REA Business Modeling Language

Toward a REA based Domain Specific Visual Language

Mohannad M. Al-Jallad

Department of Computer and Systems Sciences

Degree project 30 HE credits

Degree subject (Engineering and Management of Information Systems)

Degree project at the master level

Autumn/Spring term 2012

Supervisor: Paul Johannesson

Reviewer: Petia Wohed

Swedish title: REA Affärsmodelleringsspråk - ett REA baserat visuellt och domänspecifikt språk

Abstract

Resources Events Agents (REA) ontology is a profound business modeling ontology that was developed to define the architecture of accounting information systems. Nevertheless, REA did not manage to get the same attention as other business modeling ontologies. One reason of such abandon is the absence of a meaningful visual notation for the ontology, which has resulted in an abstruse ontology to non-academic audience. Another reason for this abandon is the fact that REA does not have a standard formal representation. This has resulted in a humble amount of researches which have focused on defining meta-models of the ontology while neglecting the wider purpose of REA-based information systems development. Consequently, the ontology was deviated away from its original purpose, and rather used in business schools.

To solve the aforementioned issues, this research presents a Model Driven Development (MDD) technique in the form of a REA-based Domain Specific Visual Language (DSVL) that is implemented within a modeling and code generation editor. This effort was taken in order to answer the question of "*How would a REA-DSVL based tool make the REA ontology implementable in the domain of information systems development?*"

In order to answer the research question, a design science methodology (DSRM) was implemented as the structure of this research. The DSRM was chosen because this research aims to develop three main artifacts. These are; a meta-model of REA, a visual notation of REA, and a REA-DSVL-based modeling and code generation tool.

The first phase of the DSRM was to identify the problems which were mentioned earlier, followed by the requirements identification phase which drew the outline of the; meta-model, the visual notation, and the tool. After that, the development phase was conducted in order to develop the aforementioned artifacts. The editor was then demonstrated using a case study of a local company in Stockholm-Sweden. Finally, the resulted artifacts were evaluated based on the collected requirements and the results from the case study.

Based on the analyses of the artifacts and the case study, this research was concluded with the result that a REA-based DSVL tool can help in boosting the planning and analysis phases of the software development lifecycle (SDLC). This is achieved by automating some of the conventional software planning and design tasks, which would lead to more accurate systems' designs; thus, minimizing the time of the planning and design phases. And it can be achieved by abstracting the direct logic of REA through providing functionalities that help users from different backgrounds (academic and professional) to embrace a business modeling editor rather than an ontology; thus, attracting a wider users base for implementing REA.

Keywords

Resource Event Agent, REA ontology, Domain Specific Visual Language, DSVL, Domain Specific Language, DSL, meta-modeling, Eclipse, Ecore, EMF, GMP, visual notation development.

Acknowledgement

I would like to think of this research as the result of all the past *events* which have resulted in me coming here to Sweden, attending my current school, meeting all the great people that I have met here, and finally finishing this work. I thank the lord for all that happened, and for what is yet to come.

This research wouldn’t have seen the light if it wasn’t for the profound support and perfect guidance of Professor. Paul Johannesson, whose wisdom and knowledge are sources that I had the honor to witness and learn from. For that sir, I thank you.

Even though my gratitude cannot be expressed by words to them, I would like to extend my gratitude to my family. To my father’s soul, my mother, sister, and brothers, thank you. Without each and every one of you, I wouldn’t be the man that I am today.

Mohannad M. Al-Jallad

Table of Content

List of Figures

List of Tables

1. Introduction

This chapter intends to provide the background of this research and the problem it addresses, and it describes the following content of this research document.

1.1 Background

Enterprises implement information systems to improve their businesses. Information systems add values to different aspects within an enterprise. These aspects include the enterprise's work procedures, its employees, and the development and implementation methodologies within that enterprise (HEVNER, Alan R. et al., 2004). Due to the fact that businesses are competitive in nature, and that an enterprise might need to change its business or parts of it to compete with other enterprises which operate in the same business domain, the impact of performing changes to a business without carefully studying the enterprise's knowledge and its business domain can be very costly; thus, enterprise models, which reflect different structural aspects within the enterprise, can be very helpful in assessing the impact of such changes (B. YU, J.A. Harding, K. Popplewell, 2000) (RAHMOUNI, M. and Lakhoua, M.N., 2011).

Enterprise modeling has emerged, amongst other reasons, to provide a source for enterprise domain knowledge. This enterprise knowledge makes it easier for different roles within an enterprise to understand and analyze different aspects of the enterprise (ZOUGGAR, N. et al., 2009). While enterprise models provide a description of the processes and business environment as they exist in a *specific* enterprise, enterprise ontologies provide definitions of the concepts and relationships that would appear in the *domain* of the enterprise's business in general. This nature of domain ontologies authorizes them to be used as blueprints of business-domains which are practiced within different enterprises. Keeping the purpose of business ontologies in mind, modeling languages are still needed to build models of the processes, business, or structure of the modeled enterprises. In this sense, ontologies and modeling languages are complementing each other in the domain of enterprise modeling (ZOUGGAR, N. et al., 2009).

Several enterprise/business modeling ontologies were introduced throughout literature and practice. The most well-accepted and used ones are e3-value, Resource-Event-Agent (REA), and Business Model Ontology (BMO) (SCHUSTER, R. and Motal, T., 2009) (SONNENBERG, C. et al., 2011 b) (SONNENBERG, C. et al., 2011 a) (GAILLY, Frederik and Poels, Geert, 2007). Some of these ontologies are suitable for developing business information systems. Others fit for supporting the knowledge base of their domains when used. REA is an ontology that was developed first to provide a financial analysis model (MCCARTHY, William E., 1982) , and then it has emerged across two decades to support the development of business information systems (GEERTS, Guido L. et al., 1996) (GEERTS, Guido L. and McCarthy, William E., 2000) (GEERTS, Guido L. and McCarthy, William E., 2002) (GEERTS, Guido L. and McCarthy, William E., 2006). Although REA, and other similar business modeling ontologies, claim their suitability in supporting the production of information systems, the implementations of such ontologies is limited in that domain.

In order to put business models into a practical perspective, conventional means for information systems development have been elevated to higher levels of abstraction. This movement toward abstractness is elaborated on the usage of models as key drivers for systems representation and development (MU, Liping et al., 2010). Model Driven Development (MDD) is an approach for describing and building software systems. Using this approach, different software modeling languages are used to model different aspects of the software. These models are then used to generate one software system (STAAB, Steffen et al., 2010). While MDD focuses on the general methodology of model-based software development, Model Driven Engineering (MDE) focuses on the design and specification of the modeling languages which are used in MDD (STAAB, Steffen et al., 2010). MDE can be considered as a generalized/global approach that is based on the Model Driven Architecture (MDA).

On its behalf, MDA is an architecture that was developed by the Object Management Group (OMG) to facilitate the separation between the business logic, system requirements, and the platform of software systems (OMG, 2003). The specifications document of MDA defines four layers[1] to accomplish this separation of concerns. MDA uses technologies developed by OMG, namely UML and MOF, at its technical base (OMG, 2003). In order to expand the benefits of MDA outside the boundaries of OMG's technologies, MDE (as a generalization of MDA) uses the four-layered architecture in general technical spaces. One of these technical spaces is the usage of ontologies as means for modeling software systems (LAFORCADE, Pierre, 2010).

MDE is practically built on top of two main concepts; *modeling languages*, which are used to produce models, and *model transformation* which is related to the transfer of models from one modeling language to another (STAAB, Steffen et al., 2010). Models are implementations (or products) of modeling languages. Modeling languages on their behalf are implementations of meta-models. Meta-models are defined as models of models, or models which are used to build other models (GONZALEZ-PEREZ, Cesar and Henderson-Sellers, Brian, 2008). Meta-models are also considered the core artifacts which are used for models transformation, and they are required to successfully deliver the benefits of MDE when generating model-driven systems (STAAB, Steffen et al., 2010).

Domain Specific Modeling (DSM) is an implementation method of MDE. Similar to ontologies, modeling languages in DSM are designed specifically for modeling particular domains and not any other. In MDA, UML can be used to model different spaces including the particular domain of any given DSM language. This nature of DSM gives domain experts more control over the correctness and completeness of their models as it gives these experts more control over the domains that they are familiar with (LAFORCADE, Pierre, 2010). For example, if a manager of some financial department wants to build a model that represents her department, a typical DSM language would provide her with elements which directly represent; employees, money, contracts…etc. On the other hand, when using UML, this manager would need to model these business entities using UML classes according to the appropriate Object Oriented (OO) relations between these entities. This of course is a difficult task to non-technical IS personnel; thus, the need for IS engineers would be inevitable in the latter scenario. This, indeed, is one of the benefits associated with using DSM languages; that is, to allow the direct participation of system owners in designing and developing their own business solutions.

One variant of DSM languages is referred to as Domain Specific Visual Languages (DSVL). A DSVL provides business modelers with visual modeling mechanisms. Domain experts can use such mechanisms, usually implemented inside MDD tools, to design models that represent their businesses. Such designing tools provide customary business domain concepts in the form of visual components (drawings or figures). These tools would then use the developed models to generate business applications for the modeled domains automatically (SPRINKLE, Jonathan and Karsai, Gabor, 2004).

[1] *The details of the four-layered architecture will be discussed in the next chapter.*

Usually, modeling languages which are used within DSVL-based tools are specifically designed to support the very specific need of a particular business. In order to bring the benefits of ontologies and DSVLs together, an ontology-based visual modeling language should be produced, and then implemented within a DSVL-based modeling tool. This, indeed, is an outline of what this research is trying to accomplish. This research tries to contribute to the movement toward models evolution, by incorporating a well-distinguished business modeling ontology, namely the Resource-Event-Agent (REA), in the domain of MDD. This attempt is carried out in order to bring business modeling ontologies, the REA ontology in this case, closer to the domain of information systems development.

1.2 Problem definition

Resource Event Agent (REA) is a business ontology which has well defined roots from the heart of the economic theory. REA was originally developed to support the creation of financial databases, and it has emerged later to support the production of business information systems (GEERTS, Guido L. and McCarthy, William E., 2002). REA faces some problems which limit the ontology's capability of reaching the level of systems development support that it was created for. The main problem is that REA is still viewed as a raw ontology (SONNENBERG, C. et al., 2011 a) (GAILLY, Frederik and Poels, Geert, 2007). REA is described in a set of papers by McCarthy and Geerts without a formal representation of the ontology[1], and it lacks a standard visual notation for it to be accepted by IS professionals who do not possess a previous knowledge of REA. These problems have suppressed the true power of REA in designing business information systems, which has resulted in a very few practical implementations of the ontology. The previous problems form the foundation of this research. This section intends to describe the identified problems in details.

REA lacks a formal representation that conveys a common and a single comprehension of the ontology among its users. There have been many attempts to represent the ontology in a formal manner (GAILLY, Frederik and Poels, Geert, 2007) (SONNENBERG, C. et al., 2011 a) (SONNENBERG, C. et al., 2011 b); though, most of these attempts were incomplete or inaccurate due to the complexity of the ontology and its numerous bifurcations. In addition to that, some of the previous attempts managed to provide different meta-models with different cardinalities. These results are invulnerable evidence to the severity of the mentioned problem.

In addition to a formal representation, REA also lacks a visual notation that places it as a modeling language rather than a rigorous business ontology. According to (SONNENBERG, C. et al., 2011 a), William E. McCarthy, the founder of REA, personally expressed to the authors that REA needs a sufficient visual notation like the one associated with e3-value, and that such a visual notation would make the ontology easier to reach business modelers. Furthermore, providing REA users with a modeling tool like the one available for e3-value might help in making such goal easier to reach.

The amount of work that has been done so far around the previously mentioned problems, tried to solve one of the identified problems, but not the other. The previous problems can be solved by building a modeling language which has an abstract syntax in the form of a meta-model; thus, solving the problem of the formal representation, and by building a concrete syntax in the form of a visual notation, which would solve the problem of the missing visual notation. These solutions might help in solving their associated problems; though, the main problem of bringing REA closer to an

[1] A formal representation in this discussion means a meta-model with clear cardinalities between the ontology's entities.

implementation context requires providing mechanisms that allow the generation of system objects, which can be used directly in building software systems. The latter problem can be solved by developing a DSVL-based MDD tool, which can be used for modeling and code generation (SPRINKLE, Jonathan and Karsai, Gabor, 2004) (LAFORCADE, Pierre, 2010).

Developing a visual notation and a meta-model for REA are direct solutions for their associated problems, and their results can be directly judged by the developed artifacts. On the other hand, using a DSVL for bringing the ontology closer to an implementation context is *theoretically* expected; nevertheless, a deeper exploration of the DSVL and its implementation is needed in order to reveal the extent to which a REA-DSVL based tool can bring the ontology closer to information systems development context.

1.3 Research Question

Based on the problems described in the previous section, REA still faces a challenge in being widely implemented in the architecture of business information systems. The main goal of this research is to develop a REA-based DSVL. A typical DSVL consists of a meta-model and a visual notation; thus, these two artifacts should be developed as a first step. The developed DSVL is then to be implemented in a tool according to DSM specifications (LAFORCADE, Pierre, 2010). This effort is carried out in order to help answering the question:

"How would a REA-DSVL based tool make the REA ontology implementable in the domain of information systems development?"

1.4 Disposition

This report is structured as follows; the first chapter described the requirements and motives behind this research, and it defined the problems that this research is intending to solve. This chapter also presented the research question that can be answered by solving the identified problems.

Chapter two provides an extended background of different topics that are covered in this research. Then chapter three presents the methodology followed in this research and its rationale, and it provides the details of how each phase of the methodology was conducted.

Chapter four represents the results and analysis of the research according to the followed methodology. Chapter five then concludes the main results of this research, and finally, chapter five discusses the results and their analysis, and it provides an overview of the limitations of the research. Then it describes how a typical set of future works can be built on top of this research.

2. Extended Background

This chapter intends to discuss some of the topics and technologies which are used in this research. First, ontologies, modeling languages and meta-models are described and their relations are discussed. Then, Business modeling ontologies are presented, followed by a description of the REA ontology. After that, MDA and MDE are discussed, and then all the previous topics are discussed together in order to explain what this research is trying to do. After that, a brief description of one MDD framework is given, followed by a discussion of the related research in the domain of REA.

2.1 On ontologies, modeling languages, and meta-models

Ontology, as a concept, is a descendant from the branch of philosophy known as Metaphysics. Metaphysics in philosophy concerns the study of essences and origins of things, and it is divided into two branches; these are general metaphysics, and specific metaphysics. Ontologies, in philosophy, are the first branch of metaphysics, or the general metaphysics. Ontologies concern the study and investigation of general *concepts* of beings which, if known, can help in revealing the essence of beings (WAND, Yair, 1996) (SÁNCHEZ, Diana Marcela et al., 2007).

More recently, ontologies have been used in different research fields of computer science like; Artificial Intelligence (AI), Object Oriented paradigm, and Databases. In computer science ontologies, things that can be *represented* are considered *concepts* of ontologies. *Concepts* are the primary components of ontologies, and they reflect the *essence* of the things that they are trying to capture. Based on that, ontologies in computer science are built on top of two main concepts, these are; *Conceptualization* and *Representation*, where conceptualization refers to the process of creating concepts of things from the ontology's domain. Representation is the process of presenting these concepts in a *simple* manner, which one can use to communicate his/her understanding of different concepts and how they are related to each other. In computer science, representation is usually depicted using models (SÁNCHEZ, Diana Marcela et al., 2007).

Models are abstract representations of complex realities (MU, Liping et al., 2010). Meta-models are models which are used to build other models, or simply, models of models (MU, Liping et al., 2010) (GONZALEZ-PEREZ, Cesar and Henderson-Sellers, Brian, 2008). Models in computer science are usually produced using modeling languages. Modeling languages, as traditional languages, consist of three main parts. These parts are; an abstract syntax, at least one concrete syntax, and semantics (STAAB, Steffen et al., 2010) (MU, Liping et al., 2010). Abstract syntax is the collection of language constructs and how they are related, and it is usually represented by meta-models. The second part, the concrete syntax, can be of a visual or textual nature. The concrete syntax is a method for presenting the language constructs to the language users. Finally, semantics of a language assign meanings to the language constructs (STAAB, Steffen et al., 2010).

A special kind of modeling languages is referred to as Domain Specific Languages (DSLs). DSLs refers to modeling languages that cover a small bounded set of concepts, in a way that these concepts together serve functionalities related to a specific domain or discipline. DSLs are different from the Generic Programming Languages (GPLs), which are languages that are used to perform *several* functionalities regardless of the domain (VAN DEURSEN, Arie et al., 2000). HTML is an example of

DSL, while Java is an example of GPL. HTML is only used for building web pages, while Java can be used for building desktop applications, network programs, and HTML pages. DSLs which use graphical notations for its concrete syntaxes are referred to as Domain Specific *Visual* Languages (DSVLs) (SPRINKLE, Jonathan and Karsai, Gabor, 2004).

As seen from the previous discussion of modeling languages and ontologies, domain ontologies define the set of concepts that cover the constructs of a specific domain, and they rely on the user of the ontology to assign a representation of these concepts according to their understanding. On the other hand, modeling languages have more organized structure, and they provide a formal representation of the language constructs that is shared among all language users. This nature of ontologies makes them the main source of knowledge for a specific domain. Modeling languages, which are built on top of such ontologies, provide a formalization of the domains that they are built to support. In this sense, a typical ontology driven modeling language will take its constructs from the *concepts* of an ontology. These constructs are then used to build the modeling language's *abstract* and *concrete* syntaxes. The modeling language would also provide a set of semantics that convey a common and a clear understanding of the language constructs; thus, the relation between ontologies and modeling languages can be seen as of a complementary nature (GUIZZARDI, Giancarlo, 2006).

This research, in one of its steps, aims to build a domain specific visual language (DSVL) based on a business modeling ontology, namely the REA ontology. The resulted modeling language will then be used as the core language for an MDE based implementation. Business modeling ontologies and Modeling Driven Engineering (MDE) will be discussed in the following sections.

2.2 Business modeling ontologies

Ontologies in computer science are categorized into three main categories according to their generality. Generality in this context means the collection of concepts that ontologies cover. These categories are; top-level ontologies, domain and task ontologies, and application ontologies (SÁNCHEZ, Diana Marcela et al., 2007). Top-level ontologies cover concepts which are not limited to a specific domain or problem (ex: time, space, distance…etc). Domain and task ontologies cover concepts from a specific domain, and at the same time they are not limited to a specific implementation of that domain. For example, business domain ontologies cover general concepts of businesses (ex: actors, roles, resources…etc) without relying on the exact nature of the business (ex: financial business, engineering business…etc). Finally, application ontologies cover concepts which are associated with a specific domain or a problem (ex: engineering management, engineering finance…etc). This research focuses on one of the ontologies which are associated with the domain of enterprises business modeling. Such ontologies are referred to as business modeling ontologies (GAILLY, Frederik and Poels, Geert, 2007).

Today, several business modeling ontologies are available. The most well-accepted and used ones among these ontologies are e3-value, Resource-Event-Agent (REA), and Business Model Ontology (BMO) (SCHUSTER, R. and Motal, T., 2009) (SONNENBERG, C. et al., 2011 a) (SONNENBERG, C. et al., 2011 b) (GAILLY, Frederik and Poels, Geert, 2007). Each of the aforementioned ontologies managed to provide a specific value to the domain of business modeling, and each of them had quite sufficient amount of related research built on top of it, either to extend the ontology, or to relate it to other domains of enterprise modeling.

Business Model Ontology (BMO) is a business ontology based on the analysis of different academic and industrial endeavors, which has resulted in an ontology that relies on four major business pillars. BMO's business pillars are; the product, the customer interface, the infrastructure, and the financial aspects of businesses (CORALLO, Angelo et al.). The foundation of BMO takes the internal perspective of businesses. It aims to model the value proposition of the business and how to make profit out of it; though, it does not directly relate the modeled business to the business network around it (SCHUSTER, R. and Motal, T., 2009).

E3-value (GORDIJN, Jaap, 2002) is another business modeling ontology which supports the modeling of networks between enterprises that exchange resources (values) among themselves. The basic constructs of e3-value represent the participants in a business network, the economic values that are exchanged between these participants, and the nature and direction of exchange (GORDIJN, J. et al., 2006). Unlike BMO and REA, E3-value has a modeling language built on top of it. Moreover, the e3-value based modeling language is supported within a tool that can be found on the original website of the ontology[1].

The third major business modeling ontology is the Resource-Event-Agent (REA) ontology. As its name suggests, the main objective of REA models is to *mainly* represent the resources, agents (actors), and events that are associated with the business of a specific enterprise. REA takes the perspective of the enterprise that is being modeled, rather than a holistic view of the business-network like in the case of e3-value (SCHUSTER, R. and Motal, T., 2009) (ZHANG, Guoqiang et al., 2010). REA ontology was also extended to cover a wider set of concepts related to business environments. Such extension helps in modeling the details of what *should be* handled in the business rather than just what is *being* handled (GEERTS, Guido L. and McCarthy, William E., 2002) (GEERTS, Guido L. and McCarthy, William E., 2006). Extended REA covers concepts like contracts and commitments, which are missing in other business modeling ontologies.

Researchers in the domain of business modeling gave REA the upper hand over e3-value and BMO when it comes to designing business information systems. BMO focuses on categorizing business aspects which are needed for the production and delivery of services, but it does not focus on conceptualizing the main elements of business environments (SONNENBERG, C. et al., 2011 a) (SONNENBERG, C. et al., 2011 b). BMO also focuses on the internal perspective of the business, without viewing the network around it; thus, leaving REA and e3-value as *better* candidates for modeling a wider range of business effective elements. (SCHUSTER, R. and Motal, T., 2009).

Some researchers placed e3-value in the same position as BMO. Due to the holistic approach of e3-value, it can be considered as a good reference for the managerial perspective to have an overview of the modeled business *network*. BMO, on the other hand, is suitable for the *internal* managerial perspective of the modeled business (GAILLY, Frederik and Poels, Geert, 2007).

Compared to e3-value and BMO, REA provides additional levels of details which can help in forming conceptual models of both the internal and external business processes and environments (GAILLY, Frederik and Poels, Geert, 2007) (SCHUSTER, R. and Motal, T., 2009). Because REA's foundation is based on real concepts from the accounting theory, REA business models are useful when developing accounting information systems (GAILLY, Frederik and Poels, Geert, 2007).

Due to its suitability for developing information systems, this research takes REA as the foundation ontology for building a modeling language, which in its turn, will be used as the base modeling

[1] *http://e3value.few.vu.nl/tools/*

language for an MDE implementation. A detailed explanation of REA will be presented in the next section.

2.3 The Resource Event Agent (REA) ontology

Resource-Event-Agent (REA) is a business modeling ontology that was initially created as an accountability framework for building accounting information systems. The framework was then extended a number of times to cover more general aspects of enterprise business modeling; thus, making the framework suitable to be viewed as a business domain ontology (GEERTS, Guido L. and McCarthy, William E., 2002).

According to the progressive nature of REA's creation, the ontology has two models that represent its overall concepts space; these are the basic and extended REA models. The basic REA model is based on REA's first accountability framework, and the extended REA model represents the additional enterprise business domain concepts (GEERTS, Guido L. and McCarthy, William E., 2002).

In REA's view of business environments, any business activity is built on top of three main pillars, these are; *Resources*, *Events*, and *Agents* (thus the name REA). Agents are entities (individuals, organizations, companies…etc) that participate in the business-related processes. These agents can be within or outside the modeled enterprise or business. Resources are things of value that are being exchanged or produced in the course of the running business. Events are the activities which lead to the interaction between different *Agents* in order to exchange of produce *Resources*. When all the previous components of REA interact with each other, they form what is known as the "business value chain" (GEERTS, Guido L. and McCarthy, William E., 2002) (HRUBY, Pavel et al., 2005). ***Figure 1*** shows the basic REA model.

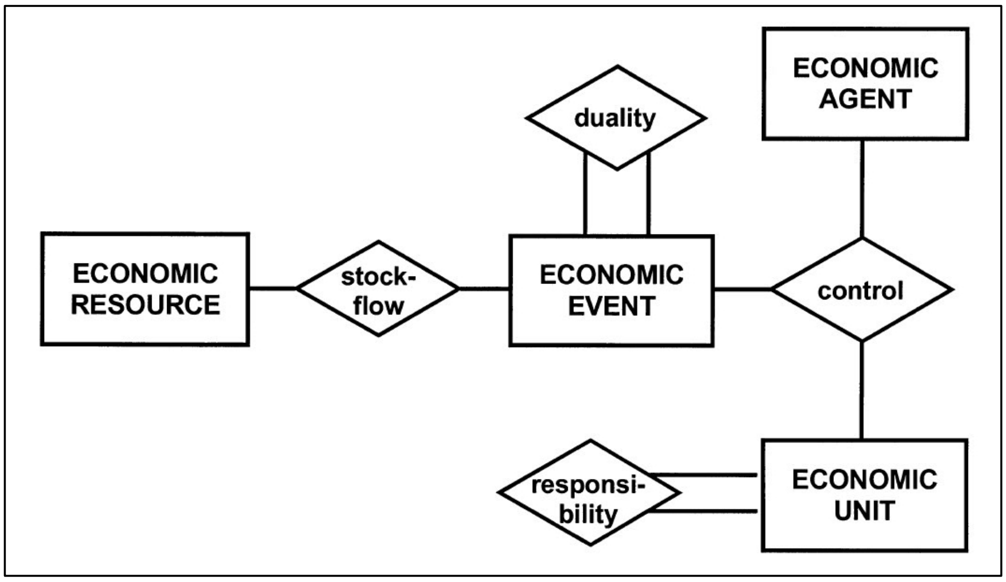


Figure 1: The basic REA model as it appears in (GEERTS, Guido L. and McCarthy, William E., 2002)

Agents in REA can be internal or external agents depending on the perspective that is taken while developing REA models. REA models are based on the internal perspective of the modeled business, which means that any agent that works for the modeled business (or enterprise) is considered an internal agent, while other participants are considered external agents (GEERTS, Guido L. and McCarthy, William E., 2002).

REA has two main types of economic events, these are; *exchange* and *conversion* events. Exchange events represent business tasks at which different resources are being exchanged between internal and external agents. Conversion events represent the tasks which lead to the creation of new resources from other resources, and they are usually executed by internal agents. Depending on the perspective of the modeled business, each of the events types can be either an increment event or a decrement event. Increment events are events that *add* resources to the business, and decrement events are events that *remove* resources out of the business's custody. In REA, all events (exchange and conversion) occur in combination of at least one increment and one decrement events. Such relation between events is referred to as *Duality* relationship (HRUBY, Pavel et al., 2005). Each *exchange* event must have exactly one internal agent and one external agent who exchange resources. Conversion events on the other hand, must have at least one internal agent who performs the conversion. Both exchange and conversion events operate on exactly one resource at a time (GEERTS, Guido L. and McCarthy, William E., 2002).

A sale operation is a typical example of exchange events. In a sale operation, a product (resource) is given to the product buyer (external agent) by the cashier (internal agent). In return, the product buyer gives money (other resource) back to the cashier. As noticed in the previous example, the task at which the cashier gives away the product is a typical example of a *decrement exchange event*. The event is considered decrement because a resource, the product in this case, is being removed out of the cashier's custody. In the same sense, the task of receiving cash from the buyer is an example of an *increment exchange event* because money is added to the cashier's custody.

As mentioned earlier, REA has an extended model that covers additional concepts of business environments. The extended model of REA adds *Commitments* as a new ontological concept (GEERTS, Guido L. and McCarthy, William E., 2002) . Commitments are *promises* of performing economic events in the future (HRUBY, Pavel et al., 2005). In the previous sale example, if the cashier arranges a delivery of the product to the buyer's house, or the buyer paid for the product using a credit-card, then these promises of performing events are modeled as commitments. The previous examples are modeled as commitments because the delivery of the *physical* product will take place at some point in the future, as well as the payment through the credit-card. In the latter case, the money will be *actually* collected by the cashier after a period of time, and not at the time of performing the credit-card payment.

Because they rely directly on economic events, commitments have the same categorization of economic events. Depending on the economic events that they promise to fulfill, commitments can be either exchange or conversion commitments, and for each type they are either increment or decrement commitments (HRUBY, Pavel et al., 2005).

In addition to Commitments, the authors of REA identified *Contracts* and *Schedules* as means to aggregate exchange and conversion commitments on the policy level (GEERTS, Guido L. and McCarthy, William E., 2000). Contracts are associated with exchange commitments, and they usually contain a set of *Terms* that define the conditions which lead to the execution of one or more commitments. Schedules are associated with conversion commitments, and they hold the same rules as for contracts (HRUBY, Pavel et al., 2005). The policy level specifications and the process level view are two more extensions of REA that were not extensively researched.

This research aims to build a modeling language for the REA ontology. The modeling language will cover the concepts from the basic and extended models. The ultimate purpose of the modeling language is to be used in a model driven development context; thus, bringing REA closer to its software implementation purposes. Model Driven Development will be discussed in the next section.

2.4 Model Driven Architecture (MDA) and Model Driven Engineering (MDE)

Model Driven Development (MDD) is a continuation to the researchers' effort in providing higher levels of abstraction to the process of software development. Software programmers used to write the details of how both the system and software should react to deliver the functionalities required from the software. With the advancements of programming languages and the introduction of compilers, software programmers focused more on the software part, leaving the details of systems manipulation to the compilers. More recently, the move toward incorporating models in the process of writing the business logic of software systems became an aspiration to software engineers due to the new challenges that came to surface with the usage of object oriented languages. The process of incorporating models in the process of software development is referred to as Model Driven Development (MDD). The most distinguished effort in the domain of MDD is realized in the goals and aims of what is known as the Model Driven Architecture (MDA) (ATKINSON, Colin and Kühne, Thomas, 2003).

