

International Journal of Cooperative Information Systems  
© World Scientific Publishing Company

## A UNIFIED APPROACH TO WORKFLOW BASED ON ONTOLOGIES

WIKUS COETSER AND JUDITH BISHOP\*

*Polelo Research Group, Department of Computer Science,  
University of Pretoria 0002 South Africa*

Received (30 November 2005)

Revised (Day Month Year)

This article considers how to incorporate ontologies into workflow in an effort to improve computer support for collaborative work and improve system interoperability. A connection is drawn between the production process of an artefact in a community of collaborators and an agent based system. In order to make this connection more concrete, knowledge representation formalisms are re-cast into workflow description tools. Then current mark-up languages are considered for standardising workflow descriptions, enabling high levels of interoperability between workflow applications, creating a union of agent and human workflow constructs.

*Keywords:* workflow; artefact; agents; ontology; OWL; RDF; computer supported collaborative work; CSCW; interoperability.

### 1. Introduction

In this article the question is addressed of how to integrate human and agent workflow processes. The definition of an artefact is used as a starting point for identifying the requirements of a workflow formalism that would allow this integration to take place. In abstracting the workflow process from an agent based system, new opportunities for interoperability is introduced. The idea is put forth that ontologies form the common ground for interoperability. In order to clarify what an ontology is, and how it relates to the user and how it relates to agents, these concepts are more closely considered in this introductory section.

#### 1.1. *Ontology*

An ontology, in the philosophical sense, is a systematic account of existence<sup>34</sup>, something by which we understand the world. In the context of information systems it can be seen as a domain specification by which data can be interpreted. What constitutes an ontology is open to broad interpretation, as illustrated by Smith<sup>32</sup>

\*corresponding author: jbishop@cs.up.ac.za

2 *Wikus Coetser, Judith Bishop*

where the meaning of the word *ontology* is interpreted as anything from a catalogue to a set of logical constraints with automated reasoning. It is important to note that this definition of *ontology* focuses on data and its interpretation (for example in an information retrieval or cataloguing system). It is also possible to define an ontology as a form of computation, an information filter that can be used to convert possibly meaningless information into an understandable or usable form. Following this line of reasoning, one could cite Devedzić<sup>7</sup> where parallels are pointed out between software engineering and ontologies, with some of the examples being object oriented design, programming languages and compilers, where the definition and specification of a system can be seen as a model for understanding the system.

### 1.2. *Artefacts in context*

Artefacts can be understood in the context of activity theory<sup>4</sup>. Activities form a context in which individual actions can be understood. An activity, such as writing a book, is executed in an environment that is defined by a community, an individual from the community and an object, or in the context of this article, an *artefact*. The community could be a community of authors who have defined a division of labour by, for example, specifying which author produces certain chapters of a book. This division of labour governs the relationship between the artefact and the community. The relationship between the artefact and the individual exists in the form of actions, which are carried out in order to produce the artefact. The relationship between the individual and the community is formed by rules made by the community. In the context of writing a book, the division of labour could require each author to write a specific chapter of the book, and the rules of the community could state that each chapter must at least have an abstract and one image. Seen as a system, the output of the community (being the artefact) can be made of sub-systems. This recursive aspects of what can be defined as an artefact complicate the definition of an artefact, in the sense that it should be possible to define a feedback loop between the artefact and the community and individual that produced it. It is possible to see a book as an artefact consisting of other artefacts, for example chapters, and images, which themselves can also consist of artefacts. The content of the book could change the perceptions and rules of the community. The feedback loop between the artefact and the entities that produced it opens the possibility of a self-aware (based on it's own ontology) and self-modifying system.

### 1.3. *Agents: an overview*

Agents have been characterised in Jennings<sup>16</sup> as autonomous entities that are capable of solving a problem, or parts of a problem, possibly in conjunction with other agents. Agents are flexible in their problem solving and are embedded in an environment that makes it possible for them to function. This implies that agents do not necessarily have well defined interfaces with the environment, but are tolerant to changing conditions and can change their internal state in order to solve the problem

at hand. In order to design a system that has some or all of these characterisations, two problems are faced: semantics and syntax, both of which are connected to the intentions represented by the agent<sup>33</sup>. The syntax problem takes the form of making agents interoperable, and can exist purely as a syntactic mismatch in information exchange. The semantic problem involves system intentions (i.e. what problem are being solved, how an agent aims to solve the problem and how the agent understands the problem), and can be solved using modal logic<sup>33</sup>. One may ask what it is that the logic will represent: this must be the agent ontology, how it views the environment and the problem it must solve. Agent goals and intentions must be described in a problem specific context relevant to the agent. This description is useful in a context where agents are used as problem solving components in a agent based system, as interactions between the agents (subsystems) can be dynamically formed<sup>a</sup> based on the problem at hand and the agent intentions and goals.

