Domain-specific modelling of the REA ontology

Zdenek Melis, Jaroslav Zacek, Frantisek Hunka

Abstract – The REA ontology is a concept for creating design of business infrastructure based on ownership and its exchange. It provides a comprehensive framework to business process modelling. This ontology has four levels of abstraction to cover greater possibility of views on business of usage compared to standard modelling methods. This paper describes creation of modelling tool based on this ontology using the domain-specific modelling. This approach uses domain-specific structure of the ontology to ensure validation of created models. Modelling tool automatically generates the resource flow simulator based on object-valued Petri nets and allows interconnections of models with different level of abstraction.

Keywords *- business process modelling; DSL; DSM; REA ontology; source code generation*

INTRODUCTION

Business process modelling is a challenging and essential discipline for business management. The business process modellers place great emphasis on the descriptive mechanism of business processes, which must be able to handle all demands that they made on it and it must be also user-friendly and it must be based on the modelling problem domain. Equally as important is to have a tool that integrates this mechanism and provides a basic environment to help create a model. The created model is not beneficial by itself and may act counterproductively if contains some mistakes. The basic requirement of the tool is to lead the modeller, check him and help him. The model itself can only represent reality. The modelling tool is often required to do more such as to give developers some feedback either in the form of reports, statistics, analyzes or usable source code. We cannot assume that the modeller has a deeper knowledge of programming or economic mathematics and statistics. All these requirements for the modelling tool practically reflect the characteristics of domain-specific modelling.

Choosing of the ontology for describing business processes is also important. Nowadays the modellers have a wide range of different ontologies to build a model, but most of them have imperfections due to the use of general concepts for modelling. Therefore they are not able to answer the question why the business activity is performed. From this perspective, it is preferable to use a value modelling that, unlike other modelling principles, focuses on modelling of the value of economic resources. The best-known representative of the value modelling is the REA ontology.

Currently, there are two similar projects that combine domain-specific modelling with the ontology REA - *The REA-DSL* [11] and *REA Policy Modelling* [9]. The first one is limited only to operational (basic) level of the ontology and uses its own notation. The usage of this tool is very limited in the practice. The second tool is focused on the policy level of the REA value model and provides, besides basic validation of the model, the set of validation attributes in the form of propositional calculus. This approach makes working with the model very confusing. Practical use of the tool is also limited by the absence of automation functions. For these reasons there was an effort to create a tool that eliminates discussed problems and that will be suitable for practical use.

This paper explains how to create a domain-specific modelling tool for business processes modelling described by the REA ontology. The first part of the paper discusses the domain-specific modelling technology and the REA enterprise ontology. The second part is devoted to the creation of modelling tool itself from the initial identification of domain concepts and rules to creating a metamodel and its transformations into the domain language of the modelling tool.

I. DOMAIN-SPECIFIC MODELLING

Domain-specific modelling (DSM) is a software engineering methodology narrowly focused on one particular specific domain. Narrow focus allows working with domain terms including their meaning. Domain-specific language (DSL) is created based on the domain terms. DSL contains the domain syntax and the semantics.

Basic advantages can be divided into several categories [3]:

- *Productivity* - Easier and more efficient application development leads to higher productivity within a single domain. The productivity gain is often in hundreds of percent.
- *Quality* - Improving the quality is given by the model validation and verification, which prevents to create inconsistent or illogical links. By using domain concepts it also eliminates the need of mapping domain concepts to different modelling language, which is a frequent source of errors. Finally full code generation guarantees the absence of implementation errors.

Zdenek Melis, Jaroslav Zacek, Frantisek Hunka are with the University of Ostrava, Department of Computer Science, 30. dubna 22, 701 03 Ostrava, Czech Republic. E-mail: zmelis@seznam.cz , {Jaroslav.Zacek, Frantisek.Hunka}@osu.cz

- *Expertise requirements* - 2 groups of people participates on the development of applications using DSM. The first group consists of domain experts, and their task is to create a modelling tool using DSM techniques. In the second group there are the users of this tool, which creates models and from them it is subsequently generated the final application. These users need only knowledge of the domain to develop applications.

DSM architecture has three layers:

- *Language (DSL)* - According to [3] DSL is a computer programming language of limited expressiveness focused to a specific domain. Domain concepts are mapped to objects modelling language, or as an object property, object links, sub-models or models of the other language.
- *Generator* - converts the generated model to a specific syntax. The output of the generator is not necessarily the source code, but it can be tests, documentation, metrics, or for example prototyping.
- *Domain framework* - forms a layer between the generated code and existing code in the target environment and reduces the complexity of the code. The other areas of its use include the elimination of duplicity from generated code, providing an interface for the generator, integration with existing code, or hiding the target environment.

The basic concept of the DSM is to focus on a fixed domain. That defines the language and model behaviour of that domain. In the case of fixed domain the DSM achieves incomparably better results compared to other modelling approaches, especially in the productivity and simplicity of development. However, if the domain is not fixed, or the changes are expected, the DSM loses its advantages. In this case, it is comparable or even it has significantly worse results than other modelling approaches.

II. Value Modelling of Business Proceses

The essence of the value modelling of business processes is to monitor the value of economic resources and their property. An economic resource is the thing or the service that is lacking and has benefits to economic agents. This modelling approach is based on the use of specific concepts, unlike other modelling methods (e.g. IDEF0 [2]), that use general concepts [8]. The best known representative of the value modelling is the REA ontology.

