within the Eclipse IDE. This procedure produces no code. The modelling workbench can be added to Eclipse as a regular plugin once it's done. A workbench, or its specialisation, could be useful.

The description of a editors is dynamically reviewed, and the workbench specifier receives instant feedback while updating the description to develop the modelling workbench within the Eclipse

IDE. There really is no code generated by this procedure. Once finished, the modelling workbench can be loaded to Eclipse as a typical plugin.

Thanks to this quick feedback, a workbench or its specialisation can be built in a matter of a few hours.

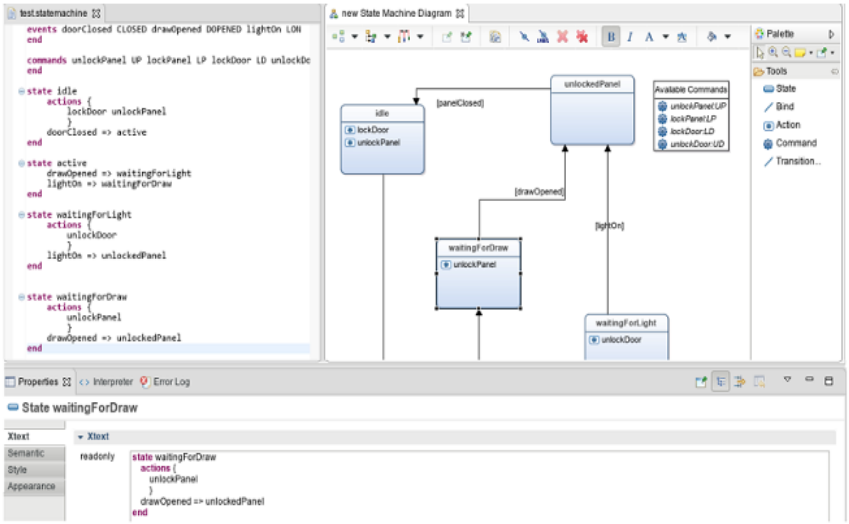


loop..

Figure 8 A diagram editor created with Sirius

Integration with other technologies

Sirius can be used with EMF Compare, Acceleo, or M2Doc to compare models (to generate MS Word documents).
Other technologies like Eclipse Xtext can also be used to create supplementary text editors for Sirius. Models may be loaded and saved using their textual syntax using Text, allowing the use of diagram, table, and textual editors all at the same time.



.

Figure 9 Sirius and Text editing the same model

Case study description

In the case study, the model comprises of mainly two scenarios, which are described as below:
In the first scenario, there is a person named Ali who goes shopping the goes through the store and views a catalog. In the catalog, we have our top product by the name skincare displayed. He then goes to the product section opens the skincare range and finds an item of vitamin C cream. Now our customers like that item and view its features. In features, we have a name, id, and description of an item that is a very good skincare product, etc., as well as the price that is 1000pkr. Now our customer adds that item to the wish list but does not buy it yet and returns it from the store.
In the second scenario the same person, Ali comes back another day. He again views the catalog He then goes to his wish list section. In his wish list item, vitamin C cream is being shown. He views the item and adds this item to the cart for purchasing. After adding the product to the cart, he selects a payment method. After selecting the payment method, he adds his address, phone number, contact details. He gets the option of courier service that is leopards for our store. After this invoice is generated showing total payment and all order details such as item, price, and customer contact. After confirming the order and checking out the customer is sent shown order date and expected delivery date

4. MODELING IN SIRIUS

The detail modeling in Sirius are discussed and explained in this section on detail with a proof of screenshots of each step.

4.1 Working in sirus

4.1.1 Implementation **of ecommerce store DSL (M2level) model**

The implementation of the ecommerce store DSL (M2level) model is shown in **Figure-10** below.