#### 1.4. *The connection between agents and artefacts*

The connection between the concept of an *agent* and an *artefact* may not at first be obvious, because agents are autonomous entities capable of computation, where artefacts are broadly defined by a history of interactions between an object, an individual and a community. This is where the first point of overlap exists between agents and artefacts: they both operate in a community. For some problems, agents need to form relationships inside a community and form a division of labour. The community rules may take the form of agent intentions, goals and domains that specify what can and cannot be done by the agents. Finally, the product of agents working in this community is an artefact, by the definition of *artefact* presented in section 1.2. The second point is partially motivated by the first: in declaring their interrelationships and constructing the community of agents, the agents and the artefacts have something in common: both use an ontology in terms of which they can be understood. The ontology of the agents defines the interoperations of the agents in an *ad hoc* manner that can adapt to a changing environment and the existence of *ad hoc*-crocies, allowing for great flexibility in designing a software system. The artefacts need to be marked up in a mark-up language which needs its own frame of reference (i.e. an ontology) in which it could make sense to external entities, possibly agents. Therefore agents and artefacts can be interrelated through a common ontology.

## 2. Workflow issues

### 2.1. *The community: A broader framework for interaction*

In section 1, the concept of an *artefact* was discussed in the framework of the theory of activities. This framework was useful as a starting point for creating a unified

<sup>a</sup>An example scenario exists in the NOMAD<sup>28 26</sup> system, which must support casually connected communities in a distributed environment.

4 *Wikus Coetser, Judith Bishop*

workflow system, but lacks details on how the creation of an artefact fits into a broader organisational framework. One can argue that the community represents the organisation (for example a commercial entity such as a company), but the concept of *community* is too vague to codify in a logic formalism. In order to better clarify collaborative concepts, computer supported collaborative work (CSCW) is considered. Groupware, which is a form of CSCW, is defined by Ellis<sup>9</sup> as

computer-based systems that support groups of people engaged in a common task (or goal) and that provide an interface to a shared environment.

Ellis<sup>9</sup> also identifies three key areas in CSCW: collaboration, communication and coordination. These three areas combine to create a shared environment that combats user isolation and supports a common task. Groupware software usually has messaging, decision support, computer conferencing, intelligent agent and coordination support<sup>9</sup>.

#### 2.1.1. *Messaging*

A list of issues pertinent to messaging in CSCW has been identified by Chesnais<sup>21</sup>. Some of these issues are:

- *Group awareness.* Group awareness is important to a unified workflow approach because when different artefacts or sub-artefacts are received from different users, contradictory artefact mark-up can be generated based on the ontologies used by the users. This situation can be avoided by the use of a common community ontology in artefact mark-up, which would also allow for community members to search for artefacts on each other's computers based on the community ontology. This searching (retrieving) and uploading behaviour can serve as a two-directional communication between community members, resulting in a better group awareness of the task at hand.
- *Private, semi-public and public messaging.* Privacy is related to confidentiality by the fact that confidentiality implies keeping private information private. Confidentiality is one of the basic computer security services, apart from integrity and authentication, and suffers from problems relating to the clear demarcation of privileges in a collaborative environment, as stated in Ahmed<sup>19</sup>. In a collaborative environment it is not always clear who is allowed to see an artefact and who is not, as this influences the group awareness of the task at hand. In order to combat this, an ontology can enforce privacy based on the division of labour and the community rules influencing the development of an artefact: if a user is not supposed to work on an artefact, the artefact must be invisible to the user.

## 2.2. Collaboration and coordination models

In order to unify human and agent workflow practices, the elements of workflow models are now considered.

### 2.2.1. Defining workflow models

In Lei<sup>24</sup>, definitions of models and meta-models in the context of workflow systems are given, and are related to organisation models, process models and so forth. A workflow model is defined as a entity that:

combines a process model and an organisational model<sup>24</sup>

On the other hand, a workflow meta-model is defined as:

a representational language in which to express workflow models<sup>24</sup>

Based on these definitions, it should be clear that, in terms of creating a unified workflow approach based on ontology, a workflow meta-model is desired rather than a workflow model. In combining a process model, which is a model of a specific process such as shipbuilding or writing a book, and an organisation model, which is a model of organisation (such as a company or community of authors) an application specific workflow model results. The definition of a workflow meta-model also shows consistency with the concept of an ontology, as both ontology and workflow meta-model are used to describe something else. One can also attempt to make an ontology of ontologies, or a workflow meta-model of workflow meta-models. In order create a unified human-agent ontology, some common workflow model elements will now be considered.

