VERSITA
Information Technology and Management Science
doi: 10.2478/itms-2013-0010
2013 / 16

Reusable Components in Knowledge-based Configuration Design Systems

Henrihs Gorskis[1], Arkady Borisov[2], [1-2] *Riga Technical University*

***Abstract* – This paper takes a look on how components for knowledge-based intelligent systems can be created for reuse. For this purpose, we use production rules as inspiration for a system that uses an ontology description for the method and the domain ontology for the knowledge about the domain the problem takes place in. In this paper we give a description of an approach that hopefully can give insight into such a system. The approach is based on previous work and other scientific publications concerning this field of study. The created ontology models are in no way guaranteed to be useful outside of this example and the approach itself might still need to be improved in the future.**

***Keywords* – Intelligent system, ontology, production rules**

I. Introduction

One if not the most expensive task of implementing an information technology solution to a problem is the development of a software solution [5]. In order to cut the expenses associated with software development, the reuse of existing solutions would be preferred.

One possibility for reuse of existing solutions would be the reuse of problem solving methods in knowledge-based systems [6].

This is often difficult since any problem-solving methods and problem solutions would be closely connected with the problem domain. An approach would have to be introduced that would disconnect domain knowledge and problem-solving methods.

This paper strives to provide a possible approach to this problem. By using production rules from an existing knowledge system, we try to create an ontology that describes these rules not only in a way that provides descriptions of all concepts within the rules, but also is reusable. By reusable we mean a description that is as independent as possible from the domain knowledge.

II. Analysis of the Production Rule Example and Preparation for Conversion

In order to find an approach we will construct several ontology models from an example of production rules.

The example used in this paper is a modification of the rules from the bagger problem. It was originally introduced by Patrick Winston of MIT [1]. All the rules are given in Table I.

TABLE I
Rule Base

Rule	IF	THEN
B1	step is check-order there is a bag of potato chips there is no soft drink bottle	add one bottle of Pepsi to order
B2	step is check-order	start bag-large-items step
B3	step is bag-large-items there is a large item to be bagged there is a large bottle to be bagged there is a bag with less than 6 large items	put the large bottle item in the bag
B4	step is bag-large-items there is a large item to be bagged there is a bag with less than 6 large items	put the large item in the bag
B5	step is bag-large-items there is a large item to be bagged	start fresh bag
B6	step is bag-large-items	start bag-medium-items step
B7	step is bag-medium-items there is a medium item to be bagged there is an empty bag or a bag with medium items bag is not yet full medium item is frozen	put the medium item in a freezer bag in the bag
B8	step is bag-medium-items there is a medium item to be bagged there is an empty bag or a bag with medium items bag is not yet full	put the medium item in the bag

B9	step is bag-medium-items there is a medium item to be bagged	start fresh bag
B10	step is bag-medium-items	discontinue bag-medium-items start bag-small-items step
B11	step is bag-small-items there is a small item to be bagged there is a bag that is not yet full bag does not contain bottles	put the small item in the bag
B12	step is bag-small-items there is a small item to be bagged there is a bag that is not yet full	put the small item in the bag
B13	step is bag-small-items there is a small item to be bagged	start fresh bag
B14	step is bag-small-items	stop

From this description we can extract an ontology that describes the item domain. This ontology holds all the concepts and individuals that describe the items from the shopping list. The rules are similar to rules used in such systems as XCON [3].

But let us first take a look at the rules and their meaning.

Rule **B1**. Since rules and every test within the rules are performed in order, the very first test is for the current active step. In case this test fails, the reasoner can immediately jump to the next rule and safe time this way.

Next, the rule wants to know if there is a "bag of potato chips" in the user's order. There are several ways of looking at some information given in these rules. There are at least 2 ways of implementing such a request. The first way is a direct check for the item "chips" not for its class or other property of the individual. The second way is to define "chips" as a sub- or super-concept of an item. For example, we could implement a concept "bag of potato chips" as a sub-concept of "medium-sized item". This way the rule would check for the existence of any potato chip product from several possible ones while at the same time working with a medium-sized item when needed. However, this would either indicate that all bags of potato chips are only ever medium sized or every item concept would need to be connected to a sub-concept of "bag of chips". The structure of the domain ontology and the connected requirements of the method ontology need to be defined in a matter that allows such tests. For the purposes of this paper we will define the test for a "bag of potato chips" as a direct search for a specific individual in order to explore the required specifications in the method ontology for such a search.