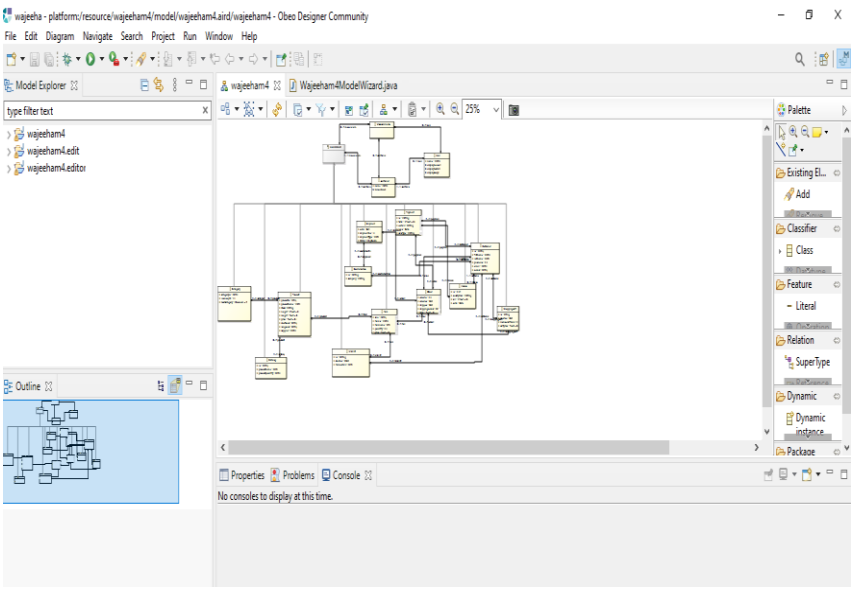


Figure 10 Implementation of ecommerce store DSL (M2level) model

4.1.2 Creating eclipse editor

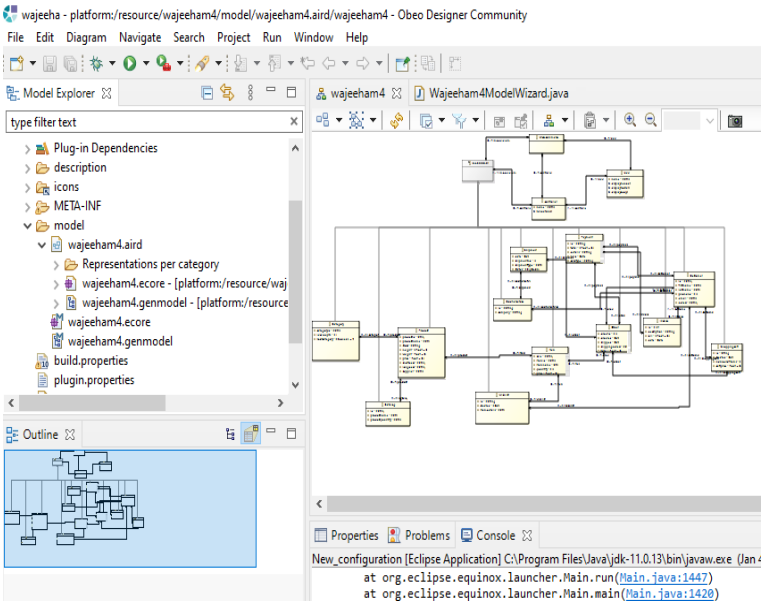


Figure 11 Creating eclipse editor

4.1.3 Editor tree view level model m1 level creation

The Editor Tree view level model m1 level creation on detail is shown in the Figure-12 below.