Model Driven Architecture (MDA) is a standard that was developed by the Object Management Group (OMG) to facilitate an architecture that is used *when* incorporating models in the process of software development. MDA describes the types of models which are needed to successfully deliver model-based software, and it describes how these models are related to each other. MDA dictates that any model-based generated software should be viewed from three different viewpoints. The first viewpoint is the Computation Independent Viewpoint, which describes the business logic and the requirements of the software system. The models that depict this information are referred to as *Computation Independent Models* (CIM). The second viewpoint is the Platform Independent Viewpoint, which describes the structure of the software (like the framework used, the scheduling techniques used…etc) without mentioning the technical specifications of the platform that the system should operate on. The models which are associated with this viewpoint are called *Platform Independent Models* (PIM). The third and final viewpoint is the Platform Specific Viewpoint, and as the name suggests, this view point is associated with the view of the system from the perspective of a specific platform. The models which are used for this purpose are referred to as *Platform Specific Models* (PSM) (OMG, 2003).

In order to achieve the goals of MDD, the OMG has identified an infrastructure that facilitates the structure of modeling languages which are used to build the models needed for MDD (ATKINSON, Colin and Kühne, Thomas, 2003) (MU, Liping et al., 2010). The OMG's standard infrastructure, also known as the four-layer architecture, relies on technologies developed by the OMG. The first layer of this architecture, referred to as M3, is represented by an OMG standard know as the Meta-Object Facility (MOF). MOF is used to define the next layer of the architecture which is referred to as M2. M2 layer is represented by the famous Unified Modeling Language (UML). The UML is a method used to represent systems in an object oriented manner using graphical notations. The third layer (M1) of the architecture holds models which are developed using UML, and the final layer (M0) holds the user data which is used to model the third layer (ATKINSON, Colin and Kühne, Thomas, 2003) (MU, Liping et al., 2010).

As it was mentioned earlier, MDE as a generalization of MDA takes the four-layer architecture to contexts outside the boundaries of the technologies used in MDA (STAAB, Steffen et al., 2010). MDE views the four-layer architecture as a sequence of; meta-meta-model, meta-model, model, and user data that correspond to the layers described earlier M3, M2, M1, and M0 respectively. Such view of the four-layer architecture makes it easier to assign different technologies to the meta-meta-model and meta-model layers. The authors in (MU, Liping et al., 2010) have identified five different technologies that can be applied to these layers; thus, showing that this structure can be useful for different MDE implementations.

As it was described earlier, MDE is based on two main concepts; these are modeling languages and model transformation (STAAB, Steffen et al., 2010). Modeling languages were discussed earlier in section **2.1**. Model transformation simply refers to the process of mapping the abstract syntax (meta-model) of one modeling language to an abstract syntax of another modeling language. A pre-requisite for model transformation is that both abstract syntaxes should be developed using the same meta-meta-model (STAAB, Steffen et al., 2010). As MDE is a generalization of MDA, then modeling languages which aim at implementing MDE should support model transformation according to MDA specifications (OMG, 2003).

This research aims to build a modeling language based on the four-layer architecture that was described earlier. The technology that will be used for building the language is based on Eclipse’s Modeling Project. Eclipse Modeling Project is a collection of frameworks, tools, and standards which aim at implementing MDA practically (ECLIPSE, 2012). The core framework of Eclipse’s modeling project is the Eclipse Graphical Modeling Framework (GMF) which is a framework accompanied with a toolset that provide the ability to generate; tools, Java code, and other applications based on *Ecore* meta-models. Ecore serves the same purpose as UML. The Eclipse Modeling Project is the project that will be used for building the REA-based modeling language in this research.

2.5 A REA-Based DSVL

The previous structure of *modeling languages* that was described in section **2.1** is the traditional structure of *languages* in general. (MU, Liping et al., 2010) has identified the same structure, and mentioned that this structure can be viewed as consisting of three main parts, these are structure, behavior, and presentation (MU, Liping et al., 2010). ***Figure 2*** shows the structure of modeling languages according to Liping’s view. The main section of a modeling language is the structure (the abstract syntax), which contains the main concepts of the language and the relations between these concepts. In the case of REA ontology, this section contains the constructs described in section **2.3** (economic resources, agents, events, commitments …etc). The second section of a modeling language is the presentation (The concrete syntax). This section contains the parts which represent the concepts from the structure. Presentation can be of visual figures, texts, tables…etc. As for the REA ontology, there is no official visual representation of the ontology; thus, a visual notation should to be developed for the ontology in order for it to be considered as a DSVL. The final part of a modeling language is the behavior (*including* semantics). This section describes how the dynamics of a language are executed. Dynamics in this context refers to how the components of the abstract syntax are linked to the visual notation components (semantics). In the case of MDE languages, the behavior section also describes how models represented by one language can be transformed to other languages.

Liping’s view of modeling languages provide a suitable representation of languages that are used for MDE based implementations rather than the conventional languages structure. This structure details the presentation as being of textual or graphical nature, and most importantly, it defines transformation which is an important characteristic of MDD modeling languages.

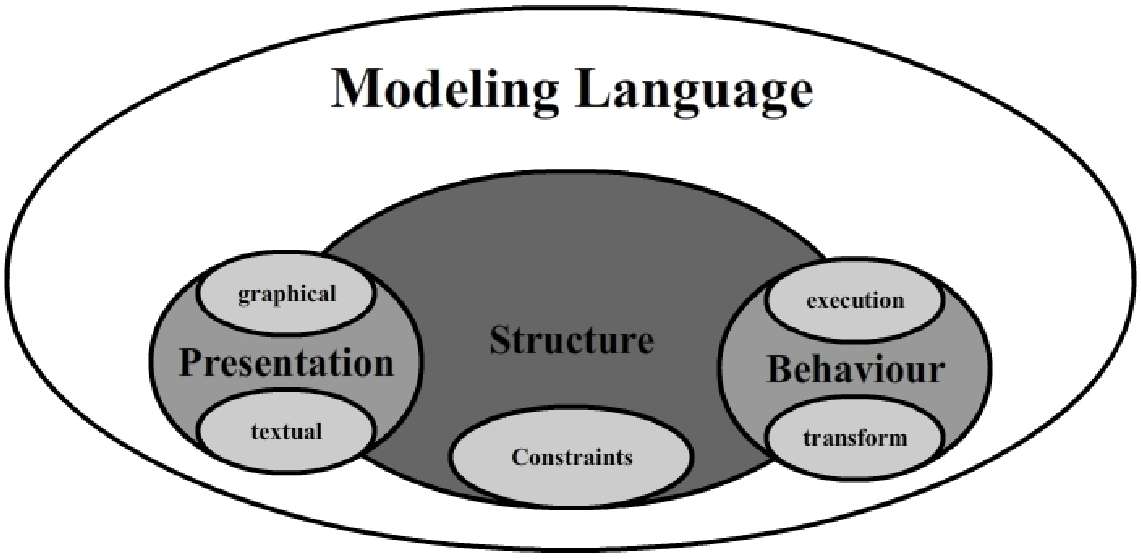


Figure 2: The aspects of a modeling language

Consequently, in order to build a DSVL which conveys the structure described in ***Figure 2***, a meta-meta-model, Ecore in this case, is used for building the meta-model (the structure or the abstract syntax) of a REA based DSVL. As the language to be produced by this research is a DSVL, then a visual notation is required. Eclipse GMF provides mechanisms for developing such a notation which will be used as the concrete syntax for this language. The behavior section has two parts, model transformation which is supported by GMF, and execution, which refers to the linkage of the visual components and the meta-model. This operation is supported by GMF, and it will be discussed in the next section. Models transformation will not be discussed in this research since the REA-DSVL will not be integrated with other languages.

2.6 Eclipse’s Graphical Modeling Framework (GMF)

This section includes technical information needed for understanding some of the technical sections of this report. The information presented in this section is based on the tutorial presented in (ECLIPSE.ORG), and it assumes that the reader had followed the first few steps of the tutorial and had created a new GMF project.

When a new project is created, a GMF project provides a dashboard that describes the relations between GMF components. ***Figure 3*** shows the dashboard associated with a GMF project. As it appears in ***Figure 3***, the GMF consists of six files, these are; Domain Model, Domain Gen Model, Graphical Def Model, Tooling Def Model, Mapping Model, and Diagram Editor Gen Model. ***Table 1*** defines the main rules of each of these files.

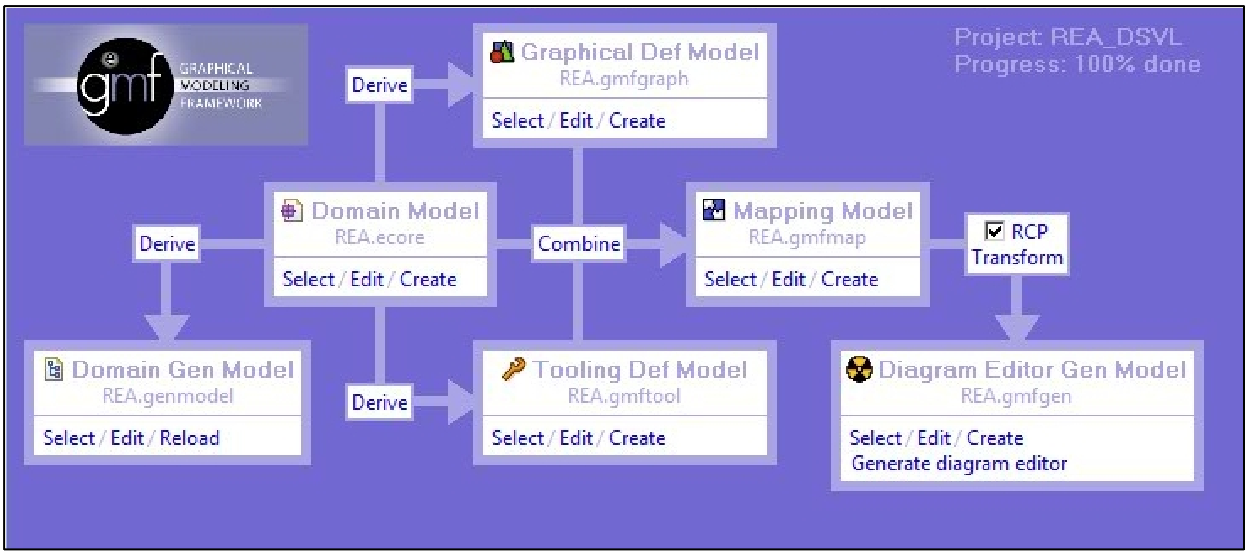


Figure 3: GMF Dashboard

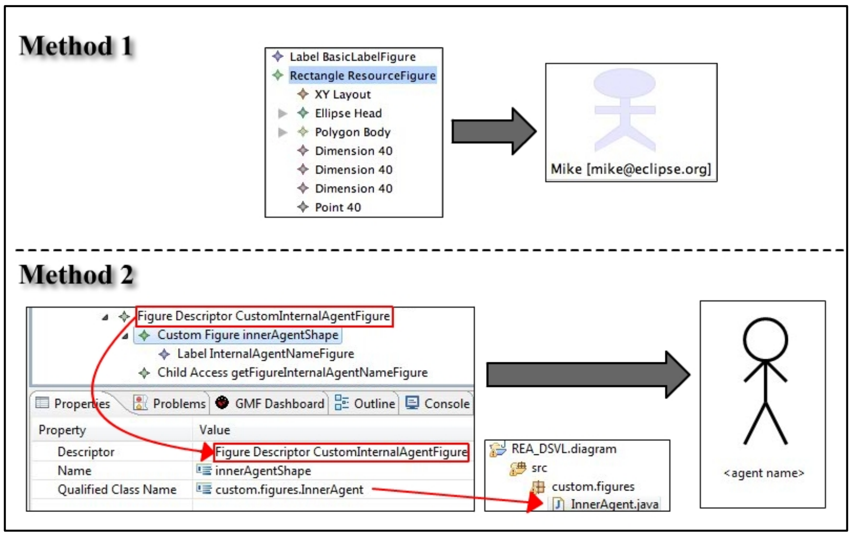
File	Purpose
Domain Model	Contains the meta-model Ecore file.
Diagram Gen Model	An auto-generated file based on the meta-model Ecore file. This file is responsible for creating the basic editor environment in the form of Eclipse plug-in.
Graphical Def Model	Contains a file that links the elements of the meta- model with their default or user-defined graphical definitions.
Tooling Def Model	Contains the file that controls the layout of the generated modeling environment.
Mapping Model	Contains the file that links the graphical definitions and the tool functions that will create these elements. Mapping Model is also responsible for defining additional validation rules, and for preparing the environment to be finally released for generation
Diagram Editor Gen Model	is responsible for generating the modeling environment (the editor)

Table 1: GMF Dashboard elements

As it appears in the dashboard, users should prepare their Ecore conceptual model as a first step, and then they can use the “Drive” boxes to generate next files. All of the newly generated files (except for the Domain Gen Model) need to be customized according to the users’ needs. Any change in the domain model requires regenerating the rest of the files. Any change in; Graphical Def Model, Tooling Def Model, or Mapping Model; requires regenerating the Diagram Editor Gen Model. Thus, users should choose wisely when to move ahead when deriving the next set of files.

Graphical Def Model is responsible for defining shapes of the domain elements. There are two options for implementing the graphical concept in GMF. One of these options is to follow the tutorial’s method. This method can be found in the second section of (ECLIPSE.ORG). The tutorial suggested building custom elements using predefined geometric objects. The definition of these objects is placed directly in the Graphical Def Model file. Geometric objects definitions are represented by sets of points, each of these points has X and Y locations on an XY plane. The XY plane represents the screen place that will be occupied by the figure.

Another option for defining graphical shapes in GMF is to use custom java classes for each element. These classes implement the Eclipse Draw2d API for building 2D graphics. Using this API, custom classes extend the super class “org.eclipse.draw2d.Shape”. This class defines two methods for drawing objects, one of them is responsible for filling the objects, and the other is responsible for drawing the outline of objects. After writing these classes, the user maps them to their corresponding definition in the Graphical Def Model file. ***Figure 4*** shows the two methods and their implementation results.



Property	Value
Descriptor	Figure Descriptor CustomInternalAgentFigure
Name	innerAgentShape
Qualified Class Name	custom.figures.InnerAgent

Figure 4: Graphical implementation methods in GMF

The GMF provides two options for generating editors. The first is to produce an Eclipse plugin that gives the Eclipse-IDE users an option for creating a modeling file. If this option is used, users will have the full Eclipse interface with all menu items. Typical Eclipse menu items include options for building java files, debugging…etc. If the user chooses to create a new domain diagram file in this case, the user will have a modeling environment within Eclipse which includes all the defined domain visual elements.

The other option is to generate the editor as a “Rich Client Platform” (RCP). If this method is used, users will have a single option for creating their domain diagrams. The interface in this case will contain only the menu items necessary for modeling. This option is very concise, yet serves the purpose of the modeling tool.

2.7 Related work

The effort to produce a meta-level support for REA is not new. McCarthy (GEERTS, Guido L. and McCarthy, William E., 2000) started this effort by extending the conceptual model of REA to contain more generic constructs than the ones found in his first model. This effort targeted the implementation of REA in the design and development of accounting information systems. Efforts continued to provide different UML profiles, web-based profiles, and XML representations of the ontology, nevertheless; efforts to represent REA as a DSL or a DSVL were limited to only one work (SONNENBERG, C. et al., 2011 a).

(SONNENBERG, C. et al., 2011 a) managed to produce a domain-specific-language based on REA (REA-DSL); nevertheless, some limitations and drawbacks were identified on different aspects of the produced language, for example, according to (SONNENBERG, C. et al., 2011 a), a duality in REA connects *stock-flows* together. According to (HRUBY, Pavel et al., 2005) though, dualities connect *events* together, and stock-flows have no means to be directly connected. The latter opinion was also supported by McCarthy in (GEERTS, Guido L. and McCarthy, William E., 2002).

Another example of the flaws found in (SONNENBERG, C. et al., 2011 a) was its proposal of new patterns to solve problems which were already solved. In their work, the authors mentioned that events of the same nature might occur over different periods of time, and they gave the example of paying for goods with partial payments. For that purpose, they have suggested "economic event series" to cover the collection of such events. Greets and McCarthy, on the other hand, had presented the elements "commitment", "term", and "contract" to solve this exact issue (GEERTS, Guido L. and McCarthy, William E., 2000), and these solutions are part of the original REA definition. On top of that, the solution covered only the basic model of REA. The extended model was considered a future work of that research.

Although the work of (SONNENBERG, C. et al., 2011 a) had some flaws, it had influenced the use of DSLs as a method for solving the identified REA problems. Literature about DSLs was reviewed in order to gain a better insight of the term and its scope. The authors of (VAN DEURSEN, Arie et al., 2000) provided a description and examples of the DSL terminology. An adequate review of meta-models, and their relations to DSLs was provided in (MU, Liping et al., 2010). The latter paper in general discussed the four layered architecture of MDA. It has provided several examples of the architecture's implementations. The idea of linking a meta-model to a visual notation was presented in this paper as well, and this idea has inspired the architecture of the DSL to be developed in this research.

Up to this point, the term DSL was used to describe the modeling language to be developed, though, after reviewing some resources like (LI, Karen et al., 2010), the term DSVL was used instead. DSVLs provided both a closer representation of the language to be developed, and it provided a different "language name" than the one used by (SONNENBERG, C. et al., 2011 a). The latter resource used the phrase "REA-DSL" to describe the language that was produced in that work, thus, a different name was needed to refer to the language to be developed in this research.

The work of Gailly and Poels in (GAILLY, Frederik and Poels, Geert, 2005) and (GAILLY, Frederik and Poels, Geert, 2007) managed to provide two different meta-models for REA. Their first work suggested a formal representation of the ontology based on OWL, and the second one suggested a UML-based meta-model which improves the overall ontology. It was obvious from the conclusion that the authors have drawn on their second paper that their first meta-model was not suitable for viewing REA as a business *modeling* ontology, but rather as a business *domain* ontology; thus, the discussion of their first meta-model can be discarded. Although the authors have claimed that their second meta-model has covered detailed aspects of REA and based on that it can be used for modeling business, they have missed a vital part of distinction, which is the separation between the two kinds of events and commitments. The authors divided their event meta-class into two sub-classes, increment-event and decrement-event meta-classes. At the same time, they did not do the same distinction for exchange and conversion events, which is considered of a higher priority because an exchange or conversion decides the type of the stock-flow to be used, and the type of the commitment to be linked to such events.

(ZHANG, Guoqiang et al., 2010) suggested a new business modeling framework that is based on REA and one management strategy framework. The framework was then implemented with OWL. This work had quite sufficient representations of REA's basic and extended models, though it did not contain any representation of their understanding of REA in the form a meta-models, but rather in the form of OWL models.

(SCHUSTER, R. and Motal, T., 2009) has presented an attempt to link e3-value models to REA, and for that purpose the authors have suggested some changes to the core representations of REA, which is outside the scope of this research,

Finally, (SONNENBERG, C. et al., 2011 b) has presented in this work a REA-based XML language. The authors of this work are the same authors of REA-DSL which was discussed earlier. This language is simply an XMl schema that represents the same REA logic that was presented in the author's first work; thus, the same flaws can be found in both places.

Away from the validity of the previous researches, it is noticeable that all of the previous attempts focused on one aspect of the identified problems from section **1.2** and neglected others. All the previously discussed researches did not consider the wider image of the problem, which is the implementation of REA in the domain of information systems development. Based on this fact, this research tries to bring the different solutions under one hood, and in addition to that, it provides a first attempt toward solving the wider problem of REA, that is bringing REA closer to its implementations purposes.

3. Methodology

Information systems (IS) research is a special kind of IT research, which in its folds concentrates on enterprises and how information is manipulated within them. In IS research; two methodologies are usually used to gain knowledge about the research domain; these are behavioral science and design science. While these two methodologies can be seen as running similar processes, behavioral science is usually associated with theories creation, and design science is associated with the creation and evaluation of artifacts (HEVNER, Alan R. et al., 2004). Based on the fact that this research aims to build something in order to solve a problem, then this research can be seen as an IS design science research.

Design science research can be conducted following different frameworks. According to (HEVNER, Alan R. and Chatterjee, Samir, 2010) there exists one widely accepted and distinguished design science methodology referred to as the *Design Science Research Methodology (DSRM)* (PEFFERS, Ken et al., 2007). This methodology consists mainly of six activities, these are:

1. Problem identification and motivation: Reasoning about the problem is presented along with possible solutions' justification in compliance with the problem roots.
2. Define the objectives for a solution: Identification of the solution objectives based on the nature of the problem. The objectives can be quantitative or qualitative, and they require knowledge of the state of the problem, and knowledge of other available solutions if any.
3. Design and Development: In this activity the identified objectives are implemented into a design; based on which the artifacts are developed.
4. Demonstration: Demonstrate that the artifact is capable of solving the whole or parts of the identified problems. Typical demonstration methods include case-studies, simulation, experimenting...etc.
5. Evaluation: Includes measuring the efficiency of the solution in solving the problem, this means checking whether the artifact succeeds in fulfilling the objectives from the second activity. The nature of the evaluation method depends on the nature of the objective, and it requires knowledge of the appropriate evaluation methods. If the evaluation showed that the artifact is not fully suited to solve the problem or parts of it; the researcher can go back to the third activity of the process, or proceed to the next activity and consider any further improvements as sub-projects of the current one.
6. Communication: Communicate all aspects of the research to the interested community. This phase will be omitted from this report, as this research is a master's degree thesis; thus, the report itself is a mean of communicating its content to the targeted audience.

This research follows the process identified by *DSRM*. This chapter covers the grounding of the methods choice used within each step of DSRM, and how such methods were implemented to answer the question of this research.

3.1 Choice of research method

Following the same structure of DSRM; this section describes the methods choice in each step of this research and the rationale behind such choices.

3.1.1 Problem identification and motivation

This research revolves around the REA ontology; thus, this activity was focused on finding the body of knowledge around REA and how well it was realized in the real world. It also tried to find the extent of REA's practical implementation in the field of information systems development. For this purpose; literature review was the natural method of choice for commencing the study. Other observatory methods did not fit in this activity because REA is a typical theoretical ontology which can be best sought in literature.

3.1.2 Define the objectives for a solution

At this point, the problems related to REA were identified, and their boundaries were drawn. Based on the identified problems, an initial set of goals for this research was set. In order to find how similar goals were approached, literature review was conducted.

Literature review was conducted as it was the main source used in identifying the problems. Some resources have identified similar or close problems, and these resources have used methods for attacking such problems; thus, it was natural to review these literature resources.

Internet searching was also conducted at this phase in order find software tools for building the DSVL. This activity included searching for commercial, open-source, and educational tools. Then a comparison between these tools was performed in order to find the best tool which fits the DSVL-development purpose.

3.1.3 Design and Development

The development process of REA DSVL went through four main successive-recurring phases. As depicted in ***Figure 5***, these phases were; *meta-model development*, *visual-notation development*, *visual notation implementation*, and *editor generation*. These phases were successive in their flow in a way that each phase had to be finished before the next one gets initiated. The phases were recurrent in case of errors or design improvements. If an error appeared in one phase, the problematic phase was reconducted, and then its following phases were reconducted in the same original successive order. This section describes the method choice in each of these phases.

Meta-Model Development

Two main options were considered for the construction of the meta-model. The first option was to use an existing meta-model of the ontology from a previous work, and then to implement it in the chosen development environment. The second option was to create the meta-model from scratch. The first option proved problematic in several ways; first, the understanding of REA is different from one researcher to another; thus, the meta-models produced by each researcher would be different from other researchers. Another problem was an ethical one, as the generated artifacts of this research were to be published online, and the original owners of the meta-models might have concerns regarding publishing their work under different projects. Thus, the second option seemed more appropriate.

A framework for the developing meta-models for DSLs was described in (SCHAFER, Christian et al., 2011). This paper managed to accumulate best practices from other frameworks targeting the same purpose. As this research tries to build a DSVL, the aforementioned framework was a suitable guideline for developing the abstract syntax of the DSVL; thus, some of the guidelines presented in that framework were implemented in building the meta-model of this DSVL.

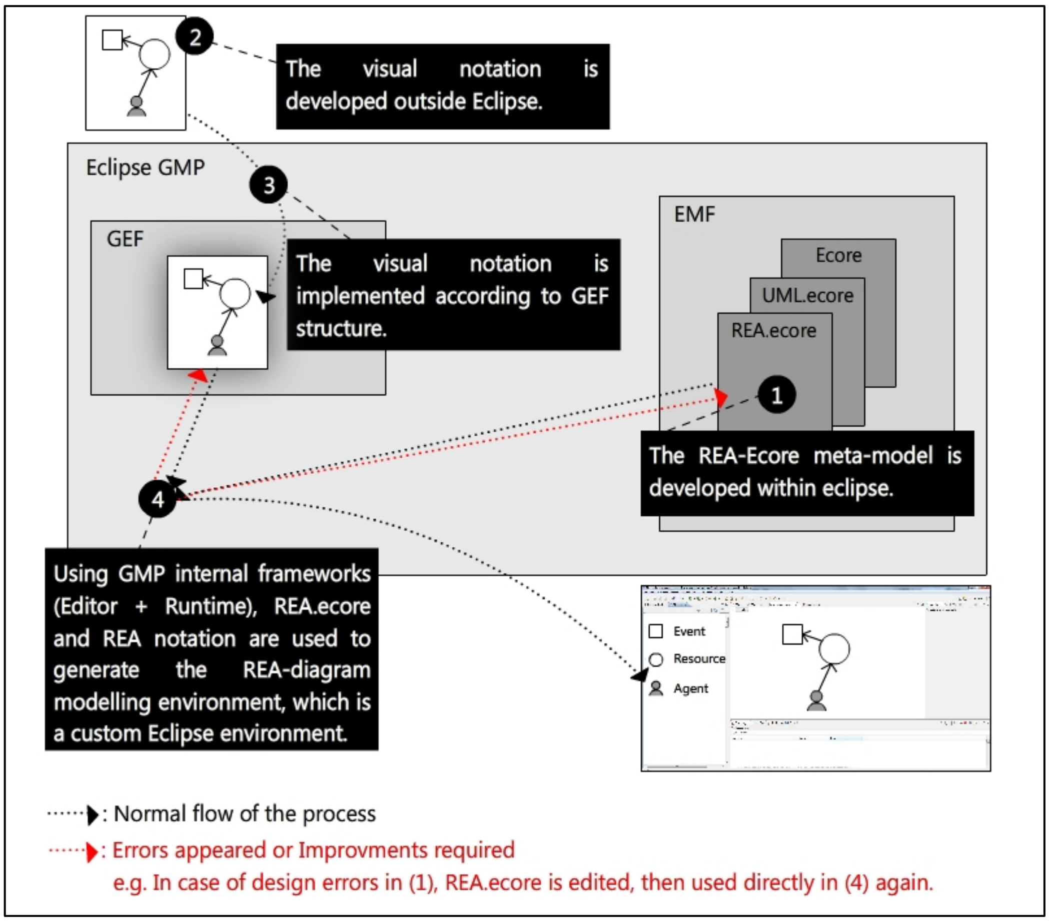

2
The visual notation is developed outside Eclipse.
Eclipse GMP
3
GEF
The visual notation is implemented according to GEF structure.
EMF
Ecore
UML.ecore
REA.ecore
1
The REA-Ecore meta-model is developed within eclipse.
4
Using GMP internal frameworks (Editor + Runtime), REA.ecore and REA notation are used to generate the REA-diagram modelling environment, which is a custom Eclipse environment.
Event
Resource
Agent
: Normal flow of the process
: Errors appeared or Improvments required
e.g. In case of design errors in (1), REA.ecore is edited, then used directly in (4) again.

Figure 5: REA-DSVL development process

Visual-notation development

This part of the research was completely dependent on the researcher's imagination and sense of creativity. Some concerns were taken into account, like the influence of previously experienced modeling languages on the imagination of the notation designer. These concerns were also considered an opportunity to overcome some of the limitations found in other modeling languages.

Another option was to take the notation from other modeling languages, and use it for REA. This option is both non-ethical, and could cause confusion to the users of other languages, who might use the notation from the developed DSVL of this research.

In order to provide the best applicable notation, a set of design-concepts were developed based on the relations and elements of REA. These design-concepts were then implemented (as row graphics) using graphical modeling software in order to have a grip on the notations' graphical aspects (dimensions, position to screen…etc).

Visual notation implementation

As mentioned in section 2.6, GMF provides two methods for implementing graphical notations. As this section is discussing the choice of research methods, it is inevitable to discuss the technical options which were presented in section 2.6 because the implementation of REA-DSVL is based on Eclipse's GMF.

The choice between the two methods which were presented in 2.6 was clear and direct. The second option with the custom java classes was chosen for four main reasons. The first reason is the flexibility and reliability provided by the Draw2d API compared to the first method. Using the API, one can define shapes with the finest level of details. The API also provides a set of classes which allow the developer to control the look and feel of shapes with guaranteed results; that is, any logically written code will produce results in the rendered shapes. The first method, however, does not have a validation mechanism; thus, one can define as many attributes from the predefined set of attributes, and only the ones that apply to the parent shape will be rendered. This might result in a number of unnecessary attributes which are bypassed by the GMF engine.

The second reason for choosing the API method was that any change in the "Graphical Def Model" file requires deriving a new "Diagram Editor Gen Model" file. While developing the notation; one needs to check how any change on the graphical definition will be reflected on the modeling environment. If the first method was used, changes would have to be applied directly to the "Graphical Def Model" file; thus, a new "Diagram Editor Gen Model" file would have to be generated in order to test any change. In the case of the second method, changes were applied to the java class, not to the "Graphical Def Model" file. "Graphical Def Model" file in this case held only the references to classes' names and locations under the graphical shapes that they define. So using the second method saves time and effort when testing the graphical definition.

The third reason for choosing the second method was maintainability. For example, "Event" domain element was broken down into four subtypes, these elements share the same shape characteristics, except for the internal symbols that define events-types (exchange or conversion) and value-types (increment or decrement). If the first method was used, all graphical definitions would have needed to reproduce the same general event shape characteristics. They would have also needed to define the distinctive characteristics of exchange, conversion, increment, and decrement. In this case, if a change was to be applied to the main event shape, all four subtypes were needed to apply the same change to their definitions. On the other hand, when the second method was used, it was easy to define a super class that handled the graphical definition for the "Event" shape, and then defined four classes that inherited the super class. Using this method, if a change is required to the main event shape, only the super class will be updated.