The term business process can be defined differently depending on areas where the term is used. For example, V. Řepa [10] defined it as follows: *The business process is a set of activities transforming a summary of inputs into a summary of outputs, which the company performs in order to fulfil business objectives.* In other words, the business process is the sequence of steps that creates products and services by transformation of inputs into outputs. It is important to say that each process must have its owner and fixed boundaries.

A. The REA Ontology

In order to explain the term of the REA ontology, it is necessary to explain the term ontology itself. According to [23] ontology is an explicit specialization of conceptualization. Specialization is a study of things that exist or may exist in a particular domain. Conceptualization is the abstraction and the simplified view of the world. Specialization means a formal and declarative representation [4].

The REA ontology (REA means Resource, Event, Agent) is a concept for creating design of business infrastructure based on ownership and its exchange. It is based on the concept of economic exchanges, increasing the enterprise value.

B. Levels distribution of the REA ontology

Depending on the level of abstraction the REA ontology can be divided into four levels [1]:

- *Value System Level Model* - This level represents the view of the resources that are exchanged between enterprises and external business partners. The high level of abstraction is used for alignment of business objectives and strategies [12].
- *Value Chain Level Model* -Illustrates interconnection of business processes with a focus to monitoring resource flows. The value chain is used primarily to diagnose competitiveness of an enterprise because it offers an overall view of the chain of processes. This view allows simulations and optimizations [16] of processes in the company.
- *REA Model Level* - Describes individual business processes and it is the most important level of the REA ontology, because here are modelled most important information relating to the company. This level answers to the question of why the process is carried out. The value model of business process shows particular changes in the value of resources that are based on the concepts of economic exchanges and conversions increasing the value of the company.
- *Task level* - Describes steps to fulfil events. Defines the lowest level of abstraction, contains the code concepts and therefore is implementation-dependent.

The REA model level is divided according to functionality into two groups [7]:

- *Operational level* - Forms the basic structure of the model. Describes events that have already happened. Basic concepts of operating levels are resource, event and agent and semantic abstractions increasing the value of the company - the exchange and the conversion.
- *Policy level* – Illustrates what could, should or shouldn't happen. It contains concepts and semantic abstraction defining rules, schedules, contracts and other extensions of the model such as grouping, typing, or commitments.

C. Use of the REA ontology

Traditional approaches to business process modelling using general concepts are not recommended because of a low concreteness of the models. Such a model is not able to provide the checking of economic errors and automation. Unlike these approaches the REA ontology uses specific concepts, which increases the amount of model information preserving the simplicity of the model.

The REA ontology contains internal rules for the verification of the consistency of the model and thereby it prevents creation of incorrect links. The result of this consistency verification is the model that has all relations properly declared and connected and answers the question why a business activity is carried out, and therefore why economic events occur. This is a significant difference and big advantage of the REA ontology compared to other solutions offered by traditional approaches.

Another feature of the REA ontology is the simplicity and clarity of models for ordinary users, who will work with it. On the other hand, the ontology is sufficiently accurate to be able to automate the process modelling.

Another major advantage of the REA ontology is that the model has an independent order of occurring of the economic events. The model is able to record everything that actually happens, and it is not limited to the sequence of events (scenario) defined at the time of design. Another advantage is the independence of the model in relation to technical aspects of the transfer of resources. Therefore it is not necessary to change the model in case of changing the technical infrastructure and due to the strict rules of the REA ontology the integrity and consistency of the model is ensured [7].

Folding individual business processes when the outputs of one process are linked into the inputs of the other processes creates a value chain. This represents a cyclic net of processes. The processes exchange the value and thus directly or indirectly contribute to the formation of desirable characteristics of the final product or service. They can be subsequently exchanged with other economic agents for a resource that has greater value for the company. The value chain helps to identify economic resources within an enterprise and provides an overview to business processes of the company.

III. Identification of key elements of the REA enterprise ontology

The first step in creating a design of modelling tool is a basic structure that behaves as the descriptive apparatus and describes the general principles of the REA value model. First it is necessary to identify individual concepts and relationships within the structural division of the REA model level and specify rules for their behaviour. These findings should lead to the creation of the most important part of the modelling tool - the metamodel. Metamodel clearly specifies and defines models created by the tool and ensures their basic validation.

A. Concepts of the operational level of the REA model level

Basic operational level of the REA model level contains 3 concepts [7]:

- *Economic resource* – Basic economical company resource that company wants to plan, monitor and control. Examples of economic resources are products, services, money, raw materials, tools, etc.
- *Economic agent* - An individual or some organization capable of having control over economic resources and able to receive or transmit further control of other agents. An example might be a customer, supplier, company, etc.
- *Economic event* – Represents either increment or decrement in the value of economic resource. This transformation could be realized immediately or in a certain time period. Examples could be work unit, using of services, renting, etc.

Following links and domain rules are defined between these concepts [13]:

- *Stack flow (Inflow/ Outflow)* - Connects economic resource and economic event
  - There must be at least one economic event for every economic resource.
  - There must be at least one economic resource for every economic event.
- *Participate (Provide/ Receive)* – Connects economic agent and economic event
  - Every event requires the participation of two agents, one in the role of the recipient and the second as a provider.
  - Every agent must be connected with an incremental event by the *Receive* link and with a decrement event by the *Provide* link.
- *Duality* - Connects an incremental and a decrement economic event
  - Every incremental event must be linked to at least one decrement event through Duality entity.
  - Every decrement event must be linked to at least one incremental event through Duality entity.

B. Semantic abstraction and concepts of the policy level of the REA model level

The limited range of this paper does not allow dealing with all concepts of the policy level. For this reason, we have to focus on processing basic and most frequently used concepts of the REA model level, namely semantic abstractions typing and group and resource management concepts - a contract, a schedule and a commitment.