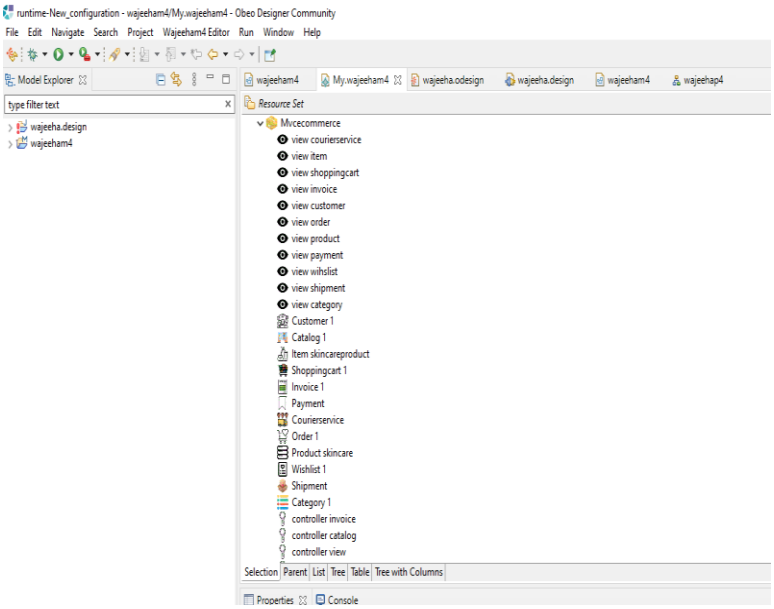


Figure 12 Editor Tree view level model m1 level creation

4.1.4 Setting up relationships

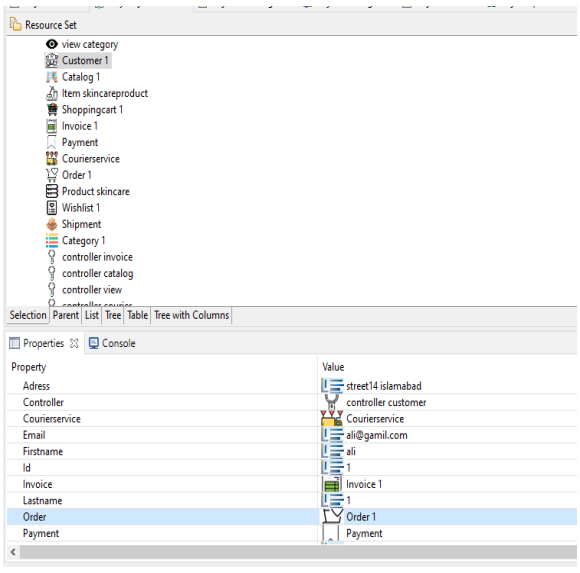


Figure 13 Setting up relationships

4.1.5 Created .design file

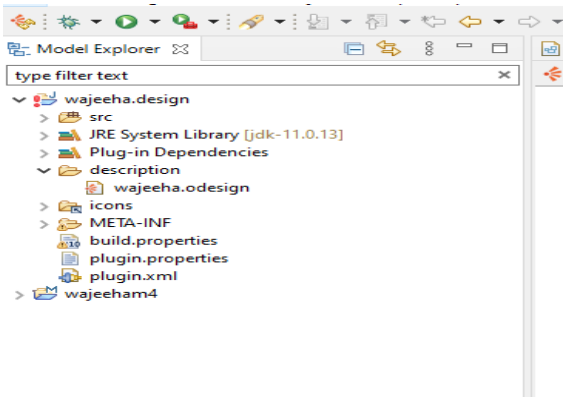


Figure 14 Created .design file

4.1.6 Created all nodes

4.1.7 All nodes created are shown in Figure-15 below.

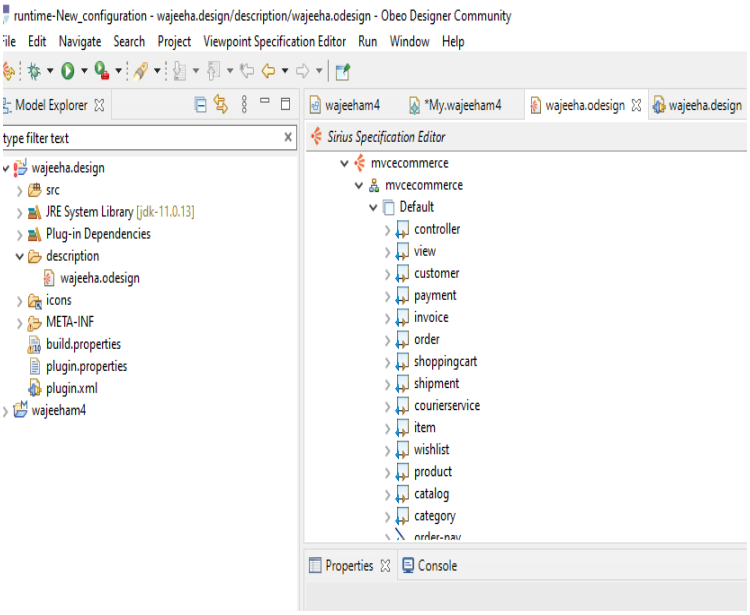


Figure 15 Created Nodes

4.1.8 Setting workspace images for each node

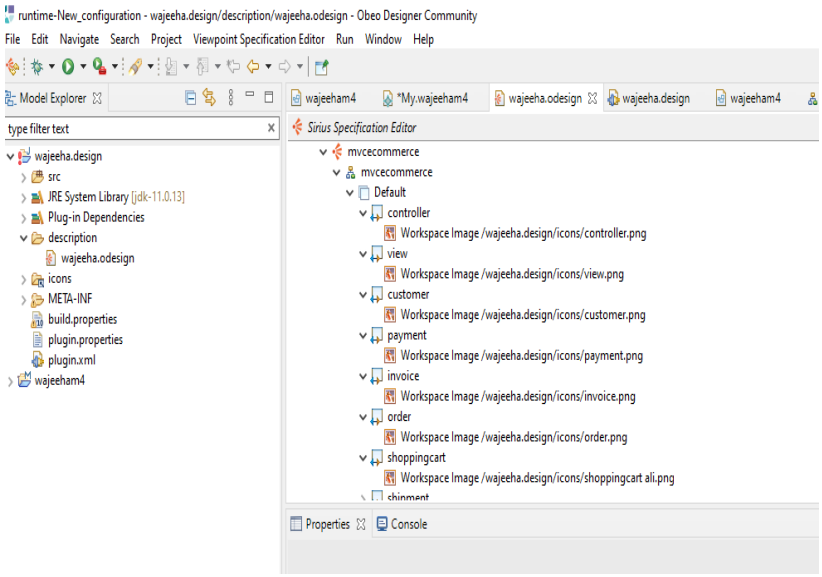


Figure 16 Setup workspace images for individual node

4.1.9 Then I created relation-based edges

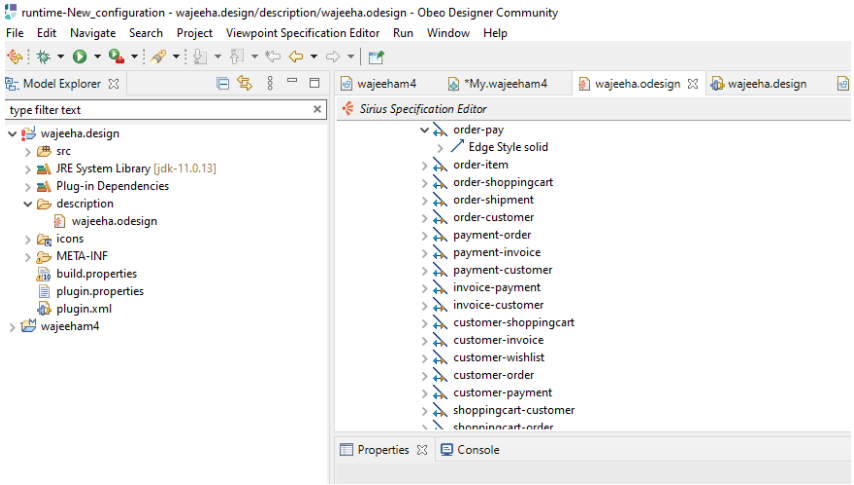


Figure 17 relation-based edges

4.10 Setting up their properties

The figure-18 shows setup and their properties.

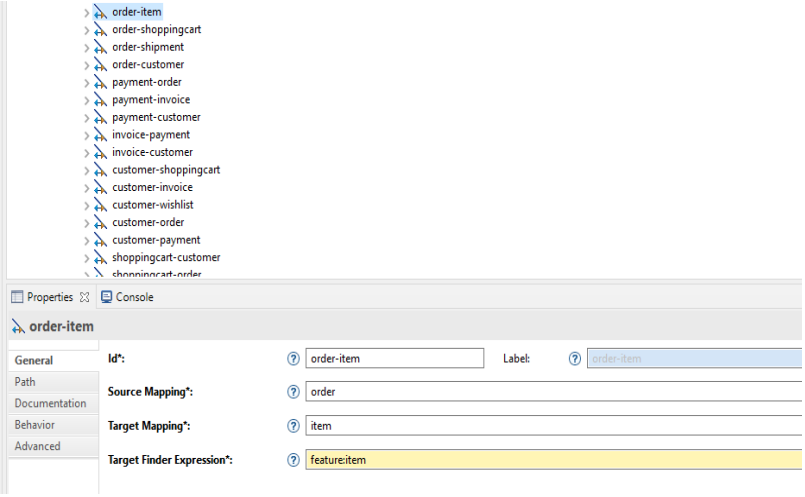


Figure 18 Setting up their properties

4.11 Model generation

The Figure-19 below shows the model generated on detail. Now, I will have to design palette for users to easily drag, drop for custom edge creation for that section is created, and in section, all nodes are created further.

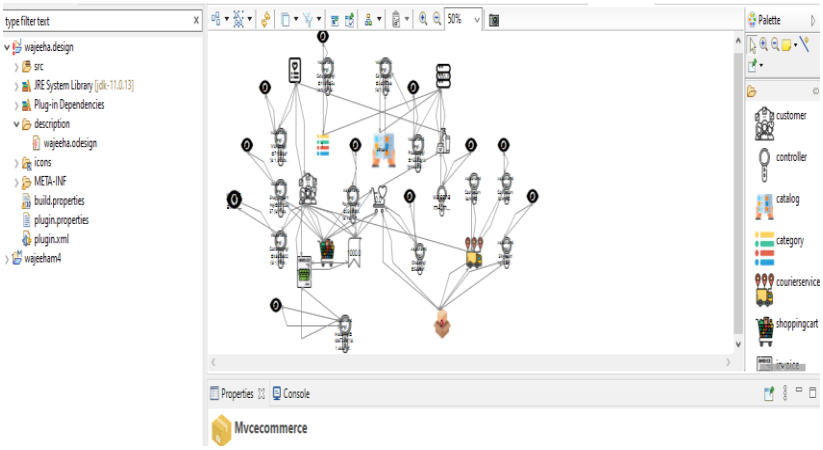


Figure 19 Model Generation

At the movement, as our model is generated it’s time to design palette for users to easily drag and drop for custom edge creation for that section is created and in section all nodes are created again.