While working with GMF editor; one can face difficulties in moving from one view to another, and in finding the attributes needed to configure different elements in each view. These difficulties become very clear when using the first method. The GMF itself is relatively new, and it is still undergoes continuous fixing and updating. The tutorial's author in section 2 of (ECLIPSE.ORG) expressed this while explaining methods to overcome the difficulties associated with using the first method. The second method has the advantage of stability in this context, because java classes have a robust and permanent structure. In the event of changing the future architecture of GMF, java classes will remain immutable. This indeed was another reason behind selecting the second method.

These were the main reasons behind choosing the programming option over the static one.

Editor generation

This section also discusses technical options from GMF. For the same reasons given in the previous section, technical information from section 2.6 will be discussed in this section. GMF provides two main methods for generating editors; these are RCP or Plugin generation.

The option with RCP was chosen for generating the REA-DSVL Editor. The tool to be developed is a business modeling tool, and some of its users might not have a technical background; thus, it was essential to provide them with a simple environment that was easy to operate. In addition to that, the core modeling functionalities are the same using the two previous options; thus, there was no need to include overhead functionalities which were not directly related to the purpose of the editor. The size of the final produced editor increases dramatically when the first option is used. The choice of the RCP option was mainly based on these factors.

3.1.4 Demonstration

This step of DSRM can be conducted using simulation, case-study, or experimentation (HEVNER, Alan R. and Chatterjee, Samir, 2010, p.30). A typical case study requires studying the subject of research in its natural environment over a period of time. Simulation is about mimicking reality using software systems, and experimentation is concerned with testing causes and effects (BHATTACHERJEE, Anol, 2012).

Experimentation was first chosen as an evaluation strategy rather than a demonstration strategy in this research; therefore, it was not performed at this phase. Simulation, on the other hand, was not a suitable choice for this task. Although the generated models by the developed tool would mimic the *structure* of the business environment, though they do not produce the same outcomes of the actual business, in other words, the produced models would not simulate the actual business.

A case study was chosen as the demonstration strategy of the developed artifact. This research provides a solution that can be best used to model running businesses; thus, a running business in Stockholm-Sweden was chosen to be modeled. It is necessary to point out that the case study was chosen as a demonstration strategy, and an interview within that case was conducted as the strategy's method. This methodological structure follows the one identified in (DENSCOMBE, Martyn, 2007).

The carried out case study did not expand over time as it would be expected by typical case studies; instead, it was a small case study performed using an unstructured one-to-one interview with the business manager. This methodology was sufficient enough for two main reasons; the first one was the small number of employees, which were seven. All business activities carried out by these seven employees were directly supervised by the business manager; thus, interviewing the business manager was sufficient enough to acquaint all performed business activities. The second reason was the purpose of the case study. According to (DENSCOMBE, Martyn, 2007, p.38), case studies can be performed for many reasons, one of which is to describe events, processes, and relationships carried out in the case study. The latter reason perfectly fits the purpose of this demonstrative task, as the used methodology was sufficient to gain an insight of the running business processes. Accordingly, other lengthy research methodologies, like observations for example, were not performed in this case.

The chosen case study was a small tires and rims trading company in Stockholm. The first factor behind choosing this company was the fact that REA targets the modeling of accounting business environments. Any running business with financial objectives would have accounting trends to measure profitability and unprofitability means in that business; thus, the main concern with finding an appropriate case study was the availability of such accounting trends, which were perfectly available in the chosen business. In this sense, this case study can be generalized to similar businesses regardless of their size or business-type, as long as these businesses have similar business profitability measures.

Another reason for choosing this company was its small business size, which at the same time, was adequate for covering the full functionalities of the developed artifact. The developed modeling tool is based on the developed visual notation and meta-model. A typical model designed by this tool, would use the visual notation for modeling, and simultaneously generate a data-models based on the original meta-model. Consequently, modeling one sufficient business process from the case study would be sufficient to demonstrate the full functionalities of the tool.

The small business size has also helped in formulating a better realization of the selected business boundaries. Larger businesses would have required longer interviews, observations, and analysis, which would have resulted in the same tool demonstration results.

Another important reason for choosing this company was the availability and willingness of the business owner to conduct this case study.

3.1.5 Evaluation

According to the structure of DSRM (HEVNER, Alan R. and Chatterjee, Samir, 2010, p.30), this activity revolves around analyzing the data that was observed during the demonstration activity, and then compare the results of the research to the original purpose of the artifacts. This activity dictates checking the artifacts against the gathered requirements. As the requirements of this research contain both quantitative and qualitative requirements, the quantitative requirements were evaluated directly according to their fulfillment, and the qualitative requirements were evaluated based on the techniques that will be presented in this section.

The meta-model’s completeness and structure is best judged by experts in the domain of REA and meta-modeling. Such an evaluation can be carried out by surveying the opinions or interviewing the mentioned experts. To find experts who were willing to review the meta-model seemed quite hard when compared to other evaluation options; thus, this evaluation was not performed using the former methods; rather, it was evaluated quantitatively based on the coverage of REA’s elements.

Experts in REA and visual composition were required to judge the visual notation’s quality in accordance to REA. As it seemed difficult to reach such opinions, this evaluative approach was not taken. Compared to the literature found on REA based meta-modeling, literature about ontological visual composition could not be found; thus, the qualitative requirements of the visual notation were evaluated based on its usage within the tool. The visual notation also had some quantitative features related to REA that has been evaluated quantitatively.

The developed tool represents the main artifact of this research due to its direct relation to the research question. As with the meta-model and the visual notation, the tool had a set of requirements associated with it; thus, the first step was to evaluate these requirements. After that, the main purpose of the tool and its relation to the research question was evaluated.

Experimentation was the first strategy chosen for evaluating the tool. Experiments target the measuring of causes and effects relationships. Causes are typically what the researcher wants to test, and effects are the desired results that the researcher wants to reach (BHATTACHERJEE, Anol, 2012, p.83). A typical experiment should include testing the subject of that experiment on treatment and control groups. A treatment group is an experimental group that uses the subject of the experiment (the cause) to perform predefined tasks. The control group is a group that performs the same tasks as the treatment group without using that subject of the experiment.

Unfortunately, an experiment was designed for evaluating the qualitative aspects of the tool, but only a partial set of the experimental groups responded to the experiment call; thus, another form of evaluation based on informed arguments was conducted for this task. Informed arguments are simply arguments which are based on the analysis of the context that these arguments are presented in. Such arguments should consider all aspects of their context, including the points that support them as well as the points that defy them (GOCSIK, Karen, 2005).

3.1.6 State of the art techniques and methods

In order to follow up the latest techniques and methods which were used for solving similar problems, the criteria for selecting literature resources included looking up recent literature that dealt with REA. This has revealed DSLs and DSVLs as means for building a REA-based modeling language. MDA and MDE in general are considered state of the art techniques for systems development, and they constitute the essence of this research.

3.1.7 Ethical and social considerations

Some of the ethical and social considerations which have been taken on the level of methods *choice* included the choice of developing the meta-model from scratch. This in turn, was based on the two factors; firstly, the previous suggested meta-models were analyzed objectively based on the reviewed REA resources, and when limitations were found, the choice for developing the meta-model was supported. Secondly, the meta-model is part of the modeling language and the editor which were planned to be available online; thus, including any artifact developed by the other authors needed their approval, and since this seemed to be a difficult task to accomplish, the idea was dropped entirely.

Ethical and social aspects were carefully considered during the implementations of the methods which were described in this sub-chapter. These considerations will be described at the end of the implementation discussion.

3.2 Application of research method

Following the same structure of DSRM; this section describes the methods used in each step of this research and how they were implemented.

3.2.1 Problem identification and motivation

Literature review was conducted on different areas related to enterprises and business modeling. In order to have a better understanding of the problem domain, many concepts were studied. These concepts included; REA ontology, meta-modeling, domain specific languages (DSL), domain specific visual languages (DSVL), Model Driven Architecture (MDA), Unified Modeling Language (UML), Meta Object Facility (MOF), Eclipse Modeling Project (EMP) and many others.

The criteria behind selecting the appropriate publication were based on three factors. The first criterion was the relevance of the paper. The second criterion was the year at which the resource was published. The newer the resource was the better its chances of being reviewed. The final criterion was the popularity (number of citations) of *relevant* papers.

These criteria proved easy for some of the mentioned topics like EMP, and in some cases, the latter criterion had to be dropped. This had to be done because some topics were slightly covered in literature, and so their citation count was low. This usually occurred when searching for composite topics (e.g. REA based DSL, REA + DSL, “REA” and “DSL”).

The sources used for locating literature were; KTH university library[1], IEEE Xplore Digital Library [2] Google scholar[3], Springer Link[4], ACM Digital library[5], Science Direct[6], and Google search engine. This research was held in KTH; thus, the first logical source to use was the KTH university library. When some of the needed concepts were looked up, the resulted relevant resources were most of the time found in IEEE Xplore, Springer, and ACM libraries. That was the main reason behind choosing the latter sources.

This research was conducted based on an initial task of building a meta-model for a business ontology. Topics like “business ontology” and “MDA” were provided with the task description. The first step was to gain an overview of these concepts. MDA was an easy topic to locate as it had a dedicated website (OMG, 2003). MDA was originally reviewed to have a better understanding of UML and MOF (OMG, 2011). Further reading into business ontologies and the problems associated with each of them revealed greater details which have finally lead to this research. Based on the reviewed resources, the problems which were presented in section 1.2 of this report were identified.

3.2.2 Define the objectives for a solution

Following the same approach of literature review that was described in the previous section, similar problems and their solutions were reviewed. Some aspects of the identified problems had clear and direct solutions; thus, the requirements for such aspects were set directly without further navigation.

[1] *http://www.kth.se/kthb*

[2] *http://ieeexplore.ieee.org*

[3] *http://scholar.google.se*

[4] *www.springerlink.com*

[5] *http://dl.acm.org*

[6] *www.sciencedirect.com*

Other problems' aspects left the door open to choose from different *technical* implementations to attack such concerns.

The problem of the formal representation of REA had a number of research papers which tackled the problem. These papers were reviewed in order to check how the problem was attacked, and if possible, check the developed meta-models. As this research is trying to bring the REA ontology to its implementation purposes, a formal representation had to be combined with a technical implementation. For this purpose, the literature was searched for terms like "MDE, MDD, meta-modeling, DSL, and REA".

The problem of the missing visual notation had quite a direct solution which was to design a visual notation. Literature and internet searching for modeling tips were conducted, though no significant results were found; thus, standard requirements were formulated for the notation.

Because there were attempts to solve some of the identified problems, the limitations of the previous solutions were narrowed down, and the original REA papers were reviewed in order to have a better understanding of what was missing in the previously suggested solutions. After the limitations were lined out, some requirements were formulated in order to overcome the limitations identified in other solutions.

The requirements for the tool to be developed were formulated based mainly on the functionalities provided by the platform which was used for developing the DSVL. The developed tool in this research can be considered a *Proof-of-Concept* tool rather than an actual commercial tool; thus, the tool requirements were not strict as if they were based on well-known software standards like ISO/IEC 9126-1:2001[1]. For the previous reason, the requirements of the tool were lightly formulated, and it focused more on its main purpose of being an MDD tool.

3.2.3 Design and Development

Meta-Model development

As mentioned earlier in this research, the Eclipse's Graphical Modeling Project (GMP) was chosen for the development of this modeling language. GMP uses Ecore as the core language for the development of its domain models; thus, in REA-DSVL, Ecore was the language used for building the meta-model of the language. The practical development process of the meta-model, including installing eclipse and the required plugins, is typical to the process in the first tutorial of (ECLIPSE.ORG). This section describes the steps that follow the installation and the creation of a new Ecore file.

When the design of the domain model started, the basic REA model was taken as the first input for building the entire meta-model. Some guidelines were taken from (SCHAFER, Christian et al., 2011) while designing REA-DSVL's meta-model. One of the guidelines was to break down the domain elements into principal components. The main components of the basic REA model are; Events, Agents, and Resources. Other components associated with the *extended REA* model are; Commitments. All the mentioned REA elements were to be modeled in the final meta-model, in addition to some additional elements from the policy level as Terms, Resource-Type, and Contracts.

Elements of the basic REA model alone were not sufficient to build a prototypical domain model. ”Events” had to be broken down into ”Exchange Events (EE)” and ”Conversion Events (CE)”; simply

[1] *http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=22749*

because these two kinds of REA events represent different kinds of events, and they have different kinds of relations with other REA elements. ”Resources” and ”Agents” were considered basic elements at this stage, so they were not broken down further. Exchange and Conversion events were broken down further into a third level; the new level represented the incremental and decremental aspects of each event type. After the last step was done, events were represented in three levels as it appears in the upper section of ***Figure 6***.

According to the guidelines in (SCHAFER, Christian et al., 2011), after decomposing the elements of a domain model; generalization relationships are drawn between the elements. The next guideline was to remove any additional generalization relationships (including elements) that do not add further information to the meta-model. After applying the previous guidelines, ”Conversion Events” and ”Exchange Events” meta-classes were removed from the meta-model, and their children were directly connected to the main ”Event” meta-class. ***Figure 6*** shows the events design process.

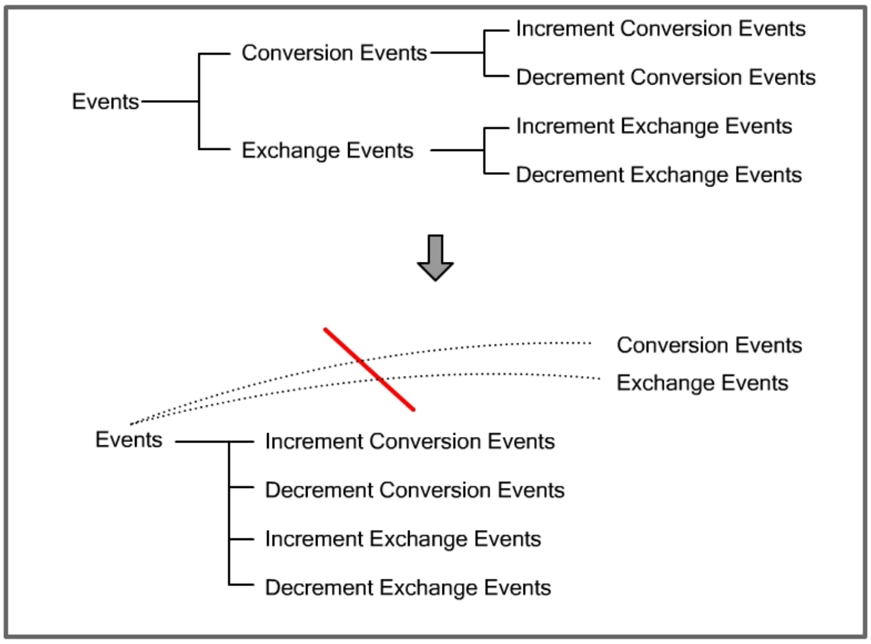

Events
Conversion Events
Increment Conversion Events
Decrement Conversion Events
Exchange Events
Increment Exchange Events
Decrement Exchange Events
Conversion Events
Exchange Events
Events
Increment Conversion Events
Decrement Conversion Events
Increment Exchange Events
Decrement Exchange Events

Figure 6: Events design process

Some aspects of REA could not be designed as features for events, resources, or agents meta-classes, but rather as separate domain elements. Such features included the relations connecting REA elements together, like; Events-Events relations (dualities), and Events-Resources relations (stock flows). Duality relationships connect multiple instances of events together; thus, they needed a dedicated model element to represent them. Events-Agents relationships were kept as default links because these relations are explicitly linking one instance of the “Event” class with one instance of the “Agent” class directly. Following the foregoing reasoning, relations between events and resources (stock flows) link an instance of an event class with another of a resources class, and so it should follow the same rule that was applied on events-agents relationships, rather than that, “stock flow” relations were given their own domain elements like “Dualities”, this in fact was the first error encountered while designing the meta-model.

To solve the problem encountered earlier, the class ”Stock flow” and its subclasses; ”Give”, ”Take”, ”Produce”, ”Use”, and ”Consume” were removed from the meta-model, and the connections between these classes and ”Event” subclasses were extended from ”Event” subclasses directly to the ”Resource” class directly. The removed subclasses (Give, Take, Produce, Use, and Consume) were replaced with links that have the same labels.

After fixing the basic meta- model, it was extended to cover REA’s extended model and some of REA’s policy level elements. These elements are; “Commitments”, “Resource Types”, “Terms”, and “Contracts”. As with events; commitments were decomposed into four sub-classes, these were; “Increment Exchange Commitment (IEC)”, “Decrement Exchange Commitment (DEC)”, “Increment Conversion Commitment (ICC)”, and “Decrement Conversion Commitments (DCC)”. Commitments were then associated with the appropriate events meta-classes. Similar to events’ dualities, increment and decrement commitments are associated via a “reciprocity” relationship. Reciprocities can link multiple instances of commitments together; thus, they had been given their own domain elements.

The rest of REA elements were added as meta-classes, and the relations between all the elements were set according to the original REA model. Cardinalities and relationships were then confirmed with the ones in (HRUBY, Pavel et al., 2005) for validity.

After the development of the meta-model, the development of the visual-notation started. This phase is described in the next section.

Visual-notation development

This phase entailed developing a set of design concepts which correspond to the set requirements which were suggested for the visual-notation. Three different design concepts of the visual notation were developed in order to provide a wider base of choices. These concepts were drafted out using a pen and a pad, and then they were implemented in Adobe Flash Professional CS5. Other graphical modeling tools could be used for the same purpose. The main idea behind using such tools is to provide a live experience of how the designed notation would feel in a computerized environment. This activity would give the designer a first impression of the notation that is being developed, and the ability to perform any improvements before implementing the design in the GMF.

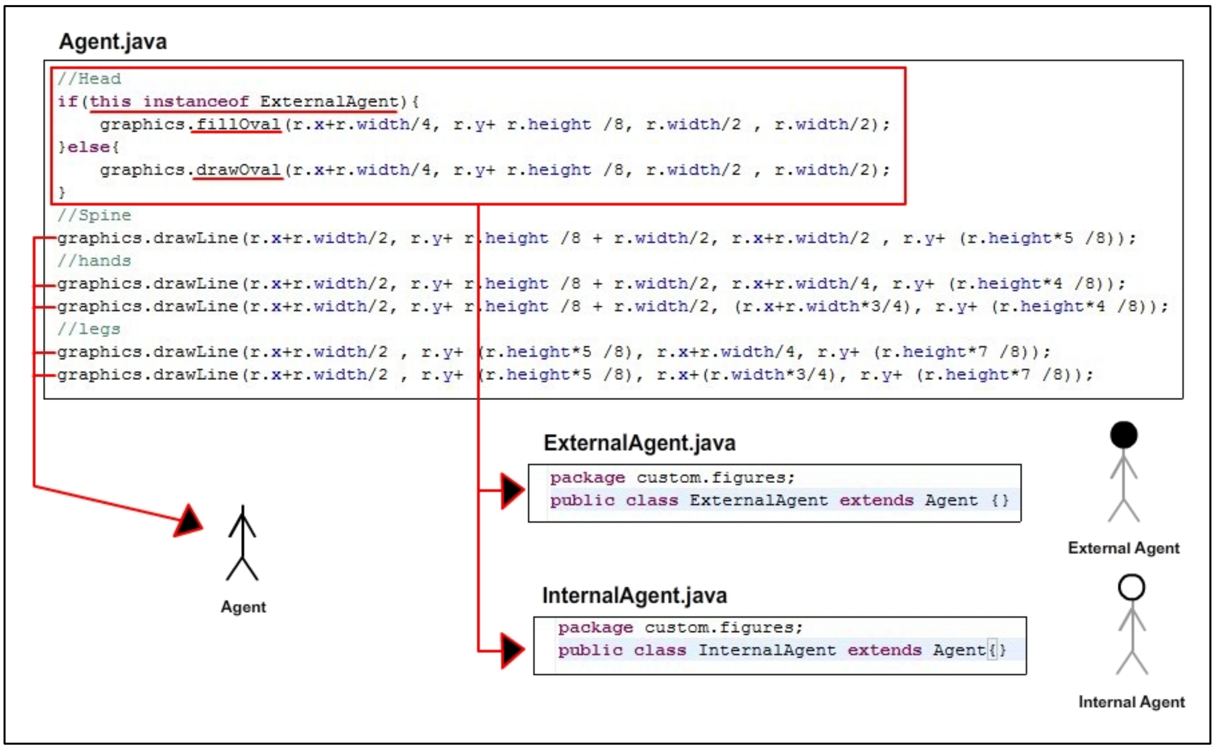
After implementing the concepts in the graphical modeling tool, one design concept was chosen, and it was implemented in the next phase.

Visual notation implementation

This phase included developing Java classes for the meta-model’s elements based on the selected concept from the previous phase. These classes implemented the Eclipse Draw2d API. Amongst other usages, this API is used for building 2D graphics programmatically. More information about the API can be found under ”GEF and Draw2d Plug-in Developer Guide” section of the official Eclipse documentation.

The first step in building the aforementioned Java classes was to determine the elements which have common shapes, and then to develop super classes which were responsible for rendering these shapes. After that, all classes with the same common shape extended the corresponding super class. The super classes have inner functionalities for determining the type of the loaded child instances, and accordingly, they draw the detailed graphics for each of their subclasses. As a rule of thumb, all classes with the same super class in the meta-model would have the same general graphical shape.

An example of the previous operation is depicted in ***Figure 7***. ”Agent” meta-class in REA’s meta-model has two subclasses. These are; ”Internal Agent” and ”External Agent”. The Agents’ figures share the same conventional stickman shape. The only difference between the two shapes is the filled head of the stickman. As it appears in the figure, ”Agent.java” (the super class) has a method that draws all parts of the stickman, and according to the instance of the subclass that is calling this method, the head will either be outlined or filled as an oval.



```
//Head
if(this instanceof ExternalAgent){
    graphics.fillOval(r.x+r.width/4, r.y+ r.height /8, r.width/2 , r.width/2);
}else{
    graphics.drawOval(r.x+r.width/4, r.y+ r.height /8, r.width/2 , r.width/2);
}
//Spine
graphics.drawLine(r.x+r.width/2, r.y+ r.height /8 + r.width/2, r.x+r.width/2 , r.y+ (r.height*5 /8));
//hands
graphics.drawLine(r.x+r.width/2, r.y+ r.height /8 + r.width/2, r.x+r.width/4, r.y+ (r.height*4 /8));
graphics.drawLine(r.x+r.width/2, r.y+ r.height /8 + r.width/2, (r.x+r.width*3/4), r.y+ (r.height*4 /8));
//legs
graphics.drawLine(r.x+r.width/2 , r.y+ (r.height*5 /8), r.x+r.width/4, r.y+ (r.height*7 /8));
graphics.drawLine(r.x+r.width/2 , r.y+ (r.height*5 /8), r.x+(r.width*3/4), r.y+ (r.height*7 /8));
```

```
package custom.figures;
public class ExternalAgent extends Agent {}
```

```
package custom.figures;
public class InternalAgent extends Agent{}
```

Figure 7: The visual implementation of "Agent" meta-class

From a programming perspective, some might argue that the subclasses can be implemented as inner classes. This might also work, but from an OO design perspective, inner classes are typically used for encapsulating structures as parts of larger classes, in other words, an inner class is a part of the encapsulating class, but in the case of this implementation, they are different classes. This method will also allow any dramatic future changes plausible and easier to implement. At the end, it is a matter of choice; any of these methods can be used.

Sections of the code in ***Figure 7*** show that the code of the “Agent” class was written from an aspect ratio perspective. This simply means writing the graphical class to maintain the right ratios between the components of the graphical shape. For example, if one wants to draw a rectangle that has a width of twice its height, then the rectangle can be said to have the dimensions of (width, width*1/2), were the first parameter is the width, and the second is the height. Aspect ratio programming for graphics is very important when resizing graphical shapes, as it guarantees the coherence of the graphical shape’s components to the overall shape.

After writing the classes for model elements, they were tested in the test editor of Eclipse. Even though the design concepts from the previous phase were implemented in a graphical tool, some elements were edited in this phase after experiencing the actual modeling elements in a live modeling environment. These elements were edited programmatically in order to improve the modeling experience. The suggested improvements covered the shapes of events and commitments elements.

After this task was finished and tested, it was time to generate the final editor. The process of generating the editor is described in the next section.

Editor generation

This by far was the shortest phase in the DSVL's development lifecycle. After the "Mapping Model" file of the GEF project was prepared, the editor's generator file was derived with the "RCP" option checked. ***Figure 8*** shows how this operation was done. After completing the third step of ***Figure 8***, a new Eclipse project was automatically generated with all the elements needed to run the editor. After that, a "Product Configuration" file was created inside the generated project, and it was configured according to the method described in (VOGEL, Lars, 2007).

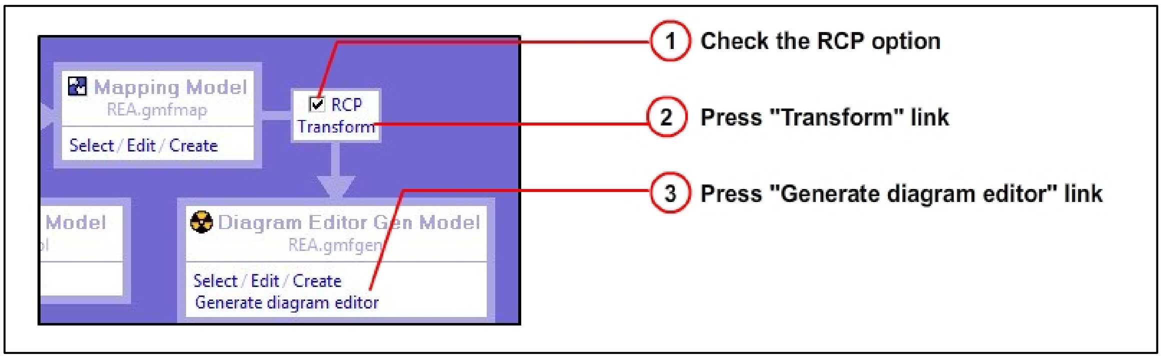


Figure 8: Generating the "gmfgen" file

By the end of this phase, the design and development task was finished, and it was time to demonstrate the generated editor in a practical scenario.

3.2.4 Demonstration

After generating the tool, it was the time to demonstrate its functions in a real case study. The case that was chosen is a tires and rims trading company in Stockholm/Sweden called ABS Wheels[1]. The business case from the case study was analyzed, then modeled based on the results of the analysis.

The first meeting with the company's manager included describing the purpose of the research, and giving a brief introduction of REA. The benefit of REA's applications and how it can help in managerial decisions were also illustrated with examples, and then the link between these applications and this research was explained. After that, the interviewed manager was asked if he would accept the indication of his company's name in this research, and if he would accept to record the interview for academic purposes, to which he answered with approval.

[1] *http://www.abswheels.se*

From there, open-ended questions were asked to the interviewed manager regarding his business in general. Then the questions took more restrictive manner, which targeted the relation between the business activities and the explained REA concepts. The questions did not take any fixed structure, because the interviewee did not have any prior knowledge of REA. Some explanations and reminders of REA reoccurred throughout the interview. For example, the interviewee first provided only three main REA processes which were carried out in his business, which was expected. The interviewee was then asked to think of all his business processes that cost him or provide him with money, as REA is about evaluating the total expenses and profits of a business. The latter question seemed to fit the interviewee's business perspective and background better than the first question. After the latter question, the interviewee started to detail all the tasks that cost or provide him with money, which raised the number of identified REA processes to ten.

After concluding the interview, its questions and answers were compiled into a case description. This case description was sent to the business manager for confirmation, to which he replied with no additional comments.

Following the interview, the business was analyzed and modeled using the developed tool. Pivotal agents and resources were identified, and then the main events and their types (exchange or conversion) were identified. After that, the identified events were modeled and linked to the related agents and resources. Commitments and contracts were identified and modeled later when they were needed.

3.2.5 Evaluation

As mentioned in the method choice section for evaluation, this activity targeted the evaluation of the meta-model, the visual notation, and the modeling tool. The meta-model had quantitative requirements and no special quantitative requirements; thus, its evaluation was based on the fulfillment of these requirements directly. The visual-notation had one quantitative and one qualitative requirement. The quantitative requirement was judged based on the fulfillment of that requirement. The qualitative requirement could not be evaluated independently, but rather as a part of the modeling tool. The modeling tool is the main artifact of this research, and it had a set of qualitative requirements. The fulfillment of the tool's requirements was planned to be evaluated based on an experiment which was not conducted due to the limited responses, and instead was judged using on informed arguments.

An experiment was designed for two groups with similar knowledge of REA, typically students with an academic knowledge of REA. The treatment group was asked to use the developed tool to model a small business scenario from the previous case study, and the control group was asked to model the same case using another designing tool like Microsoft's Visio. After that, a set of questions were compiled to measure the level of the experimental groups comfort in designing REA models using each technique. The evaluation in this case was designed to measure the correctness of the designed models, and how the groups responded to the questions.

As the tool was used for modeling the business from the case study, the experience of using the tool was used for formulating arguments which were used for judging both the fulfillment of the tool's requirements, and the extent to which the tool can answer the research question.

3.2.6 Ethical and social considerations

Ethical and social considerations which were taken into account at the problem identification phase included identifying real existing problems that if solved, would lead to tangible benefits to the domain of study. Another consideration was related to judging previous attempts to solve the problems objectively, and based on such judgment; the choice was taken to continue with studying the identified problems. Another ethical consideration was taken when existing works of a similar nature were criticized. Such criticism was based on profound evidences of the claims and with the sole purpose of validating the legitimacy of arguments.

A lot of ethical considerations had to be observed in the development phase. Starting with the meta-model, all the knowledge that was used to build this meta-model was presented. Existing REA meta-models were reviewed only for the purpose of knowledge and suitability check. None of the reviewed meta-models was used to develop the REA meta-model of this research, and all of the reviewed meta-models were listed in the background of this research.

As for the visual notation, the notation was developed based on the writer's own merit. None of the previously developed notations, if any, was used. The visual notation was also designed in a way that conceives the maximal simplicity while covering full REA concepts. Several visual concepts were designed in order to provide the DSVL users with the best possible visual notation. The notation's symbols are directly related to REA, and they do not hide or include any symbolic figures that might cause any ethical or social harm to language users.