The final part of the IF statement of the first rule is the check for a soft drink bottle in the order. This problem is similar to the "bag of chips" one; however, we can see the mentioning of a "Bottle of soft drink" and "Pepsi"; therefore, we can implement this as Pepsi being an individual of the concept "bottle of soft drink" and it, in turn, is a sub-concept of a large item. For this item we will implement a more difficult structure of concepts.

The **THEN** part of the rule adds a specific individual to the order.

The next rule **B2** exchanges the current step with the next one. The original version of rules like this made an extra step of stopping the execution of the current step before assigning the next.

Rule **B3** tests to see if the order has large bottles that need to be bagged. From the rules alone it is unclear if "Large Bottle", "Bottle of soft drink" and "Bottle" are the same concepts; a concept hierarchy is any other structure of information. For example, we could have the concept "Bottle" as a sub-concept of "Item". The concept "Large Bottle" is a sub-concept of both "Bottle" and "Large Item". And, in its turn, the concept "Bottle of soft drink" is a sub-concept of "Bottle" or "Large Bottle". Again it is unclear if in this domain bottles of soft drink are always large. If they are not large, there should be additional concepts, such as "Large Bottle of soft drink", "Medium Bottle of soft drink" and "Small Bottle of soft drink". They all would be sub-concepts of "Bottle of soft drink" and the one concepts of either "Small Item", "Medium Item" or "Large Item".

This rule also tests if there are 6 large items in the bag in order to determine if there is still room to place a large item.

Should the rule fire, the large bottle in question would be placed inside the bag.

B4 is a shortened version of B3. Since rules are always fired in order, by the time the reasoner reaches, there will not be any large bottles left in the order and only remaining large items would be needed to be bagged.

Rule **B5** is interesting since by the time this rule is reached there will be no free space in the bag left. The test for 6 items was performed in the previous rule. This way the rule system can determine when a bag change needs to be performed.

When **B6** becomes the only available rule, it is clear that the next big step in the bagging process needs to be taken.

Rule **B7** is the first rule of the next step, which bags medium items. However, in its original wording this first rule was not clearly meant to put items into a shopping bag. Instead it searched for items with the property "Frozen" (or another indicator for this) and put them only into a specialized freezer bag. Unfortunately there were several things unclear with this rule and could be interpreted very differently. If this step is meant to find all frozen items and put them into freezer

bags then the test for room inside the shopping bag is meaningless since the item is placed inside the freezer bag and not the shopping bag, and until a rule is hit that will put it into a bag, the active bag can change. If the freezer bag were immediately placed inside the shopping bag, the test for room inside the shopping bag would make sense, however, the test for the item being inside a freezer bag (as it was in its original form) would not. Any item in a freezer bag would already be also inside the shopping bag and would not need to be bagged. For this paper the rule was changed removing the IF element "the medium item is not in a freezer bag" and modifying the THEN element to indicate that the item is put inside a freezer bag and inside the shopping bag at the same time. This modification of the THEN part also ensures that there is only one freezer bag in every shopping bag. This however would mean that implementation of this element would need to tell the system to check if there already is a freezer bag inside the shopping bag and put the frozen item inside that bag.

Another thing that needs to be noted is the test for an empty bag or a bag that contains medium items. It is interesting in two ways; first, it contains two tests separated with "or". And, second, it begins with the words "there is a(n)". This wording indicates that this element is not only a test, but it also changes the environment, in which the execution takes place. It seems that if there is a bag that fulfils the criteria of this test, that bag is made to be the current shopping bag that is being filled. If the current bag which was being filled at the time this rule fired did not fulfil the requirement of being empty or containing a medium item, and at the same time there were another bag that fulfilled it, from that moment onwards that bag would be considered the current active bag.

The other rules concerning medium items, similar to the rules about the large items, continue to become more and more general until the next step concerning small items needs to become activated.

B11 is the first rule concerning small items. It searches for a bag that is not empty. It needs to be noted that "not empty" is different from "containing 6 large items" or "not empty". That makes at least 3 possible tests that can be performed on a bag concerning its fullness. The rule also searches for a bag that does not contain a bottle. However, in combination with rule **B12,** which puts a small item in any bag that is not full, this means that small items are preferably put in bags with no bottle, but can end up with one, if no empty bag is available. Still in combination with rule **B13** a situation can arise that a small item is put in a bag with a bottle, but after that a new bag is started, leaving that small item in an undesired situation with no rule to put it to another bag.