### 2.2.2. Identifying workflow model elements

The following workflow model<sup>b</sup> elements have been selected from Georgakopoulos<sup>23</sup> and Van der Aalst<sup>29</sup>:

- *Workflow*. Workflow is defined as a set of tasks that have been ordered. The concept of an ordering of tasks suggests that the underlying formalism used to express workflow must at least be capable of representing a partial ordering.
- *Tasks*. An ordering of human actions or agent actions. Drawing a parallel with systems, tasks can be seen as being larger units composed of sub-workflows. This suggests that the representation formalism for the unifying ontology must have recursive capabilities.

<sup>b</sup>The phrase *workflow model* is used in the rest of the document, even though the elements identified in this section are meta model components, rather than model components.

- *Manipulated objects.* Manipulated objects are artefacts that are produced by the community. Artefacts represent a possible connection point with the privacy issues identified below. The broad definition of an artefact raises the following questions: Can the workflow itself be considered an artefact?
- *Routing information and dynamic information.* Routing information specifies the order in which tasks are executed. Routing information requires that it must be possible, based on the representational formalism for the system ontology, to form a total ordering of tasks.

Sometimes a distinction is drawn between a workflow model and the instances of a model. In terms of an ontology-based representation of workflow, an actual workflow described through the ontology can be seen as an instance of the ontology.

### 2.3. Privacy issues and collaborative security models

Privacy has been identified as an aspect of messaging, which is related to workflow, in section 2.1.1. Privacy in CSCW systems are problematic as they often give rise to situations in which one wishes to trust and distrust the same entity in a system. Some forms of collaborative work could for example demand information sharing with the community, but at the same time one may also wish to keep artefacts currently under construction private. In order to come to grips with this situation, privacy must be considered in more detail. Palen<sup>25</sup> defines privacy in terms of a framework, which consists of three boundaries. These boundaries are the *temporal boundary*, the *disclosure boundary* and the *identity boundary*. The boundaries form lines of tension between opposite aims. In order to make the processes related to workflow less invasive in terms of privacy, this framework must be associated with the definition of an artefact used thus far. Privacy must be inserted into the boundary between the user and the community, as this is what distinguishes the individual from the collective. Each of these elements from the privacy framework<sup>25</sup> are now considered in turn, and related to the concepts of workflow and ontology.

#### 2.3.1. Disclosure and privacy

The disclosure boundary exists between publicity and privacy<sup>25</sup>: in the context of ontology supported workflow one needs to, at the same time, disclose information to the rest of the community, while keeping some of it private. Information disclosure could form a protective mechanism, it could be used to anticipate requests from the community, or it could be used to deceive or protect the user from potential scenarios inside the community. Furthermore, in order to work on an artefact privately, without corruption from other agents, it may be necessary to selectively disclose information, or keep it private. It is therefore necessary that a workflow supporting ontology support a method of selective disclosure.

### 2.3.2. Identity and privacy

The identity boundary exists between the user and the rest of the community<sup>25</sup>: In a workflow system it could happen that an agent could wish to associate itself with one community, and sometimes with another. An example given in Ahmed<sup>19</sup> shows this by referring to a group of lawyers working on the same document. It could happen that some lawyers work for company A, and some lawyers work for company B, with company A and B having conflicting interests. This forms two conflicting groups, both potentially containing the same lawyer. In designing a ontology supporting workflow, it would be necessary to make the design choice of whether the individual agent can switch loyalties, or whether the agent must stay in the same community. Possible solutions to this problem is considered in section 2.3.4 relating to collaborative security models.

### 2.3.3. The temporal aspects of privacy

Disclosure and protection of private information is not only subject to the present, but also to past and future disclosures of information<sup>25</sup>. This implies that disclosure of private information in a workflow scenario could be subjected to tests referring to the ontology which specifies what must be disclosed, and requires reasoning capabilities for determining what may and may not be disclosed.

### 2.3.4. Security models for collaborative work

Based on the preceding three privacy boundaries, existing security models are now considered for maintaining the boundary between the individual agent and the community during the artefact construction process and workflow related routing of information.