The details are shown in Figure-20 below.

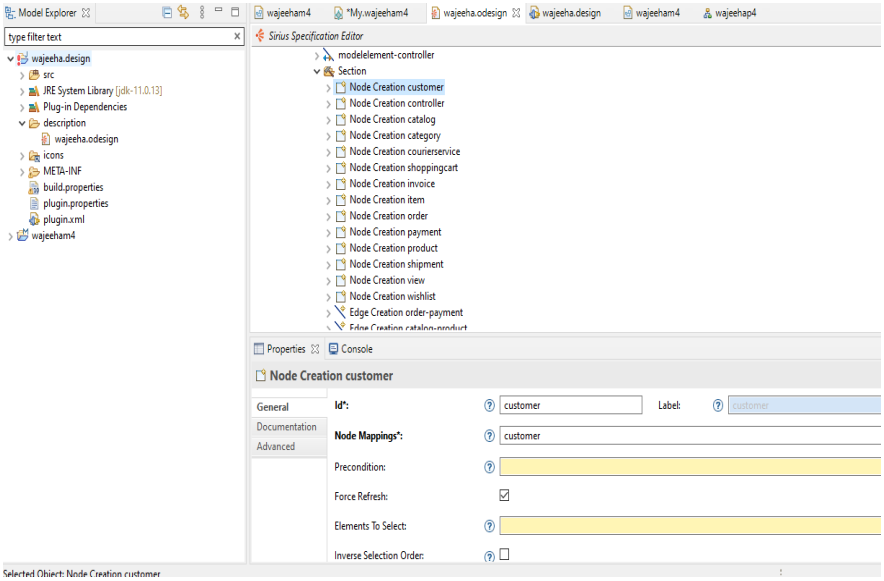


Figure 20 Design Palette

The main properties of after this all relation, based edges were created and their properties were setup in the figure 21-

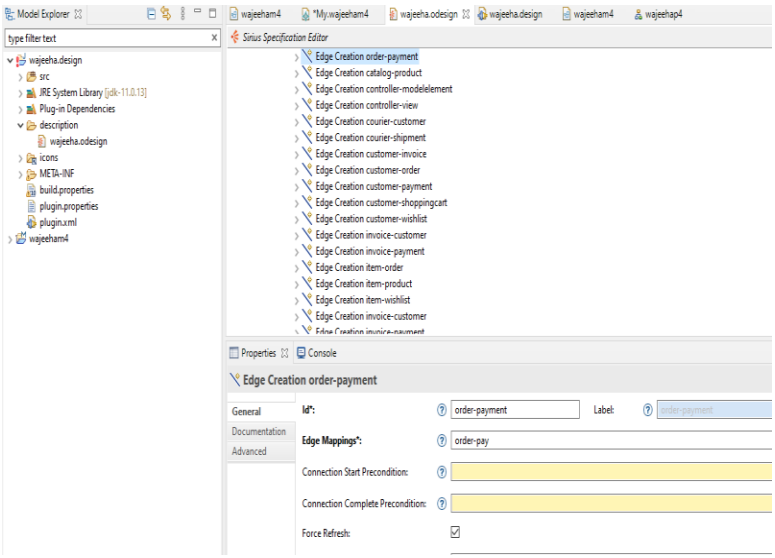


Figure 21 Relation & Base Edges Created

Final model with palette design

Now, the final design of the project is designed on detail. The figure-22 shows the final model which has been designed.

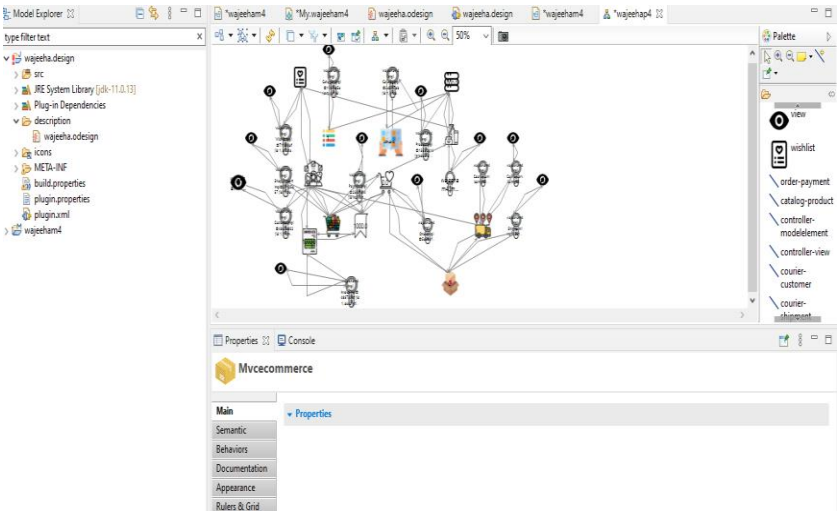


Figure 22 Final model with palette design

5. MODELING IN MPS

The below figures show the overall modeling in MPS.

First, I have created a sandbox, which is shown in Figure-23 below.

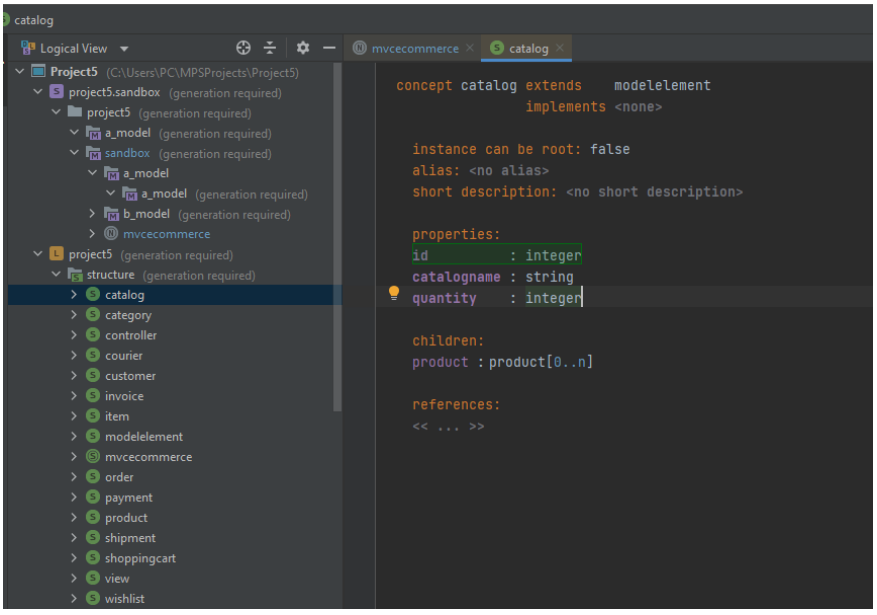


Figure 23 Sandbox

After the Sandbox creation, I have created classes and properties of each concept shown in figure-24.