Rule **B14** ends the execution of the rules switching to the step "stop".

III. The Domain Ontology

From the rule description it made sense to arrange the items around concepts describing the item sizes. It was done this way since many rules address the items in question as "Large Item", "Medium Item" and "Small Item". Only rarely an item was addressed directly or as something else. In the case of the Pepsi item, in order to take a full advantage of all its properties it was made into an instance of "Bottle", "Large Item" and "Soft drink", making it a "Large Bottle of soft drink". Figure 1 shows a graphic representation of the domain ontology. Other items "Pepsi 0.5L" and "Ice Pop" were added to show, that the domain ontology could hold other information that is not necessarily used to solve a given task. Let' us assume that the

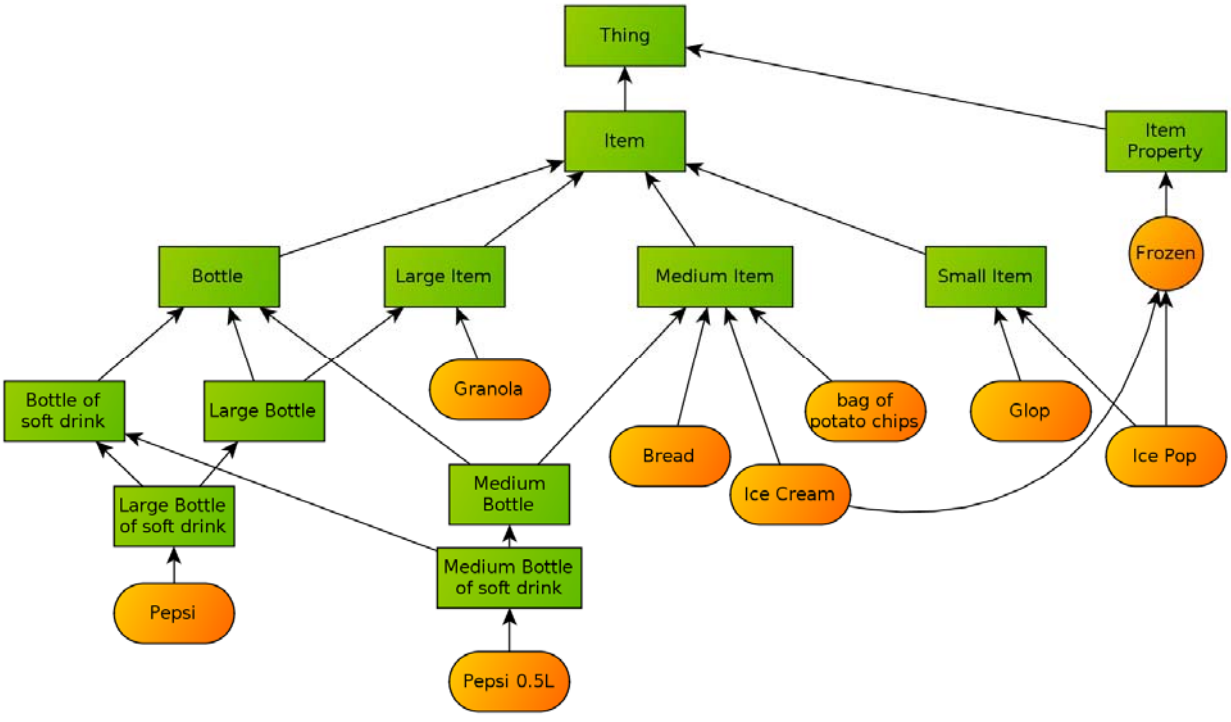

Fig. 1. Domain ontology

store's policy is not to put small frozen products into freezer bags since the shopper might like to enjoy it right away. Therefore the "Ice Pop" will not be put into a freezer bag by the current rules. The property "Frozen" in this ontology is given by using a property with the individual "Frozen" as a target. Not every domain ontology model might use this approach. For example one ontology model might have a literal value "Frozen". Every frozen item would have a property with that literal value as a target. Such problems might be solved by the mapping process if they are considered correctly.

IV. Implementation of the Rule Execution

Depending on the system that is constructed we can have several different implementations of the bagger method [8], [9]. They can range from very basic and simple ones to very specific and complex implementations. For example, it would be valid to provide a very simplistic method ontology that simply provides the rules in a concept hierarchy with the rules as individuals. Such an ontology would only describe the rules and execution would be manual or in a system that would access the individuals from the ontology and parse their names.

