An Approach to Knowledge Integration Applied to a Configuration Problem

Maria Vargas-Vera[1], Miklos Nagy[1], and Dietmar Jannach[2]

[1] The Open University, UK
[2] Computer Science Department
Technical University of Dortmund, Germany
{mvargasvera@gmail.com
M.Nagy@open.ac.uk
Dietmar.Jannach@tu-dortmund.de}

Abstract. This chapter presents a framework for knowledge integration based on mappings between similar concepts in constraint graphs associated to a configuration problem. In particular, the chapter is devoted to one of the problems which could arise when performing collaborative knowledge integration, namely detecting knowledge overlaps. The solution to the overlapping problem relies on the use of matching algorithms. To illustrate our approach we present as a case study a computer configuration problem. This problem is important as it has the promise to become an alternative approach for the current data integration solutions. Through our approach the real cost of integration can be reduced as it is not necessary to invest a great amount of resources beforehand a truly integrated system can be operational.

Key words: Knowledge integration, Semantic Web, Constraint programming

1 Introduction

An important challenge in several fields ranging from design of expert systems to collaborative design construction in engineering is to integrate several sources of knowledge created by different stakeholders. This is not a new problem as it has been around for several decades i.e. CAD (Computer-Assisted Design) systems. In our proposal, we attempt to design a new solution approach, which amalgamates research outputs from Ontology Alignment (in particular distributed solutions) and Fuzzy Logic in order to provide an alternative solution to the problem of knowledge integration.

There exists no unique definition of knowledge integration. Therefore, we need to first clarify the different meanings of the term Knowledge Integration. The definitions found in the fields of Artificial Intelligence, Ontologies, Databases and Knowledge Management vary strongly. For example, in the Artificial Intelligence community, Knowledge Integration is seen as the process of integrating knowledge into an existent body of knowledge [24]. Knowledge integration refers

to the identification of how new and prior knowledge interact while incorporating new information into a knowledge base. In contrast, the view taken from Knowledge Management is that knowledge integration is a fundamental management practice. According to Grant [15], the organization's primary concern is the integration of its dispersed knowledge resources in order to apply them to a "production of a new artifact" as a mean of creating new knowledge out of novel combinations of existing knowledge [15].

In the context of Artificial Intelligence, the problem of knowledge integration can also be seen as a scenario of building distributed knowledge bases in a collaborative way. These knowledge models need to be integrated into a single model. However, while integrating knowledge several problems could arise. One of them is the problem of overlaps and conflicts in the knowledge as subtle and unexpected interactions of knowledge could appear with the newly added knowledge [24]. However, in our opinion knowledge integration is the process of creating an unified knowledge model by means of integrating individual models made by different knowledge engineers. This integration is basically a reconciliation of the terms and relations used by each knowledge engineer while building their own model.

Furthermore, the basic argument is that knowledge cannot be viewed as a simple conceptualization of the world, but it has to represent some degree of interpretation. Such interpretation depends on the context of the entities involved in the process. This idea is rooted in the fact the different entities' interpretations are always subjective, since they occur according to an individual schema, which is than communicated to other individuals by a particular language. These schemas called mental spaces, contexts, or mental models have been investigated in the past [10] [14] [20].

The motivating scenario of our work is that we assume that large knowledge bases are typically constructed by different knowledge engineers or domain experts in an incremental and collaborative way. However, as the individual parts of the knowledge bases may also be developed independently by different teams or organizational units, one or more knowledge integration phase are required in the overall process in order to detect and resolve conflicts and overlaps. The development of a knowledge base for a product configuration system [30] is a typical example as different organizational units contribute technical and process- or marketing-related constraints on legal product constellations. The problem is even harder, when the configurable product is delivered by multiple providers in a supply-chain [4], and requires the cross-company integration of knowledge bases and interfaces. Another example, the creation of a 3D-virtual campus where several departments of the Open University (OU) collaborated in the whole design of the campus i.e. Library, Research School, Computing Department, among others [29].

Our suggested approach to knowledge integration deals, in a first instance, with the overlapping problem. The detection of the overlapping problem is performed by mappings between the knowledge models. Therefore, the main contribution of this chapter is to propose as a solution to the overlapping problem

based on matching algorithms which use Dempster-Shafer and Fuzzy Voting Model. An scenario to illustrate the knowledge integration using DSSim best methods is outlined in our case of study.

The chapter is organized as follows. Section 2 provides an overview of related work. Section 3 presents a case scenario that illustrates the overlapping problem when performing knowledge integration. Section 4 describe details of the mapping process. Section 5 shows our framework to knowledge integration. Section 6 presents an evaluation of our methodology to knowledge integration. Finally, in Section 7 we present our conclusions and describe our future work.

2 Related Work

Several research communities have investigated the information integration problem. This lead to numerous different approaches in a way in which different information sources can be integrated. After an analysis of the literature we have identified four perspectives on our literature review. These perspectives are Knowledge Based Systems, Ontologies, Databases and Knowledge Management. The first perspective deals with problems in knowledge modeling in particular in expert systems. The second perspective is the work in the Ontologies field ranging from ontology merging to alignment. The third perspective, Databases, is more related to data integration which consists of providing an unified view on the data stored in different databases with different models. The Knowledge Management perspective is not explored in too much detail as is not the main focus of this chapter.

2.1 Expert Systems - Knowledge bases

Murray [23] presents an approach to knowledge integration as a machine learning task. He implemented a system called REACT which is a computational model that identifies three activities. (1) "Elaboration": describes, how new and prior knowledge interact, although this feature is restricted to focus only on selected segments of prior knowledge. (2) "Recognition", which identifies the consequences of new information for relevant prior knowledge and (3) "Adaptation", which exploits the learning opportunities by modifying the new and prior knowledge. A learning opportunity occurs when a property of a particular object in the learning context can be generalized into a property for every instance of a class of objects. Empirical evidence indicates that indeed knowledge integration helps knowledge engineers to integrate new information into a large knowledge base.

Knowledge Integration has became an essential element in the Semantic Web Community. For example, knowledge integrations allows to access services which offer knowledge contained in various distributed databases associated with semantically described web portals. In this context Zygmunt et al., propose a framework for knowledge integration supported by using an agent-based architecture [35]. The approach relies very much on the integration of ontologies by

the gradeAgent which estimates the similarity between classes and properties in the ontology. The approach uses algorithms of lexical and structural comparison. The checking of similarity between larger parts of a graph is performed with the use of Similarity Flooding algorithm. The approach also applied additional techniques based on a thesaurus when looking for synonyms and on the use of high level ontology to adjust concepts from the ontology to a given set of concepts which identify important notions. The framework does not handle uncertainty in the similarity metrics. In principle, it seemed as a good solution but in real scenarios the notion of uncertainty limited to a crisp mappings 0 or 1 made a strong limitation in a proper identification of matching concepts and properties.

2.2 Ontologies view

The knowledge engineering community uses ontologies as the main approach for resolving semantic differences in heterogeneous data sources. Based on this approach several categories can be identified to Data Integration. One of them is to create a global ontology. In this way all the different sources share the same ontology in order to make the information integration possible. These solutions fit well when the number of sources is limited and a consensus can be achieved between partners. However, for real life scenarios, this solution is inflexible in nature and is not considered as a viable alternative in the context of knowledge integration.

Ontology merging aims to achieve semantic integration through merging different source ontologies into a consistent union of the source ontologies. These systems make use of the fact that different ontologies have overlapping fragments that is the basis of the merging process. FCAMERGE [13] offers a global structural approach to the merging process. It takes the source ontologies and extracts instances from a given set of domain-specific text documents by applying natural language processing techniques. Based on the extracted instances the system apply formal concept analysis techniques to derive a lattice of concepts as a structural result of merge process. The produced result is explored and transformed to the merged ontology by the ontology engineer. PROMPT [11] is a semi-automatic ontology merging tool that makes initial suggestions based on linguistic similarity between class names then performs automatic updates, finds new conflicts and makes new suggestions.

Ontology mapping aims to achieve semantic integration through the creation of mappings between concepts, attributes etc. between two ontology entities. Based on database schema integration solutions a wide range of techniques has been proposed from manually defined rules to semi automatic approaches that make use of machine learning, heuristics, natural language processing and graph matching algorithms. MAFRA [1], a mapping framework for distributed ontologies supports in interactive, incremental and dynamic ontology mapping process in the Semantic Web context. The main contribution of this approach is that it creates a true distributed ontology mapping framework that is different from mediator based approach. GLUE [2] evolved from a mediator based LSD [9]

data source schema matching, applies machine learning techniques and similarity measures based on joint probabilistic distributions.