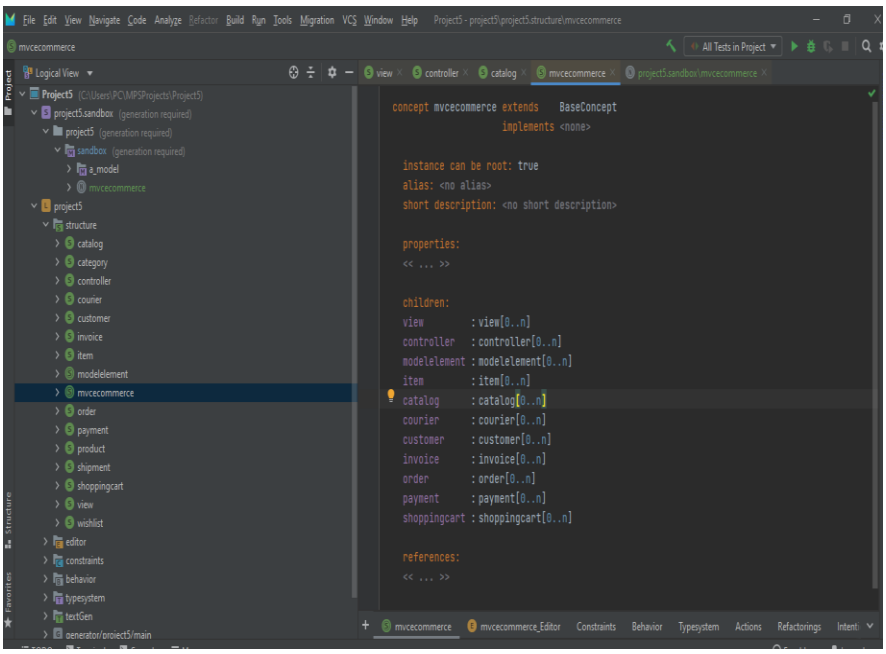


Figure 24 created children and properties

After creating classes the editors of Mvc class, item class and controller class were made as shown in fig 25:

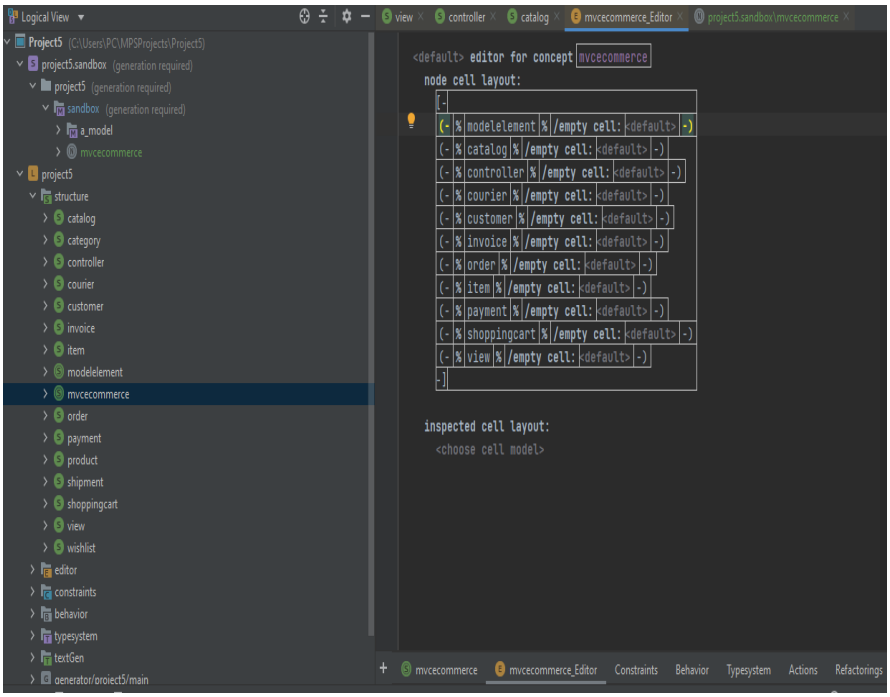


Figure 25 created children and properties

Now the final tree view was built as shown in figure 26-27:

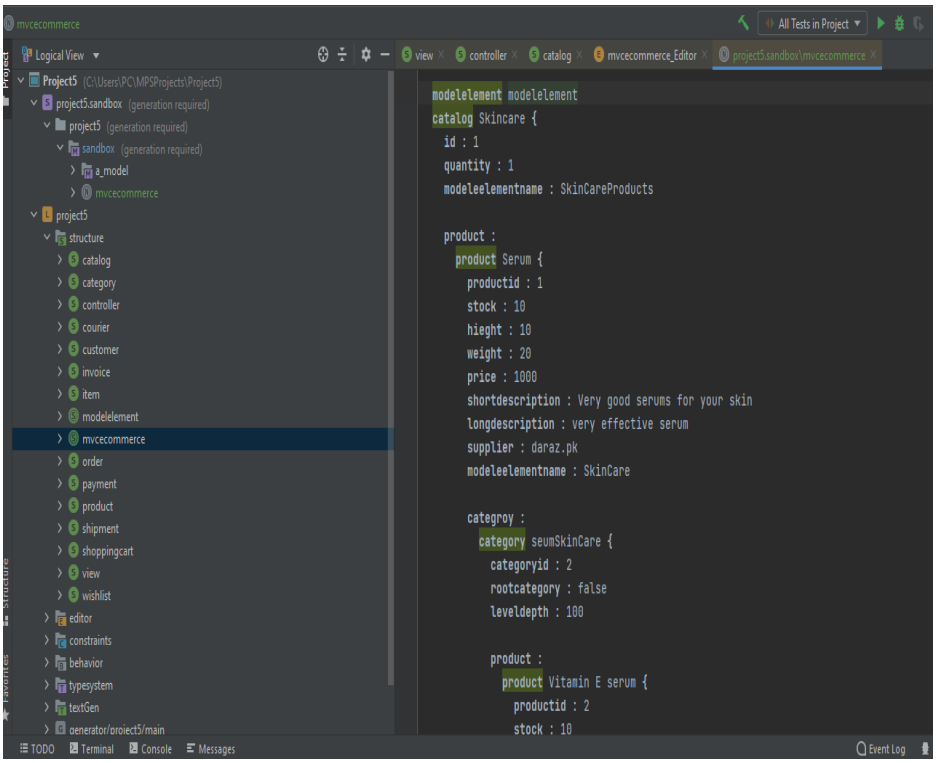


Figure 26 created children and properties

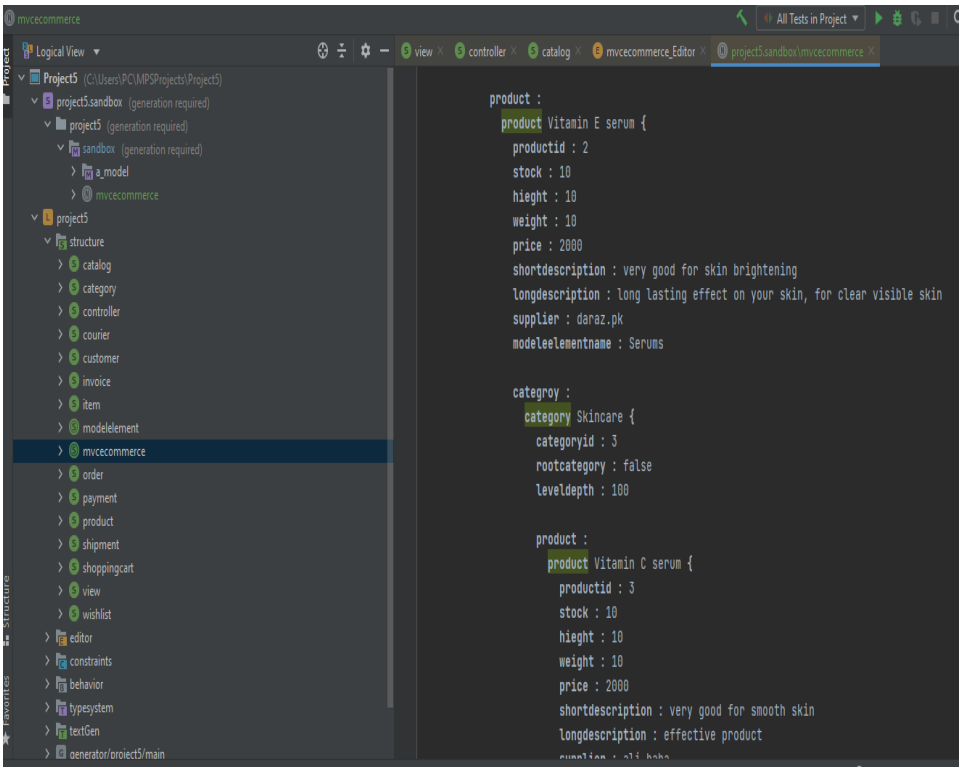


Figure 27 created children and properties

6. COMPARATIVE ANALYSIS

The below tables show the comparison of

Table 1

Task	Description	Time spent	Issue faced	Resolution of issue including working hour breakdown
Task 1	Obeo installation	15 minutes	No issue	
Task 2	Simple M2 level class diagram	20 minutes	Setting data type for money	I added data type for money java.lang.object and it was java.lang.integer
Task 3	Create a dynamic instance (M1 level) of Mortgage System class	10 minutes	No issue	
Task 4	Setting values of mortgage attributes	10 minutes	values were not setting up for the attributes	Wrong data type of money was entered changed the data type to java.lang.integer after

				that the attributes value was set
Task 5	Ocl installation	15 minutes	None	
Task 6	Execute OCL statements on instance model, in both the consoles 1) Interactive OCL Console and 2) Interactive Text Console	20 minutes	execution command errors	Checking and correcting spelling mistakes of class and names
Task 7	Using Sirius made family tree	30 minutes	None	

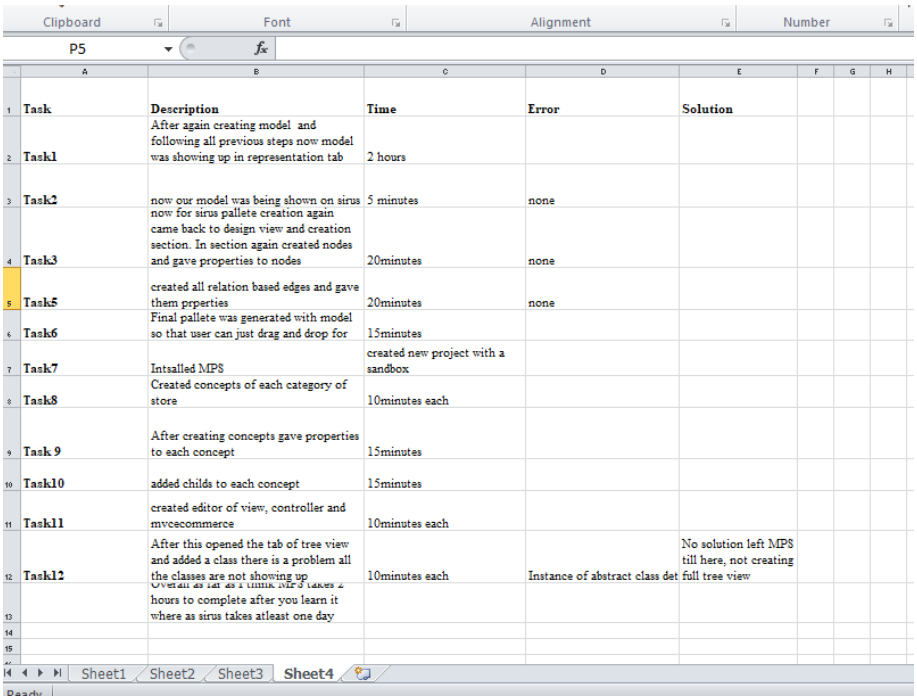
Table 2

	A	B	C	D	E
1	**Task**	**Description**	**Time spent**	**Issue faced**	**Resolution of issue**
2	**Major Task**	Created m2 level ecommerce diagram on obeo	1 hour	none	implemntation was successful
3	**Task 1**	Created all the classes of MVC ecommerce store where Model element class was abstract	15 minutes	none	
4	**Task2**	Setting attributes in each class and setting variable types	15 minutes	none	
5	**Task3**	Created relations between all classes	30 minutes	Each class had relations with multiple classes so relations were being mixed	
6					