However, in this paper we will try to construct a more complex method ontology that provides all necessary elements and data for direct execution. This study is the continuation of the theme of the previous paper about ontology construction from guidelines [10]. The previous study also provided inside into ontology models capable of execution, based on GEM [11] guidelines.

V. Method Ontology

This is a general description of one possible definition of the method ontology.

The method ontology must contain all elements and descriptions for the bagger problem to be executed. The method ontology must have a description of how it works in general. The concept "Method description" can do this. However, depending on how fine a description needs to be provided, it may be better to define several sub-concepts and a structure that is better suited for providing information of a method. The main idea is to have a specific element within the ontology that will describe what information needs to be given in the beginning and what information is returned in the end. The bagger method requires a list of items picked by a user to be provided in the beginning – the order list. In its turn, the method returns a list of bags and their content.

Next, the method ontology defines an internal and external part of the execution. In the external part, we can see parts of the structure of the domain ontology. This is needed since several IF elements require tests for specific concepts (Large item, soft drink bottle). Moreover, several individuals are in the method ontology since they are referenced directly by the rules. Besides, having the concept structure of the domain ontology in the method ontology helps finding mapping solutions in cases of new domain ontology models that do not have mapping information, since this is the part that will serve as an interface to the domain ontology after the mapping process.

In the internal part, the ontology describes all elements needed for rule execution. First, there is a list of all step values that are used by the rules. In this case they are: check-order, bag-large-items, bag-medium-items and bag-small-items.

Next, we have the rule concept and its associated IF and THEN parts. Dissection of a rule individual can be seen in Table II.

TABLE II
Definition of Individual B1

Individual B1	
Property	**Target**
is-a	Rule : *Concept*
IF_1	"step is check-order" : *Individual of IF concept*
IF_2	"there is a bag of potato chips" : *Individual of IF concept*
IF_3	"there is no soft drink bottle" : *Individual of IF concept*
THEN_1	"add one bottle of Pepsi to order" : *Individual of THEN concept*

Also we can extract from the tasks in the IF and THEN parts some useful sub-functions that can be called recurrently rather than having to give the same description of actions for several rules.

The temporal part is a special part of this ontology. During execution, individuals contained in it will change several times.

In this example, a function is a hard-coded set of static activities and does not need inputs. However, every function that is connected via an "IF_x" property has a Boolean operator output, for every such function has to be true for the rule to be true. Some IF functions and every THEN function affect the state of a working-ontology. This working-ontology holds the information necessary to describe the change during the execution of the rules.

It is necessary to note, that the IF and THEN properties of rule individuals have to be numbered, for the order of activating them can have an effect on how a rule is determined to behave.

One thing that is not given in the picture above is the element that describes the activity of every function. This can be given in the ontology as a property value or the individual itself gives a link to the resource that describes the action.

A sub-function is a simpler and frequently used function. It is used for actions that can be generalized and, therefore, save the number of definitions that need to be given for the execution of actions. A the example of sub-function' property is given in Table III.

TABLE III

DEFINITION OF SUB-FUNCTION "STEP IS"

Individual "step is (Step x)"	
Property	**Target**
is-a	Sub-function : *Concept*
uses	Step : *Individual of Step*
uses	CurrentStep : *Individual of temporal variable*

Functions that are described in the IF part of the ontology, the functions of the THEN concept and sub-functions use other elements given in the method ontology. For example, the IF function "step is bag-large-items" has to have a reference to the individual "bag-large-item" of the "Step" concept. This makes it clear how this function operates and does not have to rely solely on its personal definition. However, since tests for the current step are common in this method it also uses a sub-function – "is step" sub-function. This sub-function receives the individual "bag-large-items" as an input from the "step is bag-large-item" function. In order to test whether or not this individual is in fact the current step it needs to be connected to the "current step" individual of the concept "current variable". Having access to this variable the sub-function can examine it for its current connection to a "step" individual. If they are equal to the sub-function and the main function, both hold true.

The temporal part of the method ontology contains individuals that will change properties during execution. Also new individuals will be created. Let us take a closer look at it.

VI. TEMPORAL ELEMENTS

Here we can see the ontology that describes the current state of the bagging algorithm. It will be either part of the method ontology and actively used in it or a separate instance of this ontology can be created. The "Active element" concept and "Current Step" individual will be used in order to point to the current step, which is being performed. In order to use this ontology, it will always have to be liked in some way to the method ontology.