Since ontology mapping problem is one of the first steps in the direction of Semantic Web based data and information integration it has become an active research topic recently. As a consequence numerous ontology mapping systems have been proposed but only a handful of them have participated in the Ontology Alignment Initiative (OAEI)[3], which serves as a comparison benchmark for such systems. ASMOV) [19] is an automatic ontology mapping approach, which carries out the mapping in two phases. In the first phase different similarity measures are calculated and combined in order to establish preliminary mapping pairs. In the second phase the system carries out a semantic verification, in order to detect semantically inconsistent mappings and their causes. RiMOM[34] is an ontology mapping approach that uses the combination of different strategies in order to achieve the good results. The different strategies are selected based on the characteristics of the source ontologies and the pre-defined rules. Anchor-Flood [31] is an ontology mapping system, which has been developed in the context of International Patent Classification (IPC) in order to exploit the available taxonomy of related terms in an abstract and align it with the taxonomy of IPC ontology. The mapping is done in two phases. First part is the ontology mapping, where the concepts and properties in the different ontologies are aligned. The second part of the mapping process is the mapping of the instances of the ontologies. Anchor-Flood approach assumes that neither ontology concepts nor an instance comprises the full specification in its name or URI alone. TaxoMap [17] is an ontology mapping tool, which was designed to support information integration between different sources. The mapping process is oriented from ontologies that describe external resources (named source ontology) to the ontology (named target ontology) of different web portals. However the system design assumes that target ontology is supposed to be well-structured, whereas source ontology can be a flat list of concepts. Therefore TaxoMap heavily relies on the labels it uses a moropho-syntactic analysis for tagging text with part-of-speech and lemma information and a similarity measure which compares the trigraph of the concept labels. Lily [33] is an ontology mapping system for integrating information described by heterogeneous ontologies. The system employs hybrid matching strategies to create the mappings for both normal and large scale ontologies.

Summing up the suggested solutions ranges from creation of global views of ontologies, mapping or combining. However we believe that the creation of a global view (or global ontology) is a limited solution as it seems to work when the number of models is limited. The mapping solution is more appealing as the combining solution seems to be not scalable.

[3] http://oaei.ontologymatching.org/

2.3 Databases

In the database community several solutions have been proposed. However not all approaches [6] have been implemented in real life applications. The characteristics of these approaches are that they all have inputs and outputs, which is supplied or processed by a human designer. The inputs are usually the domain models including entity relationships, views and sometimes queries whereas the outputs are conceptual models, global schemas, mapping rules or conflicts. The majority of approaches based on a mediator architecture that involve logical database schemas, which are used as shared mediated views over the queried schemas. A number of systems have been proposed e.g. TSIMMIS [12], Information Manifold[21], InfoSleuth [7], MOMIS [8] , LSD [3] that shows the flexibility and the scalability of these approaches. In particular, MOMIS is focused a data integration from scientific data sources but it also been applied to other domains like building a tourism information provider [8]. The problem, however, is that these solutions rely on the initial idea of database schema integrations namely to create a global view, which will be used as a mediator between the different sources. According to Halevy [16] the major bottleneck in setting up a data integration framework in databases community is the effort required to create the source description and more specifically writing the semantic mappings between the sources and the mediated schema. Of course, we share the same opinion and this can be understood as we expand more in this issue.

The database integration schema's solution requires that an integrated global schema is designed from the local schemas, which refers to existing databases. This global schema is a virtual view of all databases taken together in a distributed database environment. The conceptual modelling of a database (DB) schema is mainly based on Entity-Relationship (ER) model or Unified Modelling Language (UML) class diagrams. There are two design approaches namely direct and gradual. In the direct approach, user requests are processed all at once and the whole DB schema is created directly. This approach is appropriate in the cases of designing small DB schemas, but it is inappropriate in cases when a complex DB schema should be integrated. The gradual approach is used when the number and complexity of user requests are beyond the designer's power of perception. Design of a complex DB schema is based on a gradual integration of external schemas. An external schema is a structure that, at the conceptual level, formally specifies the user's view on a DB schema. Each transaction program that supports a user request is based on an external schema, which is associated to it. After being created, external schemas are integrated into a conceptual DB schema. Nevertheless this idea has been developed further once the ontologies have been proposed as described in section 2.2.

2.4 Knowledge Management

Hung [18] present an empirical study that investigates the patterns of knowledge integration in the collaborative development of system on a chip (SoC) by semiconductor firms. The study focused on the central interactive process

for engineering applications and experimental practice to enhance knowledge integration and technology innovation for rapid product development. A process model for knowledge integration via experimental practice is presented; further explanation can be found in [18]. The process of knowledge integration is triggered by new requirements i.e. new product features or testing methods, which cannot be resolved based on the current knowledge. This integration process depends upon knowledge already existing in the organization as well as new external knowledge. The outcome of the process is a technological innovation and the fact that the knowledge of the organization is enhanced by means of knowledge integration. The Knowledge Management perspective which appears related to our work is the one based on the Distributed Knowledge Management (DKM) approach explored in the Knowledge Management community [22], in which subjective and social aspects of the real world are taken into account. However, this perspective is not going to be explored as is out of the scope of this chapter.

Summing up we could say that there are some commonalities between the four perspectives namely Knowledge Based Systems, Ontologies, Databases and Knowledge Management. The work on knowledge based systems community is concerned with interaction of knowledge when a new piece of knowledge is found and added to the knowledge base. The work on knowledge integration found in the ontology merging community appears to be suitable for detecting overlapping between different knowledge bases. The work in the Database community is more concerned with data integration. Finally, the fourth one is however more related to a processes/products specifications and organizational aspects, which need to be modified when adding new knowledge as the solution relies in adapting a known case to the new situation.

3 Scenario

The scenario presented in this chapter illustrates the problem of different views of knowledge modeled by different departments within an organization. Let us imagine the scenario where we have a Computer manufacturing firm formed with three departments, for example, Sales, Technical Design Manufacturing and Software Department. Each of these departments have already pre-established functions within the organization. These functions are not shown explicitly in Figure 1 but they are part of the box Enterprise Resource Planning (ERP). The Configuration Logic stores the CSP (Constraint Satisfaction Problem) applied to Computer Configuration.

Customers request a computer with certain specification using the interface provided by the “configuration system” then as, a second step, the “configuration system” returns “Quotes” and then the customer could make an order using the on-line system.

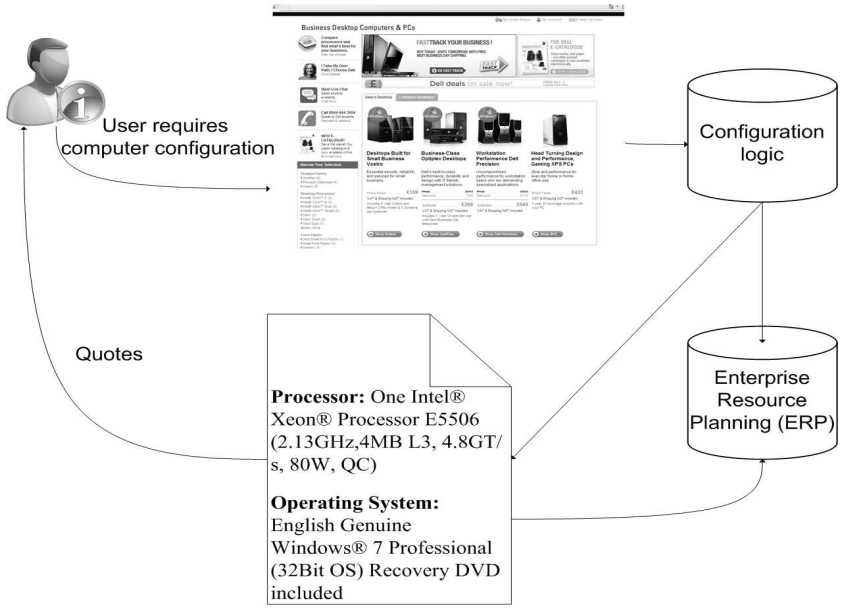

Fig. 1. Computer configuration system

3.1 Constraint Satisfaction Problem (CSP)

This section gives firstly a brief descriptions of Constraint Satisfaction Problem (CSP) and related terminology and secondly our Case Study namely computer configuration problem using the definition given in section 3.1.

Definition 1 *A CSP is a triple* $P = \{V, D, C\}$, *where* $V = \{v1, v2, ..., vn\}$ *is the set of variables called the domain variables;*

$D = \{D1, D2, ..., Dn\}$ *is the set of domains. The domain is a finite set containing possible values for the corresponding variables;*

$C = \{c1, c2, ..., cn\}$ *is the set of constraints. A constraint ci is a relation defined on a subset of* $\{vi, ..., vk\}$ *of all the variables; that is,* $\{Di, ..., Dk\} \supseteq ci$.

The structure of a CSP may be represented by a constraint graph, which is defined as follows: variables are represented with nodes, and the constraints between them are represented with edges. The labels of the edges represent the constraints and the labels of the nodes represent the domain of the variables.

Definition 2 *Assignment: It is a mapping from a set of variables to their corresponding domains. Let* v_i *be a variable and* D_i *its domain. The process that* v_i *takes a value, say* d_i *from the domain* D_i *is called assignment. Such an assignment is denoted* (v_i, d_i).

For a CSP problem which has a set of variables, say $v_1, v_2, ..., v_m$ the assignment for all the variables is denoted $\{(v_1, d_1), (v_2, d_2), ..., (v_m, d_m)\}$. When all the variables are assigned a value, the assignment is called complete, otherwise partial.