- *Typification* - It is a homogeneous collection whose elements have the same characteristics defined by the type. Typification is an abstraction of a group of objects into a certain category forming bond "*is a-*

kind-of" It is used to capture the description of the concept applied to a set of objects [5].

- *Group* - Is a structural element of the REA model level designed for creating heterogeneous collections or sets of REA entities. This is a special form of aggregation that forms a bond "*is a-member-of*" [5]. The group itself is not limited to operating level entities, but virtually any entity of the REA model level including other groups with which, however, forms intransitive dependence.
- *Commitment* - Promise or obligation to perform an economic event in a specific time. For this reason, parameters should contain the scheduled date and planned value. An example can be ordering goods, where one party agrees to provide chosen goods and the other party agrees to provide the required amount.
- *Contract/Schedule* - Is a set of obligations and rules that define conditions of performing the duality in the future and events performed if initial commitments are unfulfilled.

Between these concepts following links and domain rules are defined:

- *Typification* - Connects resource/event/agent type with corresponding concept of operational level
  - Every entity type must be linked with just one entity it represents.
- *Fulfillment* - Connects commitment and economic event
  - Every incremental commitment must be linked to incremental economic event.
  - Every decrement commitment must be linked to decrement event.
- *Specification* - Connects commitment and agent (or agent type)
  - Every commitment must specify the agent or the agent type who is responsible for it.
- *Reciprocity* - Connects incremental and decrement commitment
  - Every decrement commitment must be linked with at least one incremental commitment.
  - Every incremental commitment must be linked with at least one decrement commitment.
- *Party* - Connects contract/schedule and economic agent or agent type
  Each contract/schedule must declare two parties (Agents or Agent types).
- *Clause* - Connects commitment with contract or schedule.
  - Every contract must contain decrement and incremental commitment.
  - Every commitment must be declared by some contract or schedule.
- *Reservation* - Connects commitment and economic resource or resource type
  - Every commitment has to reserve at least one type of resource.
- *Group* - Connects group with any entity of the model
  - It has no restrictions.

Fig 1 illustrates an example of the model structure presenting basic concepts, semantic abstractions and links between them. For better clarity the model is divided into the operational level and the policy level.

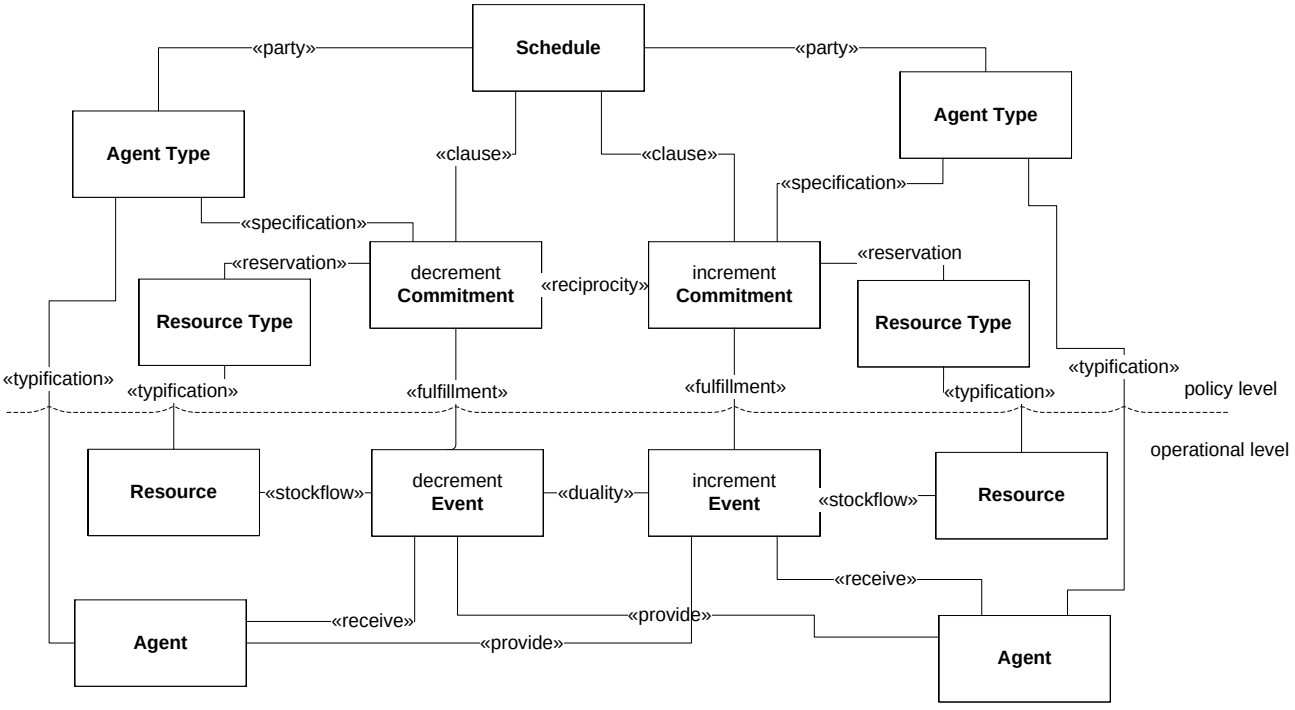

Fig. 1. An example of the general structure of business process

IV. METAMODEL OF THE REA MODEL LEVEL

The core of the REA ontology is the REA model level. In order to create a domain-specific language that will provide a validation of the model, it is necessary to identify elements and their relations in the model and on this basis create the metamodel.

According to [6] the metamodel is a model that is used for formal specification of other models.