The editor was designed and generated to ensure a minimum usage of the platform resources. The packaged editor was checked for malicious code and infected files as they are forbidden by both; ethics and law. The generated tool was also produced in a way that covers full modeling needs, and at the same time in a way that occupies the minimal set of users' machine resources.

When the case study was conducted, the business owner was asked if he would agree to publish the details of the case study and his company name before publishing them. Based on the manager's approval, the details of this information were revealed in this report. The analyzed business of the case study was presented in a way relevant to this research, and in order to save the privacy of the business, other business related information that did not affect the purpose of the case study was not included. As for the reader, the case study was described in a summarized manner while covering all business details related to this research.

As for the evaluation, a typical experiment was designed to evaluate the developed language and tool. Unfortunately, the experiment could not be completed due to the unsatisfactory number of participants. No attempts to forge the results were undertaken, as it is completely unethical, and that it would not help the author of this report in realizing the true value of his work.

4. Results and Analysis

In this chapter, the results of this research are presented according to the phases of the design science which were discussed in the previous chapter.

4.1 Problem identification and motivation

The results of the problem identification and motivation phase have been presented in section ***1.2*** of this report; thus, this section will just list the identified problems and the results of the literature review that was conducted for identifying the problems and constructing the requirements.

The major problems which have been identified were:

- REA is a business modeling ontology that was originally created to support the development of business information systems for enterprises. Although the ontology has been developed since the 1980's, it is still not widely used in developing information systems, this is due to
    - The ontology does not have a common formal representation, which has resulted in many researches focusing on defining meta-models of the ontology while ignoring the wider image of the ontology's purpose for developing information systems.
    - The ontology does not have a formal visual notation that takes the ontology to the next level of being a modeling language. The lack of a visual notation limited the ontology's expansion, which has resulted in containing the ontology within a scientific context.

This research has evolved on top of an idea for creating a meta-model of REA. Literature review revealed that the problem was wider than simply creating a meta-model of the ontology. For this purpose, literature about the topics which has been used throughout this research has been reviewed. The set of the total reviewed resources for this research is available in (***Appendix A.1***).

4.2 Define the objectives for a solution

Literature review revealed a general method for solving the identified problems. The general solution suggested building a DSVL for REA which can be implemented in a MDD tool. DSVLs, as modeling languages, consist of three parts, the abstract syntax in the form of meta-model, the concrete syntax in the form of visual notation, and the language's semantics in the form of mapping the visual notation to the elements of the meta-model. ***Table 2*** presents the set of requirements which have been gathered for the meta-model, the visual notation, and the MDD tool (the REA-DSVL Editor).

After setting the requirements for the general solution, the tools and standards to be used for development were narrowed down to tools which allow the development of both meta-models and visual notations together. This insured that the selected tools would generate a language that conforms to the structure of DSVLs. These tools were then analyzed according to some factors. ***Table 3*** shows the set of modeling tools which were considered for developing the DSVL. The table also shows the main factors which were taken to determine the final toolset to be used in the DSVL development.

Requirement Reference	Requirement
MMR1	Develop a Meta-Model of the REA ontology
MMR1.1	The meta-model should cover the elements defined in the basic REA model including; “Resource”, “Event”, “Agent”, “Duality”, and “Stock-flow”.
MMR1.2	The meta-model should cover the elements defined in the extended REA model “Commitment”, ”Contract”, and “Reciprocity”.
VNR1	Develop a REA based visual notation
VNR1.1	The visual notation should be simple.
VNR1.2	The visual notation should provide symbols that cover the elements of REA identified in MMR1.
MTR1	Develop a REA based modeling tool
MTR1.1	The tool should provide functionality for designing REA diagrams.
MTR1.2	The tool should provide functionality for generating REA based data-models.
MTR1.3	The generated data-models should be written in a machine independent format.
MTR1.4	The tool should provide convenient and easy to use functionalities, by providing a comprehensive user interface and minimum set of functionalities.
MTR1.5	The tool should be reachable by different users with different operating systems.

Table 2: REA-DSVL and REA-DSVL editor Requirements

Tool Name	Meta-modeling support	Visual modeling support	Dedicated editor generation	License type
Poseidon for DSLs [1]	Yes	Yes	Yes	Commercial
MOFLON [2]	Yes	Limited to UML profiles	No	Open source
ADOCUS [3]	Yes	Limited to UML profiles	No	Commercial
MetaEdit+[4]	Yes	Yes	No	Commercial
Microsoft Domain-Specific Language (DSL) Tools[5]	Yes	Yes	Yes	Commercial
Eclipse GMP	Yes	Yes	Yes	Open Source

Table 3: DSVL modeling tools

[1] http://www.gentleware.com

[2] http://www.moflon.org

[3] http://www.adocus.com

[4] http://www.metacase.com

[5]http://www.microsoft.com/en-us/download/details.aspx?id=2379#overview

MetaEdit+ and ADOCUS provide functionalities for building domain-models at the meta-meta-level, and then make instances of them within the same tool. They do not possess functionalities for generating dedicated modeling environments like GMP. In addition to that, MetaEdit+, Poseidon, and ADOCUS have their own implementations of UML. MOFLON has its own implementation of MOF. Eclipse Modeling Framework (EMF) of the GMP, on the other hand, has a unique standard of its own, the Ecore. This standard is recognized by OMG as it has provided an Ecore-based UML file in its website that can be used within EMF to build UML models based on Ecore.

Microsoft Domain-Specific Language provides a powerful toolset for building DSLs, but in a comparison between Microsoft's tool and Eclipse's GMP, the latter provide more choices of platforms that can run the generated editors. GMP supports MAC OS, Solaris, Linux, in addition to Windows, which was the only supported platform by Microsoft's toolset. In addition to that, Microsoft's visual studio is needed in order to use the DSL toolset, and the visual studio has a commercial license. Eclipse, on the other hand, is an open-source. Poseidon for DSLs has also a commercial license; thus, it was not considered.

At the end of this stage, the GMP was selected as the toolset to be used for development.

4.3 . Design and Development

This section will present the results of the four main phases which have been held under the development activity as described in section **3.2.3**.

4.3.1 Meta-Model development

As mentioned earlier, the first phase of the development process covered the development of the REA-DSVL meta-model. The first meta-model that was developed went through major design changes, and is available in ***Appendix A.3***. After applying the method and design changes which were mentioned in section **3.2.3**, the final version of the meta-model became the one presented in ***Figure 9***.

In this discussion, it is inevitable to discuss some of the major design decisions that helped in shaping the final design of the meta-model. While designing the meta-model, it was extremely important to keep the final purpose of the meta-model in perspective. This meta-model is the backbone of the DSVL to be developed. This idea provided a guideline while designing the domain model. For example, it was not necessary, if needed at all, to add a super meta-class that aggregates top-level meta-classes of the domain model. In this meta-model though, it was necessary to add such an element even though it did not add further information to the business domain of REA. REA-DSVL meta-model contains an element called "REA_Model". This element is the uppermost meta-class of the model, and it acts as the container of all model elements. One can think of it as the *canvas* of a painting, while other model elements are the shapes that make a complete painting. The foregoing metaphor is not only an explanatory example of this element's task, but rather a practical explanation of what it does. The *REA_Model* meta-class is the canvas of the designed diagrams which are based on this meta-model.

Other elements of the meta-model were modeled based on the decisions described in section **3.2.3**, and according to the relationships matrix that is available in ***Appendix A.2;*** thus the cardinalities will not be discussed in this section as they are can be found in any REA resource. As depicted in ***Figure 9***, elements of the basic and extended REA models were modeled in the meta-model.

Events were broken down into decrement and increment, and exchange and conversion. Facilitating events to exchange and conversion events was important, because in REA, exchange events must have exactly one internal agent and one external agent, but conversion events *can* have at max one internal agent, and they do not associate directly with external agents.

Increment and decrement divisions were added because in REA, any duality must contain at least one increment and one decrement event of the same nature (exchange or conversion). If these were not facilitated, one could make a duality between only two incremental events for example, or worst, make a connection between two incremental or decremental events of the same type. Accordingly, it was necessary to break down both the event-type (exchange or conversion) and its incremental value (increment or decrement) on the meta-class level. For the same previous reasons, commitments followed the same structure.

Dualities and Reciprocities were facilitated to indicate the connection between events which belong to the same category. If a duality or reciprocity was modeled using one meta-class, then there would have been a way at which users can connect exchange events to conversion events for example, which violates the rules of REA severely.

These were the major decisions which lead to formulate the major structure of this meta-model.

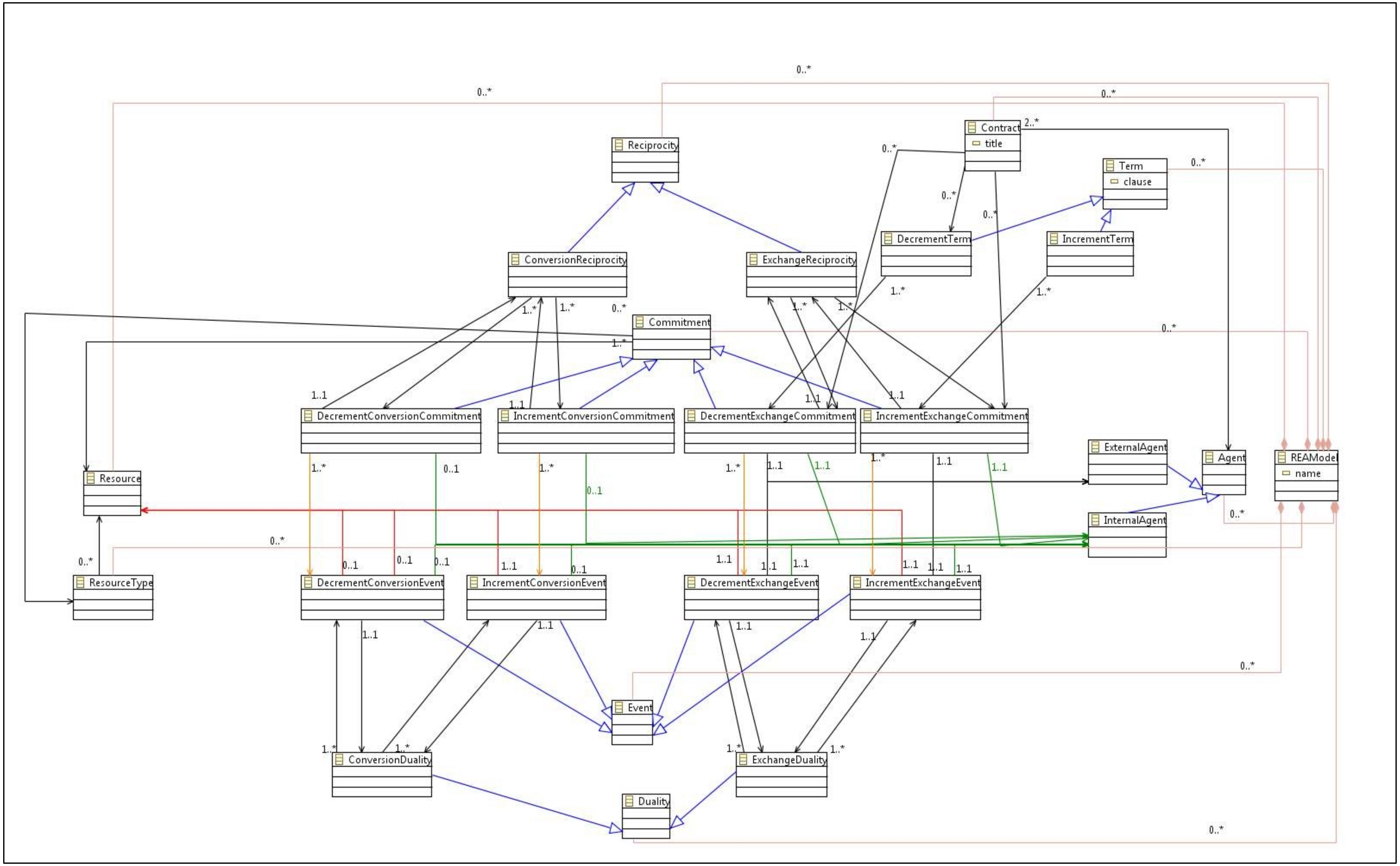


Figure 9: REA-DSVL's meta-model

4.3.2 Visual notation development

As mentioned in section **3.2.3**, three main design concepts were prepared in order to provide a wider base of choices for a better visual notation. These design-concepts are depicted in ***Figure 10***, ***Figure 11***, and ***Figure 12***. One of these concepts was chosen for implementation in the next phase.

The first concept in ***Figure 10*** has the theme of linear duality representation. As it appears in the figure, dualities were represented as containers which hold all the participating events. Each event is then linked to the resource that it will (give, take…etc), and the agent who **provide** such resource.

Two main problems were identified in Concept 1. The first problem concerned the practicality of the notation, and the second problem concerned the representation of agents.

If this notation was to be applied to large business domains, the whole model would have to take a vertical structure consisting of linear dualities and reciprocities. This might cause difficulties for modelers when tracking their models; thus, affecting the notation's practicality.

The model should also allow its readers to read its content in their common language; for example, the increment exchange event in ***Figure 10*** should be easily read as "The Customer Pays Money to the Cars Dealer". As it appears in the figure, the reader should interpret the exchange relationship in order to reach such a result. The notation was designed in this way in order to minimize the number of relations (links) that associate agents with events. The number of agents' links in a typical REA duality is twice the number of events in that duality. By designing the notation this way; the number of agents' links would have dropped to the half, unfortunately, it was not practical.

Concept 2 in ***Figure 11*** represents events in hexagonal shapes. The idea was to link the associated events together by their borders. The final business value chain would take the shape of a beehive according to this concept. Practicality issues also limited the

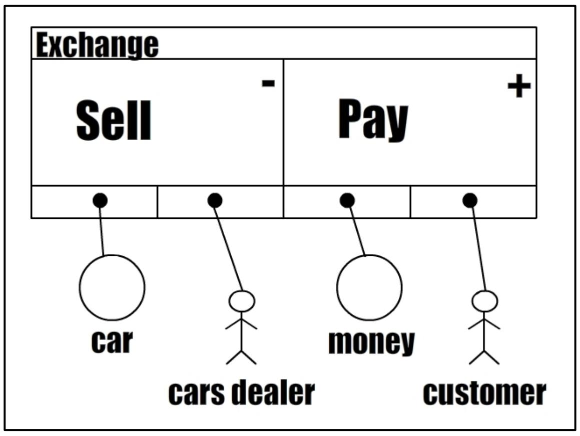


Figure 10: Concept 1

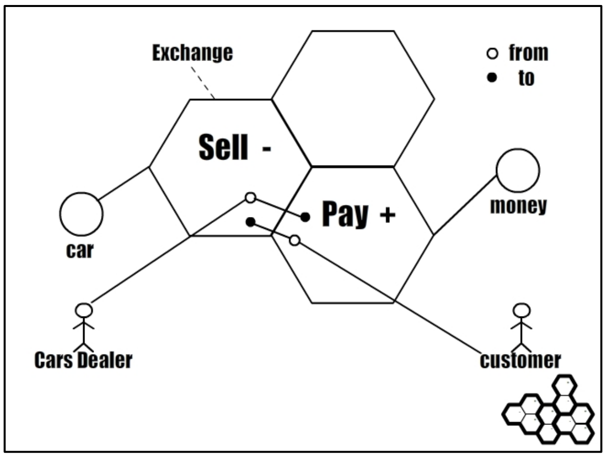


Figure 11: Concept 2

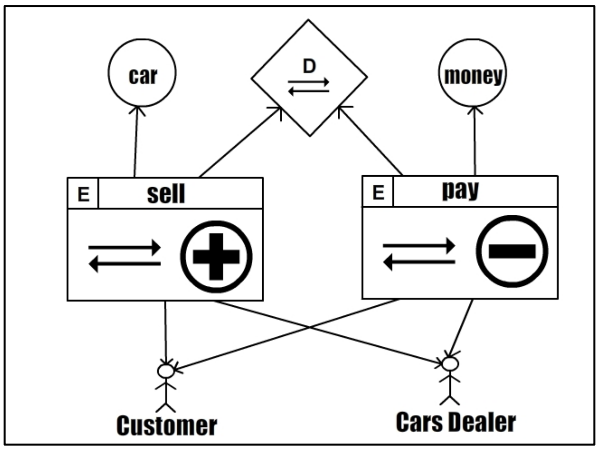


Figure 12: Concept 3

implementation of this concept. First, the concept defines a new rule on top of REA's. The figure shows that there are two points; "to" and "from". These points describe the flow of resources within events. The agent who is linked to the "from" point is the one who is providing the associated event with the linked resource. These rules might be confusing to some of the intended audience, typically the nontechnical ones. The coupling of events by borders was another reason for discarding this concept. The latter limitation can be visualized when thinking about resizing the model or editing it. If only one of the events was resized, then all of its adjacent events would have been resized accordingly.

Concept 3 in ***Figure 12*** is a simple representation of REA elements and concepts using familiar shapes. As it appears in the figure, dualities have their own figures. As with the earlier concepts, incremental characteristics are represented using (+) sign, and decremental ones with (-) sign. An *Exchange*, as a concept, is depicted with opposing arrows which reflect the give-and-take nature of exchanges. Events are depicted as normal rectangles with headers that hold the "E" character (E for Event) and the title of the event. In the body of each event, there is a representation of the type (exchange or conversion) and the value-effect nature (increment or decrement) of the event. Among the three design concepts, concept 3 seemed to provide the best representation; thus, it was chosen for implementation in the next phase.

Dualities and Reciprocities were depicted as lozenges (diamond shapes) with the character "D" or "R" to represent each of them respectively. Dualities and Reciprocities serve the same purpose for events and commitments; thus, it was quite reasonable to give them the same shape. As with events and commitments, dualities and reciprocities have the symbols which indicate "exchange" or "conversion".

The "Resource" element was modeled as a circle, and "Resource Type" was modeled as double circles due to its tight relation to the "Resource" element. "Internal agent" and "External agent" were depicted using the famous stickman shape, with minor difference in filling the head of external agent.

The "Contract" element was depicted in the form of a paper with a folded tip to give an impression of using a document. Terms were depicted as triangles to give a sense of direction from the contract element that is linked to them, and they have the (+) and (-) to indicate the effect of the term.

4.3.3 Visual notation implementation

This phase covered the development of the graphical Java classes for the concept's elements from the previous phase. As mentioned in section **3.2.3**, after implementing the visual notation and testing it in a live modeling environment, "Event" and "Commitment" shapes were edited for a better optimization of the designed models. ***Figure 13*** shows the nature of these improvements. Such changes were easily implemented due to the described aspect-ratio programming style.

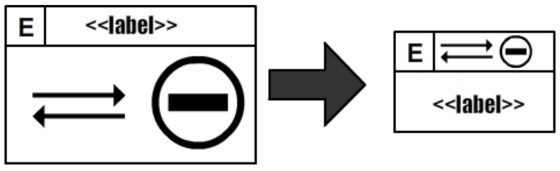


Figure 13: Concept 3 improvements

The final sets of the rendered graphics is depicted in ***Figure 14***, and *structure* of the classes written for this task are presented in ***Appendix A.5***.

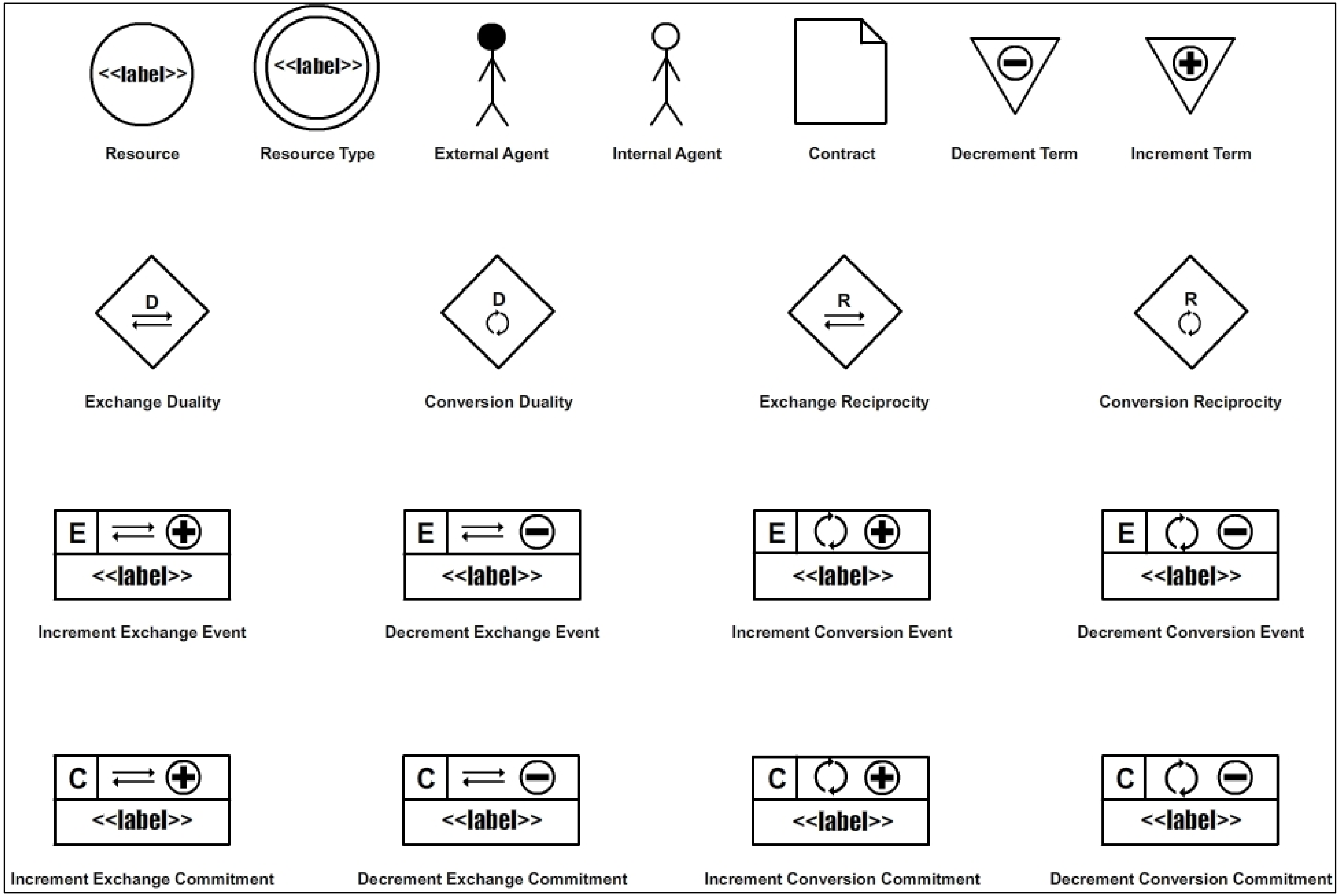
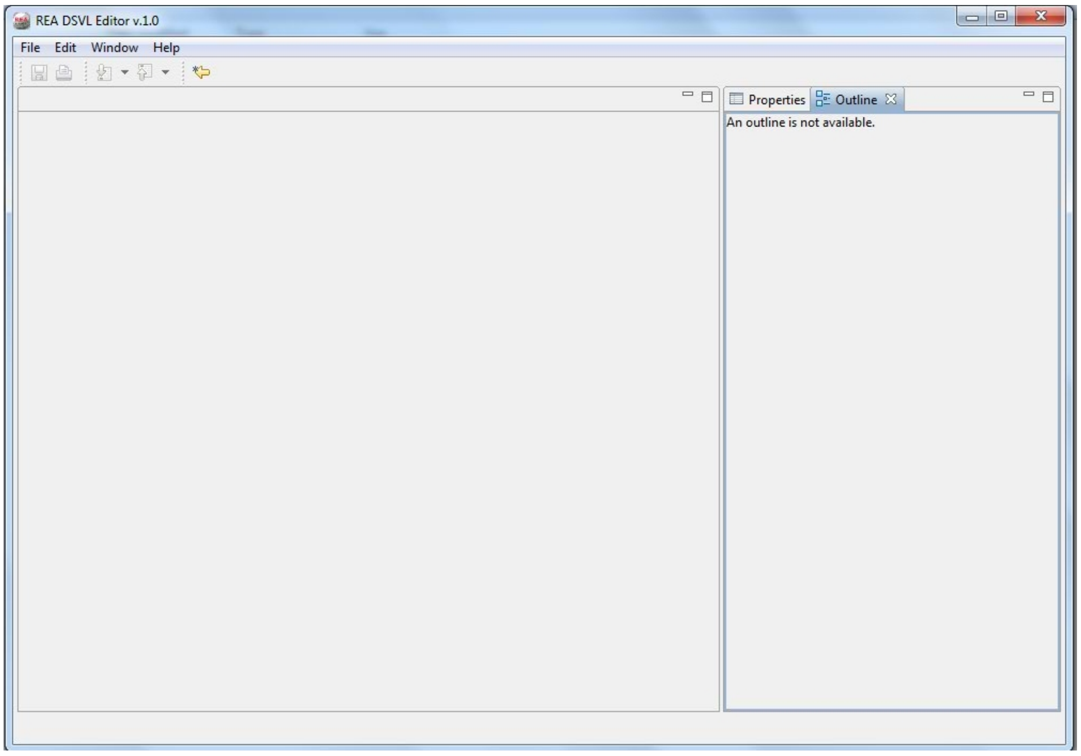


Figure 14: The final visual notation after implementation

4.3.4 Editor generation

This phase ended with generating the DSVL's editor that is called "REA DSVL Editor". This tool is available on online for both the Windows platform[1] and the Mac OS X[2]. When the editor is launched for the first time, it will open a window similar to the one that appears in ***Figure 15***. Users can customize their editor view by moving the "outline" and "Properties" windows into the desired screen location.



An outline is not available.

Figure 15: REA DSVL First Run

As it appears in ***Figure 15***, the editor has four main menus; these are; "File", "Edit", "Window", and "Help". These menus provide customary functionalities like (save, open, exit…etc), which are provided by most of software systems. ***Figure 16*** shows the file menu of REA DSVL Editor. The editor allows its users to create "REA Diagrams" as the only available option.

When the user selects "REA Diagram" from the menu in ***Figure 16***, a new diagram wizard is launched as shown in ***Figure 17***. The first window of the wizard allows the user to create files with the ".rea_diagram" extension. This file type represents the *graphical* model to be designed, or simply the diagram. After typing the name of the diagram file, the user has the choice to either create a domain-model file for the diagram, or to finish the wizard without creating such a model.

Domain-model files with the extension of ".rea" are XML data models based on the ".rea_diagram" files. These data models conform to the relations identified in the original REA meta-model that was developed earlier. The content of a domain-model (also referred to as data-model) are auto-generated at runtime while modeling the ".rea_diagram" diagrams.

[1] *https://docs.google.com/open?id=0B1dkJd7veJw7SS1OZG4zekRQeWM*

[2] *https://docs.google.com/open?id=0B1dkJd7veJw7NDlpRV9tekhDUFE*

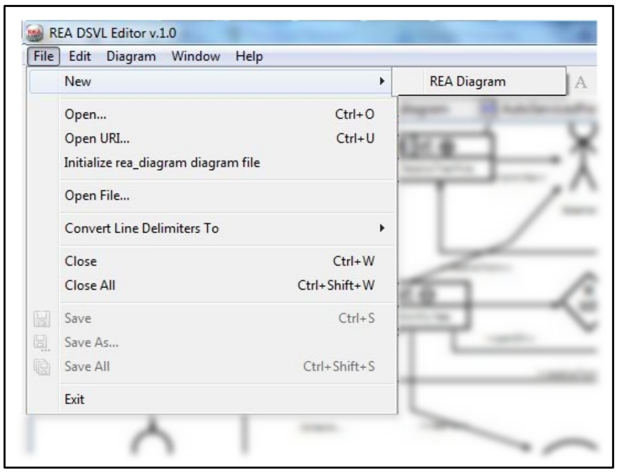


Figure 16: New REA Diagram

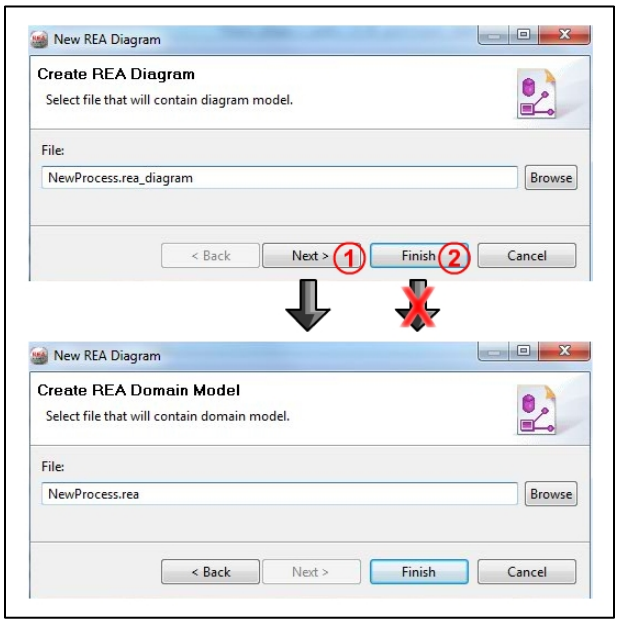


Figure 17: New REA Diagram Wizard

After creating the diagram file for the user, the tool opens the modeling environment as depicted in ***Figure 18***. The part of the editor that is marked as (1) represents the menu of REA's modeling elements. Users click on their desired element from that menu, and then click inside the modeling area to create an instance of that element like the one shown in section (3).