3.2 Case Study: Computer Configuration Problem

The case of study proposed in this chapter is a restricted version of a computer configuration problem. The problem of configuration is defined as a CSP (Constraint Satisfaction Problem) problem where we define our model using Variables, Domain for the variables and Constraints over the variables. The main goal is to obtain an assignment (i.e. a value for all the variables). The CSP is defined formally as follows:

Variables:

Table 1. Variables

V1	OS
V2	Memory
V3	Hard_disk_size
V4	CPU
V5	Monitor
V6	Mouse
V7	Video_card
V8	Graphics_card
V9	Gaming_PC
V10	Keyboard
V11	Monitor_resolution

Table 2. Domain

D1	OS	Vista, XP, MAC-OS, Windows_7,Linux
D2	Memory	512_MB, 1024_MB, 2048_MB, 3072_MB
D3	Hard_disk_size	160_GB, 180_GB, 320_GB
D4	CPU	Pentium_4, Intel_Centrino
D5	Monitor	14_inches,18_inches,19_inches,20_inches
D6	Mouse	Logitech, Magic_mouse
D7	Video_card	NVIDIA_600series, NVIDIA_700series, NVIDIA_800series
D8	Graphics_card	GeForce_7600series, GeForce_7800series, GeForce_7900series
D9	Gaming_PC	yes,no
D10	Keyboard	Win_keyboard,Mac_Keyboard
D11	Monitor_resolution	low,medium,high

Constraints:

Table 3. Constraints

C1	$IF\ OS = \text{“}XP''\ THEN\ Memory \leq 2048\_MB$
C2	$IF\ Monitor = \text{“}20\_inches''\ THEN\ Graphics\_card = \text{“}GeForce\_7800\_series''$
C3	$IF\ OS = \text{“}XP''\ THEN\ CPU = \text{“}Pentium\_4''$
C4	$IF\ OS = \text{“}Vista''\ THEN\ CPU = \text{“}Pentium\_4''$
C5	$IF\ OS = \text{“}XP''\ THEN\ Hard\_disk\_size \geq \text{“}500\_MB''$
C6	$IF\ Gaming\_PC = \text{“}yes''\ THEN\ Graphics\_card = \text{“}NVIDIA\_8000series''$
C7	$IF\ Gaming\_PC = \text{“}yes''\ THEN\ Memory \geq \text{“}2048''$
C8	$IF\ Gaming\_PC = \text{“}yes''\ THEN\ Hard\_disk\_size \geq \text{“}160GB''$
C9	$IF\ Monitor \geq 20inches\ THEN\ Monitor_Resolution = \text{“}high''$
C10	$IF\ OS = \text{“}MAC - OS''\ THEN\ Keyword = \text{“}Mac\_Keyboard''$

3.3 Constraint Graph

In order to represent the problem we use graphs, which can be defined as follows:

Definition 3 *Constraints are represented in a graph called constraint graph. Each node in this graph is labelled by a variable name together with a set of possible values for that variable. A directed constraint connected(i,j) connects a pair of nodes i and j if the value of the variable labeling i is constrained by the value of the variable labeling j.*

To illustrate our approach to mapping, we have taken the initial constraints graphs built by different engineers using different knowledge models. These original graphs hold by our individual departments are depicted in Figure 2 and 3. These graphs use standard computer jargon although, they have discrepancies on the name of variables used. The term *Video_card* and *Graphics_card* (variable names) were used by different knowledge engineers to refer to the same concept. The latest problem suggested that in order to perform knowledge integration we have to perform mappings between nodes in the constraints graphs. For the sake of clarity, we only presents overlaps in one node of the graph but this is not always the case. Figure 2 uses as variable name called *Video_card* whilst in Figure 3 the variable name is *Graphics_card*.

A unified view of two constraints graphs was produced (manually) by joining two initial constraints graphs. This unified view is depicted in Figure 4. In one hand, Figure 4 partially shows a constraint graph with nine variables namely OS, Memory, Hard_disk_size, CPU, Monitor, Gaming_PC, Video_card, Graphics_card and Monitor_Resolution.

4 Mapping Process

The main objective of the mapping is to identify that nodes in the constraint graphs are equivalent e.g. in graph 4 the “Video_card” is equivalent with the

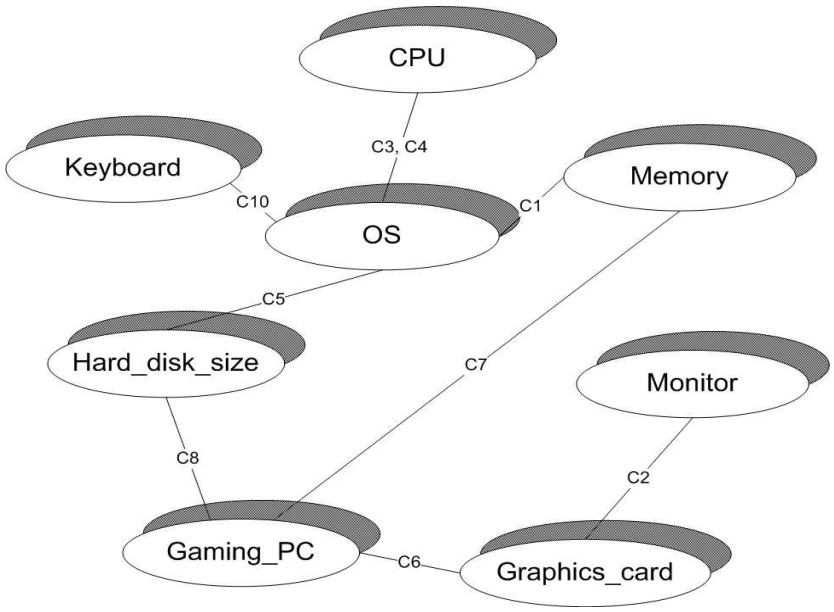

Fig. 2. A Constraint graph for the computer configuration problem using variable Graphics_card

"Graphics_card" in graph 5. In order to proceed with the comparisons we need to compare all possible node combinations of graph 4 and 5 and select the ones, which are the most similar or nothing if there is no similarity between the nodes.

Once the constraint graphs have been established the system need to establish mappings between the hardware items represented as nodes in the graph. The problem can be represented as the ontology-mapping problem in order to find correspondences between the items. The objective of the ontology mapping is to use different similarity measures in order to establish the mappings. However in practice one similarity measure or some technique can perform particularly well for one pair of concepts or properties and particularly badly for another pair of concepts or properties, which has to be considered in any mapping algorithm. In our ontology-mapping approach we use different software agents where each agent carries only partial knowledge of the domain and can observe it from its own perspective where available prior knowledge is generally uncertain. Our main argument is that knowledge cannot be viewed as a simple conceptualization of the world, but it has to represent some degree of interpretation. Such interpretation depends on the context of the entities involved in the process. In order to represent these subjective probabilities in our system we use the Dempster-Shafer theory of evidence [32], which provides a mechanism for modelling and reasoning uncertain information in a numerical way, particularly when it is not possible to assign belief to a single element of a set of variables.

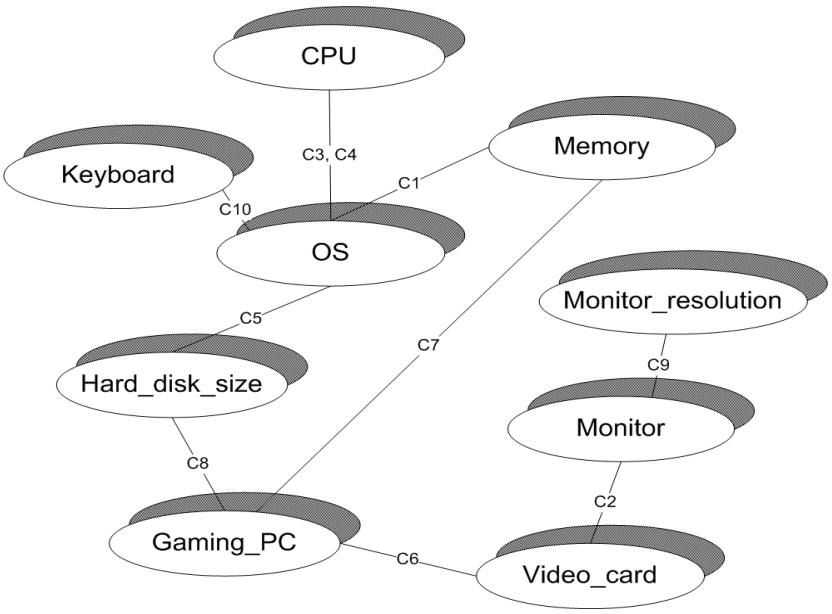

Fig. 3. A Constraint graph using variable Video_card

Further our proposed solution involves consultation of background knowledge, assessment of similarities, resolving conflicts between the assessments and finally the selection of possible mappings i.e. items that are named differently but are the same in practice. As an example consider that we need to determine that the "Video_card" is equivalent to the "Graphics_card". In this case our hypothesis (H) is that these items are equivalent but we need to find evidences that support or contradict our initial hypothesis. In our case we create several hypothesises comparing each element of the constraint graph to each other. As an example consider that the following three hypothesises were selected from all available ones:

$$H_1(equivalent) = \{video\_card\} \Leftrightarrow \{graphics\_card\}$$
$$H_2(equivalent) = \{video\_card\} \Leftrightarrow \{mouse\}$$
$$H_n(equivalent) = \{video\_card\} \Leftrightarrow \{term_n\}$$

Further it is advisable that during the similarity assessment we use different similarity algorithms i.e. use different agents that are specialised in a particular similarity assessment. Since the hierarchy of the constraint graph cannot be exploited for similarity assessment the only way is to utilise the nodes in order to detect the mappings. As such consider that we use three agents using different string similarity measures. The steps to produce the mappings are as follows:

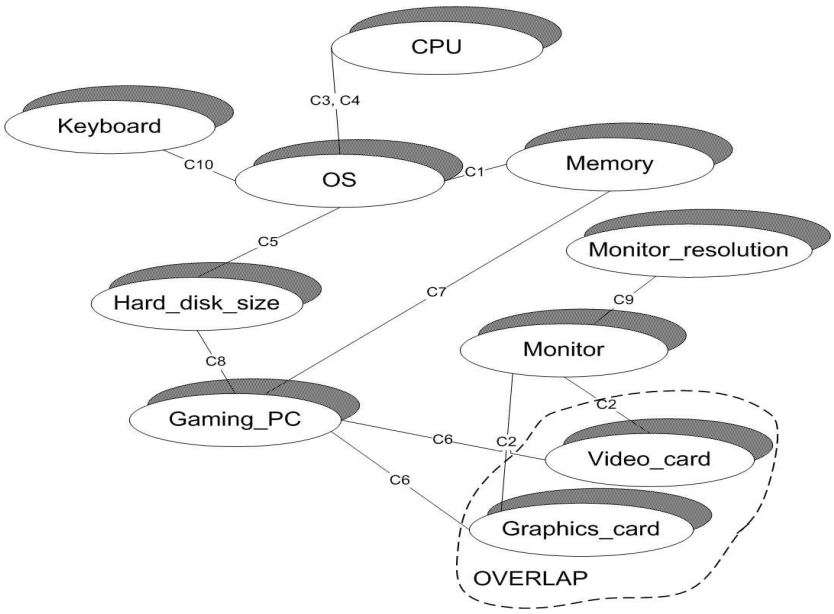

Fig. 4. A Constraint graph for the computer configuration problem with 9 variables

Step 1 consult background knowledge: In this step using general background knowledge e.g. WordNet we try to determine the meaning of the terms. Our case is specialised as the computer shop only sells electronics therefore other meanings e.g. art context of graphics can be excluded from the process. After consulting background knowledge we can extend our initial terms using sister terms and direct hypernyms with the following computer science related terms:

$$Video\_card = \{videodisplay, graphics, picture, graph\}$$
$$Graphics\_card = \{picture, movie, video, image, visual\ representation\}$$
$$Mouse = \{trackball, rotatableball, cursor\ control\ device\}$$

Step 2 similarity assessments: Using different string similarities e.g. Jaccard, Jaro-Winkler, Monge-Elkan we have found that

$$Agent1 : H_1(mapping) = 0.80; H_2(mapping) = 0.3$$
$$Agent2 : H_1(mapping) = 0.72; H_2(mapping) = 0.2$$
$$Agent3 : H_1(mapping) = 0.64; H_2(mapping) = 0.2$$

After the belief assessments we can establish that H1 is the preferred choice between the available hypothesises and that H2 does not contain contradictory

beliefs. However H1 contains contradictions because Agent 2 belief does not support sufficiently that H1 can be selected. The different strategies for selecting the contradicting belief is out of the scope of this paper but for our scenario we use the rule of thumb that in an ordered list of beliefs at least 2 agents should have the same belief otherwise there is a contradiction. In our framework all the numerical values represent the belief mass function that each agent can deduce from the similarity calculations. The represented beliefs are the interpretation of each agent and such they are subjective. Once the beliefs in similarities have been established agents need to select the hypothesis with the highest belief. In our example this corresponds to the H1 namely that the "Video_card" and "Graphics_ card" could be similar. Before the mapping is selected we need to verify that the original beliefs are not contradicting.

Step 3 verification and resolution of contradictions: It is important to point out that our proposed approach does not utilise thresholds for defining what is contradicting or not. e.g. if the difference is greater than 0.5 then there is a contradiction. Our solution makes use of comparisons between each agent's belief and eliminates the one that can be contradictory with the majority of the beliefs. The strategies for selecting, which agent should start evaluating trust is a complex issue and is out of the scope of this paper. However in our scenario we consider a basic rule that tries to establish similar beliefs of at least two agents. Therefore the beliefs in similarities need to be ordered and the agent whose belief function value is the smallest (smaller then the highest and greater than the smaller) will start to the trust evaluation process. In our example Agent 2 is in the position of detecting such contradiction as both Agent 1 and Agent 3 has different belief on the similarity. The question in this case is to trust Agent 1 and support that "Video_card" and "Graphics_card" is equivalent or trust Agent 3 whose belief is lower and probably discharge the mapping.

In order to resolve the contradiction we use the fuzzy voting model [5] because the different beliefs in similarity can be resolved if the mapping algorithm can produce an agreed solution, even though the individual opinions about the available alternatives may vary. We propose a solution for reaching this agreement by evaluating trust between established beliefs through voting, which is a general method of reconciling differences. Voting is a mechanism where the opinions from a set of votes are evaluated in order to select the alternatives that best represent the collective preferences. Unfortunately deriving binary trust like trustful or not trustful from the difference of belief functions is not so straightforward since the different voters express their opinion as subjective probability over the similarities.

For a particular mapping this always involves a certain degree of vagueness hence the threshold between the trust and distrust cannot be set definitely for all cases that can occur during the process. Additionally there is no clear transition between characterising a particular belief highly or less trustful. Therefore our argument is that the trust membership or belief difference values, which are expressed by different voters can be modelled properly by using fuzzy representation. Before each agent evaluates the trust in other agent's belief over

the correctness of the mapping it calculates the difference between its own and the other agent's belief. Depending on the difference it can choose the available trust levels e.g. if the difference in beliefs is 0.08 (belief of Agent 2 - Agent 3 and belief of Agent 3 - Agent 2) then the available trust level can be high and medium. We model these trust levels as fuzzy membership functions. In fuzzy logic the membership function (x) is defined on the universe of discourse U and represents a particular input value as a member of the fuzzy set i.e. $\mu(x)$ is a curve that defines how each point in the U is mapped to a membership value (or degree of membership) between 0 and 1. Our membership functions are as follows:

Definition 4 *Similarity is an input variable and is the result of some syntactic or semantic similarity measure between two terms/nodes in the ontology. These similarity measures can be obtained using a wide variety of standard techniques like Jaccard distance or node distance in source and target graphs that represent the ontology fragments. In terms of fuzzy representation we propose three values for the fuzzy membership value* $\chi(x) = \{low, average, high\}$

Definition 5 *Belief is an input variable, which describes the amount of justified support to A that is the lower probability function of Dempster, which accounts for all evidence* E_k *that supports the given proposition A. Consequently the belief value is equivalent to the normalised sum of similarity values that is calculated based on the evidences that support the hypothesis. We propose two values for the fuzzy membership value* $\beta(x) = \{weak, strong\}$

Definition 6 *Belief difference is an input variable, which represents the agents own belief compared to the other agents' belief over the correctness of a mapping in order to establish mappings between concepts and properties in the ontology. Therefore during conflict resolution we calculate the level of difference by comparing agent x belief to agents x-1, x+1 beliefs over the similarity. We apply three values for the fuzzy membership value* $\mu(x) = \{small, average, large\}$

Definition 7 *Trust is the output variable and represent the level of trust we can assign to the combination of our input variables. The trust is therefore calculated by applying the fuzzy rules on the fuzzy input variables. We propose three values for the fuzzy membership value* $\tau(x) = \{low, medium, high\}$

Once each input and output variables have been initialised we run the fuzzy system[4] that defuzzifies the result defined by our output variable i.e. trust. During fuzzy reasoning we have the linguistic output variables, which need to be translated into a crisp (i.e. real numbers, not fuzzy sets) value. The objective is to derive a single crisp numeric value that best represents the inferred fuzzy values of the linguistic output variable. Defuzzification is such inverse transformation, which maps the output from the fuzzy domain back into the crisp domain. In our ontology mapping system we have selected the Center-of-Area (C-o-A) defuzzification method. The C-o-A method is often referred to as the Center-of-Gravity

[4] http://jfuzzylogic.sourceforge.net/

method because it computes the centroid of the composite area representing the output fuzzy term. In our system the trust levels are proportional with the area of the membership functions therefore other defuzzification methods like Center-of-Maximum (C-o-M) or Mean-of-Maximum (M-o-M) does not correspond well to our requirements.

Consider the set of words $\{Low\_trust(L\_t), Medium\_trust(M\_t), High\_trust(H\_t)\}$ as labels of a linguistic variable trust with values in U=[0,1]. Given a set "m" of voters where each voter is asked to provide the subset of words from the finite set T(L), which are appropriate as labels for the value u. The membership value $\chi\mu(w)(u)$ is taking the proportion of voters who include u in their set of labels, which is represented by w. The main objective when resolving conflict is to have sufficient number of independent opinions that can be consolidated. To achieve our objective we need to introduce more opinions into the system i.e. we need to add the opinion of the other agents in order to vote for the best possible outcome. Therefore we assume for the purpose of our example that we have 10 voters (agents). Formally, let us define

$$\begin{aligned} V &= \{A1, A2, A3, A4, A5, A6, A7, A8, A9, A10\} \\ T(L) &= \{L_t, M_t, H_t\} \end{aligned} \tag{1}$$

The number of voters can differ however assuming 10 voters can ensure that

1. The overlap between the membership functions can proportionally be distributed on the possible scale of the belief difference [0..1]
2. The work load of the voters does not slow the mapping process down

Let us start illustrating the previous ideas with a small example. By definition consider three linguistic output variables L representing trust levels and T(L) the set of linguistic values as $T(L) = \{Low\_trust, Medium\_trust, High\_trust\}$. The universe of discourse is U, which is defined as U=[0,1]. Then, we define the fuzzy sets per output variables $\{(Low\_trust), (Medium\_trust), (High\_trust)\}$ for the voters where each voter has different overlapping trapezoidal, triangular or Gauss membership functions as depicted on Figure 5.