The concept "Bag" describes any bag that is used in the bagging problem. Another "Active element" concept, that is "Current Bag", is linked to the bag that is being used by the algorithm at the time. During execution, the property of the individual "Current Bag", which points to one of the concepts "Bag", will change several times.

The concept "Freezer Bag" holds individuals of any freezer bag used during the bagging process.

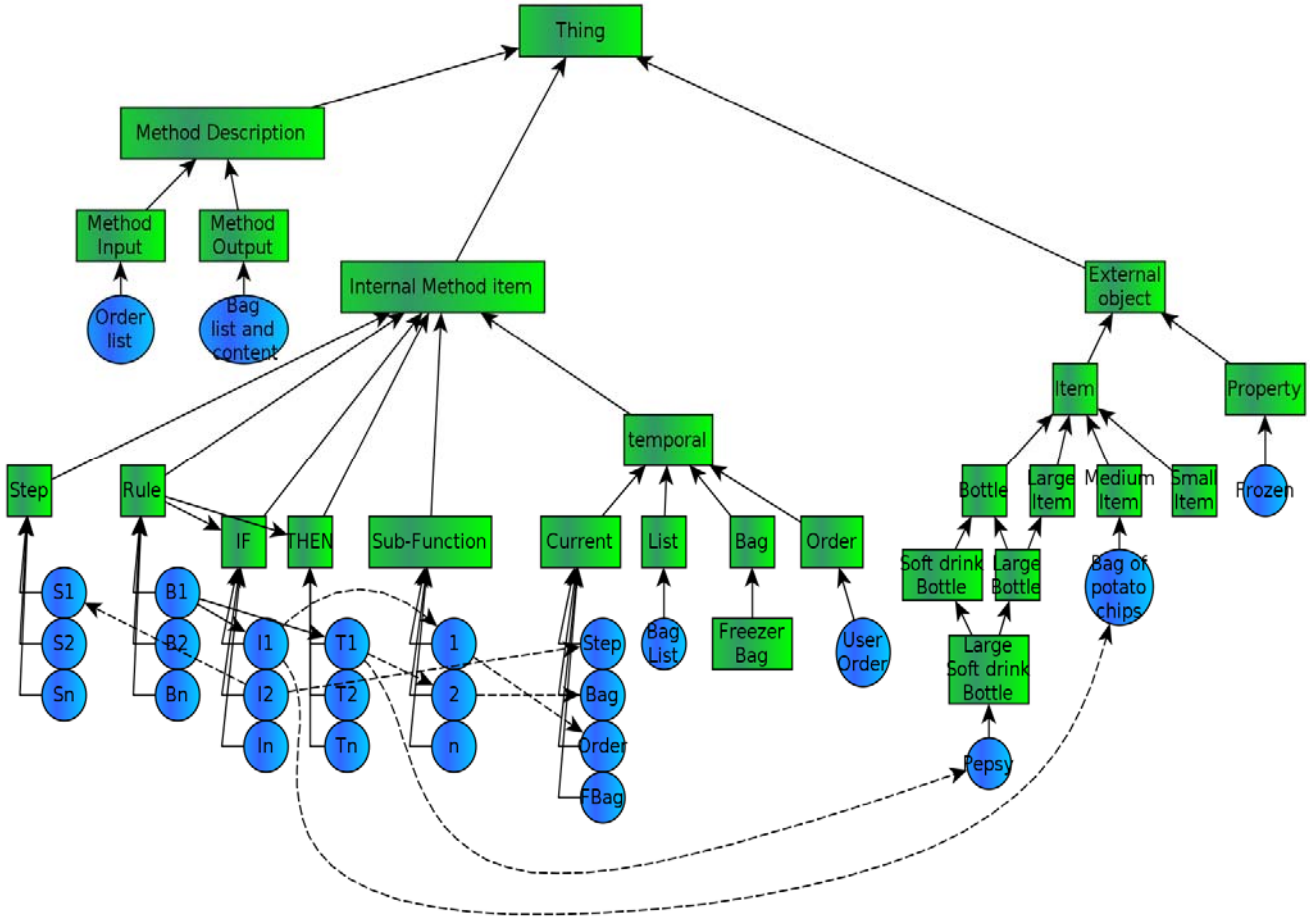

Fig. 2. Method ontology

The concept "Cart" (or order) holds only one individual. This individual describes the current cart or order of the user. This individual needs to be linked with the items described in the domain ontology. During execution, the cart will lose items by giving them to bags. By the end the user's cart or order will be empty.