The creation of the metamodel is carried by analogous way as the formation of the model itself. Individual model elements are sequentially added to the metamodel and subsequently interconnected by specific links that correspond to possible connection in the model. The multiplicity based on domain rules is determined for these links. It is a way to capture the behaviour of the model, its elements and links in the metamodel and to ensure the validation of the model. By successive steps there were added individual concepts into the metamodel and based on previous analyzes were defined by their interactions and multiplicity. These steps have led to a metamodel shown in Fig. 2. Group links are not shown at the figure just for the clarity, because the group can connect with any model element (including itself).

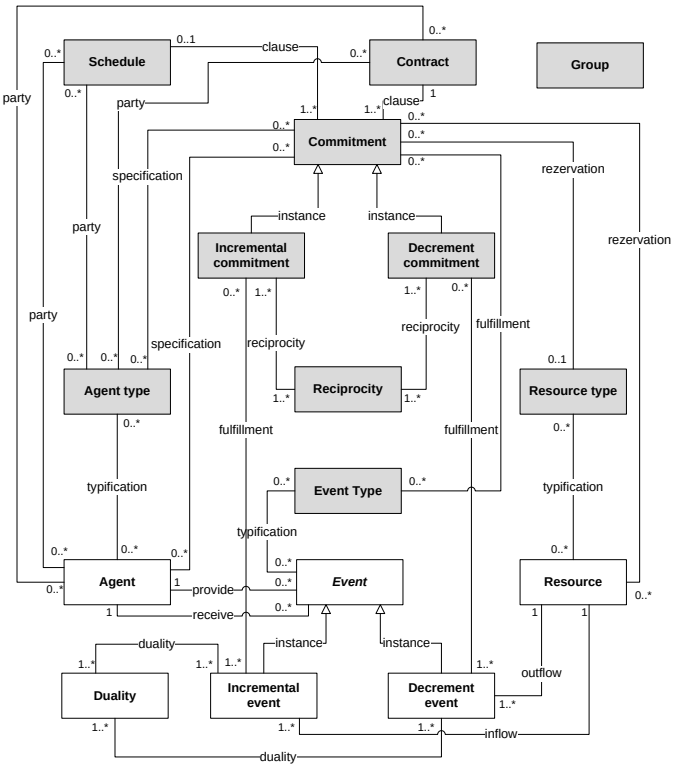

Fig. 2. The metamodel of selected concepts and semantic abstractions of the REA model level

V. RESOURCE FLOW FORMALISM

In order to practically implement the value chain and ensure not only their validation, but also the ability to simulate the flow of resources, it is necessary to use an appropriate formalism. The Petri net theory matches best for solving this problem type (e.g. [17], [18]). To be able to work with such complex structures, such as the resource, the object-value Petri nets (OV-PN) must be used.

A. Object-valued Petri net

Object-valued Petri net is an extension of P/T Petri net. This extension has been introduced in [22]. Object-valued Petri nets are used as formalism for validation and synchronization of complex object models [14].

Definition: Object-valued Petri net

Petri net is extended to a 6-tuple (P, T, F, V, R, C), where:

- P *is a finite nonempty set of places,*
- N *is a finite nonempty set of transitions,*
- $T \cap P = \emptyset$ *(P and T are disjoint),*
- $F \subseteq (P \times T) \cup (T \times P)$ *is a finite set of arcs (flow relation),*
- *V is a finite set of object data types,*
- *R is a finite set of transforming functions* $R: P \cup T \rightarrow \psi(V)$*, where* $\psi(V)$ *is the power of the set of object data types.*
- *C is a set of capacity function.* $C: P \rightarrow N \cup \omega, N \subseteq \mathbb{N}$ *and* ω *denotes infinite.*
- $M_0: P \rightarrow V_{MS}$ *is the initial marking of the token.* $\forall p \in P: M_0(p) \in R(p)_{MS}$*, where* $R(p)_{MS}$ *is the multiple set of the object data type tokens in p.*

The main idea of the Object-valued Petri net is an object-valued token that provides adequate expressivity to describe resources represented by complex object structures. The token carries basic information to identify the specific object instance. Initial marking consists of the multiple set of object data types deployed across the net. Firing of each token means change in marking of the net and also change of the token type. However token identification remains and therefore we can identify the token in every step of the simulation process. If the model is partly linked with the Object-valued Petri net theory we have to define the path of tokens. Formalism itself defines necessary basis to create the model, unfortunately that does not ensures the sequence of movements into desirable result. Object-valued Petri net realizes transition as soon as the transition is feasible. Nevertheless the real model can require other conditions to realize the transition (for instance lazy constructions). Therefore we have to state the new definition of the path and pass of the model.

Definition: Path of the OV-PN

Let OV-PN = (P, T, F, V, R, C) be an Object-valued Petri net with initial marking M_0*. The path from the place* $u_1 \in P \cup T$ *to following place* $u_n \in P \cup T$ *is the sequence* $(u_1, u_2, \ldots, u_n)$*, where* (u_i, u_{i+1}) *for* $1 \leq i \leq n$*.*

Definition: Pass of the OV-PN

Object-valued Petri net OV-PN = (P, T, F, V, R, C) with initial marking M_0 is feasible when:

1. *Must exist an initial place $i \in P$ where $\bullet i = \emptyset$.*
2. *Must exist exactly one final place $o \in P$, where $o\bullet = \emptyset$.*
3. *Every place $u \in P \cup T$ lies on the pass between initial place **i** and final place **o**.*

In this context the first condition is understood as a marking of the input of the model that can be represented by more than one input parameter. If this condition is set to be strict to the value of input marks the model cannot realize calling of the method with more than one input parameter. The output of the model is usually one because of the standard method construction in object-oriented paradigm [19]. Third condition expresses the fact that every place and every transition exists on the path between the initial place and the final place. Therefore the Object-valued Petri net should not have blind paths and every call in the model should be reachable form initial place by passing finite number of transitions representing a flow relation *F*. Similarly to the initial place every place in the model exists in the flow relation *F* is able to reach the final place of the model.