The difference in the membership functions represented by different vertices depicted in Figure 5 ensures that voters can introduce different opinions as they pick the possible trust levels for the same difference in belief. The possible set of trust levels L=TRUST is defined by the Table 4. Note that in the table we use a short notation L_t stands for Low_trust, M_t stands for $Medium\_trust$ and H_t stands for $High\_trust$. Once the input fuzzy sets (membership functions) have been defined the system is ready to assess the output trust memberships for the input values. Both input and output variables are real numbers on the range between [0..1]. Based on the difference of beliefs, own belief and similarity of the different voters the system evaluates the scenario.

The evaluation includes the fuzzification which converts the crisp inputs to fuzzy sets, the inference mechanism which uses the fuzzy rules in the rule-base to

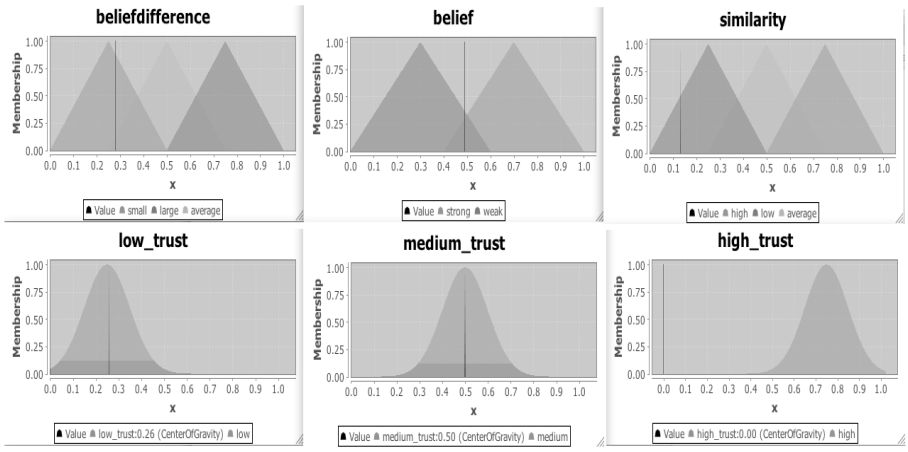

Fig. 5. Possible membership functions

produce fuzzy conclusions (e.g., the implied fuzzy sets), and the defuzzification block which converts these fuzzy conclusions into the crisp outputs. Therefore each input (belief difference, belief and similarity) produces a possible defuzzified output (low, medium or high trust) for the possible output variables. Each defuzzified value can be interpreted as a possible trust level where the linguistic variable with the highest defuzzified value is retained in case more than one output variable is selected. As an example consider a case where the defuzzified output has resulted in the situation described in Table 4. Note that each voter has its own membership function where the level of overlap is different for each voter. Based on a concrete input voting agent nr 1 could map the defuzzified variables into high, medium and low trust whereas voting agent 10 to only low trust.

Table 4. Possible values for the voting

A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
L_t	L_t	L_t	L_t	L_t	L_t	L_t	L_t	L_t	L_t
M_t	M_t	M_t	M_t	M_t	M_t				
H_t	H_t	H_t							

Note that behind each trust lever there is a real number, which represents the defuzzified value. These values are used to reduce the number of possible linguistic variables in order to obtain the vote for each voting agent. Each agent retains the linguistic variable that represents the highest value and is depicted in table 5.

Table 5. Voting

A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
H_t	M_t	L_t	L_t	M_t	M_t	L_t	L_t	L_t	L_t

Taken as a function of x these probabilities form probability functions. They should therefore satisfy:

$$\sum_{w \in T(L)} Pr(L = w|x) = 1 \tag{2}$$

which gives a probability distribution on words:

$$\sum Pr(L = Low\_trust|x) = 0.6 \tag{3}$$

$$\sum Pr(L = Medium\_trust|x) = 0.3 \tag{4}$$

$$\sum Pr(L = High\_trust|x) = 0.1 \tag{5}$$

Therefore applying the appropriate input variables and the basic fuzzy rules the system will determine that as a result of voting and given the difference in belief x = 0.08 (belief of Agent2 - Agent 3 and belief of Agent 3 - Agent 2) the combination should not consider the belief of the third agent since based on its difference compared to another beliefs it turns out to be a distrustful assessment. The before mentioned process is then repeated as many times as many different beliefs we have for the similarity i.e. as many as different similarity measures exist in the ontology mapping system.

Step 4 belief combination: Once the conflicts have been resolved and the distrustful beliefs have been eliminated the individual beliefs can be combined to a more coherent belief. This belief combination is done using the Dempster's combination rule:

$$m_{ij}(A) = m_i \oplus m_j = \sum_{E_k E_{k'}} m_i(E_k) * m_j(E_{k'}) \tag{6}$$

where we have two individual belief mass functions $mi(Ek)$ and $mj(Ek')$ and i and j represent two different agents. After the belief combination the belief in H1 equals 0.79 and for H2 equals 0.25. As a consequence we can deduce that the "Video_card" and "Graphics_card" are the same component.

5 Knowledge Integration Framework

Ontologies offer interoperability and the possibility of a real integration of heterogenous sources. The vision of the Semantic Web predicts that existing resources i.e. databases, data on web pages will be described using ontologies.

These ontologies would either be created by individuals, organisations or as a result of converting existing thesaurus like WordNet. These resources will be user by software applications in order to determine the meaning of concepts, properties and to exchange these meanings in a certain context. Therefore in our Knowledge Integration framework we have included two ontologies containing the background knowledge for the Computer Configuration Problem. We assess similarities in the provided ontologies in order to find similar structures and terms. This background knowledge is used later at the level of the CSP solver so overlapping knowledge can be detected effectively using DSSim created background knowledge from domain ontologies.

The integration framework used in this work is depicted in Figure 6 where the flow of control between modules is shown. The picture shows the knowledge models used in the framework. In our particular case, we are using as knowledge models ontologies from PC online shops. These ontologies were built by ourselves and they are written in OWL the standard ontologies language. Then, we populate our ontologies using the values of two on-line shops. The suggested framework is generic as a first instance (for testing our ideas) the two models are OWL ontologies however, other formalisms could be used for building the models, for example, the formalisms which could be easily converted into the OWL ontology language.

We assume in our Knowledge Integration framework that each model has been built by different knowledge engineers and therefore, overlapping and contradictions of knowledge might occur. Our solution to the overlapping knowledge is performed by using DSSim. Although, for the purpose of this chapter, DSSim is presented as a black box as it has been described elsewhere [25] [26] [28] [27]. DSSim is an ontology alignment system based on Dempster Shafer and a fuzzy voting model. Besides DSSim uses background knowledge and WordNet to assess similarity between classes an properties in ontologies. The models are the input to our DSSim system which produces the mappings detected between the models. Currently, for the purpose of the experiment shown in the next section we are dealing only with two ontologies (i.e. two models). However, the framework can be extended to deal with multiple knowledge models. Preliminary results suggest that one problem addressed in our knowledge integration framework namely the detection of overlapping knowledge can be solved using the mappings obtained by DSSim.

The integrator module uses DSSim outputs and makes requests for approval of mappings to the knowledge engineer or human expert. The "mapping information" obtained by DSSim is passed to the Integrator module which then modifies the constraints graphs using the approval information provided by the knowledge engineer. Finally, the modified Constraint Satisfaction Problem (modified constraint graph) is passed to the choco solver which solves the Constraint Satisfaction Problem and returns results to the final user. In our research we used choco as it is a standard constraint solver which can be integrated with Java. Therefore, it fits very well with our overall knowledge integration framework.

The "Configuration System Interface" is the front-end to our users. Currently is not an elaborated interface as our priority was to test our ideas on the knowledge integration framework. Next section presents an experiment using two knowledge models on our case of study (a restricted version of a configuration problem).

5.1 Algorithms for Detecting and Correcting Overlappings

Two different algorithms play a key role in our framework as they deal with detection and correction of overlaps i.e. mappings and the resolution of the constraint satisfaction program (CSP). Our strategy of combining these algotithms is to create a clear split between the steps considering performance reasons. Knowledge integration on real domains is a challenging and complex task, involving two computationally expensive operations:

1. Mapping terms between different sources: In order to detect overlapping in the configurations we need to create mappings between all possible items in the inventory (all instances in both ontology 1 and ontology 2). This operation typically involve comparisons on a state space that is the cartesian product of the different inventory items.

$$INV_n \times INV_m = \{(item_n, item_m) | item_n \in INV_n \; and \; item_m \in INV_m\} \quad (7)$$

   Consequently the possible comparisons increases as the number of inventory items increases therefore a real time comparison can easily become unfeasible. Further in real life scenarios the inventory cannot change between proposing two configurations to the user therefore the mapping can be and should be done only when the inventory in the sources are changing. As a result we create the mappings between the inventory items first and we run the mapping algorithm when necessary during the knowledge integration.
2. CSP search strategy and model solving: A key ingredient of any constraint solving approach is the appropriate construction of the search tree and the definition of the search algorithm. A common approach is by assigning variables to certain values however the size of the search tree is proportional to the number of variables and domain values we use in the model. Therefore finding a solution for a certain problem can easily become unfeasible in real time if the number of components for the configuration is increasing. Unfortunately contrary to the mapping generation the constraint satisfaction problem need to be resolved in the real time every time the user asks for a certain quote from the system. Therefore in our approach we need to put an upper limit on the time the algorithm can run in order to provide answer to the user' query.