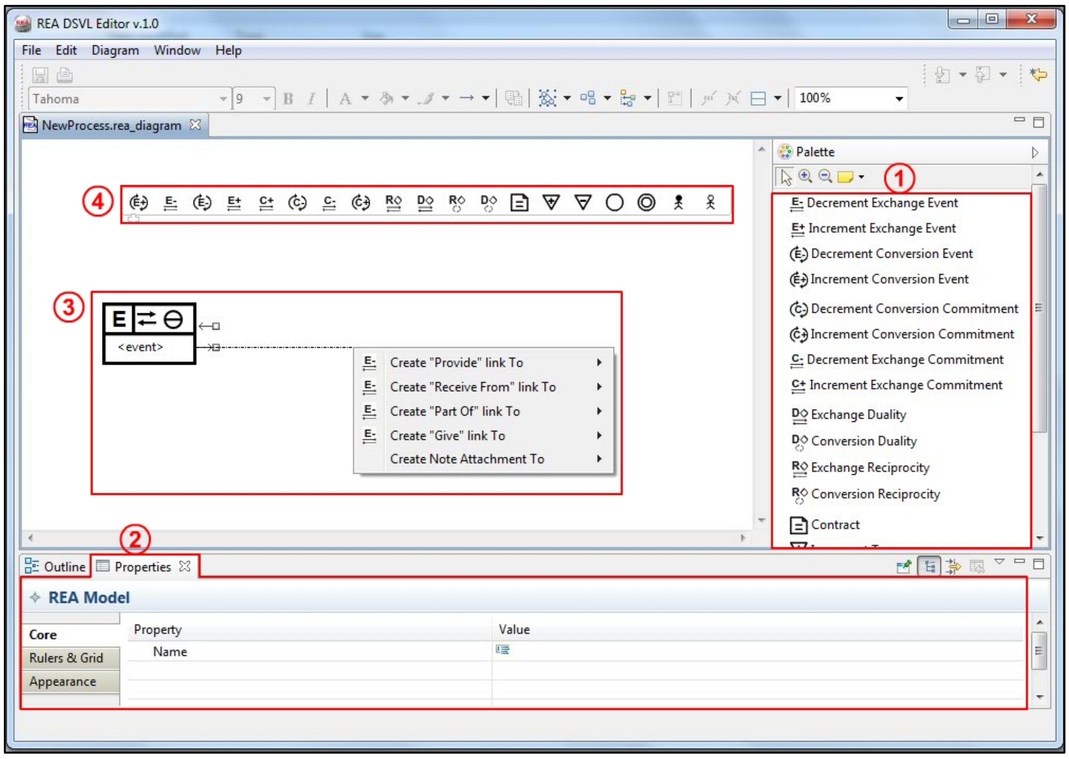
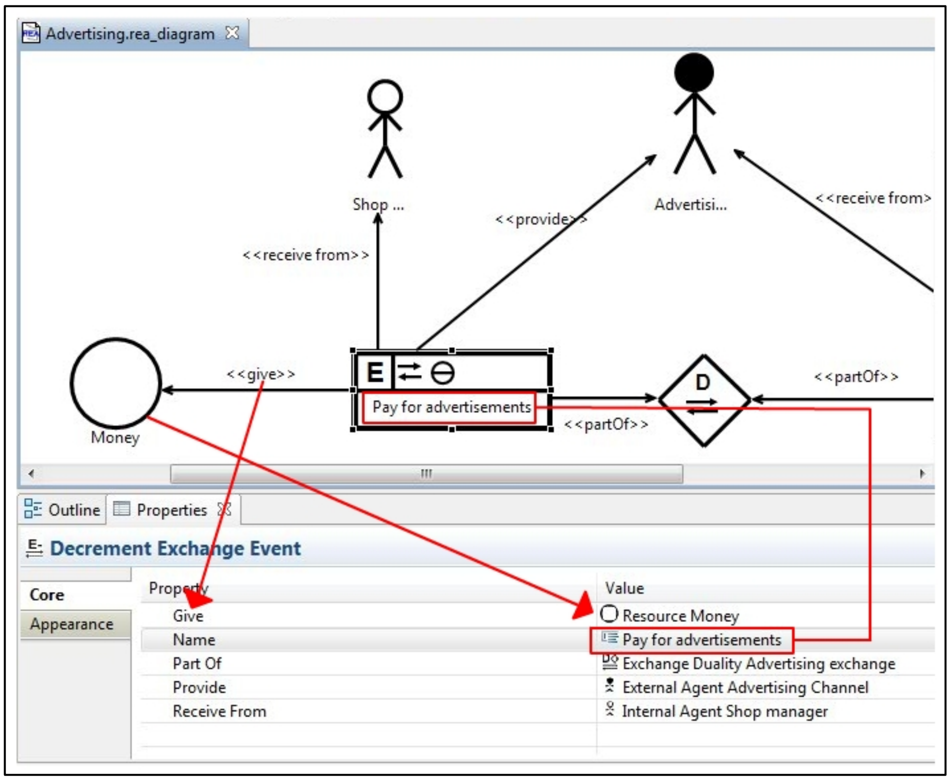


Figure 18: REA-DSVL Editor

The editor provides modelers with other options for creating elements on the modeling area. Part (4) of ***Figure 18*** represents a quick flyer menu. The flyer menu appears after few seconds when the mouse is pointed to a blank space in the modeling area. Another option for creating elements is the context menu associated with each of the diagram's elements. Part (3) of ***Figure 18*** shows the context menu associated with a decrement exchange event. When the modeler points her mouse over an element in the modeling area, two arrows will appear on the border of that element. One of these arrows is directed toward the element, and the other is directed outside the element. The arrow directed inwards represents the links (relationships) ***to*** that element from other diagram elements, and the outer arrow represents the opposite. When one of these arrows is dragged to an empty area of the diagram, a context menu with the appropriate relationships appear. Modelers can choose the relationship type that they want to draw from the context menu, and then they can connect the relationship end to an already existing element, or they can create new elements according to the relationship type.

Part (2) of ***Figure 18*** shows the "properties" window. This window is used to edit; names, relationships, and the look and feel of modeling elements. The "properties" window has two tabs; these are "Core" and "Appearance" tabs. The properties window changes its content according to the selected modeling element. The "Core" tab contains pairs of "Property" and "Value" elements. The "Property" elements represent the relations of the selected modeling element which are based on the meta-model, and the "Value" elements represent the implementation of these relations in the diagram. ***Figure 19*** shows an example of how this window would look like when selecting an exchange event.



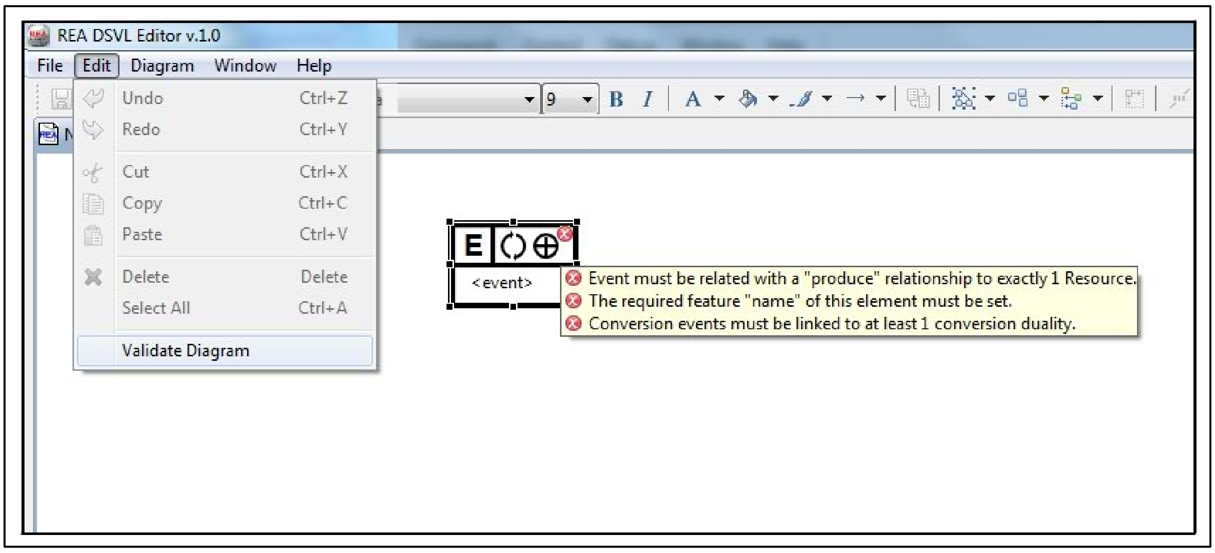
Property	Value
Give	Resource Money
Name	Pay for advertisements
Part Of	Exchange Duality Advertising exchange
Provide	External Agent Advertising Channel
Receive From	Internal Agent Shop manager

Figure 19: The properties window for an exchange event

It is worth mentioning here that the relation between the properties window and the designed models is bidirectional, this means that, if the relationships were set on the designing area, the properties window will change accordingly, and if the relationships were set using the properties window, the model will draw the links between its elements based on the configured relationships.

The other tab of the properties menu, namely “Appearance”, has options that allow the user to change the look and feel of the modeled elements. These options include changing the font’s type, size, and color of the model labels. It includes also functionalities for changing the colors of the diagram elements. Some valuable appearance options are also available to the links of the model. If a link between two elements was selected, the editor provides options for the link to avoid obstacles (other links or elements).

After creating a new “REA Diagram” file from the “File” menu, a new menu entitled “Diagram” will appear in the tool bar of the editor. In addition to “Diagram”, a new menu item with the label “Validate diagram” will appear in the “Edit” menu as shown in ***Figure 19***. The “Diagram” menu provides the same functionalities as the ones provided by the properties view. When the user clicks on the “Validate Diagram” menu item, the diagram will be validated according to the relations identified in the first developed REA meta-model. In case of violations to the meta-model, an error sign will appear on the upper right corner of the diagram element, and when the user points her mouse over that error sign, a list of errors will be listed as a tooltip.



Event must be related with a "produce" relationship to exactly 1 Resource.
The required feature "name" of this element must be set.
Conversion events must be linked to at least 1 conversion duality.

Figure 20: Diagram validation

In addition to the views and functionalities described earlier, the “outline” window of the editor shows an overview of the complete model, and it provides a highlighted blue rectangle of the viewable part of the model that appears on the screen.

Another feature provided by this tool is its ability to produce images from the designed models. This can be achieved by showing the context menu of the diagram. To do that, the user presses the right mouse button on an empty area of the diagram, then chooses the context element “File >> Save image as”.

These were the major functionalities provided by “REA DSVL Editor”. A quick summary of these functionalities is provided in ***Table 4***. The next section will present the results from the next phase of the DSRM, namely, the demonstration phase.

Functionality Name	Place in editor	Functionality Description
Create new REA diagram file	File >> New >>REA Diagram	Used for creating new REA diagram file with the extension ".rea_diagram".
Create a REA diagram from a data-model	File >> Initialize rea_diagram diagram file	Used for creating ".rea_diagram" files from ".rea" files. The ".rea" files are xml files based on the main REA meta-model. The ".rea_diagram" files are also XML files, though they hold the visual REA models information.
Modeling and Editing diagrams	Modeling canvas and properties view tab	The modeling canvas provides three main methods for creating new elements, these are: • Elements palette. • Context menu of the created elements. • Flyer menu. For editing the relationships of elements, one can manipulate the relations directly through drag and drop, or can use the "properties" tab for editing both the relationships and the "Look and Feel" of elements.
Validate Diagrams	Edit >> Validate Diagram	When the "Validate Diagram" from the edit menu is clicked, the modeled diagram will be validated against the meta-model that was developed earlier.
Models Overview	Overview view tab	Shows the complete designed model and the current viewable section of the diagram or model.
Create Images of the designed Diagrams	File >> Save image as	Saves the modeled diagram as an image.

Table 4: Summary of REA DSVL Editor's main functionalities

4.4 Demonstration

After interviewing the manager of ABS Wheels, a business description of the company's business activities was compiled and sent to the manager for confirmation. The business description of ABS Wheels is available in ***Appendix A.6***. After the manager's reply with no further comments, the analysis of the business was conducted according to REA, and the main REA elements of the business were identified.

The business description in ***Appendix A.6*** was written in a way that makes it easy for the reader to identify the REA processes in ABS Wheels; thus, instead of discussing the details of why the business was modeled in a specific manner, which is not the purpose of this section, the results of this analysis are directly provided in ***Appendix A.7***. Following this step, business models of ABS Wheels were designed using the REA DSVL Editor. The processes diagrams and their associated data-models (XML files) are available in ***Appendix A.8*** and ***Appendix A.9*** respectively.

For the purpose of demonstration, the modeling of one process from the case study will be described in this section. The following text is taken from the business description in ***Appendix A.6***, and it describes the process of publishing advertisements for the ABS Wheels.

> ***"The company occasionally places advertisements for its business on radio stations, magazines, and Internet websites. The payment for such advertisements is done at the time when the ad is requested."***

To start modeling, a new ".rea_diagram" file was created with its associated ".rea" data-model. The events from the previous description are clear and simple. The *company's manager* (Internal agent) pays for the *Advertising channel* (External agent) some *money* (Resource), and the advertising channel publishes *advertisements* (resource) for ABS Wheels (Internal agent). For that purpose, an increment exchange event with the name "Get advertised" was modeled, then another decrement exchange event with the name "Pay for advertisements" was modeled, and both of these events were linked to an exchange duality element.

After that, the agents and resources which were previously identified were modeled using the appropriate elements from the editor. After completing the model, an image of the model was generated. The generated image is used in this report under ***Figure 21***.