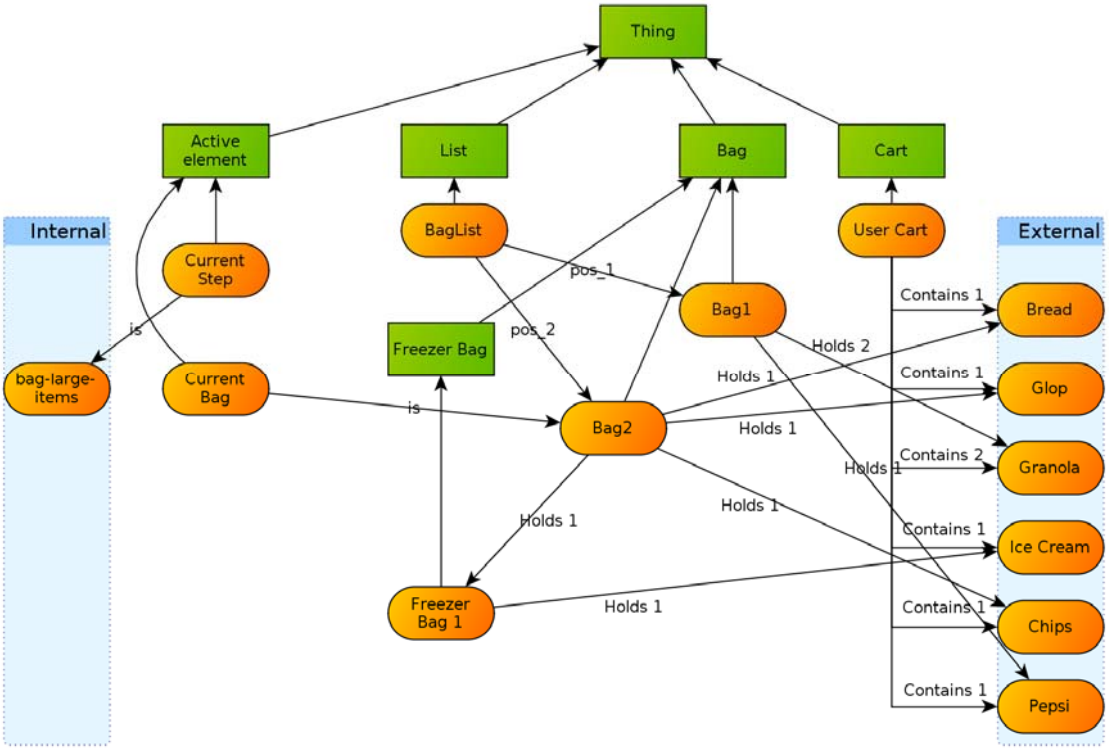

Fig. 3. Temporal working ontology

In order to operate with several items, the properties of bags and carts will have to hold a numeric value, which indicates the number of the same item they hold, for example, "Contains 2". A more specific graphical representation of the temporal ontology is given in Fig. 3.

VII. Using the Method Ontology

If we desire that the method ontology is used as a set of instructions that describe specific actions of a system, we must define a way that would make that possible.

One possible way would be to let the structural representation of the ontology speak for itself. A user or sufficiently intelligent software agent could understand the described actions from the element names alone. Another solution would be to give every function element an additional description for the actions that need to be taken.

Also some sort of language could be introduced in order to make it machine readable and executable. For example, the sub-function "set step" could have an additional description that would state:

```
Set (target at property "uses_1") to
(target at property "uses_2");
```

In such a language the system would have to know the commands "Set" and "to", and understand what other individuals are referenced. This way it would be possible to introduce system specific commands that would carry out the required actions. Using a system that provides the possibility for plug-in development that has a component-based ontology [2], [4], usage in mind would be recommended.

VIII. Conclusion

In this paper we described one possible way of creating and implementing a reusable method ontology that fulfils the bagger algorithm. Reusability would arise from the possibility of mapping the method ontology, which describes the actions, to a new domain ontology model [7]. Some aspects of this approach need to be tested further. In the provided example a specific hierarchy of concepts and even some individuals were given in the method ontology. How will the mapping process be done, when a new domain ontology model is mapped to this method ontology? Solving the problem of concept names not being the same would be easy enough, but what would happen if the structure were not the same. However it seems that in the case of the bagger, successful execution would be possible with a strange domain ontology, as long as only generic items are used and none of the specific cases happen. There also needs to be a more specific description of the reasoning system and how it operates with the ontology models.