The ontology mapping process that includes the fuzzy voting is described in Algorithm 1. The input of the algorithms are the similarity matrixes that contain various similarity measures. The output of this algorithm is the mapping file in the OAEI format. This mapping file describes, which items are equivalent

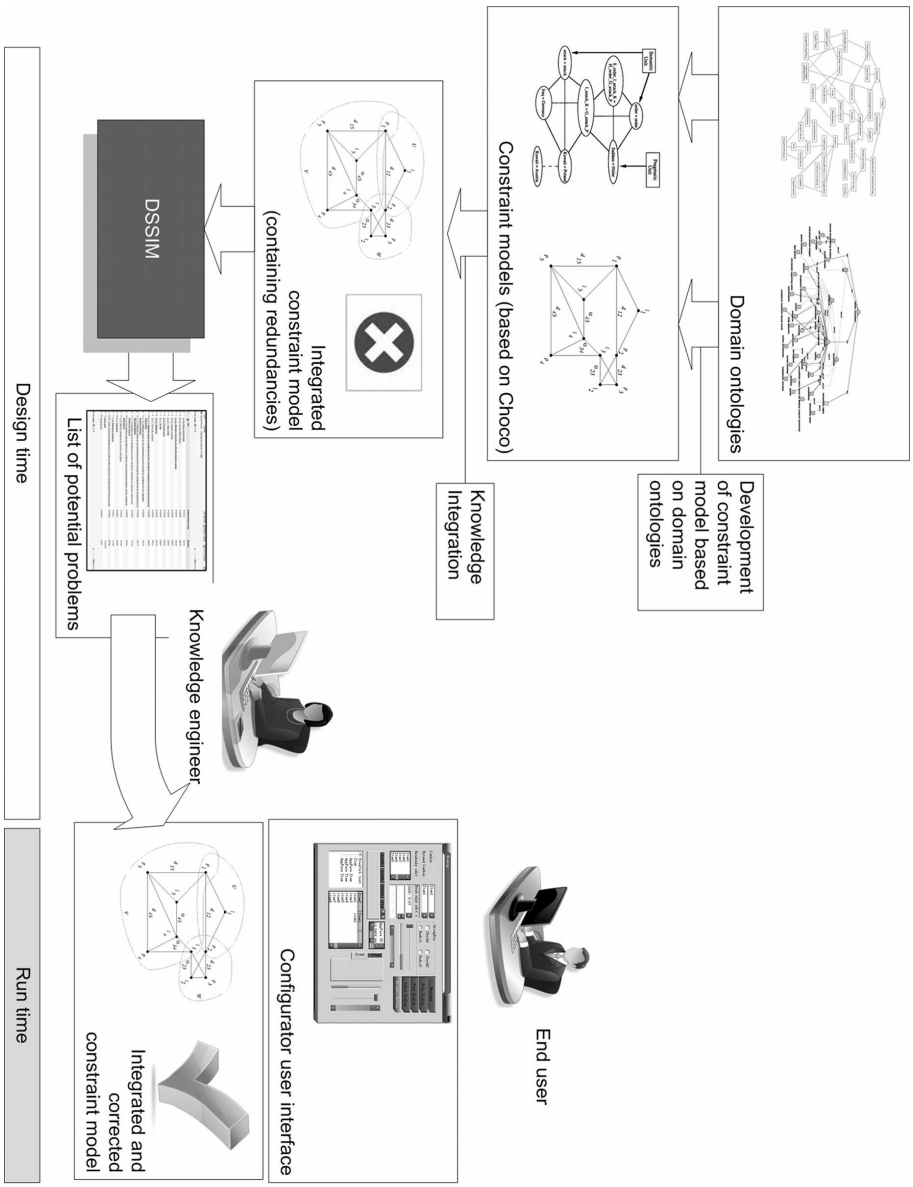

Fig. 6. Knowledge Integration Framework for a Configuration Problem

in the different inventories e.g. "Video_card and Graphics_card". As we have described before the algorithms iterates through the similarity matrixes and tries to establish and combine the Dempster belief functions using scenario and evidences (line 1-7). In case the evidences are contradictory a number of voters are created in order to determine, which belief can be trusted (line 8-14). Once the trusted beliefs have been selected the algorithm combines the beliefs and creates the mapping file based on these beliefs (line 16-17). This iterative process finishes once all items from inventory 1 have been compared with all the items in inventory 2. In our scenario these inventories contain all the instances in the ontologies.

```
   Input: Similarity belief matrixes S_{n×m} = {S_1, .., S_k}
   Output: Mapping candidates
 1 for i=1 to n do
 2 |   BeliefVectors BeliefVectors ← GetBeliefVectors(S[i, 1 − m]) ;
 3 |   Concepts ← GetBestBeliefs(BeliefVectors BeliefVectors) ;
 4 |   Scenario ← CreateScenario(Concepts) ;
 5 |   for j=1 to size(Concepts) do
 6 |   |   Scenario ← AddEvidences (Concepts) ;
 7 |   end
 8 |   if Evidences are contradictory then
 9 |   |   for count=1 to numberOf(Experts) do
10 |   |   |   Voters ← CreateVoters(10) ;
11 |   |   |   TrustValues ← VoteTrustMembership(Evidences) ;
12 |   |   |   ProbabilityDistribution ←
   |   |   |   CalculateTrustProbability(TrustValues) ;
13 |   |   |   Evidences ← SelectTrustedEvidences(ProbabilityDistribution)
   |   |   |   ;
14 |   |   end
15 |   end
16 |   Scenario ← CombineBeliefs(Evidences) ;
17 |   MappingList ← GetMappings(Scenario) ;
18 end
```

Algorithm 1: Creating mapping for component overlap detection

On the other side the configuration selection based on CSP solution is not an iterative process, however solving the problem itself can pose challenges in terms of computational complexity. The algorithm is described on Algorithm 2. The inputs of the algorithm are the inventory items in different sources, the mapping file and the requested items from the user if any.

The first step is to create a sample configuration using information from the requested items that has been specified by the user e.g. user wants Intel processors only. The sample configuration will contain items from both inventory 1 and 2 therefore it may contain similar items. These similar items are eliminated from the configuration (line 2) using the mapping file that was generated by the ontology mapping algorithm. The next step is to create the CSP model that we

wish to solve using the sample configuration (line 3). After we iterate through on each item in the configuration and transform it to a CSP variable together with its possible domain (line 4-6). Then we assign the constraints for the problem (line 7) and solve the CSP problem using our established model(line 8). Finally we can read out the suggested configuration from the model, which will be used as a quote to send back to the user.

Input: $Inventory_n, Inventory_m, Mappingfile, Requesteditems$
Output: Suggested configuration
1 SampleConfiguration ← `GetSampleConfiguration` (InventoryItems, RequestedItems) ;
2 SampleConfiguration ← `EliminateSimilarNodes` (Mappings) ;
3 CPMModel ← `CreateCPMModel` (SampleConfiguration) ;
4 **for** *i=1* **to** *NumOfComponents* **do**
5 | CPMVariable$_i$ = `CreateVariables` (`GetComponentDomain` (SuggestedConfiguration $_i$)) ;
6 **end**
7 Constraints ← `CreateConstraints` () ;
8 CPMModel ← `SolveCPMProblem` () ;
9 SuggestedConfiguration ← `GetSuggestedConfiguration` (CPMModel) ;

Algorithm 2: Resolving the configuration problem

6 Evaluation

In order to evaluate our approach we have carried out experiments (process is depicted on Figure 7) using two ontologies that were created from two on line PC (Personal Computers) store. Both ontologies contain categories and instances of items that are sold in the on-line shop. The main objective of our experiment was to evaluate how accurate our knowledge integration approach is. During the experiments we have generated 100 random configurations that simulates a customer choice and we have evaluated the correctness of the configurations after the two ontologies were mapped into each other.

The main idea of our experiment was to show the integration of our sample ontologies and then to use the integrated model for solving a computers configuration problem. The evaluation was performed in two phases. In the first phase we integrated two knowledge models (i.e. ontologies) from two online PC shops. The evaluation comprises to perform mapping between classes and properties of the two ontologies. In this task we had used our DSSim system [26] which is a mapping system based on Dempster-Shafer Theory described in detail in [25] [26] [27]. Although, the ontologies used in the experiment (presented in this chapter) are rather small our DSSim is equipped to deal with large ontologies [28]. In fact, DSSim has been tested with very large ontologies from the Ontology Alignment Evaluation Initiative (OAEI).