Boundedness and safeness

Object model synchronized by Petri net mechanism can be bounded at the places level as in ordinary Petri net. Every method can produce more than one output during the simulation. Places may store these outputs as Object-valued tokens (similar to colour evaluation in [21]). By applying safeness rule the places in model store only one object-valued token and the model becomes less complex.

Conservation

Created model cannot be strictly conservative. For example the *Purchase process* consumes several inputs and produces one output. The model cannot have a constant token count for every marking from set of the reachability set

$$\Re(M_0): \sum_{\rho_i \in S} M(p_i) \neq \sum_{\rho_i \in S} M_0(p_i).$$

Liveness and deadlock

All methods in object-oriented paradigm can be executed more than once [20]. However by executing some method an internal state of the object can be altered. That means if we need to apply liveness property to whole model, every method must be considered as an atomistic operation.

Generally the transition $t \in T$ is alive if:

$$\forall p \in \bullet t: M(p) \neq \emptyset \text{ and } p \in t\bullet: M(p) = \emptyset.$$

It means that transition becomes active, if there are tokens on all transition's entrances and the place that follows the transition is empty. The net is alive if there is at least one live transition in every step of simulation process otherwise a deadlock occurs. Deadlock is solved on a higher abstraction level and requires user intervention.

B. Object-valued Petri net extensions

The main condition of the value chain is cyclicality. However the Object-valued Petri net has two definitions that limit the path of the net and pass of the net. First definition says that O-V Petri net with a specific marking M_0 has a specific sequence from one place to another. The value chain has also specific sequence that defines the path of the chain. Moreover the cyclic chain consists of many single paths connected to each other. To express a general value chain principle with the Petri net theory we have to define a cyclic Petri net and its inverse net:

Definition: Cyclic Object-valued Petri net

A marked Petri net (OV-PN, M_0) is cyclic Petri net if from every reachable marking M it is possible to return into M_0 (i.e. $M \in \Re(OV-PN, M_0) \Rightarrow M_0 \in \Re(OV-PN, M)$.

Definition: The inverse of an Object-valued Petri net

For a Petri net OV-PN, its inverse $\overline{OV-PN} = (P, \bar{T}, \bar{F})$ is given by:

- $\bar{T} = \{\bar{t} | t \in T\}$ *and*
- $\bar{F}(\bar{t}, p) = F(p, t)$ *and* $\bar{F}(p, \bar{t}) = F(t, p)$ *for every* $p \in P$ *and* $t \in T$

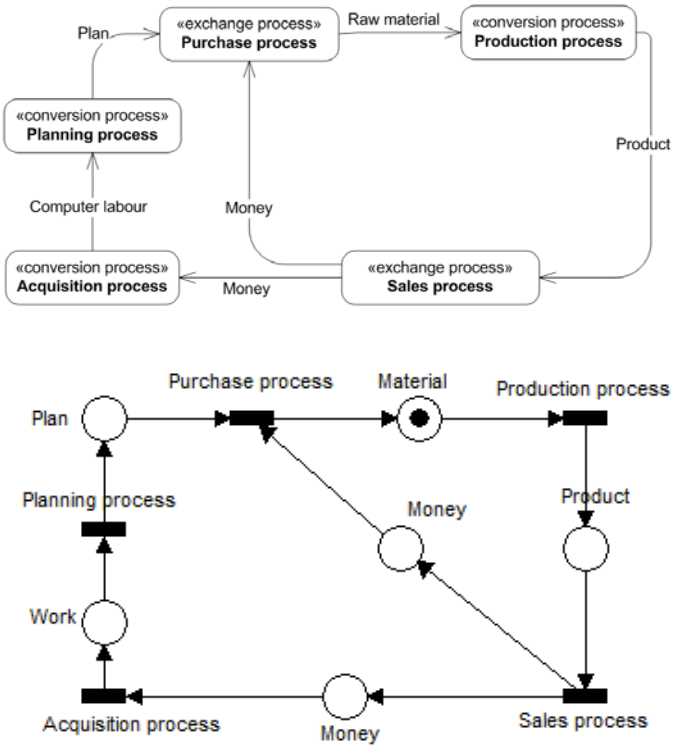

Fig. 3. Value chain and its OV-PN representation

Cyclic Object-valued Petri net is the basic formalism for creating, synchronizing and managing cyclic models of the value chain. Fig. 3 illustrates the example of the value chain and its OV-PN representation.

VI. DESIGN PROCESS OF THE VISUAL MODELLING TOOL

The tool was created in a Visual Studio by Microsoft with the DSM Tools extension. This domain-specific development environment offers an excellent compromise between the availability, price and automation support.

A. Proposal of the DSL of the REA model level

The creation of the DSL of the tool corresponds to the metamodel mentioned before. Individual concepts are replaced by domain classes. The highest class is a domain model. It has a composite link to *REAElement*, which is the super class of all REA model level concepts and specifies their basic structure and parameters. Between various elements of the metamodel there are created unique relations. DSL Tools, however creates only one-way links. For this reason links management is solved programmatically. This solution also provides an automatic determination of links. Every element of the model has an associated class with a visual representation of the element whose visual parts are connected to parameters of the domain class. After adjusting parameters related to properties of the tool (such as the structure of the toolbar), the tool can be started and in a limited way also used.