- *The Chinese Wall Model*. The aim of the Chinese Wall Security Model<sup>20</sup> is to control information sharing in environments where conflicts of interest exist. This makes it suitable for applications where the identity security boundary contains a conflict between opposing communities in artefact mark-up. The Chinese Wall Security Model uses a three level system to categorise objects (artefacts). These layers are the objects layer, the company dataset layer and the conflict of interest classes. Artefacts exist in the objects layer. The company dataset layer groups objects from the same company together. Based on the concepts associated with artefact construction in section 1.2, communities would take the place of companies. This is necessary in order to make information about conflicts of interest explicit in order to enforce the model. There is no *natural*<sup>c</sup> method to match the

<sup>c</sup>In order to define conflicts of interests, additional constructs need to be added to the definition of an artefact. Compare this to role based access control in the next point, which fits into the definition of an artefact, and the workflow concepts identified.

conflict of interest problem based on the definition of an artefact. For this reason, the logical structures of the ontology representation formalism will have to be used to create and enforce these classes. In the Chinese Wall Security Model, access is granted if an object being accessed is in the same community group or in a different set of conflicting classes.

- *Role based access control.* Role based access control is described in Ferraiolo<sup>22</sup>, based on the definition of a role. A role is seen as a set of transactions that a user can do in an organisation. Utilising the concept of a community, a hypothetical role based access control mechanism would take the form of a division of labour, in the context of the definition of an artefact. The community can decide on the division of labour, and, based on the artefact annotation utilising the workflow ontology, return only the applicable artefact that an agent may see.

### 3. Possible formalisms for ontology based workflow representation

Based on section 2, it is possible to create a list of requirements for a workflow ontology formalism. Some of these requirements include:

- It must be able to represent a partial ordering of tasks
- Algorithms must exist for creating a total ordering of tasks, essentially building a sequential plan for execution
- The formalism must support recursive ontology based workflow descriptions in order to systematically decompose tasks into subtasks.
- The formalism must support temporal reasoning: as tasks are completed, new total orderings of tasks may need to be created based on a changing environment. This also has security implications, as discussed in section 2.
- The formalism must have an exclusion mechanism for making artefacts internal to certain tasks private, in order to support privacy aspects.

The concepts found in ontology literature can be divided into three layers: the mathematical layer, the standards layer and the application layer. The mathematical layer includes the formal theories and algorithms that are used to provide services to the other layers. The standards layer exists for the purpose of interoperation between ontology technologies and also serves as a guideline for building applications utilising ontological concepts. The application layer consists of programs utilising the mathematical and standards layer.

#### 3.1. *The theoretical foundation*

Accept for the requirements stated above, the following services have been identified<sup>6 8</sup> which should form part of an ontology supporting system:

- (1) *Concept satisfiability.* The aim of concept satisfiability is to prove that there is a set of artefacts that fits into an ontology.

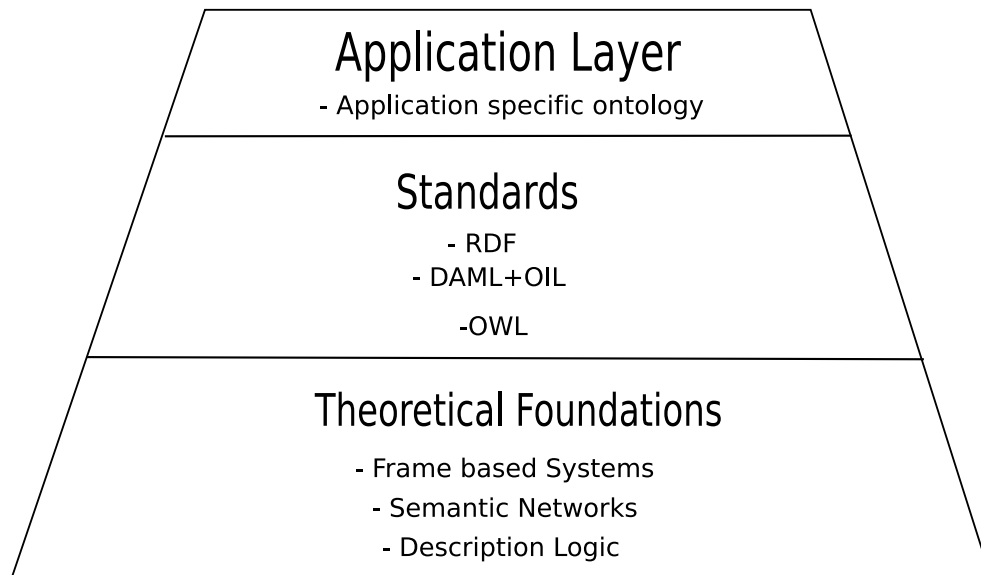


Fig. 1. The three layers of the ontology world

- (2) *Subsumption*. The aim of subsumption is to prove that one concept from an ontology is more general than another concept.
- (3) *Model consistency*. The aim of model consistency checking is to ensure that, given a set of concepts in an ontology, the concepts do not lead to contradictions.
- (4) *Instance checking*. The aim of this service is to find a collection of artefacts that are instances of a set of concepts from an ontology. This is also referred to as the retrieval problem.