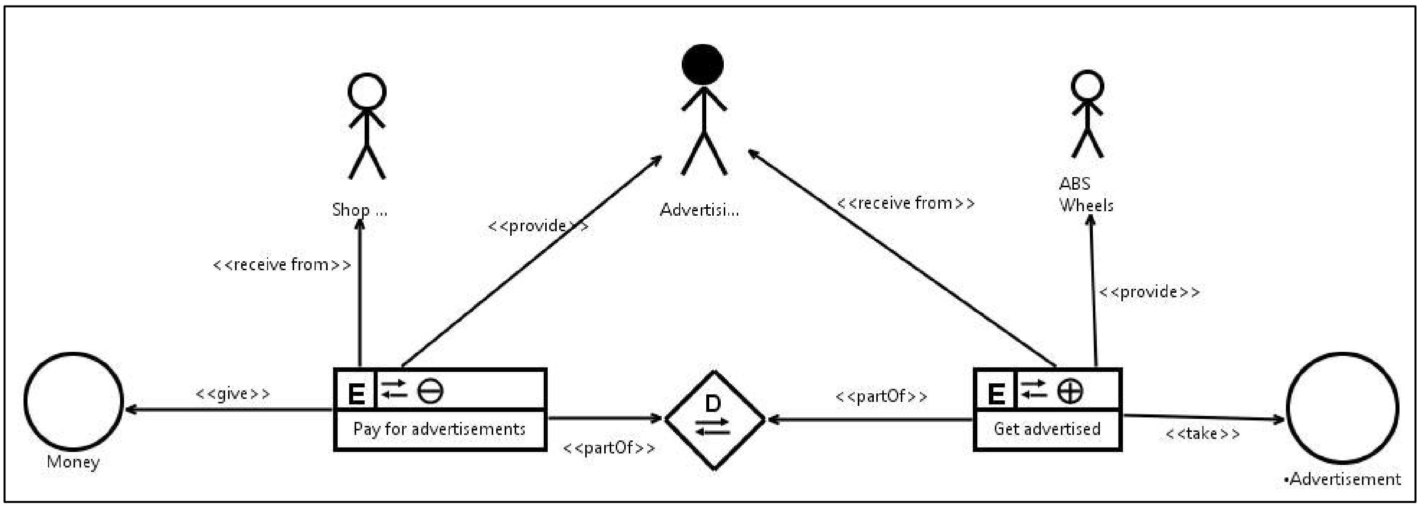


Figure 21: ABS Wheels Advertising process as generated from REA-DSVL Editor

As described in the previous section, the ".rea" file content is created automatically while developing the model. The content of the ".rea" file which was created for this process is presented bellow in its XML format:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<rea:REAModel xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:rea="http://rea_extended/1.0">
  <hasAgents xsi:type="rea:InternalAgent" name="ABS Wheels"/>
  <hasAgents xsi:type="rea:ExternalAgent" name="Advertising Channel"/>
  <hasAgents xsi:type="rea:InternalAgent" name="Shop manager"/>
  <hasEvents xsi:type="rea:DecrementExchangeEvent" name="Pay for
advertisements" give="//@hasResources.0" partOf="//@hasDualities.0"
provide="//@hasAgents.1" receiveFrom="//@hasAgents.2"/>
  <hasEvents xsi:type="rea:IncrementExchangeEvent" name="Get advertised"
take="//@hasResources.1" partOf="//@hasDualities.0"
receiveFrom="//@hasAgents.1" provide="//@hasAgents.0"/>
  <hasResources name="Money"/>
  <hasResources name="Advertisement"/>
  <hasDualities xsi:type="rea:ExchangeDuality" name="Advertising
exchange" containsDecrementEvent="//@hasEvents.0"
containsIncrementEvent="//@hasEvents.1"/>
</rea:REAModel>
```

The XML schema that is called "rea" is an XML schema that is based on the REA meta-model that was created for this modeling language. All the tags available in this XML file represent elements from the schema file, and they are simply depicted in the meta-model as the relationships between the root element (REAModel) and all other sub elements.

When the modeling of ABS Wheels started, the intention was to develop all the process under one model. This, however, seemed to be a hard task to achieve. This point will be elaborated on during the evaluation of the tool. There were no other significant limitations that appeared during modeling the process of ABS Wheels. The following section will discuss the evaluation of the tool.

4.5 Evaluation

As mentioned earlier, the evaluation was done based on the extent to which the collected requirements were fulfilled using the developed artifacts. For this purpose, each of the requirements will be revisited and compared with its associated results which were presented in this chapter so far. The editor was evaluated using informed arguments in order to check its suitability for answering the research question.

4.5.1 The meta-model

The meta-model requirements were of a quantitative nature based on the reasons provided in section **3.1.5**. The first requirements MMR1.1 and MMR1.2 suggested that the meta-model should cover the elements of the basic and extended REA models. As seen in the meta-model that was presented earlier, the meta-model covers these concepts, and it adds additional concepts from REA's policy level specifications (Contracts, Terms, and Resource-type). By fulfilling these requirements, the fulfillment of the main requirement (MMR1) was confirmed.

The meta-model has covered its purpose as being the structure (abstract syntax) of the modeling language. A full meta-model of REA is necessary for producing full REA models. McCarthy in (GEERTS, Guido L. et al., 1996) has emphasized the importance of full REA models when building enterprise information systems. Full REA models help in increasing the knowledge base of the modeled enterprises. This is achieved by revealing the full enterprise's value chain; thus, showing the full business activities and business processes of the modeled enterprise. Based on that, according to McCarthy, the obtained business knowledge would typically help in advancing the planning and analysis phases of the software engineering lifecycle. McCarthy also mentioned that if full REA models were to be implemented as data-models, then they would provide a *skeleton schema* that is useful for other phases of the software engineering lifecycle.

Although this meta-model covers both the basic and extended REA models, other ontological concepts of REA were omitted from this meta-model. The omitted concepts included the full REA policy level representations, and REA process view. The omitted concepts were described in the work of McCarthy as "less occurring" or rare. These elements were omitted because the conception which was taken while designing the meta-model targeted the representation of business processes rather than complete enterprise business network.

4.5.2 The visual notation

Literature that discusses visual composition of modeling languages could not be found in order to provide a base for developing the visual notation. For the same purpose, it was hard to formulate scientific informed arguments based on literature. For this purpose, the evaluation of the visual notation was based directly on the use of REA DSVL within the Editor. Based on the aforementioned facts, this section tends to evaluate the visual notation based on the fulfillment of the requirements, and the experience of using the notation within the REA DSVL Editor.

The final visual notation was presented in section **4.2.3**. As described earlier, the visual notation represents the concrete syntax that language users directly interact with; thus, a basic requirement of simplicity was needed. This requirement is available under VNR1.1. To fulfill this requirement, simple geometric symbols were used while developing the notation to generate easily recallable figures. For the same purpose, elements of REA with close characteristics were designed in similar forms with minor differences. For example, commitments and events have the same relationships' nature with agents and resources; thus, they were designed using the same shape with the difference of the letters "E" and "C".

As with events and commitments, dualities and reciprocities were modeled using the same shapes with minor differences in the characters "D" and "R". The symbols which represent "increment", "decrement", "exchange", and "conversion" are shared among all elements of the visual notation, which would makes it easier for the notation users to get used to the notation. The "exchange" symbol was modeled as two opposing arrows to represent the "give and take" relationship, and the "conversion" symbol was depicted close to the "recycling" symbol to indicate the transformation from one state to another.

The other requirement of the visual notation, VNR1.2, simply suggested that the visual notation should cover all the meta-model's elements, which as seen in ***Figure 14*** was fulfilled.

Although the previous claims are legitimate, a proper evaluation should be based on an opinion other than the author's. For this reason, the requirement VNR1.1 cannot be considered as fulfilled, but rather as plausible. The second requirements VNR1.2 is of a quantitative nature; thus, it is safe to assume its fulfillment.

4.5.3 The REA-DSVL Editor

The requirements which were formulated for the editor were clear and simple. The editor's requirements targeted the very basic functionalities that should be available in any MDD editor. Requirements MTR1.1 and MTR1.2, which mandated the tool should provide functionalities for developing REA based diagrams and data-models, were fulfilled as it was described in section 4.2.4.

As seen in the data-models (***AppendixA.9***), the format of the generated files is XML. XML supports the development of Platform Independent Models (PIM) of MDA that were described in section 2.4. When thinking about code generation, using XML allows the implementation of the business in different programming languages and techniques. This in turn fulfills the requirement MTR1.3.

Although it is not a requirement of MDD, the tool was generated for two platforms, the Microsoft windows and Mac OS X. Eclipse's GMP allows generating the tool for Linux and Solaris platforms as well. This implementation fulfilled the requirement MTR1.5. This requirement was added to allow different platform users to use the editor on different operating systems; thus attracting a wider base of users.

The easy-to-use functionalities which were described in requirement MTR1.4 were assumingly fulfilled in different ways. The REA-DSVL Editor operates on two file types only. One of these files is directly manipulated by the user (the .rea_diagram file), and the other is auto-edited (the .rea file). This allows users to focus on their designs rather than the development of complex data-models.

As mentioned earlier in the section **4.3**, REA-DSVL Editor was used to model a real business scenario. The diagrams and data-models generated for the case study are listed in ***Appendix A.8*** and ***Appendix A.9*** respectively. When comparing the business description and the generated models, one can see that the tool managed to build diagrams that cover the complete business processes of the studied company. Based on the developed models of the case study, users of the editor may have the conception that their models should encapsulate the complete business in one diagram. Such technique can be used, thought it would result in complex and large diagrams. A typical technique for business modeling using the REA-DSVL Editor would be to divide the processes among several diagrams for simplicity. Either way, the REA DSVL Editor supports both modeling techniques.

As an example of complex diagrams, the first four processes from the case study were modeled in one diagram as shown in ***Figure 22***. As it appears in the figure, this diagram might be difficult to read. These difficulties can be overtaken by applying some changes to the appearance of the diagram elements. Events and dualities which belong to one process could be painted with unique colors using the properties view that was described in section **4.2.4**. When the latter modification is applied to the model in ***Figure 22***, the resulted model would look like the one in ***Figure 23***. This functionality of the editor supports the requirement indentified in MTR1.4 too.

Other editor functionalities like "links optimization", "validation", and "image generation" support the requirement MTR1.4. This requirement; though, cannot be considered as fulfilled because it was not evaluated based on an experiment like the one that was described in section **3.2.5**. Accordingly, this requirement is considered plausibly fulfilled.

Before moving to the conclusion chapter of this report, it is necessary to elaborate on the issue of the complex diagrams. The aforementioned issue with the representation part triggered another issue with data-models which are generated based on complex diagrams. In their current forms, different modeling diagrams generate different data-models. Accordingly, a business that is modeled using different diagrams would have its inner structure distributed among different data-models. The latter issue might result in redundant data objects if data-models were not interpreted carefully by system developers. This latter issue can be solved by defining an element that aggregates each process specifications. This would be achieved by implementing the process level specifications of REA. Such implementation would solve both the problem of the complex diagrams, and the problem of data-models' unity. The aforementioned solution is discussed in greater details in the last chapter of this report. The final set of the requirements and their status is provided in ***Table 5***.

Requirement Reference	Status	Requirement Reference	Status
MMR1	Fulfilled	MTR1	Fulfilled
MMR1.1	Fulfilled	MTR1.1	Fulfilled
MMR1.2	Fulfilled	MTR1.2	Fulfilled
VNR1	Fulfilled	MTR1.3	Fulfilled
VNR1.1	Plausible	MTR1.4	Plausible
VNR1.2	Fulfilled	MTR1.5	Fulfilled

Table 5: The final status of the requirements

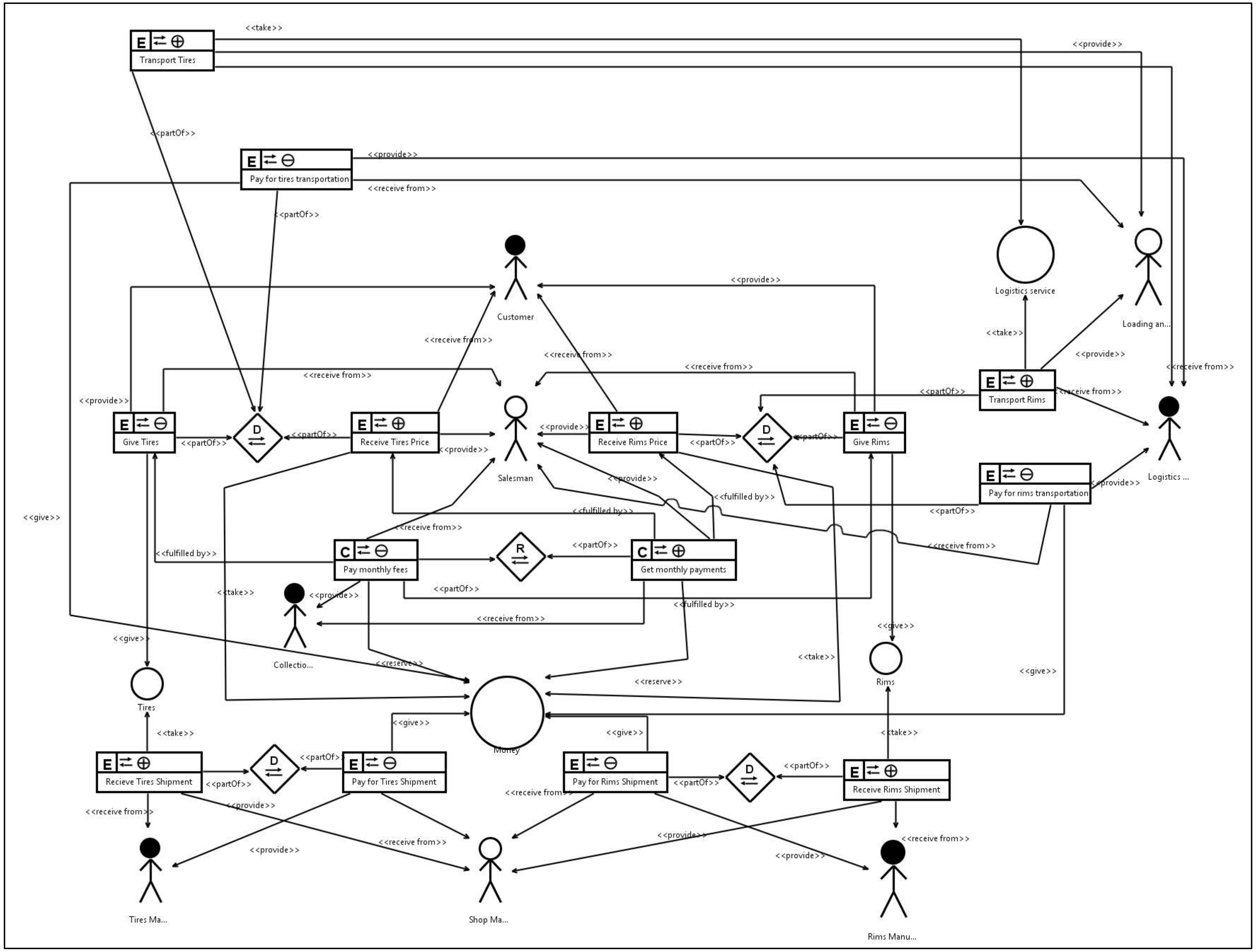


Figure 22: Multiple processes per diagram

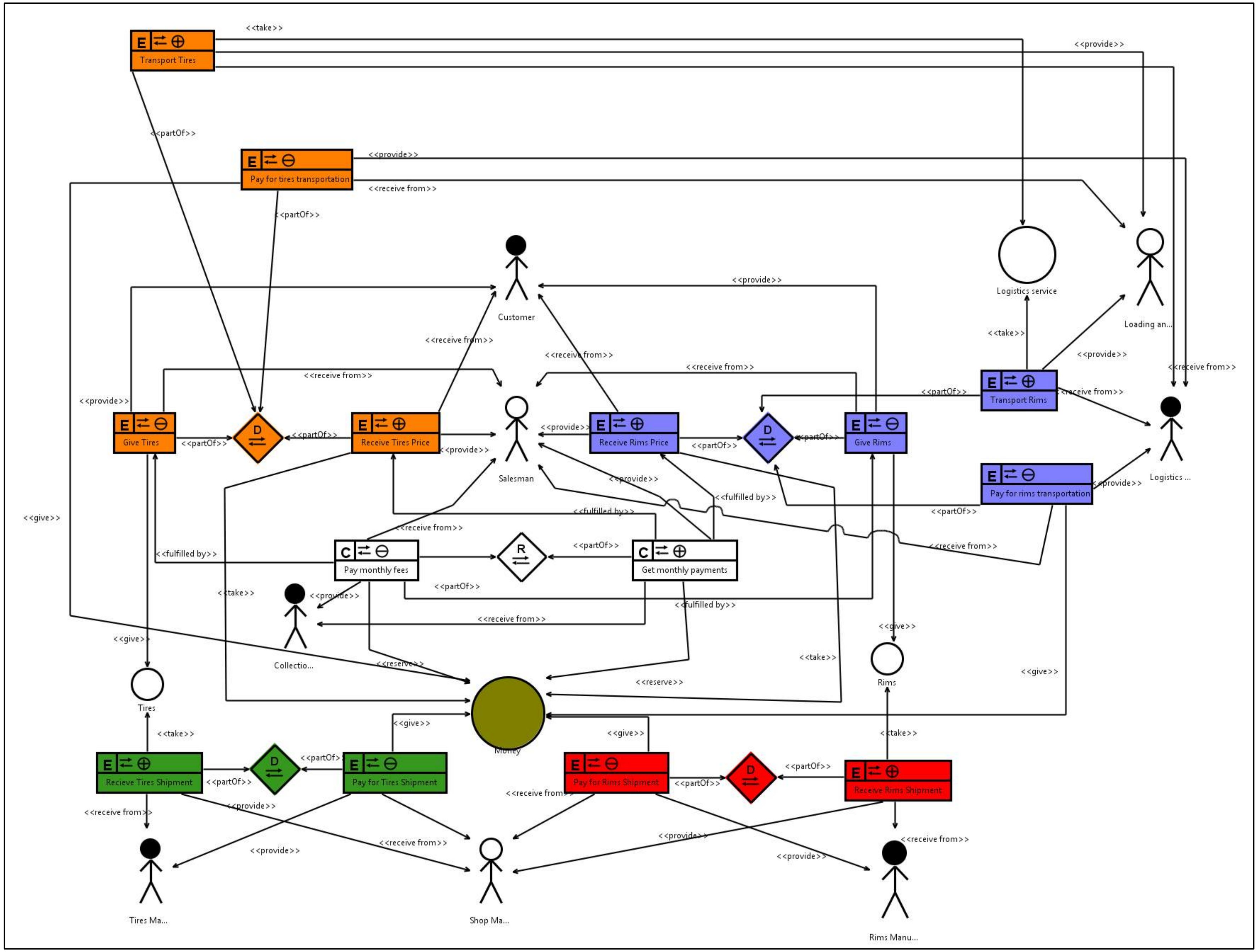


Figure 23: colored Multiple processes diagram

4.6 Results Validity and limitations

Based on the practical limitations identified earlier, the REA-DSVL Editor can be useful for modeling businesses of the same or smaller size than the business in the case study of this research. Larger business can still be modeled using the tool, though this might result in great number of diagrams and data-models.

The meta-model has covered its requirements, and when compared to other meta-models of REA, the meta-model provided additional details that other meta-models have missed. Saying that, the meta-model needs an evaluation based on the opinions of people who are experienced in the domain of REA modeling and meta-modeling.

The visual notation's requirements were simple and limited. This was reached due to the absence of literature resources dealing with visual composition. The same reason resulted in an evaluation of the visual notation symbols that might be biased.

The evaluation of the REA-DSVL editor was based on the analysis of one case study, and the perspective taken in the evaluation is of the language developer. Although the author has tried to provide a subjective analysis of the language, a better evaluation would have been provided if the experiment in section **3.2.5** was conducted. Such evaluation is needed for better judgment of the visual notation, and for collecting additional requirements from the users who might request functionalities that the author might not have thought about.

5. Conclusion

The developed tool proved that REA can be practically used in business modeling. The analysis of the language revealed that such DSVLs can indeed help in promoting the usage of REA in systems development; though, with its current limitations, the developed tool needs additional functionalities to make such goal even more practical.

Ordinary business owners with standard knowledge of software tools can be requested to model their businesses using the REA-DSVL Editor. This might provide an initial step toward engaging such owners in the process of software development for their business. Business analysts would have a better view of the business when the models are built by business owners, which in turn would lead to less analysis time, and more accurate analysis process.

The last point might hold true when the barrier of REA knowledge is neglected. The tool allows its users to connect the elements of their models according to the relations of the meta-model. In this sense, users are not allowed to makes "REA-errors". REA-errors in this sense means, for example, connecting some elements to the wrong set of other elements. Users, though, might miss some of the mandatory REA relations. These missing relations can be solved by using the validation functionality of the tool, which provides user-friendly messages that can direct the user toward solving such issues. Under these circumstances, users do not need to have an extensive knowledge of REA, but rather, a brief introduction to the tool itself and its functionalities.

The tool also allows business analysts to produce their correct and complete REA based business models. From there, they forward their ".rea" files to software architects. In this scenario, software architects will not need to interpret business documents and produce data-models anymore. The data-models would be generated accurately according to the designed business models. This might overcome some of the typical problems associated with the analysis and design phases of the SDLC; thus, it would typically lead to an accurate systems design, and less development time.

Implementing the process level specifications of REA in its meta-model would provide a base for representing the enterprise along its processes in one model. This would typically help in viewing the complete network of the *systems* around the modeled enterprise; thus, making any integrations easier.

Implementing the process level specifications would also make it easier to link the current REA-DSVL to other standards like BPMN. BPMN based MDD tools provide mechanisms for modeling direct systems implementations. The intervention of software developers would not be necessary for writing the actual software code in this case. The last implementation would bring the idea of MDD REA-based systems closer to reality.

This research has revealed that the research question can be answered by the main following points:

- A REA based DSVL tool can help in speeding up the analysis and design phases of the traditional SDLC. The previous advantages can be achieved by :
    - Automating some of the customary software design tasks. This is achieved by automatically generating data-models in the form of XML files for the modeled processes; thus, saving the time needed for transferring analysis documents to software objects.
    - Providing non-REA experts with simple modeling environments that they can understand; thus, attracting more users to the ontology. This would also support the practice of agile systems development.
- The research has also revealed that in order to support further phases of the SDLC using a REA-DSVL based modeling tool, the process level specifications of REA should be supported by the DSVL and the tool.

6. Discussion

This research provided a step toward implementing REA in business software using a model driven development technique. The research revealed that such goal can be achieved using REA based DSVL tools. Although the developed tool lacks the practicality to model large businesses, its core purpose was fulfilled, and REA based data-models were generated from visual models; therefore, providing a wider prospect of methods that can be used to implement REA as a true business architecture.

This project succeeded in proposing a meta-model, a visual notation, and a tool capable of generating data-models based on the previous two artifacts. Other works managed to provide one or two of these three artifacts under one work. Resources like (ZHANG, Guoqiang et al., 2010) (SONNENBERG, C. et al., 2011 b) (GAILLY, Frederik and Poels, Geert, 2007) managed to develop REA based meta-models. Regardless of the completeness or correctness of these meta-models, their work was limited to only proposing these meta-models. Other works managed to build direct implementations of REA as software packages; though this is far from the purpose of this project. Some works managed to build a meta-model and a visual notation based on REA like (SONNENBERG, C. et al., 2011 a); though, the tool produced for that language was a proof of concept, targeting the modeling perspective of the language, without supporting the generation of data-models. The same authors of REA-DSL have produced a different XML language (SONNENBERG, C. et al., 2011 b) based on the meta-model developed in their previous work. The purpose of the language was to provide a mechanism for REA based models transformation. Regardless of the problems that have been identified in the previous work (discussed in section **2.7**), the suggested XML schema requires its users to write their REA models using XML, which is quite a hard task for non-domain experts. The tool provided in this project; on the other hand, generates standard XML files that are based on an XML schema, and that XML schema was generated from the developed meta-model. In this sense, users do not need to worry about the complexity associated with writing XML files. On top of that, users have a visual notation that can be used for modeling business diagrams. Users of the visual notation do not need to have a profound knowledge of REA, as the tool provides functionalities to help users produce correct and complete business models.

One of the points which support the validity of the developed meta-mode of this work appears clearly in the work of (GAILLY, Frederik and Poels, Geert, 2005) and (GAILLY, Frederik and Poels, Geert, 2007). The authors of the aforementioned resources produced their first meta-model which depicted “events” of REA as a single meta-class. On their second meta-model at which they claimed a *wider* representation of REA over their first model, they have added a new level of details for “events” by suggesting “increment events” and “decrement events”. The meta-model of this research managed to provide this level of details, and added a further new level for facilitating the “exchange” and “conversion” levels of events specifications.

As for the limitations of this work, the DSVL tool was not evaluated properly. Away from the functionalities provided by the tool, a proper evaluation of the tool would reveal the limitations of the visual notation. It might be true that the visual notation at its current state covers all REA concepts, but its simplicity and expression suitability are best judged by users of the tool, and by visual designs experts.

Another limitation of this research is the extensibility of results. This limitation is associated with the analyses that were based on the analysis of one case study. Other evaluation techniques might reveal additional limitations of the DSVL, or new requirements for the tool. Such evaluation techniques should emphasize the involvement of technical personnel; who can judge the applicability of the generated artifacts to fulfill their designated purposes. An experiment like the one described in section **3.2.5** would provide a good baseline for evaluating both the visual notation and the editor.

Possible ethical and social consequences of the conclusions would obviously include cases when any of the developed artifacts is used practically. As mentioned in the conclusion chapter, the editor supports faster planning and design phases of the SDLC. The previous conclusion holds true when a careful interpretation of the generated data-models is done. Due to the separation of REA models, the generated data-models will contain different objects for the same entities which have been used between different REA models. If not carefully managed, these entities could be represented in different *system* objects (DB objects, Java objects…etc), which might result in poor systems designs.

Another important consequence of the conclusions is their suitability for ontologies like REA. The conclusions were based on implementing the REA ontology. REA has special characteristics as it was designed on the first place to support the development of information systems. Other business ontologies like; e3-value, BMO, or any other business modeling ontology might need its own careful study in order to reach the same conclusions. It is a common mistake to think that any business modeling ontology should have the same conclusion as REA's; though, this might be true for some business modeling ontologies, but not others.

The future work that can be built on top of this research would include evaluating the meta-model and the visual notation in a better way. This includes performing the experiment that was described in section **3.2.5**. One major work would include expanding the REA-DSVL and the editor to support large business diagrams. This can be achieved by using "diagram partitioning" technique of GMP. In its current specifications, the tool builds all REA processes in one diagram, and the representation of a process as an entity is abstract as processes are implicitly modeled using REA events and commitments. The new solution suggests providing a dedicated element for REA processes, and this new element would typically encapsulate its associated REA constructs. ***Figure 24*** depicts how such solution would solve some of the identified language and editor issues.

All the previous suggested work was based on the analyses' results of this research. A typical future work; though, should focus on the main purpose of this research, which is the implementation of REA in developing business information systems.

This research has concluded that DSVLs can help in engaging REA in software development. Future work can proof that a REA based DSVL would reach the ultimate goal of MDD, by building software based completely on models. At the beginning, this might sound like a long shot, but practically it is not. The following is a brief discussion of how this can be achieved.

The first step of such a solution would be to build a database infrastructure which is based on REA. Up until now, this task cannot be reached without human intervention. A proof of concept was conducted (aside from this research) in the form of converting the generated ".rea" files into java objects. This was achieved by using a method similar to the one found in (MCNEILL, Ken, 2010) with some modifications. Using the same functionality, one can annotate the generated java classes as JPA[1] objects. From there, generating the database structure can be easily automated using models.

The next step would be to define the business logic. As known, REA is a static ontology that does not support flow of events. Luckily, Eclipse has a running proposal[2] to extend its previous support of BPMN 1.0 to version 2.0. Both of Eclipse's BPMN specifications are based on EMF, which at its core is based on Ecore. As described earlier, models transformation would be an easy task in this case as both languages (REA-DSVL and Eclipse's BPMN 2.0) are based on the same meta-meta-language, Ecore. The latter operation would typically result in business process models based on the original ".rea_diagram". Running the new business process models using Eclipse's engine would result in an operational REA processes; thus, a complete REA based MDD implementation.

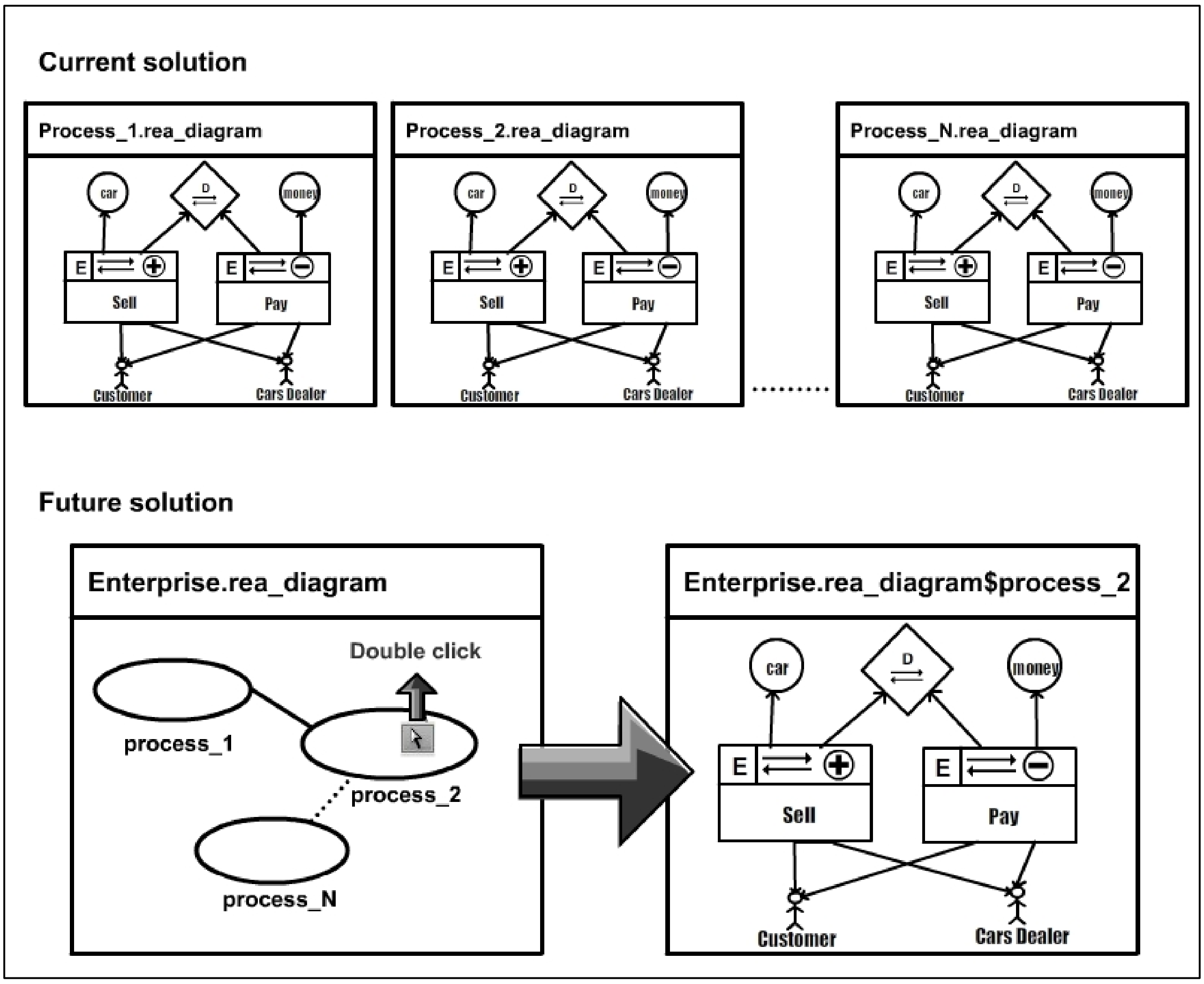


Figure 24: Current and future solutions