The second part of the experiment is the solution of the CSP problem using the mappings generated in the first phase. To illustrate our solution we started

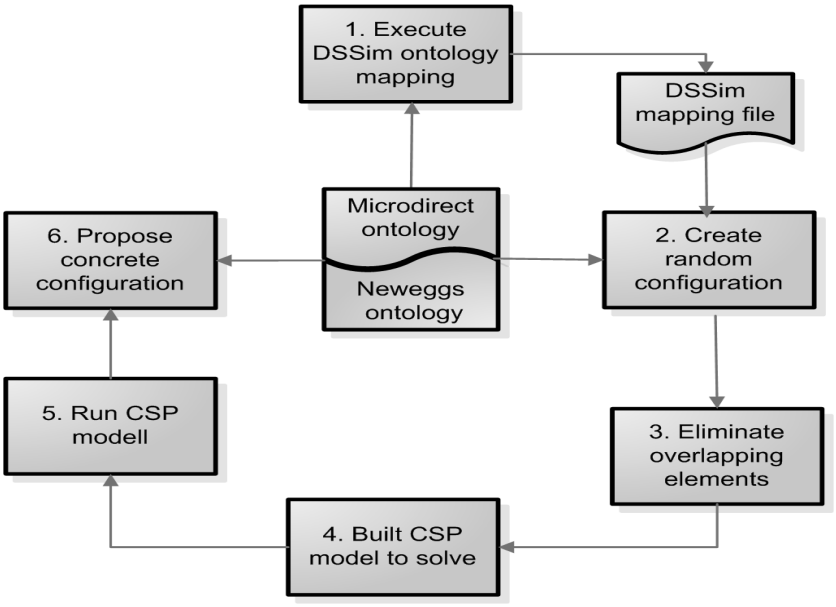

Fig. 7. Experimental process

by selecting just one PC configuration (basic configuration). Once the solution to the basic configuration of computers was obtained. Then, we carried out the experiments using the two remaining configurations namely medium and expensive configuration which were solved in a similar fashion. Therefore, in a first instance, we focussed our attention to the constraints associated to the basic configuration. These constraints are $C1...C10$ defined in section 3.2. Our solution used a constraint solver choco which is widely used in the CSP community. The notion basic, medium and expensive configurations have been represented with the number of components assuming that the more expensive a configuration is the more components the configuration will contain. In our experiment the basic configuration has 30, the medium has 50 and the expensive has 70 components.

We have carried out experiments based on the computer configuration problem described in the section 3.2. In order to make it as close to real situation as possible we have created 2 ontologies based on two online computer shops that sell a wide variety of PC Components and Accessories. One shop is the Micro Direct Ltd [5] from the UK and the second is Newegg [6] from the US. For the experiments we have created ontologies that contain only partial component list from both sites. The number of classes, properties and instances included in the ontologies are described on Table 6.

[5] http://www.microdirect.co.uk
[6] http://www.newegg.com/

Table 6. Example ontology complexities

	Microdirect.co.uk	Newegg.com
Classes	102	121
Properties	47	46
Individuals	197	242
Subclass axioms	96	118
Equivalent classes axioms	19	5

6.1 Mapping quality

The first step of or experiment is to create a mapping file using DSSim in order to detect overlapping elements from the two ontologies. The idea behind our scenario and experiments is to integrate two data sources through ontology mapping. In practice this means that our solution should make it possible to create configurations from two different shops without physically integrating the databases. The mapping file generated by our algorithm contains 93 mappings. These mappings range from the very obvious to hidden correspondences between concepts and properties e.g. Memory - Memory, ATi_Graphics_Card Video_ card. In addition we have created manually a mapping file between the ontologies in order to compare with the one that is generated by the system. This evaluation was measured with recall and precision, which are useful measures that have a fixed range and meaningful from the mapping point of view.

There are two typical measures for assessing the performance:

Definition 8 *Precision: A measure of the usefulness of a hit list, where hit list is an ordered list of hits in decreasing order of relevance to the query*

Definition 9 *Recall: A measure of the completeness of the hit list and shows how well the engine performs in finding relevant entities*

Recall is 100% when every relevant entity is retrieved. However it is possible to achieve 100% by simply returning every entity in the collection for every query. Therefore, recall by itself is not a good measure of the quality of a search engine. Precision is a measure of how well the engine performs in not returning non relevant documents. Precision is 100% when every entity returned to the user is relevant to the query. There is no easy way to achieve 100% precision other than in the trivial case where no document is ever returned for any query. Both precision and recall has a fixed range: 0.0 to 1.0 (or 0% to 100%). A good mapping algorithm must have a high recall to be acceptable for most applications. The most important factor in building better mapping algorithms is to increase precision without worsening the recall.

Before we present our evaluation let us discuss what improvements one can expect considering the mapping precision or recall. Most people would expect that if the results can be doubled i.e. increased by 100% then this is a remarkable achievement. This might be the case for anything but ontology mapping.

In reality researchers are trying to push the limits of the existing matching algorithms and anything between 10% and 30% is considered a good improvement. The objective is always to make improvement in preferably both in precision and recall.

Table 7. Mapping quality

	Value
Precision	0.66
Recall	1.0

Based on the result (depicted on Table 7) we can conclude that the recall rate is 100% therefore all the possible mappings have been found by the system. However the precision is 66 %, which indicates that some additional mapping were found and they are incorrect. The precision rate is high and indeed the manual mapping has resulted in the mapping file that contain only the equivalence relationships e.g. CPU - CPU between items. Our algorithm also identified not equivalence relations e.g. Motherboards - Server_Motherboard and this decreases the precision of the system.

6.2 Configuration quality

In the second experiment we create random configurations using components from both shops i.e. ontologies. For example we take the memory from Microdirect and select the Monitor from Neweggs. The number of components can range between 30 and 70 depending on the configuration type.

In step three using the mapping file created in step 1 we eliminate the overlapping components from the configuration. For example if Video_card was selected from ontology 1 and Graphics_card was also selected to the configuration we leave only one of them in the configuration.

During step four we take the available prices for each component in the configuration. In practice we take all instances of each component and add them as variables for the CSP problem. For example for the Video card we take Sapphire_Radeon_HD_5850_1GB or XFX_ATI_RADEON_4650. All these variables will feed into our CSP solver engine as textual variables.

In step 5 we run the CSP solver in order to get what are the process we can spend on each component in order to produce the suggested configuration. Given the fact that there is no guarantee that the CSP problem can be resolved in a timely manner we put a 10 second constraint on the choco solver in order to limit the available time for each experiment. In case the solver cannot find an optimal solution the random configuration will be returned.

In Step 6 we select the concrete components that fit into our maximum amount we can spend on each component.

The process from step 2 to 6 is repeated 100 times in order to obtain reasonable amount of data that can be analysed. We are interested in measuring how well our proposed approach can perform in order to integrate knowledge from different sources. We measure how often overlapping elements are removed from random configurations and how often overlapping items have to be evaluated from the random configurations.

The experimental results are depicted on 8.

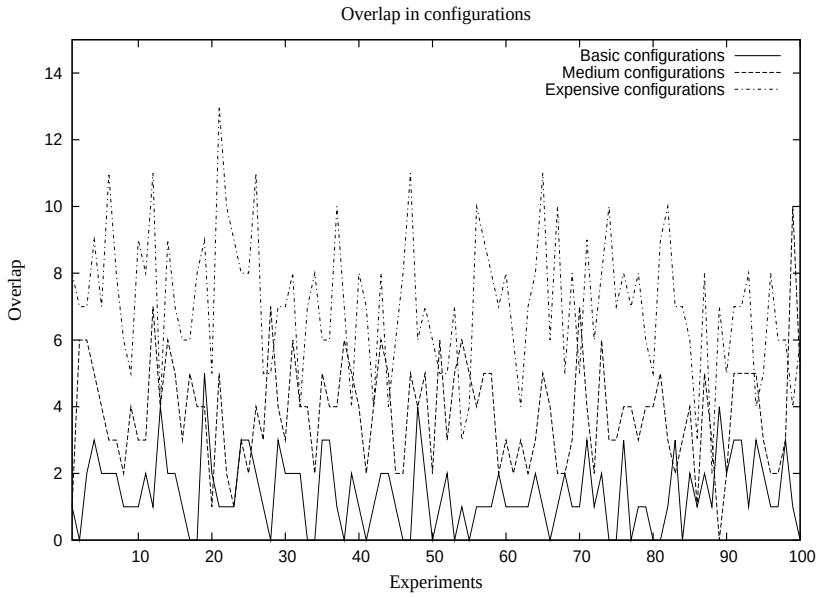

Fig. 8. Overlapping components

Table 8. Ovelapping component statistics

	Min overlap	Max overlap	Average overlap	No overlap
Basic configuration	0	5	1.45	21
Medium configuration	0	10	3.83	1
Expensive configuration	2	13	7	0

As depicted on Figure 8 and Table 8 the number of overlapping elements per configuration varies from 0-13. According to the experiments 79 % of the basic configuration, 99 % of the medium configuration 100 % for the expensive configuration represents cases where the overlapping elements have to be removed.

In practice it means the knowledge integration occurs between 79-100 % percent of the cases. This is remarkable and in operational systems where users are involved this represents a considerable percentage. Based on our experiments we can establish that knowledge integration can improve the PC configuration precision in the majority of the cases.

Our experiments have showed that the result of constraint satisfaction problem for the PC configuration improves if the number of components for the configuration increases. This can be explained with the fact that with the more complex a configuration is the more overlapping in the CSP graph can occur. This is encouraging as our main objective is to establish a solution for the knowledge integration problem.