Baader<sup>2</sup> also identifies disjointness and equivalence, where the disjointness service tests whether two concepts are disjoint, and the equivalence service tests whether two concepts are equivalent. In an ontology supported workflow system, it would be useful to have an equivalence service, to aid in the cross translation between ontologies used in disjoint communities. The realization service is also identified<sup>2</sup>, which, given an artefact, or a set of sub-artefacts, must find the most specific concept in which those artefacts exists. In using the realization service, the subsumption service can be used in order to test whether one matched concept is more general than another matched concept. Other concepts applicable to the mathematics layer are:

- *Soundness*. In a sound knowledge representation, all conclusions drawn by an inference system in answering queries are correctly drawn, no incorrect results are given<sup>27</sup>.
- *Open and closed worlds assumption*. A formalism making the closed worlds assumption assumes that all information available to the system is complete.

Therefore, the assumption can be made that the negation of a statement is true, if the statement is absent<sup>1</sup>. This is not the case with an open worlds assumption, where it is assumed that the information in the system is incomplete.

- *Completeness*. A complete knowledge representation system is capable of drawing all possible correct conclusions<sup>27</sup>.

Except for the above concepts, the following properties has been defined for a knowledge representation formalism to become accepted<sup>11</sup>:

- *Expressive power*. A knowledge representation formalism must have sufficient expressive power to enable proper definition of domain concepts. In a workflow system, the ontology formalism must be sufficiently expressive to allow broad workflow descriptions.
- *Understandability*. The formalism must be understandable to the implementer describing the domain ontology.
- *Accessibility*. The system must be able use the ontology representation. Some representations are computationally infeasible when a trade-off exists between expressive power and accessibility, as discussed in the description logic section below.

In the following subsections, potential representation formalisms are discussed for workflow ontologies, and then compared based on the above requirements.

### 3.1.1. *Frame based systems*

Frame based systems form a knowledge representation method that shares various properties with object orientation. Concepts are depicted as frames with slots serving to frames as properties serve to objects<sup>11</sup>. The following extensions are defined for the frame-based method:

- *Specialisation and inheritance*. Frames have the ability to inherit taxonomic information from other frames. In this way class-subclass relationships can be established in order to indicate that one concept is a specialised subset of another concept. This shares properties with subsumption in description logic, as one concept name subsumes another concept name if it is a subset of that concept<sup>d</sup>, establishing a super-set subset-type relationship similar to a parent-child relationship used in the frame based ontology representation. This creation of a taxonomy with *is-a* relationships is also presented in Lassila<sup>17</sup>.
- *Production rules*. Users construct knowledge bases more easily when using a declarative style of describing domain ontologies than using a procedural approach<sup>11</sup>. Productions provide a way of expressing ontology concept behaviour in a pattern-response fashion. Productions are also characterised as

<sup>d</sup>Concepts are represented in description logic as sets. See the section on description logic for more information