Fig. 4 shows a fragment of the DSL with fundamental structural separation of elements.

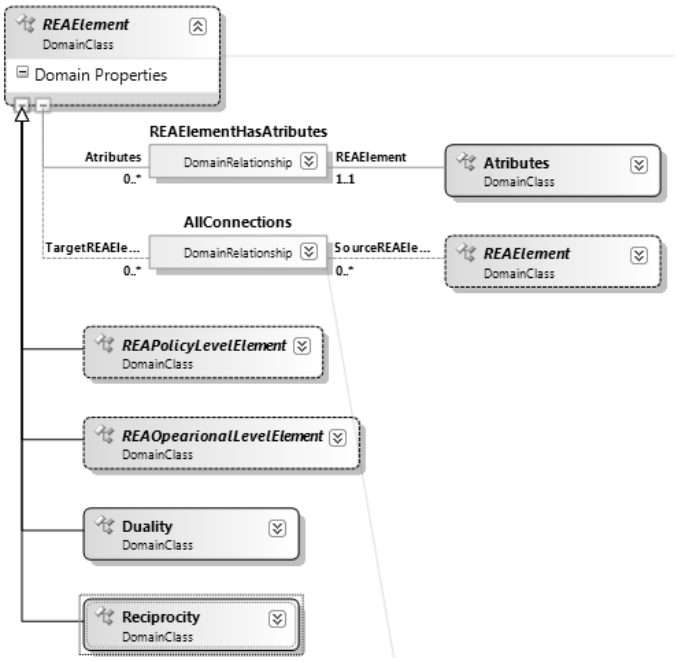

Fig. 4. Part of the DSL

B. Model validation

Model validation is controlled by the validation framework. The validation logic can be any method that is registered in the framework. Individual domain rules are placed into appropriate classes according to way of use (validation of the link, element or model) using partial class technology. While any domain rule is broken, the user is alerted to the error by description of the problem as well as the reference to the problem element.

The following code shows an example of the validation method ensuring domain rule "*Every commitment has to reserve at least one resource or resource type*".

```
[ValidationState(ValidationState.Enabled)]
public partial class Commitment
{
  [ValidationMethod(
ValidationCategories.Open |
ValidationCategories.Save |
ValidationCategories.Menu)]
  private void
ValidateConnectionWithResource(ValidationC
ontext context)
{
  if (this.ResourceTypes.Count < 1 &&
this.Resources.Count < 1)

context.LogError(ValidationResources.Commi
tmentMustConnectResourceType, "Reservation
Link", this);
  }
}
```

C. Interoperation of different levels of abstraction

Connecting different levels of abstraction allows the user to combine the value chain model with an individual representation of business processes - REA model levels. Different DSM are merged under one platform and their mutual communication is established. This connection is based on the *ModelBus* technology. The project containing the domain language, that provides its elements to other languages, is extended by an adapter. Adapter allows access to individual elements of the model, as well as the whole model itself. Domain language that uses services of another language must integrate new data type *ModelBusReference* that encapsulates information about the reference of an external model and its elements. Fig. 5 illustrates the principle of interconnection of two DSL.

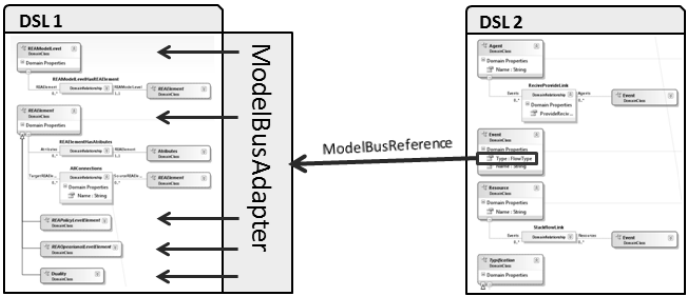

Fig. 5. The principle of interconnection of two DSL

D. Methodology of DSL interconnection

The principle of the value chain is the flow of resources across business processes. Therefore it is necessary to ensure that the resource on the output of one business process is identical to the resource on the input of process connected in the value chain. This can be achieved by creating a unique ID for each resource in the model. Interconnection of two processes in the value chain must synchronize the ID of the resource on their connector.

The problem of synchronization of resources is that before an interconnection the resources are independent of each other. They may vary in structure, characteristics and ways of use. For example the resource *Product* in the *Production process* has other important parameters than the same resource in the *Sales process* and the other meaning can have identically named attributes. The cost in the *Production process* may indicate the cost of producing one piece, whereas in the Sales process it is the final price to customers. For this reason it is necessary to distinguish which attributes will be converted and mutually synchronized, alternatively which attributes will be renamed to avoid conflict states. It is not possible to automate this step and therefore it is necessary for a model creator to perform these modifications in the user interface.

The methodology for working with attributes describes solving of the basic operations within value chains:

- Creating a new element:
  1. The user creates a new value chain element.
  2. The user is prompted for a name of the linked value model (REA model level). If the model does not exist empty one is created.
  3. Validator will check whether the model is not already used by another business process.
  4. Every resource in the selected model gets the new generated ID.
- Connecting elements:
  1. The user creates a new link.
  2. The user selects a resource from the source and target process to ensure interconnection of resources with different names.
  3. The resource ID in target process is set according to the source process. This change is transferred to transitively linked resources.
  4. The user is asked to transform attributes, change them or change its interconnection. Attributes can be transformed to both directions. The user is informed about the change of attributes only in the case the modified resource is linked to other processes.
- Removing the link:
  1. User removes the link.
  2. The new ID for the originally target resource is automatically generated.
  3. The ID is transitively transferred to all linked processes.
- Removing the element:
  1. User removes the element.
  2. For all deleted links the procedure described above is applied.
- Changing the reference:
  1. User changes the reference of the resource.
  2. The new ID is generated in the original resource.
  3. The resource in the target process obtains ID from the source process.
- Double-click on the element:
  1. User double-clicks on the element.
  2. The linked model opens.