Table 3

	A	B	C	D	E	F
1	**Task**	**Description**	**Time spent**	**Issue faced**	**Resolution of issue**	
2	**TASK 1**	Generated model representation of e-commerce store on obeo	5mins	None		
3	**Task2**	Saved gif icons in edit folder of project and reloaded sirus after code generation and icons were replaced	5 minutes	None		
4	**TASK 2**	Created child nodes of each class	15mins	None		
5	**TASK 3**	Generated all child nodes and assigned values of each attributes	20minutes	Date was not being entered in attributes	Changed date format type in model diagram	
6	**Task4**	In the design folder first saved the metamodel then created nodes of each class	10minutes			
7	**Task5**	Saved the png icons in build tab source fi	5 minutes	By mistake I was placing my icons folder on desktop that was creating error	Saved icons folder in workspace where my project was saved then gave path of folder on build tab.	
8	**Task6**	Gave path of each workspace icon that was loaded to each node	10minutes			
9	**Task7**	Nextly created all the relation based edges	20minutes	none		
10	**Task8**	Had to choose representation of model from .arid file		model was not showingup in representation	Again created a new project and followed all previous steps	
11						
12						

Table 4



Task	Description	Time	Error	Solution
Task1	After again creating model and following all previous steps now model was showing up in representation tab	2 hours		
Task2	now our model was being shown on sirus	5 minutes	none	
Task3	now for sirus pallete creation again came back to design view and creation section. In section again created nodes and gave properties to nodes	20minutes	none	
Task5	created all relation based edges and gave them prperties	20minutes	none	
Task6	Final pallete was generated with model so that user can just drag and drop for	15minutes		
Task7	Intsalled MPS	created new project with a sandbox		
Task8	Created concepts of each category of store	10minutes each		
Task 9	After creating concepts gave properties to each concept	15minutes		
Task10	added childs to each concept	15minutes		
Task11	created editor of view, controller and mvcecommerce	10minutes each		
Task12	After this opened the tab of tree view and added a class there is a problem all the classes are not showing up	10minutes each	Instance of abstract class det	No solution left MPS till here, not creating full tree view
	Overall as far as I think MPS takes 2 hours to complete after you learn it where as sirus takes atleast one day			

.

From running and completing the overall process through Sirius and MPS, I get a conclusion about both of these through comparative analysis that:
Sirius is a very complex tool and produces more errors in almost every step for developers. In addition, you may go back to start your model from scratch when facing an error.
While in comparing MPS with Sirius, then MPS is far better than Sirius in almost all perspectives.

7. THREATS TO VALIDITY

DSLs have the disadvantage of being difficult to develop and deploy, as well as having higher initial costs. In this case, MDE comes into play: DSLs and MDE go together like a dream. The MDE technique makes defining a DSL a lot easier. The term "domain-specific modelling" (DSM) is used in MDE to describe the process of creating a DSL (DSM). DSM, unlike DSL, use models to explain the domain system's numerous components. Models that are based on a graphical representation are frequently aided by graphic design tools. Users can use the DSM tool to construct domain models and, in most circumstances, generate particular code based on those models. It takes a long time to create a graphical tool for DSM. In the case of GMF and Sirius, it's required to demonstrate the editor's many features by building one or more models, which can get rather intricate, huge, and difficult to create and manage for non-experts, and which frequently require the usage of awkward tree-based editors. If we don't have code generation, we may have to rely on Sirius for a lot of customisation options. Adding custom edge decorators other than those provided by Sirius, for example, would entail code changes in the Sirius core, which is a visible

constraint. A T-shaped arrowhead cannot be used to signify and-refinement in the editor. Because constructing an editor from scratch is practically impossible, it takes a long time to build and typically relies on reliable frameworks. Aside from the technical challenges, one of the most significant issues with most graphical language workbenches is the requirement to first construct a meta-model, and then describe the features of the concrete syntax and the modelling environment in a second stage using a technical language or notation. This strategy prevents domain experts from actively participating in the DSL creation process since they may prefer to work with examples rather than meta-models and may lack the technical expertise to create complex environment requirements.

8. CONCLUSION

While encapsulating GMF, Sirius has been proven to simplify the product, reduce design time, and increase overall productivity quickly.As a result, the Sirius platform's core strengths are: an open and widely used industry standard - EMF; adaptability to any EMF-compatible DSM; a strong separation between semantic and representation models; support for various domain model representations; ease of use and rapid development; and a high level of extensibility. To put it another way, As a framework built on top of GMF, Sirius is pretty flexible and dynamic, and it offers a way to rapidly develop graphical tools without requiring any prior knowledge of backend processes. The design and implementation of a Sirius-based Graphical Editor for RESTful Sensor Web Network have shown designers with no prior experience in a solution domain may be provided a tool that allows users to easily define jobs, develop network architecture, and create RESTful services. I mentioned e-commerce with the addition of its various viewpoints in the intended project. Then we discussed the various languages and software programmes, as well as their benefits and drawbacks, in order to assist us with our modelling. Working with Sirus is a terrific alternative, and it offers a lot of good features, but MPS is easier to work with and has more versatility. When working with Sirius, there is a greater probability of making mistakes at any level. Many java bugs or many failures can occur in your setup, making it difficult to locate and diagnose. Errors do arise in MPS, but they are simple to locate and comprehend. MPS is more user-friendly, whereas Sirus is solely for programmers. Sirius creates a diagram from nodes or containers that correspond to Ecore classes and queries, which fetches the instances to display. Compostable language definitions are supported by MPS. Sirius is a sophisticated technology that causes developers to make more mistakes in practically every step. In addition, if you encounter an issue, you can restart your model from the beginning. When comparing MPS to Sirius, MPS is considerably superior to Sirius in practically every way.