These were some of the future contributions that could be built on top of this project. Other implementations of the DSVL would include choreographing the DSVL with other aspects of

[1] *http://docs.oracle.com/javaee/5/tutorial/doc/bnbqa.html#bnbqb*

[2] *http://www.eclipse.org/proposals/soa.bpmn2-modeler/*

enterprise strategy and planning techniques, which would result in a professional enterprise package that incorporates REA; thus, promoting the ontology as a major player in the domain of enterprises information systems.

Bibliography

1. ATKINSON, Colin and Thomas KÜHNE. 2003. Model-Driven Development: A Metamodeling Foundation. *IEEE Software*. **20**(5), pp.36-41.
2. B. YU, J.A. Harding, K. Popplewell. 2000. A reusable enterprise model. *International Journal of Operations & Production Management*. **20**(1), pp.50 - 69.
3. BHATTACHERJEE, Anol. 2012. Data Collection Strategies. *In*: *SOCIAL SCIENCE RESEARCH:PRINCIPLES, METHODS, AND PRACTICES*, pp.83-103.
4. CORALLO, Angelo, Fabrizio ERRICO, Marco DE MAGGIO, and Enza GIANGRECO. A methodology aimed at fostering and sustaining the development processes of an IE-based industry. *In*: *Evolving Towards the Internetworked Enterprise*, Springer, pp.28-30.
5. DENSCOMBE, Martyn. 2007. *The Good Research Guide for small-scale social research projects*.
6. ECLIPSE. 2012. *Eclipse Modeling Project*. [online]. [Accessed Feb 2012]. Available from World Wide Web: <http://www.eclipse.org/modeling/>
7. ECLIPSE.ORG. *Graphical Modeling Framework*. [online]. [Accessed 17 May 2012]. Available from World Wide Web: <http://wiki.eclipse.org/Graphical_Modeling_Framework/Tutorial#Get_started>
8. GAˇSEVI´C, Dragan, Dragan DJURI´C, and Vladan DEVEDˇZI´. 2006. *Model Driven Architecture and Ontology Development*. Berlin: Springer.
9. GAILLY, Frederik and Geert POELS. 2005. Development of a formal REA-ontology Representation. *In*: Michele MISSIKOFF and Antonio DE NICOLA, (eds). *CEUR Workshop Proceedings*. Porto (Portugal): CEUR-WS.org.
10. GAILLY, Frederik and Geert POELS. 2007. *Ontology-driven Business Modelling: Improving the Conceptual Representation of the REA Ontology*. Belgium.
11. GEERTS, Guido L. and William E. MCCARTHY. 2000. *The Ontological Foundation of REA Enterprise Information Systems*. Paper presented to the American Accounting Association Conference, Philadelphia.
12. GEERTS, Guido L. and William E. MCCARTHY. 2002. An ontological analysis of the economic primitives of the extended-REA enterprise information architecture. *International Journal of Accounting Information Systems*. **3**(1), p.1–16.
13. GEERTS, Guido L. and William E. MCCARTHY. 2006. Policy-Level Specifications in REA Enterprise Information Systems. *JOURNAL OF INFORMATION SYSTEMS*. **20**(2), p.37–63.
14. GEERTS, Guido L., William E. MCCARTHY, and Stephen R. ROCKWELL. 1996. Automated integration of enterprise accounting models throughout the systems development life cycle. *Intelligent Systems in Accounting, Finance and Management*. **5**(3), p.113–128.
15. GOCSIK, Karen. 2005. *Dartmouth Writing Program*. [online]. [Accessed July 2012]. Available from World Wide Web: <http://www.dartmouth.edu/~writing/materials/student/ac_paper/what.shtml#argument>
16. GONZALEZ-PEREZ, Cesar and Brian HENDERSON-SELLERS. 2008. Software Development Methodologies and Metamodelling. *In*: *Metamodelling for Software Engineering*, pp.1-20.
17. GORDIJN, Jaap. 2002. *E3value in a Nutshell*. Lausanne , Switzerland.
18. GORDIJN, J., E. YU, and B. VAN DER RAADT. 2006. E-service design using i* and e3-value modeling. *Software, IEEE*. **23**(3), pp.26-33.
19. GUDAS, Saulius, Audrius LOPATA, and Tomas SKERSYS. 2005. Approach to Enterprise Modelling for Information Systems Engineering. *INFORMATICA*. **16**(2), pp.175-192.
20. GUIZZARDI, Giancarlo. 2006. On Ontology, ontologies, Conceptualizations,Modeling Languages, and (Meta)Models. *In*: *Proceedings of the 2007 conference on Databases and Information Systems IV: Selected Papers from the Seventh International Baltic Conference.*, pp.18-39.
21. HEVNER, Alan R. and Samir CHATTERJEE. 2010. *Design research in information systems : theory and practice*. New York: Springer.

22. HEVNER, Alan R., Salvatore T. MARCH, Jinsoo PARK, and Sudha RAM. 2004. Design science in Information Systems research. *MIS Quarterly*. **28**(March), pp.p.75 -105.

23. HOSKING, John and John GRUNDY. 2007. Meta tools for implementing domain specific visual languages. *In*: *roceedings of the twenty-second IEEE/ACM international conference on Automated software engineering*. ACM.

24. HRUBY, Pavel, Jesper KIEHN, and Vibe SCHELLER. 2005. *Model-Driven Design Using Business Patterns*. Berlin: Springer.

25. LAFORCADE, Pierre. 2010. A Domain-Specific Modeling approach for supporting the specification of Visual Instructional Design Languages and the building of dedicated editors. *Journal of Visual Languages & Computing*.

26. LI, Karen, John HOSKING, John GRUNDY et al. 2010. Augmenting DSVL Meta-Tools with Pattern Specification, Instantiation and Reuse. *In*: *Proceedings of the Second International Workshop on Visual Formalisms for Patterns*.

27. MCCARTHY, William E. 1982. The REA Accounting Model: A Generalized Framework for Accounting Systems in a Shared Data Environment. *The Accounting Review*. **57**(3), pp.554-578.

28. MCNEILL, Ken. 2010. *Metamodeling with EMF: Generating concrete, reusable Java snippets*. [online]. [Accessed Apr 2012]. Available from World Wide Web: <http://www.ibm.com/developerworks/library/os-eclipse-emfmetamodel/>

29. MU, Liping, Terje GJøSÆTER, Andreas PRINZ et al. 2010. Specification of modelling languages in a flexible meta-model architecture. *In*: *Software Architecture: Proceedings of the Fourth European Conference*., pp.302-308.

30. OMG. 2003. *MDA Guide Version 1.0.1*. [online]. Available from World Wide Web: <http://www.omg.org/cgi-bin/doc?omg/03-06-01.pdf>

31. OMG. 2011. *OMG Meta Object Facility (MOF) Core Specification*. formal/2011-08-07.

32. OSTERWALDER, Alexander and Yves PIGNEUR. *Business Model Generation*.

33. PEFFERS, Ken, Tuure TUUNANEN, Marcus A. ROTHENBERGER, and Samir CHATTERJEE. 2007. A Design Science Research Methodology for Information Systems Research. *Journal of management information systems*. **4**(3), pp.45-77.

34. RAHMOUNI, M. and Lakhoua, M.N. 2011. State of the art of enterprise modeling. *In*: *Logistics (LOGISTIQUA), 2011 4th International Conference on*., pp.311 -316.

35. SÁNCHEZ, Diana Marcela, José María CAVERO, and Esperanza Marcos MARTÍNEZ. 2007. The Road Toward Ontologies. *In*: *Ontologies for Software Engineering and Software technology*, Springer, pp.3-20.

36. SCHAFER, Christian, Thomas KUHN, and Mario TRAPP. 2011. A Pattern-based Approach to DSL Development. *In*: *The 11th Workshop on Domain-Specific Modeling*. Portland.

37. SCHUSTER, R. and T. MOTAL. 2009. From e3-value to REA: Modeling multi-party eBusiness Collaborations. *In*: *Commerce and Enterprise Computing, 2009. CEC '09. IEEE Conference on*., pp.202-208.

38. SONNENBERG, C., C. HUEMER, B. HOFREITER, and D. MAYRHOFER. 2011 b. REA-XML: An Unambiguous Language for REA Business Models. *e-Business Engineering (ICEBE), 2011 IEEE 8th International Conference on*., pp.44-51.

39. SONNENBERG, C., C. HUEMER, B. HOFREITER et al. 2011 a. The REA-DSL: A Domain Specific Modeling Language for Business Models. *In*: *Advanced Information Systems Engineering: 23rd International Conference, CAiSE 2011*. London, UK, pp.252-266.

40. SPRINKLE, Jonathan and Gabor KARSAI. 2004. A domain-specific visual language for domain model evolution. *Journal of Visual Languages & Computing*. **15**(3–4), p.291–307.

41. STAAB, Steffen, Tobias WALTER, Gerd GRÖNER, and Fernando PARREIRAS. 2010. Model Driven Engineering with Ontology Technologies. *In*: Uwe AßMANN, Andreas BARTHO, and Christian WENDE, (eds). *Reasoning Web. Semantic Technologies for Software Engineering*, Berlin / Heidelberg: Springer, pp.62-98.

42. VAN DEURSEN, Arie, Paul KLINT, and Joost Norris VISSER. 2000. Domain-specific languages: an annotated bibliography. *ACM SIGPLAN Notices*. **35**(6), pp.26-36.

43. VOGEL, Lars. 2007. *Eclipse RCP Tutorial*. [online]. [Accessed April 2012]. Available from World Wide Web: <http://www.vogella.com/articles/EclipseRCP/article.html>

44. WAND, Yair. 1996. Ontology as a foundation for meta-modelling and method engineering. *Information and Software Technology*. **38**(4), pp.281-287.

45. ZHANG, Guoqiang, Suling JIA, Qiang WANG, and Qi LIU. 2010. REA-based Enterprise Business Domain Ontology Construction. *Journal of Software*. **5**(5), p.522.

46. ZOUGGAR, N., D. CHEN, and B. VALLESPIR. 2009. Semantic Enrichment of Enterprise Modelling – Use of Ontology. *In*: *Interoperability for Enterprise Software and Applications China, 2009. IESA '09. International Conference on*., pp.252 -258.

Appendix A

A.1 Literature Resources

1. The Practice of Enterprise Modeling. Paul Johannesson; John Krogstie; Andreas L. Opdahl
2. Conceptual Modeling for Advanced Application Domains. ER 2004 Workshops CoMoGIS, CoMWIM, ECDM, CoMoA, DGOV, and eCOMO. Shanghai, China, November 8-12, 2004 Proceedings
3. Enterprise Modeling and Computing with UML. Peter Rittgen
4. Lecture Notes in Computer Science Commenced Publication in 1973. Gerhard Goos; Juris Hartmanis; Jan van Leeuwen
5. Approach to Enterprise Modelling for Information Systems Engineering. Saulius GUDAS; Audrius LOPATA
6. Formal methods in object oriented business modelling. Michalis Glykas; George Valiris
7. Semantic Enrichment of Enterprise Modelling – Use of Ontology. Nabila Zouggar; David Chen; Bruno Vallespir
8. State of the Art of Enterprise Modeling. Mouna Rahmouni; Mohamed Najeh Lakhoua
9. Success factors for strategic information systems. Helmut Krcmar; Henry C. Lucas, Jr.
10. E3-value in a Nutshell. Jaap Gordijn
11. e-Service Design Using i* and e3value Modeling. Jaap Gordijn; Eric Yu; Bas van der Raadt
12. Evolving Towards the Internetworked Enterprise : Technological and Organizational Perspectives. Ronald Maier
13. Business Model Generation. Alexander Osterwalder; Yves Pigneur
14. A Design Science Research Methodology for Information Systems Research. KEN PEFFERS; TUURE TUUNANEN; MARCUS A. ROTHENBERGER; SAMIR CHATTERJEE
15. A domain-specific visual language for domain model evolution. Jonathan Sprinklea; Gabor Karsaib
16. Meta Tools for Implementing Domain Specific Visual Languages. John Hosking; John Grundy
17. On Marrying Ontological and Metamodeling Technical Spaces. Fernando Silva Parreiras; Steffen Staab; Andreas Winter
18. A meta-model for formulating knowledge-based models of software development. Peiwei Mi; Walt Scacchi
19. Metamodeling in EIA/CDIF—Meta-Metamodel and Metamodels. RONY G. FLATSCHER
20. Evaluation of Novel Approaches to Software Engineering. Leszek A. Maciaszek; César González-Pérez; Stefan Jablonski (Eds.)
21. A powertype-based metamodelling framework. Cesar Gonzalez-Perez; Brian Henderson-Sellers
22. The Rationale of Powertype-based Metamodelling to Underpin Software Development Methodologies. Brian Henderson-Sellers; Cesar Gonzalez-Perez
23. Software Development Methodologies and Metamodelling. Cesar Gonzalez-Perez, (Book chapter)
24. Ontology as a foundation for meta-modelling and method engineering. Yair Wand
25. Specification of Modelling Languages in a Flexible Meta-model Architecture. Liping Mu; Terje Gjøsæter; Andreas Prinz
26. Modelling software development methodologies: A conceptual foundation. Cesar Gonzalez-Perez; Brian Henderson-Sellers
27. A Meta Model for Representing Arbitrary Meta Model Hierarchies. Bernhard Volz
28. A Taxonomy of Metamodel Hierarchies. Ralf Gitzel; Tobias Hildenbrand
29. Metamodeling with Eclipse. Luis Pedro; Matteo Risoldi
30. A Pattern-based Approach to DSL Development. Christian Sch¨afer; Thomas Kuhn; Mario Trapp

31. From **domain** ontologies to modeling ontologies to executable simulation models. Gregory A. Silver; Osama Al-Haj Hassan; John A. Miller

32. An ontological analysis of the economic primitives of the extended-REA enterprise information architecture. Guido L. Geerts; William E. McCarthy

33. From e3-value to REA: Modeling multi-party eBusiness Collaborations. Rainer Schuster; Thomas Motal

34. Positioning REA as a business domain ontology. Frederik Gailly; Wim Laurier; Geert Poels

35. REA-based Enterprise Business Domain Ontology Construction. Guoqiang Zhang; Suling Jia; Qiang Wang; and Qi Liu

36. REA-XML: An unambiguous language for REA business models. Dieter Mayrhofer; Christian Huemer; Birgit Hofreiter; Christian Sonnenberg

37. The Ontological Foundation of REA Enterprise Information Systems. Guido L. Geerts

38. The REA-DSL: A Domain Specific Modeling Language for Business Models. C. Sonnenberg; C. Huemer; B. Hofreiter; D. Mayrhofer; A. Braccini

39. Innovations in Information Systems Modeling: Methods and Best Practices. Terry Halpin; John Krogstie; Erik Proper

40. An empirical study on the efficiency of different design pattern representations in UML class diagrams. Gerardo Cepeda Porras ; Yann-Gaël Guéhéneuc

41. Business modelling withUML: the implementation of CRM systems for online retailing. Pauline A. Wilcoxa

42. EMF: Eclipse Modeling Framework,Second Edition. Dave Steinberg; Frank Budinsky; Marcelo Paternostro; Ed Merks

43. Business Modeling for Service Descriptions:A Meta Model and a UML Profile. Gregor Scheithauer; Guido Wirtz

44. Model-Driven Design Using Business Patterns. Pavel Hruby ; Jesper Kiehn ; Christian Vibe Scheller

45. Electronic Communications of the EASST Volume X (2010) : Proceedings of the Second International Workshop on Visual Formalisms for Patterns.

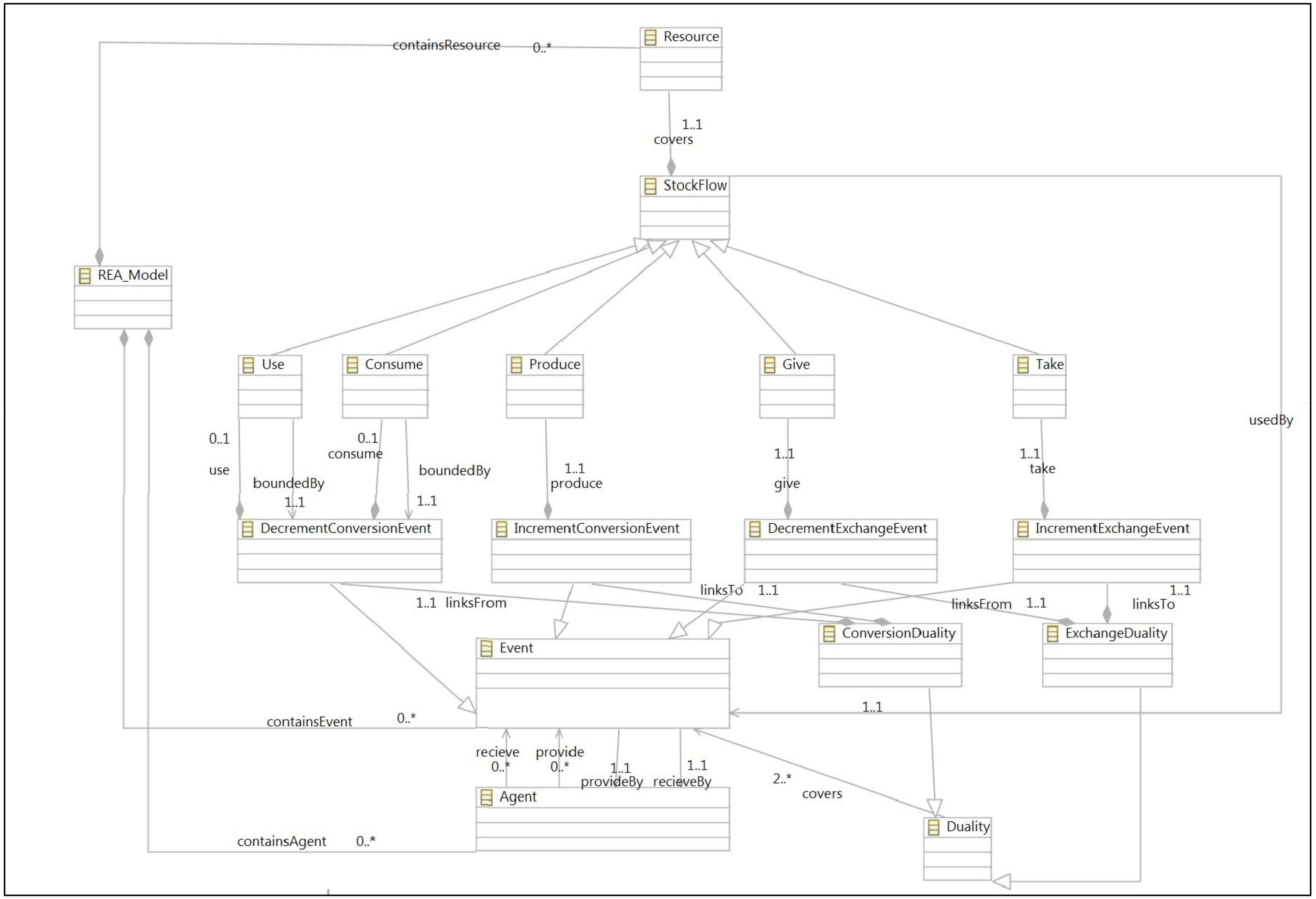
46. Domain-Specific Languages: An Annotated Bibliography. Arie van Deursen; Paul Klint; Joost Visse

A.2 REA elements relationships matrix

Elements	Event	Resource	Agent	Commitment	Term	Contract	Resource Type
Event	Duality <*,*>	Give <1,1> Take <1,1> Consume <1,0..1> Use <1,0..1> Produce <1,1>	Provide <1,1> Receive from <1,1>	Fulfillment <0,*>	-	-	-
Resource	Given to <0,*> Taken from <0,*> Consumed by <0,*> Used by <0,*> Produced by <0,*>	-	-	Reserved By <0,*>	-	-	Specified By <0,*>
Agent	Provide <0,*> Receive from <0,*>	-	-	Provide <0,*> Receive from <0,*>	-	Party <0,*>	-
Commitment	Fulfillment <1,*>	Reserves <0,*>	Provide <1,1> Receive from <1,1>	Reciprocity <*,*>	Initiated by <0,1>	Initiated by <0,*>	Reserves <0,*>
Term	-	-	-	Initiates <1,*>	-	Initiated by <1,1>	-
Contract	-	-	Has Parties <2,*>	Initiates <2,*>	Initiates <2,*>	-	-
Resource Type	-	Specifies <0,*>	-	Reserved by <0,*>	-	-	-

- This table is read from left to right.
- The numbers represent cardinalities associated with the corresponding relationship.
- A relation is pronounced as (Row Element (RE) + Relationship (R) + Cardinality (C) + Column Element (CE)).
- The highlighted example relation is pronounced as “**Resource Type**(RE) is **reserved by**(R) **0 or more** (C) **Commitments** (CE)”.

A.3 First basic meta-mode



A.4 REA-DSVL elements relationship s matrix

Meta-model Elements	IEE	DEE	ICE	DCE	IEC	DEC	ICC	DCC	ED	CD	ER	CR	Internal Agent	External Agent	Resource	Resource Type	Increment Term	Decrement Term
IEE									PO				P	RF	T			
DEE									PO				RF	P	G			
ICE										PO			PB		PROD			
DCE										PO			PB		C,U			
IEC	F										PO		P	RF	Reserves	Reserves		
DEC		F									PO		RF	P	Reserves	Reserves		
ICC			F									PO	PB		Reserves	Reserves		
DCC				F								PO	PB		Reserves	Reserves		
Resource Type															Specifies			
Increment Term					I													
Decrement Term						I												
Contract																	has	has

Symbol	Meaning	Symbol	Meaning
IEE	Increment Exchange Event	**PROD**	Produce
DEE	Decrement Exchange Event	**U**	Use
ICE	Increment Conversion Event	**C**	Consume
DCE	Decrement Conversion Event	**PB**	Processed By
IEC	Increment Exchange Commitment	**I**	Initiates
DEC	Decrement Exchange Commitment	**G**	Give
ICC	Increment Conversion Commitment		
DCC	Decrement Conversion Commitment		
ED	Exchange Duality		
CD	Conversion Duality		
ER	Exchange Reciprocity		
CR	Conversion Reciprocity		
F	Fulfillment		
PO	Part Of		
P	Provide		
RF	Receive From		
T	Take		

- The relations are pronounced in the same way as in A.1.
- This table contains only the relations that appear in the meta-model and implementation of REA-DSVL.

A.5 Visual implementation classes

Class Name	Super Class	Referenced shape
Agent	-	-
ExternalAgent	Agent	
InternalAgent	Agent	
Contract	-	Contract
Commitment	-	-
IncrementExchangeCommitment	Commitment	Increment Exchange Commitment
DecrementExchangeCommitment	Commitment	Decrement Exchange Commitment
IncrementConversionCommitment	Commitment	Increment Conversion Commitment
DecrementConversionCommitment	Commitment	Decrement Conversion Commitment
Event	-	-
IncrementExchangeEvent	Event	Increment Exchange Event
DecrementExchangeEvent	Event	Decrement Exchange Event
IncrementConversionEvent	Event	Increment Conversion Event
DecrementConversionEvent	Event	Decrement Conversion Event
Term	-	-
DecrementTerm	Term	
IncrementTerm	Term	
Resource	-	Resource
RsourceType	-	Rsource Type
BindingShape	-	-
ExchangeDuality	BindingShape	Exchange Duality
ConversionDuality	BindingShape	Conversion Duality
ExchangeReciprocity	BindingShape	Exchange Reciprocity
ConversionReciprocity	BindingShape	Conversion Reciprocity

A.6 ABS Wheels business

“ABS Wheels” is a high-level distributor; this means that the company imports its storage of tires and rims directly from manufacturers. When the company buys what it needs, the full payment is done at the time of order, this means that the manufacturer will be prepare the shipment as soon as payment is done. Such deals are typically handled by the “ABS Wheels” manager directly.

Rims and tires trading follow similar processes. The process starts when customers make their reservations either online, or through the salesman in the office by phone. Customers can pay directly through the company’s website or directly to the salesman if they were in-shop customers. If the customer wants to pay on partial payments, then the company provides the option of partial payments through a 3thd party collection company. This option is also available for online users. In the latter case of partial payments, “ABS Wheels” pays a monthly percentage to the collection company.

When an order is done over the Internet or by phone, the customer will be requested to choose a delivery option, the customer can either come to the company’s store and pick up the order, or pay an extra fee for order shipment. If the customer chooses to pay for shipment, the company will send a request to one of the available logistics companies and pay it to deliver the shipment for the customer.

When an order is finalized (either through Internet, phone or in-shop), the company’s employees who are responsible for mounting and balance prepare the order from the company’s warehouse. The company signs contracts with its employees. It pays them monthly based salaries

Auto-repairing services consist of changing cars oil and cars washing. These tasks are done by dedicated company employees. The raw materials needed for these tasks like; oil, oil filters, cleaning liquids…etc; are bought from spare-parts shops. The payment from for these raw materials is done partially over predefined time. The customer who receives the auto-repair service pays directly after receiving the service.

The company is renting its office and warehouse from a property owner. It has a contract signed with the property owner, and it pays the rent on a monthly basis through a bank. The company also pays monthly invoices to the telephone, Internet, electricity, and water companies.

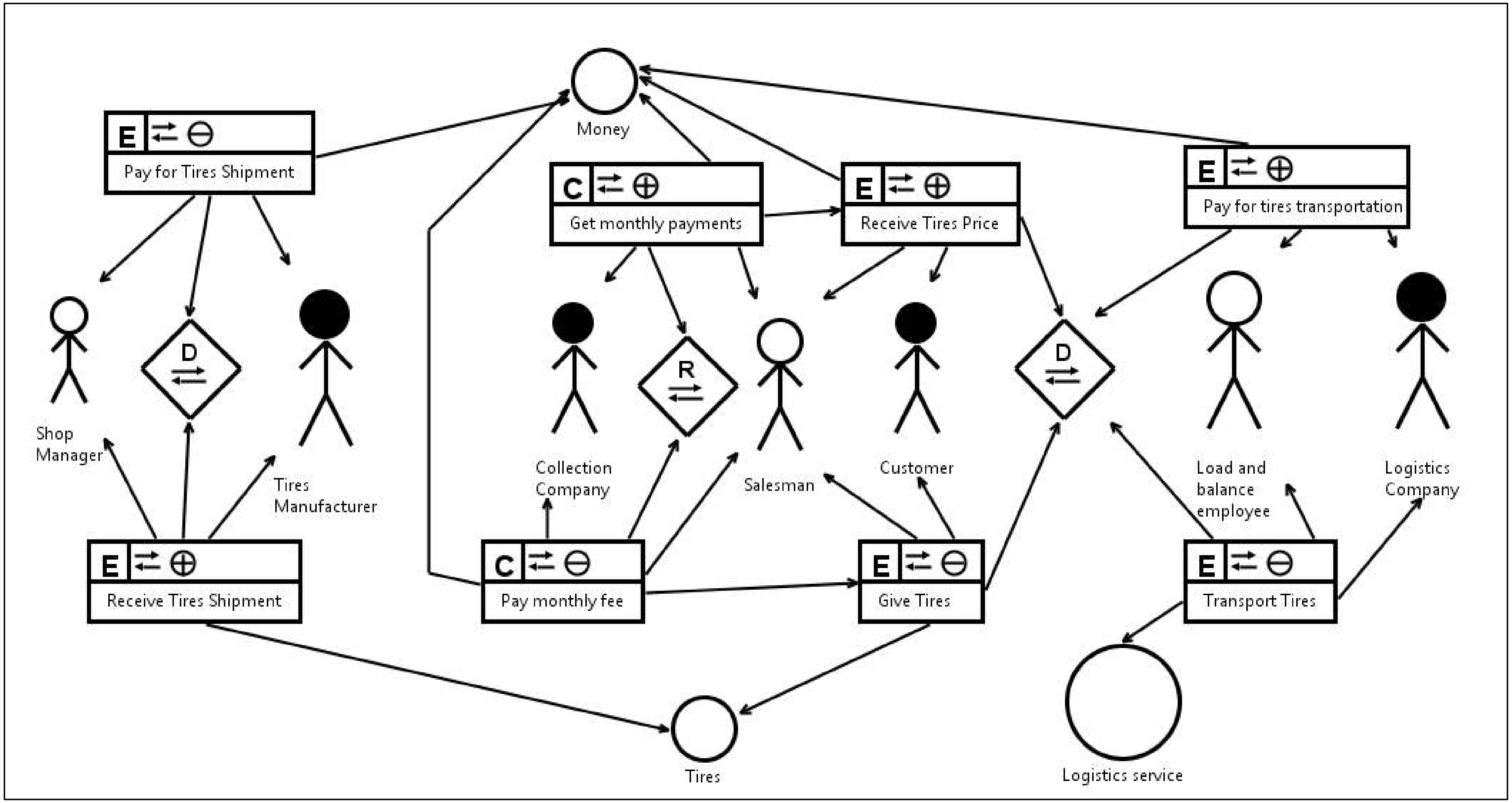
The company occasionally places advertisements for its business on radio stations, magazines, and Internet websites. The payment for such advertisements is done at the time when the ad is requested.

A.7 ABS Wheels’ REA Processes

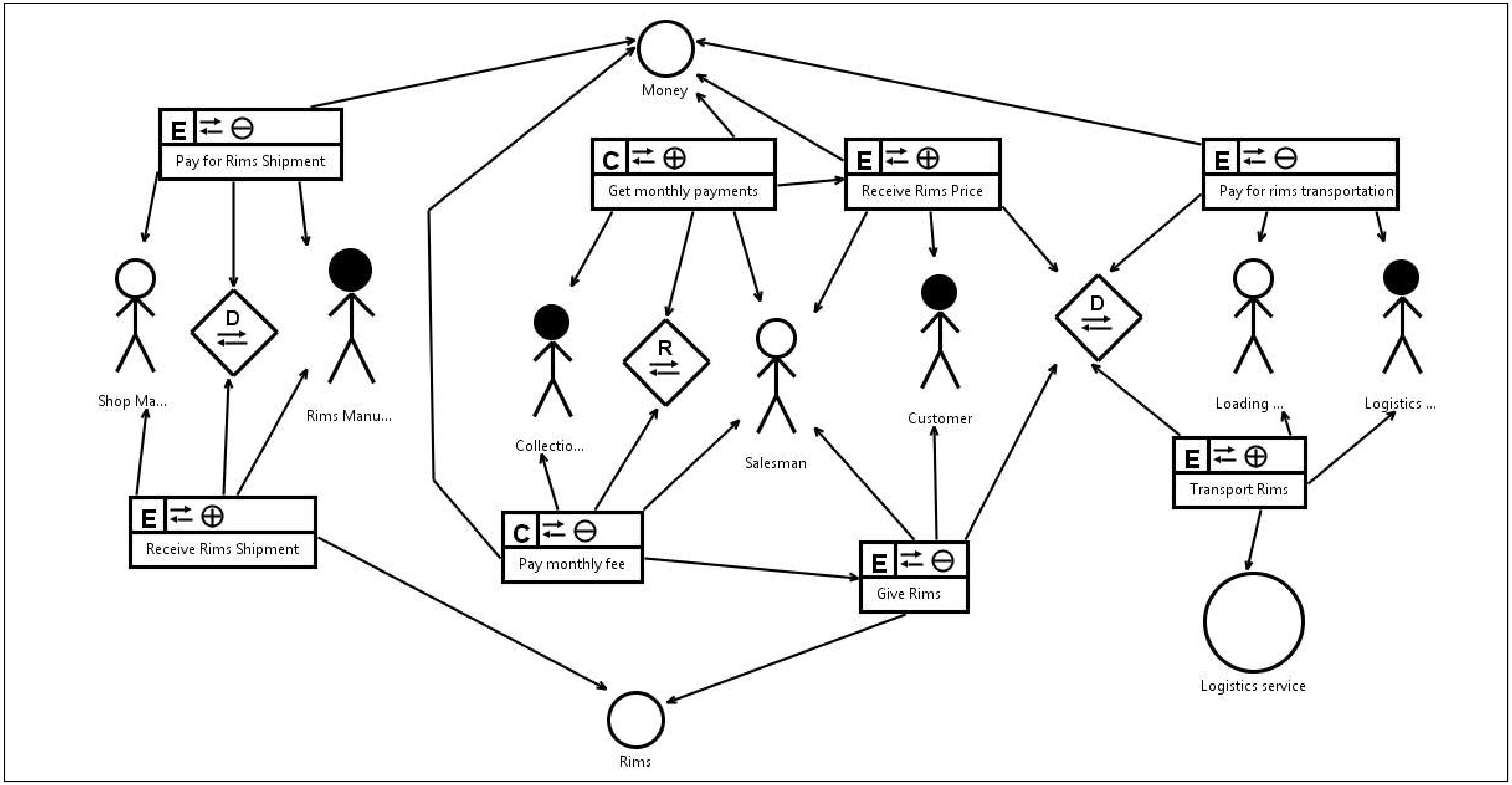
Process	Agents	Resources	Events	Commitments
Rims Importing	• Rims manufacturer (E) • Shop manager (I)	• Rims • Money	• Pay for Rims Shipment (DEE) • Receive Rims Shipment (IEE)	
Tires Importing	• Tires manufacturer (E) • Shop manager (I)	• Tires • Money	• Pay for Tires Shipment (DEE) • Receive Tires Shipment (IEE)	
Rims sale	• Customer (E) • Collection company (E) • Logistics company (E) • Salesman (I) • Loading and balance employee (I)	• Logistics Service • Money • Rims	• Give Rims (DEE) • Receive Rims Price (IEE) • Pay for rims transportation (DEE) • Transport Rims (IEE)	• Pay monthly fees (DEC) • Get monthly payments (IEC)
Tires sale	• Customer (E) • Collection company (E) • Logistics company (E) • Salesman (I) • Loading and balance employee (I)	• Logistics Service • Money • Tires	• Give Tires (DEE) • Receive Tires Price (IEE) • Pay for tires transportation (DEE) • Transport Tires (IEE)	
Spare parts acquirement	• Spare parts provider (E) • Salesman (I)	• Spare parts • Money	• Pay for spare parts (DEE) • Get spare parts (IEE)	
Auto services	• Customer (E) • Service worker (I) • Salesman (I)	• Auto-service • Money • Spare Parts	• Consume parts in service (DCE) • Apply Service to Vehicle (ICE) • Provide auto-service (DEE) • Get service fees (IEE)	
Place Renting	• Property owner (E) • Bank (E) • Shop Manager (I)	• ABS Wheels Building • Money	• Pay rent (DEE) • Get The property (IEE)	• Pay Monthly Rent (DEC) • Get monthly ownership (IEC) • Rental Contract (***Contract***)
Maintenance	• Service provider (E) • Bank (E) • Salesman (I) • ABS Wheels(I)	• Building Requirements • Money	• Pay for services (DEE) • Get services (IEE)	• Pay monthly fees (DEC) • Get Building Running needs (IEC)
Advertising	• Advertising channel (E) • ABS wheels (I) • Shop manager(I)	• Advertisement • Money	• Pay for advertisements (DEE) • Get advertised (IEE)	
Employment	• Service provider (E) • Shop manager(I)	• Labor Service • Money	• Pay salary (DEE) • Get labor service (IEE)	• Pay monthly salary (DEE) • Reserve labor service (IEC) • Employment Contract (***Contract***)

A.8 ABS Wheels' REA models

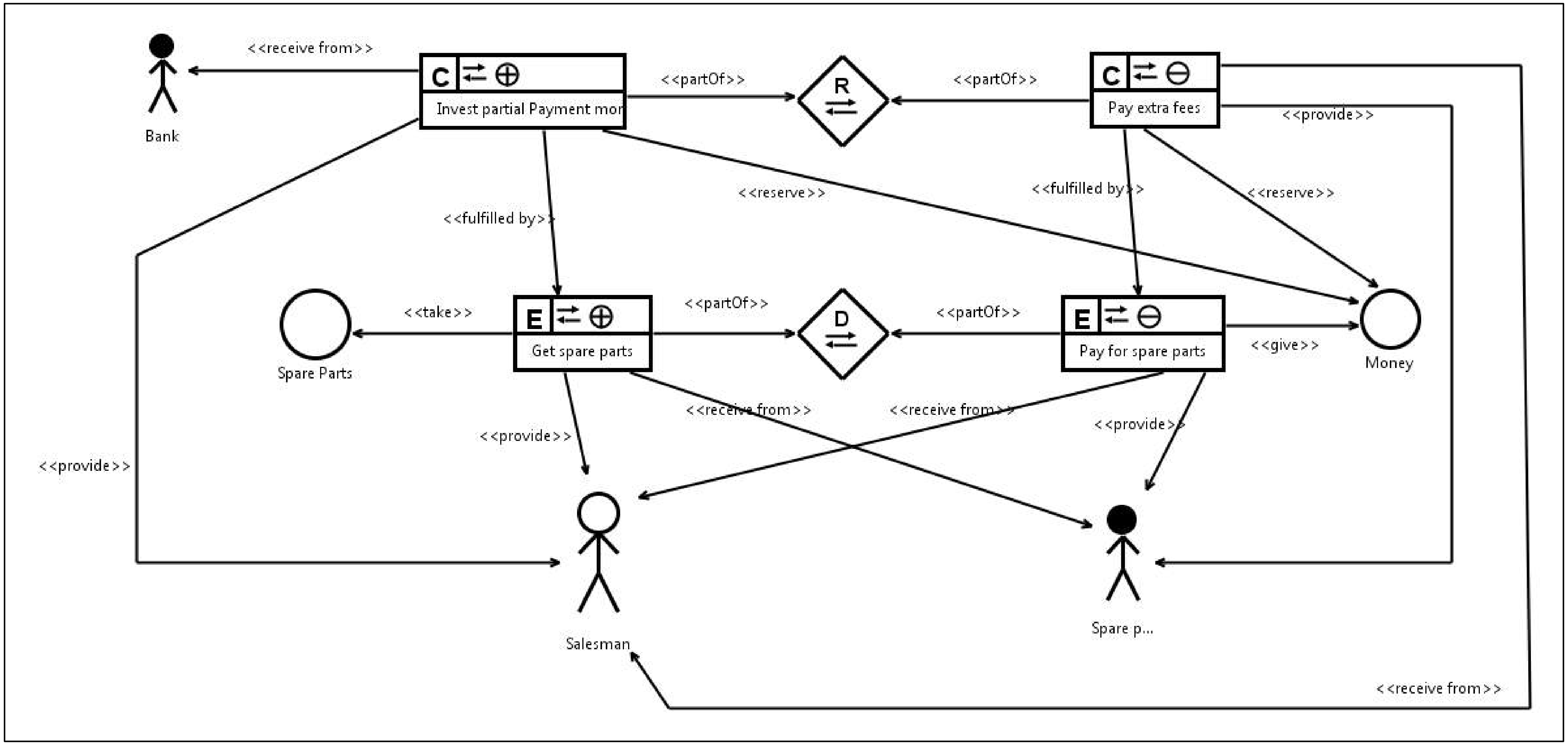
- **Tires importing and sales processes**



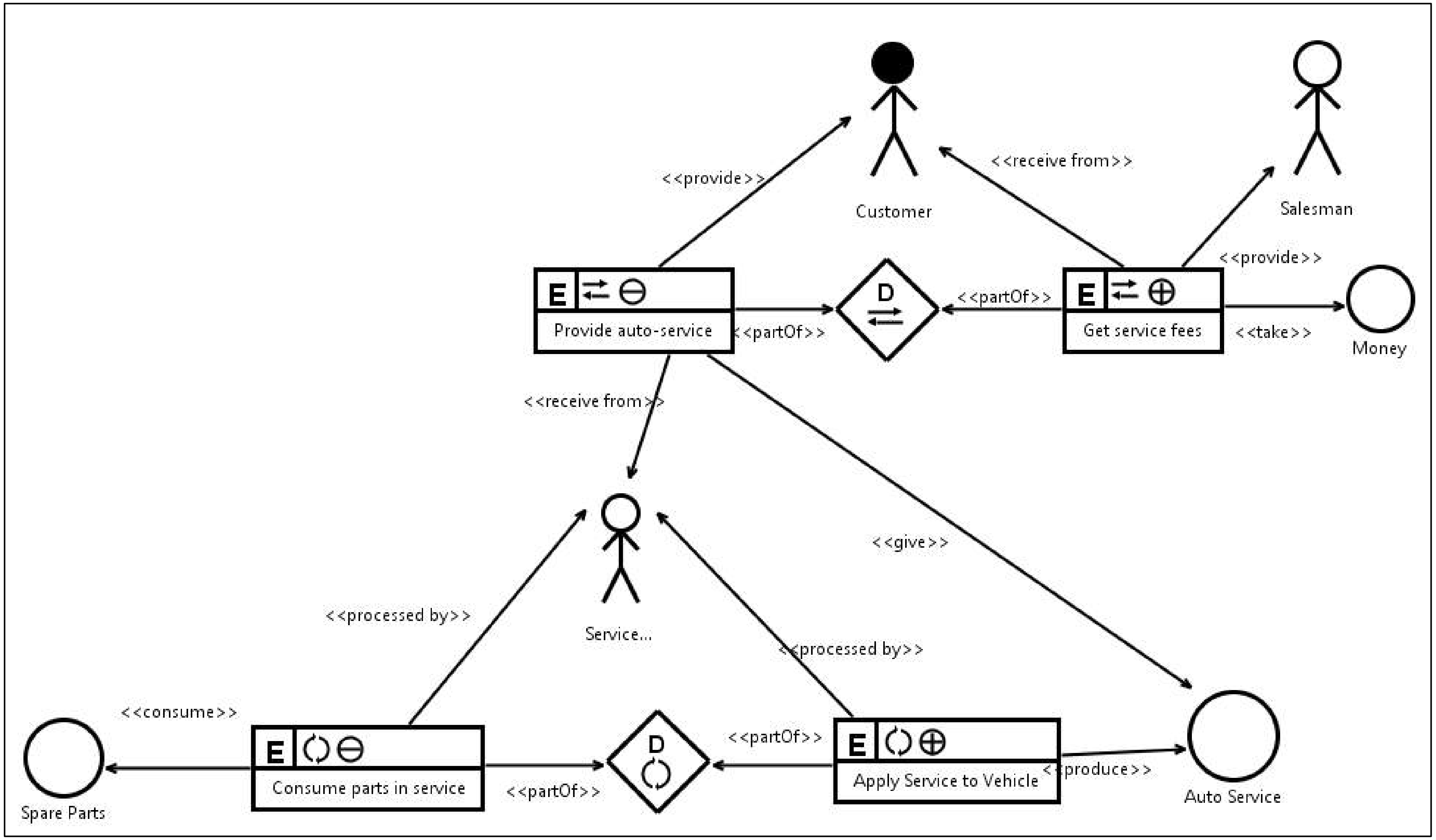
- **Rims importing and sales processes**



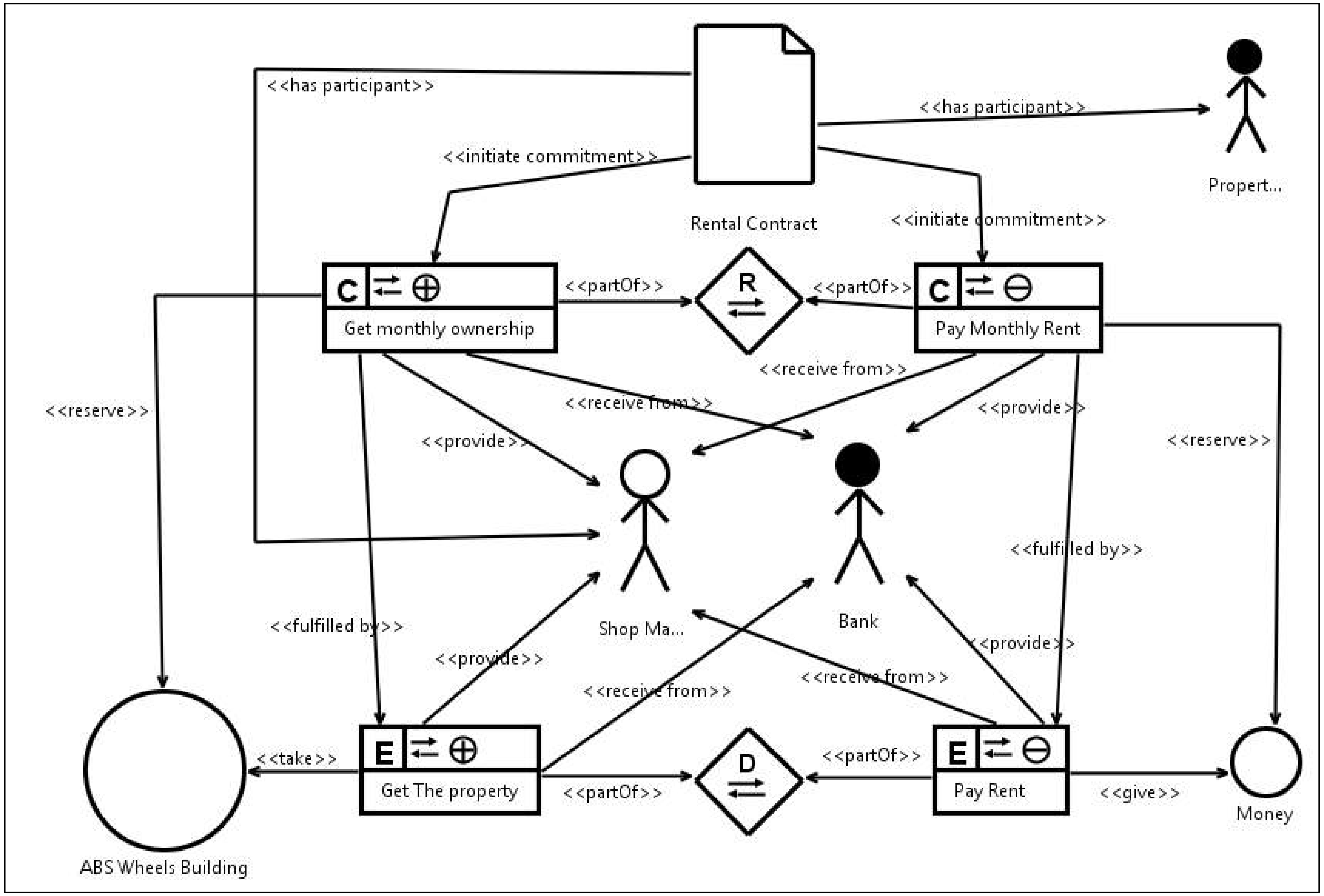
- **Spare parts acquirement processes**



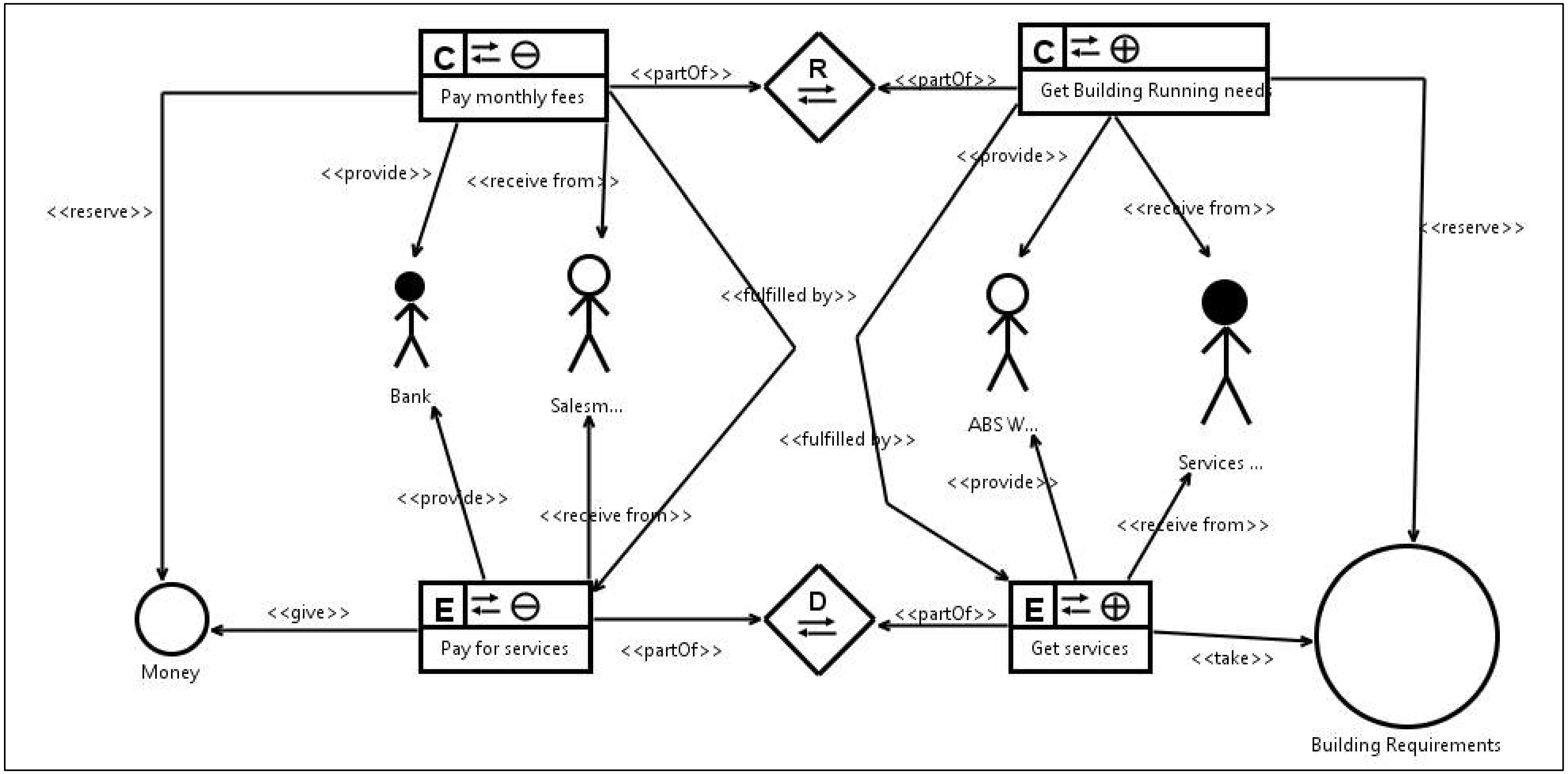
- **Auto-servicing process**



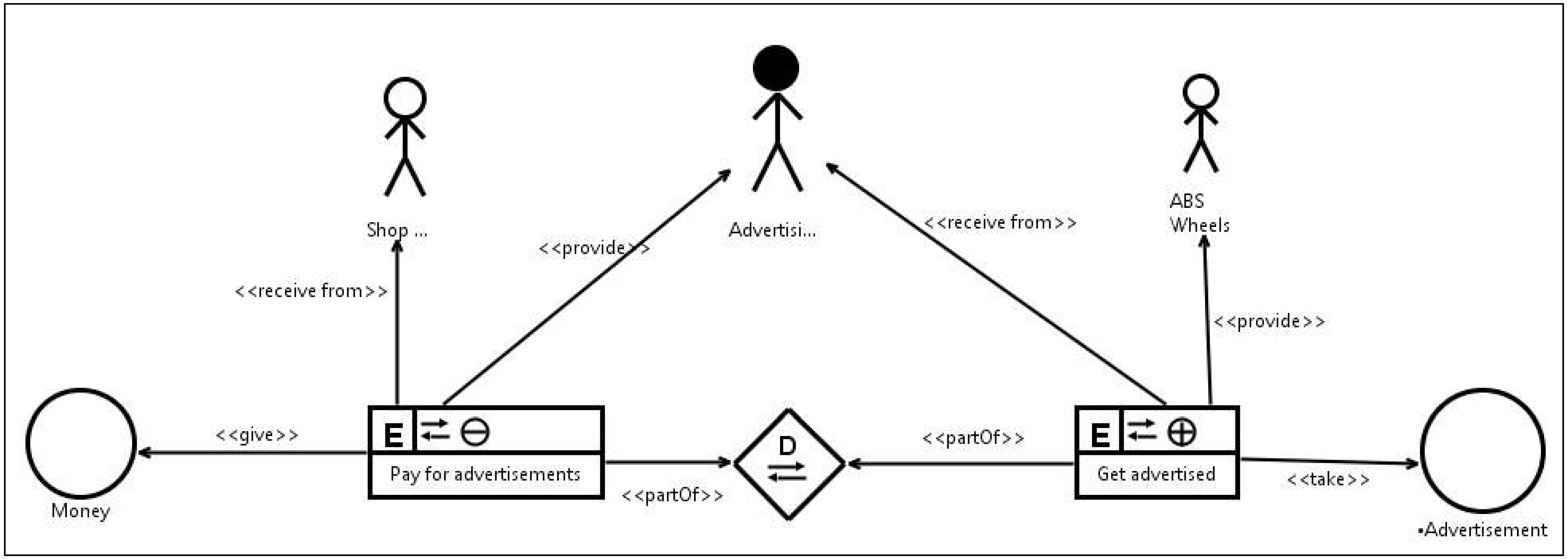
- **Place renting process**



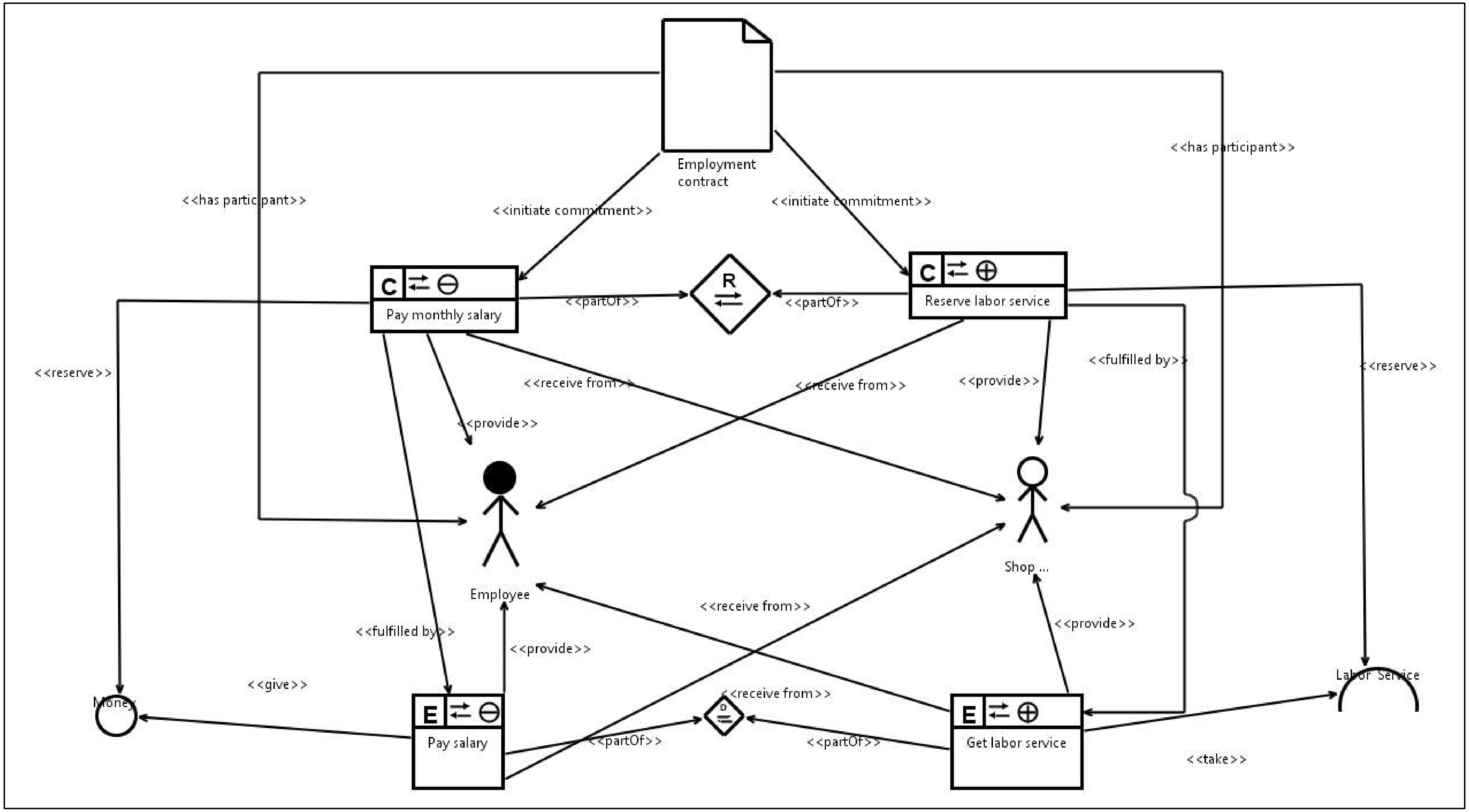
- **Maintenance process**



- **Advertising process**



- **Employment process**



A.9 ABS Wheels’ REA data-models

- **Tires importing and sales processes**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<rea:REAModel xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:rea="http://rea_extended/1.0"
name="TiresTradingAndImportingProcesses">
  <hasAgents xsi:type="rea:ExternalAgent" name="Logistics Company"/>
  <hasAgents xsi:type="rea:ExternalAgent" name="Customer"/>
  <hasAgents xsi:type="rea:ExternalAgent" name="Tires Manufacturer"/>
  <hasAgents xsi:type="rea:ExternalAgent" name="Collection Company"/>
  <hasAgents xsi:type="rea:InternalAgent" name="Salesman"/>
  <hasAgents xsi:type="rea:InternalAgent" name="Load and balance
employee"/>
  <hasAgents xsi:type="rea:InternalAgent" name="Shop Manager"/>
  <hasEvents xsi:type="rea:DecrementExchangeEvent" name="Pay for Tires
Shipment" give="//@hasResources.0" partOf="//@hasDualities.1"
provide="//@hasAgents.2" receiveFrom="//@hasAgents.6"/>
  <hasEvents xsi:type="rea:DecrementExchangeEvent" name="Give Tires"
give="//@hasResources.2" partOf="//@hasDualities.0"
provide="//@hasAgents.1" receiveFrom="//@hasAgents.4"/>
  <hasEvents xsi:type="rea:DecrementExchangeEvent" name="Transport Tires"
give="//@hasResources.1" partOf="//@hasDualities.0"
provide="//@hasAgents.0" receiveFrom="//@hasAgents.5"/>
  <hasEvents xsi:type="rea:IncrementExchangeEvent" name="Receive Tires
Shipment" take="//@hasResources.2" partOf="//@hasDualities.1"
receiveFrom="//@hasAgents.2" provide="//@hasAgents.6"/>
  <hasEvents xsi:type="rea:IncrementExchangeEvent" name="Receive Tires
Price" take="//@hasResources.0" partOf="//@hasDualities.0"
receiveFrom="//@hasAgents.1" provide="//@hasAgents.4"/>
  <hasEvents xsi:type="rea:IncrementExchangeEvent" name="Pay for tires
transportation" take="//@hasResources.0" partOf="//@hasDualities.0"
receiveFrom="//@hasAgents.0" provide="//@hasAgents.5"/>
  <hasResources name="Money"/>
  <hasResources name="Logistics service"/>
  <hasResources name="Tires"/>
  <hasCommitments xsi:type="rea:DecrementExchangeCommitment"
reservesResource="//@hasResources.0" name="Pay monthly fee"
fulfillment="//@hasEvents.1" partOf="//@hasReciprocities.0"
receiveFrom="//@hasAgents.4" provide="//@hasAgents.3"/>
  <hasCommitments xsi:type="rea:IncrementExchangeCommitment"
reservesResource="//@hasResources.0" name="Get monthly payments"
fulfillment="//@hasEvents.4" partOf="//@hasReciprocities.0"
provide="//@hasAgents.4" receiveFrom="//@hasAgents.3"/>
  <hasDualities xsi:type="rea:ExchangeDuality" name="TiresTradingDuality"
containsDecrementEvent="//@hasEvents.1 //@hasEvents.2"
containsIncrementEvent="//@hasEvents.4 //@hasEvents.5"/>
  <hasDualities xsi:type="rea:ExchangeDuality" name="TiresImportingDuality"
containsDecrementEvent="//@hasEvents.0"
containsIncrementEvent="//@hasEvents.3"/>
  <hasReciprocities xsi:type="rea:ExchangeReciprocity"
name="PaymentsCollectionReciprocity"
containsDecrementCommitment="//@hasCommitments.0"
containsIncrementCommitment="//@hasCommitments.1"/>
</rea:REAModel>
```