Most of the above operations can be implemented in the form of domain rules, which are applied automatically during execution of the specified event.

E. The code generation

The code generator uses a template engine T4 (Text Template Transformation Toolkit), which is part of Visual Studio. This engine transforms the created template into usual class containing created methods and code fragments. Subsequently this class is compiled and executed and the output is the generated code.

Created modelling tool generates the simulator of the resource flow in the value chain. It shows the flow between interconnected business processes in the chain. The code generation is performed automatically when a valid model is saved and the resulting code of the simulator is immediately after compilation executable and it doesn't need any necessary modifications.

Practical realization of the flow of resources can be achieved by the formalism of cyclic object-value Petri nets. The value chain contains at least one token - a resource that is gradually transformed by business processes to different resources. Initial tokens are manually set by the user during creating the model.

At each step of the simulation run, the output of each process is sent to the input of connected process. A resource cannot be inserted to the input if the input is occupied by token from the previous simulation step - in this case, a resource remains at the output of the previous process. If the process has all required resources available (all required entries are filled), the conversion of these resources occurs (exchange), and the result resource is placed on the output of the process. The conversion cannot be performed if the output is occupied by a resource. The simulation assumes compliance with safety rules of Petri nets to represent the transformation process of one resource.

Validation of the simulation corresponds to the type of simulation (e.g. [15]). In case of resource flow simulation the OV-PN rules are applied.

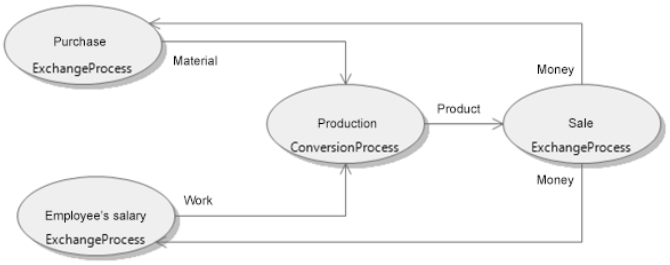

Fig 6. Example of the value chain

Fig. 6 shows a simple example of the value chain. The simulation of the flow of resources is as follows:

- *Step 1*: The token in the *Production process* is transformed to the *Product* and is sent to the *Sales process*.
- *Step 2*: The *Product* in the *Sales process* is exchanged for the *Money*.
- *Step 3a*: The *Money* is sent to the *Purchase process*, where they are exchanged for *Material*.
- *Step 3b*: The *Money* is sent to the *Employee's salary process* applies to employees, where it serves as an appropriate reward for the *Work*, which is one of resources needed for the production.
- *Step 4*: *Material* and *Work* are delivered to the *Production process*, where they are converted to the final *Product*.

The simulator consists of three main parts - the simulation core, processes and resources. This partly corresponds to the structure of generator files.

- *SimulationGenerator.tt* - Acquire and prepare the individual data from the model and then runs the subgenerators.
- *ResourcesGenerator.tt* - Generates a class structure for an economic resource.
- *ProcessesGenerator.tt* - Generates class structure of a business process.
- *CoreGenerator.tt* – Generates the simulation core.
- *SupportMethods.tt* - It includes support methods (e.g. for modifying names).

Support methods for modifying of names are used to ensure the correct naming of the generated classes. The name used by the user in the model may not meet required conventions for the class name. Spaces, special characters and accents are removed. The name is also modified if the name is a keyword of C# language, or it begins with a digit.

It was also necessary to solve the problem with generating multiple files. T4 generally assumes that one template generates just one file. Therefore, it was necessary to create a method that saves generator context (containing generated code) to specified file and after that it clears the context.

```
private void saveFile(string name)
{
   string path =
   Path.GetDirectoryName(Host.TemplateFile
   );
   string outputFilePath =
   Path.Combine(path, name);
   Directory.CreateDirectory(Path.GetDirec
   toryName(outputFilePath));
   File.WriteAllText(outputFilePath,
   this.GenerationEnvironment.ToString());
   this.GenerationEnvironment.Remove(0,
   this.GenerationEnvironment.Length);

}
```

The final step in creating generator is its adaptation to a custom tool to ensure automatic code generation. It also allows the generator to be distributed with the modelling tool. For this purpose the special class ensuring modifications of the template and its execution and the file performing the registration of generator during installation of modelling tool was created.

VII. Comparison of existing DSM tools for the REA ontology modelling

Created tool provides an integrated environment for creating models of the REA ontology, which provides validation of links, elements and the model as a whole. The tool allows creating models at both levels of an abstraction - the REA model level and the value chain level, and if necessary it can interconnect these models. At the value chain level the source code generator of the simulator of the flow of resources is integrated.

Table 1 shows a comparison of created tool with similar existing tools.