REFERENCES

[1] "Production Rules, Bratko ed. 4, chapter 15, page 343" September 2013. [Online]. Available: http://www.cse.unsw.edu.au/~billw/cs9414/notes/kr/rules/rules.html

[2] M. Musen, R. Fergerson, W. Grosso, N. Noy, M. Crubézy, J. Gennari, "Component-Based Support for Building Knowledge-Acquisition Systems", In Proceedings of the Conference on Intelligent Information Processing of the International Federation for Information Processing World Computer Congress, 2000, pp. 18 - 22.

[3] B. Neumann, "Configuration Expert Systems: A Case Study and Tutorial", 1988.

[4] M. Crubézy, E. Motta, W. Lu, M. Musen, "Configuring Online Problem-Solving Resources with the Internet Reasoning Service" IEEE Intelligent Systems, vol. 18, 2002.

[5] J. Gennari, S. Tu, T. Rothenfluh, M. Musen, "Mapping Domains to Methods in Support of Reuse", International Journal of Human-Computer Studies, vol. 41, 1994, pp. 399 - 424.

[6] J. Park, J. Gennari, M. Musen, "Mappings for Reuse in Knowledge-Based Systems", 1998.

[7] R. Studer, H. Eriksson, J. Gennari, S. Tu, D. Fensel, M. Musen, "Ontologies and the Configuration of Problem-Solving Methods", in Proceedings of the 10th BANFF Knowledge Acquisition for Knowledge-based System Workshop, SRDG Publications, 1996, pp. 11 - 31.

[8] H. Park, A. Yoon, H. Kwon, "Task Model and Task Ontology for Intelligent Tourist Information Service" International Journal of U- & E-Service, Science & Technology, Vol. 5 Issue 2, Jun. 2012, pp. 43 - 59.

[9] R. Mizoguchi, J. Vanwelkenhuysen, M. Ikeda, "Task Ontology for Reuse of Problem Solving Knowledge" Towards Very Large Knowledge Bases: Knowledge Building & Knowledge Sharing, 1995, pp. 46 - 59.

[10] H. Gorskis, Y. Chizhov, T. Zmanovska, "Semi-automatic approach to domain ontology building", International Conference on Applied Information and Communication Technologies, 2013.

[11] R. Shiffman, B. Karras, A. Agrawal, „GEM: A Proposal for a More Comprehensive Guideline Document Model Using XML", J Am Med Inform Assoc 7, 2000, pp. 488 - 498.

Henrihs Gorskis is the first-year doctoral student majoring in information technology at Riga Technical University (RTU). He received his Mg.sc. ing. degree in 2013. His research interests include data mining, ontology engineering, evolutionary computing and programming. In 2012 he wrote a paper on ontology development and data mining. In 2013 he participated in the AICT conference and also provided a paper on ontology engineering. He is especially fond of the Java programming language and uses it for both work and personal application development.
E-mail: henrihs.g@gmail.com

Arkady Borisov received his Doctoral Degree in Technical Cybernetics from Riga Polytechnic Institute in 1970 and Dr.habil.sc.comp. degree in Technical Cybernetics from Taganrog State Radio Engineering University in 1986.
He is a Professor of Computer Science at the Faculty of Computer Science and Information Technology, Riga Technical University (Latvia). His research interests include fuzzy sets, fuzzy logic and computational intelligence. He has 235 publications in the field.
He has supervised a number of national research grants and participated in the European research project ECLIPS.
He is a member of IFSA European Fuzzy System Working Group, Russian Fuzzy System and Soft Computing Association, honorary member of the Scientific Board, member of the Scientific Advisory Board of the Fuzzy Initiative Nordrhein-Westfalen (Dortmund, Germany).
E-mail: arkadijs.borisovs@cs.rtu.lv