- **Rires importing and sales processes**

```
 <?xml version="1.0" encoding="UTF-8"?>
<rea:REAModel xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:rea="http://rea_extended/1.0" name="RimsTradingProcess">
  <hasAgents xsi:type="rea:InternalAgent" name="Shop Manager"/>
  <hasAgents xsi:type="rea:ExternalAgent" name="Rims Manufacturer"/>
  <hasAgents xsi:type="rea:InternalAgent" name="Salesman"/>
  <hasAgents xsi:type="rea:ExternalAgent" name="Logistics Company"/>
  <hasAgents xsi:type="rea:InternalAgent" name="Loading and balance
employees"/>
  <hasAgents xsi:type="rea:ExternalAgent" name="Customer"/>
  <hasAgents xsi:type="rea:ExternalAgent" name="Collection Company"/>
  <hasEvents xsi:type="rea:IncrementExchangeEvent" name="Receive Rims
Shipment" take="//@hasResources.1" partOf="//@hasDualities.0"
receiveFrom="//@hasAgents.1" provide="//@hasAgents.0"/>
  <hasEvents xsi:type="rea:DecrementExchangeEvent" name="Pay for Rims
Shipment" give="//@hasResources.0" partOf="//@hasDualities.0"
provide="//@hasAgents.1" receiveFrom="//@hasAgents.0"/>
  <hasEvents xsi:type="rea:DecrementExchangeEvent" name="Give Rims"
give="//@hasResources.1" partOf="//@hasDualities.1"
provide="//@hasAgents.5" receiveFrom="//@hasAgents.2"/>
  <hasEvents xsi:type="rea:IncrementExchangeEvent" name="Receive Rims
Price" take="//@hasResources.0" partOf="//@hasDualities.1"
receiveFrom="//@hasAgents.5" provide="//@hasAgents.2"/>
  <hasEvents xsi:type="rea:DecrementExchangeEvent" name="Pay for rims
transportation" give="//@hasResources.0" partOf="//@hasDualities.1"
provide="//@hasAgents.3" receiveFrom="//@hasAgents.4"/>
  <hasEvents xsi:type="rea:IncrementExchangeEvent" name="Transport Rims"
take="//@hasResources.2" partOf="//@hasDualities.1"
receiveFrom="//@hasAgents.3" provide="//@hasAgents.4"/>
  <hasResources name="Money"/>
  <hasResources name="Rims"/>
  <hasResources name="Logistics service"/>
  <hasCommitments xsi:type="rea:DecrementExchangeCommitment"
reservesResource="//@hasResources.0" name="Pay monthly fee"
fulfillment="//@hasEvents.2" partOf="//@hasReciprocities.0"
receiveFrom="//@hasAgents.2" provide="//@hasAgents.6"/>
  <hasCommitments xsi:type="rea:IncrementExchangeCommitment"
reservesResource="//@hasResources.0" name="Get monthly payments"
fulfillment="//@hasEvents.3" partOf="//@hasReciprocities.0"
provide="//@hasAgents.2" receiveFrom="//@hasAgents.6"/>
  <hasDualities xsi:type="rea:ExchangeDuality" name="RimsImportingDuality"
containsDecrementEvent="//@hasEvents.1"
containsIncrementEvent="//@hasEvents.0"/>
  <hasDualities xsi:type="rea:ExchangeDuality" name="RimsTradingDuality"
containsDecrementEvent="//@hasEvents.2 //@hasEvents.4"
containsIncrementEvent="//@hasEvents.3 //@hasEvents.5"/>
  <hasReciprocities xsi:type="rea:ExchangeReciprocity"
name="PaymentsCollectionReciprocity"
containsDecrementCommitment="//@hasCommitments.0"
containsIncrementCommitment="//@hasCommitments.1"/>
</rea:REAModel>
```

- **Spare parts acquirement processes**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<rea:REAModel xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:rea="http://rea_extended/1.0">
  <hasAgents xsi:type="rea:ExternalAgent" name="Spare parts supplier"/>
  <hasAgents xsi:type="rea:InternalAgent" name="Salesman"/>
  <hasAgents xsi:type="rea:ExternalAgent" name="Bank"/>
  <hasEvents xsi:type="rea:DecrementExchangeEvent" name="Pay for spare
parts" give="//@hasResources.1" partOf="//@hasDualities.0"
provide="//@hasAgents.0" receiveFrom="//@hasAgents.1"/>
  <hasEvents xsi:type="rea:IncrementExchangeEvent" name="Get spare
parts" take="//@hasResources.0" partOf="//@hasDualities.0"
receiveFrom="//@hasAgents.0" provide="//@hasAgents.1"/>
  <hasResources name="Spare Parts"/>
  <hasResources name="Money"/>
  <hasCommitments xsi:type="rea:DecrementExchangeCommitment"
reservesResource="//@hasResources.1" name="Pay extra fees"
fulfillment="//@hasEvents.0" partOf="//@hasReciprocities.0"
receiveFrom="//@hasAgents.1" provide="//@hasAgents.0"/>
  <hasCommitments xsi:type="rea:IncrementExchangeCommitment"
reservesResource="//@hasResources.1" name="Invest partial Payment money"
fulfillment="//@hasEvents.1" partOf="//@hasReciprocities.0"
provide="//@hasAgents.1" receiveFrom="//@hasAgents.2"/>
  <hasDualities xsi:type="rea:ExchangeDuality"
name="SpareToolsPossession" containsDecrementEvent="//@hasEvents.0"
containsIncrementEvent="//@hasEvents.1"/>
  <hasReciprocities xsi:type="rea:ExchangeReciprocity"
name="PartialPaymentsCommitments"
containsDecrementCommitment="//@hasCommitments.0"
containsIncrementCommitment="//@hasCommitments.1"/>
</rea:REAModel>
```

- **Auto-servicing process**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<rea:REAModel xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:rea="http://rea_extended/1.0">
  <hasAgents xsi:type="rea:InternalAgent" name="Service worker"/>
  <hasAgents xsi:type="rea:ExternalAgent" name="Customer"/>
  <hasAgents xsi:type="rea:InternalAgent" name="Salesman"/>
  <hasEvents xsi:type="rea:DecrementConversionEvent" name="Consume parts
in service" consume="//@hasResources.0" partOf="//@hasDualities.0"
processedBy="//@hasAgents.0"/>
  <hasEvents xsi:type="rea:IncrementConversionEvent" name="Apply Service
to Vehicle" produce="//@hasResources.1" partOf="//@hasDualities.0"
processedBy="//@hasAgents.0"/>
  <hasEvents xsi:type="rea:DecrementExchangeEvent" name="Provide auto-
service" give="//@hasResources.1" partOf="//@hasDualities.1"
provide="//@hasAgents.1" receiveFrom="//@hasAgents.0"/>
  <hasEvents xsi:type="rea:IncrementExchangeEvent" name="Get service
fees" take="//@hasResources.2" partOf="//@hasDualities.1"
receiveFrom="//@hasAgents.1" provide="//@hasAgents.2"/>
  <hasResources name="Spare Parts"/>
  <hasResources name="Auto Service"/>
  <hasResources name="Money"/>
  <hasDualities xsi:type="rea:ConversionDuality"
name="AutoServiceConversion" containsDecrementEvent="//@hasEvents.0"
containsIncrementEvent="//@hasEvents.1"/>
```

```
  <hasDualities xsi:type="rea:ExchangeDuality"
name="AutoServiceExchange" containsDecrementEvent="//@hasEvents.2"
containsIncrementEvent="//@hasEvents.3"/>
</rea:REAModel>
```

- **Place renting process**

```
<?xml version="1.0" encoding="UTF-8"?>
<rea:REAModel xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:rea="http://rea_extended/1.0">
  <hasAgents xsi:type="rea:ExternalAgent" name="Bank"/>
  <hasAgents xsi:type="rea:InternalAgent" name="Shop Manager"/>
  <hasAgents xsi:type="rea:ExternalAgent" name="Property Owner"/>
  <hasEvents xsi:type="rea:DecrementExchangeEvent" name="Pay Rent"
give="//@hasResources.0" partOf="//@hasDualities.0"
provide="//@hasAgents.0" receiveFrom="//@hasAgents.1"/>
  <hasEvents xsi:type="rea:IncrementExchangeEvent" name="Get The
property" take="//@hasResources.1" partOf="//@hasDualities.0"
receiveFrom="//@hasAgents.0" provide="//@hasAgents.1"/>
  <hasResources name="Money"/>
  <hasResources name="ABS Wheels Building"/>
  <hasCommitments xsi:type="rea:DecrementExchangeCommitment"
reservesResource="//@hasResources.0" name="Pay Monthly Rent"
fulfillment="//@hasEvents.0" partOf="//@hasReciprocities.0"
receiveFrom="//@hasAgents.1" provide="//@hasAgents.0"/>
  <hasCommitments xsi:type="rea:IncrementExchangeCommitment"
reservesResource="//@hasResources.1" name="Get monthly ownership"
fulfillment="//@hasEvents.1" partOf="//@hasReciprocities.0"
provide="//@hasAgents.1" receiveFrom="//@hasAgents.0"/>
  <hasContracts hasDecrementCommitment="//@hasCommitments.0"
betweenParties="//@hasAgents.1 //@hasAgents.2" title="Rental Contract"
hasIncrementCommitment="//@hasCommitments.1"/>
  <hasDualities xsi:type="rea:ExchangeDuality" name="RentExchange"
containsDecrementEvent="//@hasEvents.0"
containsIncrementEvent="//@hasEvents.1"/>
  <hasReciprocities xsi:type="rea:ExchangeReciprocity" name="Monthly
Rent Commetments" containsDecrementCommitment="//@hasCommitments.0"
containsIncrementCommitment="//@hasCommitments.1"/>
</rea:REAModel>
```

- **Maintenance process**

```
<?xml version="1.0" encoding="UTF-8"?>
<rea:REAModel xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:rea="http://rea_extended/1.0">
  <hasAgents xsi:type="rea:InternalAgent" name="ABS Wheels"/>
  <hasAgents xsi:type="rea:ExternalAgent" name="Services providers"/>
  <hasAgents xsi:type="rea:InternalAgent" name="Salesman"/>
  <hasAgents xsi:type="rea:ExternalAgent" name="Bank"/>
  <hasEvents xsi:type="rea:DecrementExchangeEvent" name="Pay for
services" give="//@hasResources.0" partOf="//@hasDualities.0"
provide="//@hasAgents.3" receiveFrom="//@hasAgents.2"/>
  <hasEvents xsi:type="rea:IncrementExchangeEvent" name="Get services"
take="//@hasResources.1" partOf="//@hasDualities.0"
receiveFrom="//@hasAgents.1" provide="//@hasAgents.0"/>
  <hasResources name="Money"/>
  <hasResources name="Building Requirements"/>
  <hasCommitments xsi:type="rea:DecrementExchangeCommitment"
reservesResource="//@hasResources.0" name="Pay monthly fees"
```

```
fulfillment="//@hasEvents.0" partOf="//@hasReciprocities.0"
receiveFrom="//@hasAgents.2" provide="//@hasAgents.3"/>
  <hasCommitments xsi:type="rea:IncrementExchangeCommitment"
reservesResource="//@hasResources.1" name="Get Building Running needs"
fulfillment="//@hasEvents.1" partOf="//@hasReciprocities.0"
provide="//@hasAgents.0" receiveFrom="//@hasAgents.1"/>
  <hasDualities xsi:type="rea:ExchangeDuality"
containsDecrementEvent="//@hasEvents.0"
containsIncrementEvent="//@hasEvents.1"/>
  <hasReciprocities xsi:type="rea:ExchangeReciprocity"
containsDecrementCommitment="//@hasCommitments.0"
containsIncrementCommitment="//@hasCommitments.1"/>
</rea:REAModel>
```

- **Advertising process**

```
<?xml version="1.0" encoding="UTF-8"?>
<rea:REAModel xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:rea="http://rea_extended/1.0">
  <hasAgents xsi:type="rea:InternalAgent" name="ABS Wheels"/>
  <hasAgents xsi:type="rea:ExternalAgent" name="Advertising Channel"/>
  <hasAgents xsi:type="rea:InternalAgent" name="Shop manager"/>
  <hasEvents xsi:type="rea:DecrementExchangeEvent" name="Pay for
advertisements" give="//@hasResources.0" partOf="//@hasDualities.0"
provide="//@hasAgents.1" receiveFrom="//@hasAgents.2"/>
  <hasEvents xsi:type="rea:IncrementExchangeEvent" name="Get advertised"
take="//@hasResources.1" partOf="//@hasDualities.0"
receiveFrom="//@hasAgents.1" provide="//@hasAgents.0"/>
  <hasResources name="Money"/>
  <hasResources name="•&#x9;Advertisement"/>
  <hasDualities xsi:type="rea:ExchangeDuality" name="Advertising
exchange" containsDecrementEvent="//@hasEvents.0"
containsIncrementEvent="//@hasEvents.1"/>
</rea:REAModel>
```

- **Employment process**

```
<?xml version="1.0" encoding="UTF-8"?>
<rea:REAModel xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:rea="http://rea_extended/1.0">
  <hasAgents xsi:type="rea:ExternalAgent" name="Employee"/>
  <hasAgents xsi:type="rea:InternalAgent" name="Shop Manager"/>
  <hasEvents xsi:type="rea:DecrementExchangeEvent" name="Pay salary"
give="//@hasResources.0" partOf="//@hasDualities.0"
provide="//@hasAgents.0" receiveFrom="//@hasAgents.1"/>
  <hasEvents xsi:type="rea:IncrementExchangeEvent" name="Get labor
service" take="//@hasResources.1" partOf="//@hasDualities.0"
receiveFrom="//@hasAgents.0" provide="//@hasAgents.1"/>
  <hasResources name="Money"/>
  <hasResources name="Labor  Service"/>
  <hasCommitments xsi:type="rea:DecrementExchangeCommitment"
reservesResource="//@hasResources.0" name="Pay monthly salary"
fulfillment="//@hasEvents.0" partOf="//@hasReciprocities.0"
receiveFrom="//@hasAgents.1" provide="//@hasAgents.0"/>
  <hasCommitments xsi:type="rea:IncrementExchangeCommitment"
reservesResource="//@hasResources.1" name="Reserve labor service"
fulfillment="//@hasEvents.1" partOf="//@hasReciprocities.0"
provide="//@hasAgents.1" receiveFrom="//@hasAgents.0"/>
```

```
  <hasContracts hasDecrementCommitment="//@hasCommitments.0"
betweenParties="//@hasAgents.1 //@hasAgents.0" title="Employment
contract" hasIncrementCommitment="//@hasCommitments.1"/>
  <hasDualities xsi:type="rea:ExchangeDuality"
containsDecrementEvent="//@hasEvents.0"
containsIncrementEvent="//@hasEvents.1"/>
  <hasReciprocities xsi:type="rea:ExchangeReciprocity"
containsDecrementCommitment="//@hasCommitments.0"
containsIncrementCommitment="//@hasCommitments.1"/>
</rea:REAModel>
```