7 Conclusions

This chapter presented an approach to knowledge integration of several knowledge models. These knowledge models were created by different stakeholders. As a case of study to demonstrate our approach we introduced a restricted version of the computer configuration problem. Our case of study was modeled as a Constraint Satisfaction Problem and the constraints graphs were produced. The detection of overlapping pieces of knowledge and its solution was performed by means of DSSim, a agent-based system which uses similarity algorithms coupled with a fuzzy voting model.

The experiment shown was performed using two knowledge models and it was divided in two phases. The first phase was detection of overlapping knowledge and correction using our DSSim system. The second phase is the Constraint Satisfaction Problem using choco (the constraint solver). Our preliminary findings are encouraging and they are the baseline for assessing the usefulness of our Knowledge Integration. Of course, more work needs to be done in order to fulfill our expectations of a generic framework for Knowledge Integration. Future work comprise to carry out experiments using more knowledge models.

We have established a set of initial experiments and measures that combines ontology mapping and constraint satisfaction in a real word scenario. Our proposed experimental context for knowledge integration is the logical federation of two on-line PC stores, without physically creating a unified database. The federation is carried out only the overlapping elements of the two different data sources in order to being able to eliminate the number of equivalent components for the proposed configuration. Our ontologies used during the experiment contain only a fraction of the information that can be extracted from the two on line stores. Nevertheless our results are encouraging since even these relatively small ontologies produce 79-100 % of overlaps in the configurations. The more elements we include in our ontologies the higher overlapping components will emerge in these configurations. Therefore based on our current experiments we can conclude that the knowledge integration can occur in the majority of the cases and such approach can improve the overall situation of the system. However in the future we intend to investigate further what influences the number of

overlapping elements that occur in random configurations. In terms of constraint satisfaction our experiments have showed that only the expensive configuration performs well as the medium and basic contains far too much configuration that do not match the users criteria. One explanation is the limited number of instances in the two ontologies. We expect that the more instances we will include into our ontologies i.e. more PC components the better our constraint can be met for the basic and medium configuration. In general our experiments have showed that our approach is promising however it requires more experiments with larger ontologies in order to further assess the strengths and weaknesses of our approach.

References

1. Alexander, M., Boris, M., Nuno, S., Raphael, V.: Mafra - a mapping framework for distributed ontologies. In: EKAW '02: Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web. pp. 235–250. Springer-Verlag, London, UK (2002)
2. AnHai, D., Jayant, M., Pedro, D., Alon, H.: Learning to map between ontologies on the semantic web. In: WWW '02: Proceedings of the 11th international conference on World Wide Web. pp. 662–673. ACM, New York, NY, USA (2002)
3. AnHai, D., Pedro, D., Alon, H.: Reconciling schemas of disparate data sources: a machine-learning approach. SIGMOD Rec. 30(2), 509–520 (2001)
4. Ardissono, L., Felfernig, A., Felfernig, E., Friedrich, G., Goy, A., Jannach, D., Petrone, G., Schäfer, R., Zanker, M.: A framework for the development of personalized, distributed web-based configuration systems. AI Magazine 24, 93–108 (2003)
5. Baldwin, J.F.: Mass assignment fundamentals for computing with words. In: IJCAI '97: Selected and Invited Papers from the Workshop on Fuzzy Logic in Artificial Intelligence. pp. 22–44. Springer-Verlag, London, UK (1999)
6. Batini, C., Lenzerini, M., Navathe, S.B.: A comparative analysis of methodologies for database schema integration. ACM Computing Surveys 18(4), 323–364 (1986)
7. Bayardo, Jr., R.J., Bohrer, W., Brice, R., Cichocki, A., Fowler, J., Helal, A., Kashyap, V., Ksiezyk, T., Martin, G., Nodine, M., Rashid, M., Rusinkiewicz, M., Shea, R., Unnikrishnan, C., Unruh, A., Woelk, D.: Infosleuth: agent-based semantic integration of information in open and dynamic environments. In: SIGMOD '97: Proceedings of the 1997 ACM SIGMOD international conference on Management of data. pp. 195–206. ACM, New York, NY, USA (1997)
8. Beneventano, D., Bergamaschi, S., Guerra, F., Vincini, M.: The momis approach to information integration. In: ICEIS. pp. 194–198 (2001)
9. Doan, A., Domingos, P., Halevy, A.: Reconciling schemas of disparate data sources: A machine-learning approach. In: In SIGMOD Conference. pp. 509–520 (2001)
10. Fauconnier, G.: Mental Spaces: Aspects of Meaning Construction in Natural Language. MIT Press, Cambridge, MA (1985)
11. Fridman, N.N., Mark, M.: Prompt: Algorithm and tool for automated ontology merging and alignment. In: Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence. pp. 450–455. AAAI Press / The MIT Press (2000)

12. Garcia-Molina, H., Papakonstantinou, Y., Quass, D., Sagiv, Y., Ullman, J., Vassalos, V., Widom, J.: The tsimmis approach to mediation: Data models and languages. Journal of Intelligent Information Systems 8, 117–132 (1997)
13. Gerd, S., Alexander, M.: Fca-merge: bottom-up merging of ontologies. In: IJCAI'01: Proceedings of the 17th international joint conference on Artificial intelligence. pp. 225–230. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2001)
14. Ghidini, C., Giunchiglia, F.: Local models semantics, or contextual reasoning=locality+compatibility. Artificial Intelligence 127(2), 221 – 259 (2001)
15. Grant, R.M.: Toward a knowledge-based theory of the firm. Strategic Management Journal 17, 109–122 (1996)
16. Halevy, A., Rajaraman, A., Ordille, J.: Data integration: The teenage years. In: VLDB. pp. 9–16 (2006)
17. Hamdi, F., Niraula, B.S.N.B., Reynaud, C.: Taxomap in the oaei 2009 alignment contest. In: Proceedings of the 4th International Workshop on Ontology Matching (OM-2009). CEUR Workshop Proceedings, vol. 551, pp. 230–237 (2009)
18. Hung, H.F., Kao, H.P., Chu, Y.Y.: An empirical study on knowledge integration, technology innovation and experimental practice. Expert Systems with Applications 35(1-2), 177–186 (2008)
19. Jean-Mary, Y.R., Shironoshita, E.P., Kabuka, M.R.: Asmov: Results for oaei 2009. In: Proceedings of the 4th International Workshop on Ontology Matching (OM-2009). CEUR Workshop Proceedings, vol. 551, pp. 152–159 (2009)
20. Johnson-Laird, P.: Mental Models. Harvard University Press (1983)
21. Levy, A.Y.: The information manifold approach to data integration. IEEE Intelligent Systems 13, 12–16 (1998)
22. Matteo, B., Paolo, B., Paolo, T.: Enabling distributed knowledge management: Managerial and technological implications. Tech. rep., Ingegneria e Scienza dell'Informazione, University of Trento (2002)
23. Murray, K.S.: Learning as Knowledge Integration. Ph.D. thesis, The university of Texas, Austin (1995)
24. Murray, K.S.: Ki: A tool for knowledge integration. In: AAAI/IAAI, Vol. 1. pp. 835–842 (1996)
25. Nagy, M., Vargas-Vera, M., Motta, E.: Multi agent ontology mapping framework in the aqua question answering system. In: International Mexican Conference on Artificial Intelligence (MICAI-2005). pp. 70–79 (2005)
26. Nagy, M., Vargas-Vera, M., Motta, E.: Multi-agent ontology mapping with uncertainty on the semantic web. In: Proceedings of the 3rd IEEE International Conference on Intelligent Computer Communication and Processing. pp. 49–56 (2007)
27. Nagy, M., Vargas-Vera, M., Motta, E.: Managing conflicting beliefs with fuzzy trust on the semantic web. In: The 7th Mexican International Conference on Artificial Intelligence (MICAI 2008). pp. 827–837 (2008)
28. Nagy, M., Vargas-Vera, M., Stolarski, P.: Dssim results for oaei 2008. In: Proceedings of the 3rd International Workshop on Ontology Matching. pp. 147–159 (2008)
29. Rapanotti, L., Barroca, L., Vargas-Vera, M., Minocha, S.: deepjthink: a second life campus for part-time research students at a distance. Tech. rep., The Open University (2009)
30. Sanjay, M., Felix, F.: Towards a generic model of configuraton tasks. In: IJCAI'89: Proceedings of the 11th international joint conference on Artificial intelligence. pp. 1395–1401. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1989)

31. Seddiqui, M.H., Aono, M.: Anchor-flood: Results for oaei 2009. In: Proceedings of the 4th International Workshop on Ontology Matching (OM-2009). CEUR Workshop Proceedings, vol. 551, pp. 127–134 (2009)
32. Shafer, G.: A Mathematical Theory of Evidence. Princeton University Press (1976)
33. Wang, P., Xu, B.: Lily: Ontology alignment results for oaei 2009. In: Proceedings of the 4th International Workshop on Ontology Matching (OM-2009). CEUR Workshop Proceedings, vol. 551, pp. 186–192 (2009)
34. Zhang, X., Zhong, Q., Shi, F., Li, J., Tang, J.: Rimom results for oaei 2009. In: Proceedings of the 4th International Workshop on Ontology Matching (OM-2009). CEUR Workshop Proceedings, vol. 551, pp. 208–215 (2009)
35. Zygmunt, A., Koźlak, J., Siwik, L.: Agent-based environment for knowledge integration. In: ICCS 2009: Proceedings of the 9th International Conference on Computational Science. pp. 885–894. Springer-Verlag, Berlin, Heidelberg (2009)