TABLE I. Comparison table

	Created tool	**The REA-DSL**	**REA Policy modelling**
Automatic links determination	Yes	No	No
Links validation	Yes	Yes	Yes
Entity validation	Yes	Yes	Yes
Model validation	Yes	Yes	Yes
Code generation	Value chain simulation	SQL	C# classes
Policy level support	Yes	No	Yes

CONCLUSION

This paper described the procedure of creating the domain-specific modelling tool from the initial analysis of domain, its elements and rules to the implementation of the solution. The proposal tool is fully capable compare to existing tool mentioned in the introduction. Moreover the proposal tool solves some issues discussed before. To create the tool we had to indentify the key elements, links and rules and create a metamodel. The metamodel itself becomes a foundation formalism to define a domain specific language. Domain rules are part of the metamodel as well. These rules are used to model validation. The result of combining REA ontology and DSM is the tool that is able to create valid models of the REA model level and in a higher level of abstraction generates simulations of the resources flow.

ACKNOWLEDGEMENT

The paper was supported by the grant reference no. SGS08/PRF/2013 provided by Ministry of Education, Youth and Sports.

REFERENCE

[1] DUNN CH. L, CHERRINGTON J. O., HOLLANDER A. S, *Enterprise Information Systems – a Pattern-Based Approach*, 3 edition, 2004, McGraw-Hill/Irwin, ISBN-13: 978-0072404296

[2] Federeal Information Processing Standards (FIPS) Publication No. 183, *Integration Definition for Function Modelling (IDEF0)*, 1993, U.S. Dept. of Commerce, Washington DC.

[3] FOWLER M., *Domain-Specific Languages*, Addison-Wesley Signature Series (Fowler), 2011, ISBN 978-0-321-71294-3

[4] GAŠEVIĆ, D., DJURIĆ D., DEVEDŽIĆ V., *Model Driven Architecture and Ontology Development*, Springer Berlin Heidelberg New York, 2006, ISBN-13 978-3-540-32180-4

[5] GEERTS G. L.; MCCARTHY W. E., *Policy-Level Specifications in REA Enterprise Information Systems*, Journal of Information Systems, 20(2):37–63, 2006, ISSN: 1365-2575

[6] GONZALEZ-PEREZ C., HENDERSON-SELLERS B., *Metamodelling for Software Engineering*, *Wiley*, 2008, ISBN 987-0-470-03036-3

[7] HRUBÝ P., *Model-Driven Design Using Business Patterns*, Springer, 2006, ISBN-13 978-3-540-30154-7

[8] HRUBÝ P. A kol., *Víceúrovňové modelování podnikových procesů (Systém REA)*, VŠB-TU Ostrava, 2010, ISBN: 978-80-248-2334-8

[9] Instant T E a - Tracing Enterprise Architectures, Webové stránky společnosti, 2012, Available: http://www.instanttea.com, [10/2012]

[10] ŘEPA V., *Podnikové procesy: Procesní řízení a modelování*, Grada Publishing, a.s., Praha, 2006, ISBN 80-247-1281-4

[11] SONNENBERG CH., HÜMER C., HOFREITER B., MAYRHOFER D., *The REA-DSL: a domain Specific Modelling Language for Business Models*, Proceedings of CAISE 2011, 2011, Available: http://publik.tuwien.ac.at/files/PubDat_198185.pdf

[12] HUŇKA F., ŽÁČEK J., MELIŠ Z., ŠEVČÍK J., *REA Value Chain and Supply Chain*, Scientific Papers of the University of Pardubice. 2011, no. 21 (3/201), p. 68-77

[13] MELIŠ Z., ŠEVČÍK J., ŽÁČEK J., HUŇKA F., *Identification of key elements of REA enterprise ontology*, Procedia-Information Technology and Computer Science Journal, 2012

[14] ŽÁČEK, J., MELIŠ, Z., HUŇKA, F. *Modeling the value chain with object-valued Petri nets*. Recent Advances in Electronics and Communication Systems. Rhodes, Greece, 2013. ISBN 978-1-61804-201-9

[15] VYMĚTAL, D., ŠPERKA, R., SLANINOVÁ, K., SPIŠÁK, M. *Towards the Verification of Business Process Simulation in a JADE Framework*, International Journal of Economics and Statistics, Issue 1, Volume 1, 2013

[16] CHANG, J, LIN Y., *Exploring the dynamics of cross-level value chain relationships*, International Journal of Economics and Statistics, Issue 1, Volume 1, 2013

[17] CAPEK, J., HUB, M., MYSKOVA R., *Basic Authentication Procedure Modelled by Petri Nets*, International Journal of Computers and Communications, Issue 4, Volume 4, 2010, University Press, ISSN: 2074-1294

[18] STAINES A. S., *A Colored Petri Net for the France-Paris Metro*, International Journal of Computers, Issue 2, Volume 6, 2012, ISSN: 1998-4308

[19] SHILLING, J.: *Three Steps to Views: Extending the Object-Oriented Paradigm*, OOPSLA 89 Proceedings, 1989.

[20] ZACEK J., HUNKA F.: *CEM: Class executing modeling*, World Conference on Information Technology, 2010, page 1597-1601, ISSN 1877-0509.

[21] ZHAO, X., WEI, C., LIN, M., FENG, X., LAN, W: *Petri Nets Hierarchical Modelling Framework of Active Products Community*, Advances in Petri Proceedings of the 2013 International Conference on Electronics and Communication Systems Net Theory and Applications, 2010, page 153-174, ISBN 978-953- 307-108-4.

[22] ZACEK, J., HUNKA, F.: *Object model synchronization based on Petri net*, 17th International Conference on Soft Computing MENDEL 2011, Brno University of Technology, Faculty of Mechanical Engineering, 2011, ISBN 978-80-214-4302-0.

[23] GRUBER T. R., *A translation approach to portable ontology specifications*, Knowledge Acquisition, vol. 5, no. 2, pp. 199–220, 1993, ISSN 1042-8143