Henrihs Gorskis, Arkadijs Borisovs. Atkārtoti lietojamās komponentes konfigurācijas dizaina sistēmās, kas balstītas uz zināšanām
Viens no lielākiem izmaksu cēloņiem informācijas tehnoloģiju risinājumu ieviešanā, ir jaunas programmatūras izstrāde. Tas turklāt ir laikietilpīgs process. Viens no iespējamiem risinājumiem būtu jau esošo risinājumu atkārtota izmantošana. Šajā darbā bija veikts mēģinājums piedāvāt uz ontoloģijas balstītas zināšanu sistēmas atkārtoti izmantojamas komponentes. Tas tiek panākts, atdalot visas zināšanas, kas ir saistītas ar risinājumu, no zināšanām, kas apraksta vidi, kurā tiek veikta darbība. Dotajā darbā par zināšanu pamatu tika ņemta uz noteikumiem balstīta zināšanu sistēma. Tā tika pārveidota par ontoloģiju, kurā šie noteikumi ir doti kā konceptu indivīdi. Šādā veidā tika panākta atkārtoti izmantojama metodes ontoloģija. Piemērs dotajā darbā bija balstīts uz noteikumiem, kas aprakstīja iepakošanas procesu. Šādā veidā pats iepakošanas process bija pārveidots par neatkarīgu no iepakojamām precēm. Iegūto ontoloģiju ir iespējams savienot ar citu domēna ontoloģiju un izmantot tās sniegto informāciju par citām precēm vai lietām iepakošanas uzdevumā. Turklāt būtu arī iespējams lietot tikai noteiktas funkcijas no visām ontoloģijā aprakstītajām. Lai būtu iespējams atkārtoti izmantot iepakošanas uzdevuma ontoloģiju vai jebkuru citu uzdevuma ontoloģiju, ir jādefinē ontoloģiju savienošanas vai uzklāšanas darbības. Tas tiek saukts par „mapping" uzdevumu un savā būtībā ir apraksts, kas nosaka, kādi elementi no vienas ontoloģijas atbilst citas ontoloģijas elementiem. Pašlaik joprojām ir daži neatbildēti jautājumi. Piemēram, kā var paredzēt un izlabot iespējamās kļūdas, kuras var rasties ontoloģiju savienošanas rezultātā. Vēl viens līdz galam neizpētīts faktors ir noklusētās īpašības. Rakstot likumus, dažreiz tiek noklusēta svarīga informācija, jo tā ir pašsaprotama noteiktā domēnā. Kā ar to apieties, atdalot izpildi no šāda domēna? Neskatoties uz šīm nepilnībām, darbs parādīja šādas pieejas izmantošanas iespēju. Atkārtoti izmantojamas komponentes var atvieglot jaunu risinājumu ieviešanu uz zināšanām balstītās sistēmās. Šādā veidā ir iespējams izmantot ontoloģijās sniegto informāciju un likumu izpildes funkcionalitāti.

Генрих Горский, Аркадий Борисов. Компоненты многократного использования в системах конфигурационного дизайна, основанных на знаниях
Одним из самых больших источников трат при введении информационно-технологических решений является разработка нового программного обеспечения. Этот процесс, в том числе, занимает много времени. Одним из возможных решений является повторное использование уже существующих компонентов. В этой работе была сделана попытка предложить подход, использующий онтологию. Это было достигнуто путём отделения знаний, связанных с выполнением задачи, от знаний, описывающих среду, в которой задача выполняется. В данной работе основой для знаний послужила система продукционных правил. Она была переделана в онтологию, в которой частями правил выступали индивиды соответствующих концептов. Таким образом, была построена повторно используемая онтология метода. Пример, рассмотренный в данной работе, основан на правилах, которые описывают процесс упаковки товаров. В результате этого процесс упаковывания стал независимым от товаров, которые обрабатываются. Полученную онтологию метода возможно соединить с другой онтологией домена, описывающей другие вещи, которые становятся предметами процесса упаковки. Также возможно использование только некоторых отдельных подфункций в новой области. Для того чтобы было возможно повторное использование онтологии упаковки или любой другой онтологии метода, необходимо описать процесс соединения. Этот процесс называется «mapping», он описывает, какие элементы одной онтологии соответствуют элементам другой онтологии. На данный момент существует несколько вопросов, не имеющих ответа. Например, как предусмотреть и исправить возможные ошибки, которые могут возникать в результате объединения двух онтологий? Ещё одним до конца неисследованным фактором являются скрытые свойства. При написании правил иногда пропускают важную информацию, так как она понятна сама по себе в данной среде, и что делать, отделяя метод от такой среды? Несмотря на данные вопросы, работа показывает возможности такого подхода. Повторно используемые компоненты могут облегчить введение новых решений в системах, основанных на знаниях. Таким образом, возможно использовать информацию, заложенную в онтологиях, и функциональность выполнения правил.