REFERENCES

1.Mahadevan, B. (2000). Business models for Internet-based e-commerce: An anatomy. California management review, 42(4), 55-69.

2. Shahriari, S., & Mohammadreza, S. (2015). E-COMMERCE AND IT IMPACTSON GLOBAL TREND AND MARKET. International journal of research-Granthaalayah, 3(4), 49-55.

3. Al-Qirim, N. (2005). An empirical investigation of an e-commerce adoption-capability model in small businesses in New Zealand. Electronic Markets, 15(4), 418-437.

4. Leitner, P., & Grechenig, T. (2008). Collaborative shopping networks: Sharing the wisdom of crowds in E-commerce environments. BLED 2008 Proceedings, 21.

5. Jewels, T. J., & Timbrell, G. T. (2001). Towards a definition of B2C & B2B e-commerce.

6. Swani, K., Brown, B. P., & Milne, G. R. (2014). Should tweets differ for B2B and B2C? An analysis of Fortune 500 companies' Twitter communications. Industrial marketing management, 43(5), 873-881.

7. Niranjanamurthy, M. (2014). E-commerce: Recommended online payment method-Paypal. International journal of computer science and mobile computing, 3(7), 669-679.

8. Allen, E., & Fjermestad, J. (2001). E-commerce marketing strategies: an integrated framework and case analysis. Logistics information management.

9. Goel, R. (2007). E-commerce. New Age International.

10. Weiss, R. M., & Mehrotra, A. K. (2001). Online dynamic pricing: Efficiency, equity and the future of e-commerce. Va. JL & Tech., 6, 1.

11. Jelassi, T., & Martínez-López, F. J. (2020). The Strategic Approach of the World's Biggest e-Tailing Companies: Amazon and Alibaba. In Strategies for e-Business (pp. 467-500). Springer, Cham.

12. Savic, D., da Silva, A. R., Vlajic, S., Lazarevic, S., Antovic, I., Stanojevic, V., & Milic, M. (2014, September). Preliminary experience using JetBrains MPS to implement a requirements specification language. In 2014 9th International Conference on the Quality of Information and Communications Technology (pp. 134-137). IEEE.

13. Voelter, M., & Solomatov, K. (2010). Language modularization and composition with projectional language workbenches illustrated with MPS. Software Language Engineering, SLE, 16(3).

14. Voelter, M. (2011, July). Language and IDE Modularization and Composition with MPS. In International Summer School on Generative and Transformational Techniques in Software Engineering (pp. 383-430). Springer, Berlin, Heidelberg.

15. Hudak, P. (1998, June). Modular domain specific languages and tools. In Proceedings. Fifth international conference on software reuse (Cat. No. 98TB100203) (pp. 134-142). IEEE.

16. Shen, L., Chen, X., Liu, R., Wang, H., & Ji, G. (2021). Domain-Specific Language Techniques for Visual Computing: A Comprehensive Study. Archives of Computational Methods in Engineering, 28(4), 3113-3134.

17. Sharma, G., & Lijuan, W. (2015). The effects of online service quality of e-commerce Websites on user satisfaction. The Electronic Library.

18. Smith, A. D., & Rupp, W. T. (2003). Strategic online customer decision making: leveraging the transformational power of the Internet. Online information review.

19. Niranjanamurthy, M., Kavyashree, N., Jagannath, S., & Chahar, D. (2013). Analysis of e-commerce and m-commerce: advantages, limitations and security issues. International Journal of Advanced Research in Computer and Communication Engineering, 2(6), 2360-2370.

20. Rich, M. K. (2003). Business-to-Business Marketing: Strategies and Implementation. Journal of Business & Industrial Marketing.

21. Lucking-Reiley, D., & Spulber, D. F. (2001). Business-to-business electronic commerce. Journal of Economic Perspectives, 15(1), 55-68.

22. Kincaid, J. W. (2003). Customer relationship management: getting it right!. Prentice Hall Professional.

23. Reinartz, W., Wiegand, N., & Imschloss, M. (2019). The impact of digital transformation on the retailing value chain. International Journal of Research in Marketing, 36(3), 350-366.

24. Savic, D., da Silva, A. R., Vlajic, S., Lazarevic, S., Antovic, I., Stanojevic, V., & Milic, M. (2014, September). Preliminary experience using JetBrains MPS to implement a requirements specification language. In 2014 9th International Conference on the Quality of Information and Communications Technology (pp. 134-137). IEEE.

25. Voelter, M., & Solomatov, K. (2010). Language modularization and composition with projectional language workbenches illustrated with MPS. Software Language Engineering, SLE, 16(3).

26. Voelter, M., Siegmund, J., Berger, T., & Kolb, B. (2014, September). Towards user-friendly projectional editors. In International Conference on Software Language Engineering (pp. 41-61). Springer, Cham.

27. Bettini, L., & Crescenzi, P. (2015, July). Java-meets eclipse: An IDE for teaching Java following the object-later approach. In 2015 10th International Joint Conference on Software Technologies (ICSOFT) (Vol. 2, pp. 1-12). IEEE.

28. Völter, M. (2011). Language and IDE modularization, extension and composition with MPS. *Pre-proceedings of Summer School on Generative and Transformational Techniques in Software Engineering (GTTSE)*, 395-431.

29. Vacchi, E., & Cazzola, W. (2015). Neverlang: A framework for feature-oriented language development. Computer Languages, Systems & Structures, 43, 1-40.

30. Voelter, M. (2011, July). Language and IDE Modularization and Composition with MPS. In International Summer School on Generative and Transformational Techniques in Software Engineering (pp. 383-430). Springer, Berlin, Heidelberg.

31.Vujovic, Vladimir & Maksimović, Mirjana & Perisic, Branko. (2014). Sirius: A Rapid Development of DSM Graphical Editor. 10.1109/INES.2014.6909375.