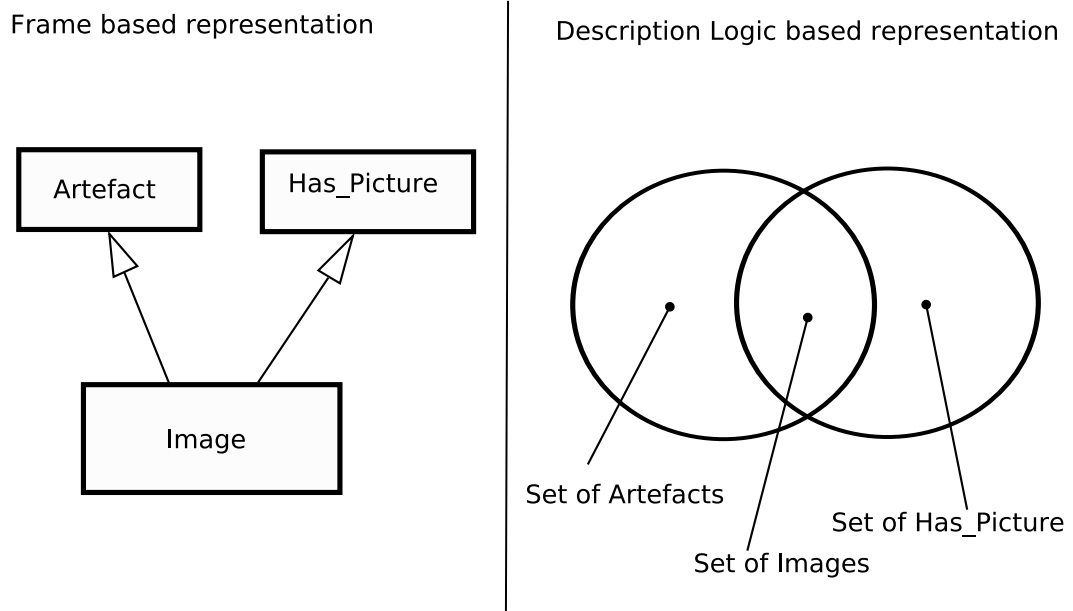


Fig. 2. A frame inheritance vs. description logic subsumption

a way of expressing *rules of thumb* over the concepts of a knowledge base<sup>3</sup>.

- *Attribute descriptions or value restrictions.* Constraints on slots can be expressed as *type information*, *cardinality* and so forth, of the frame slots which help to maintain the knowledge base integrity.
- *Procedural attachments.* Frames can support the message passing behaviour of object orientation through a daemon type service<sup>e</sup> that monitors frame slot values and invokes methods when they change, possibly updating other slot values.

Except for the subsumption type service, frames could also provide a realization type service by checking a given situation against the attribute restrictions defined on frame definitions, and in so doing match a concept from an ontology to an artefact. This concept matching service can, for example, be used to match patient information to concepts in a medical database<sup>12</sup>. This service should be extendable to matching artefacts to ontology concepts in providing input to a workflow process. Fikes<sup>11</sup> points out that constructing a knowledge base using a frame based representation is easier for domain experts to understand than a predicate logic representation. Lassila<sup>17</sup> also supports this view, by pointing out that a frame based ontology representation should make it easy to encode ontology information on the semantic web.

<sup>e</sup>A monitoring service running on a different thread

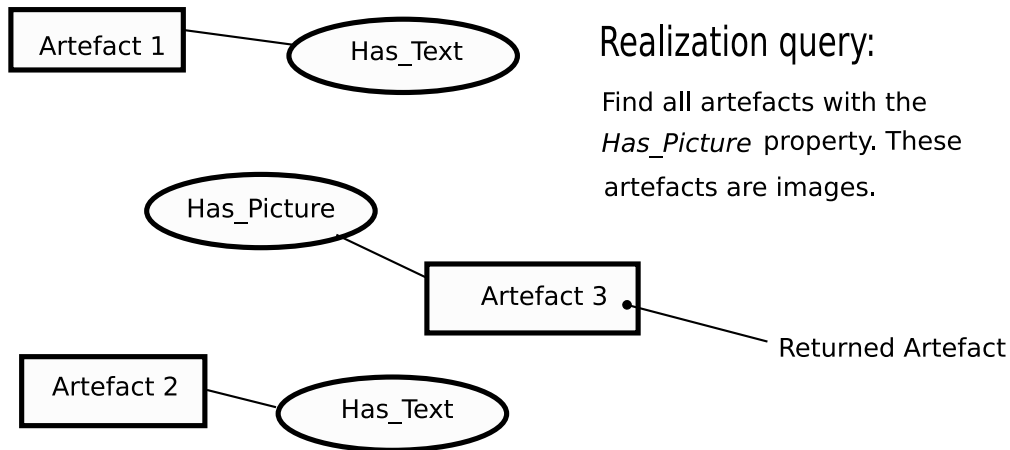


Fig. 3. Frame based realization with value restrictions

### 3.1.2. Semantic networks

Semantic networks represent objects and the relations between objects. When representing a semantic network as a graph, objects are represented as nodes, and the relations between objects are represented as labelled edges<sup>5</sup>. The semantic network representation is attractive to workflow representation due to its simplicity, when compared to the other formalisms.

### 3.1.3. Description logic

Description logic is a knowledge representation formalism that defines its semantics in terms of sets. A concept in description logic is defined as a subset of the problem domain. In an ontology supported workflow system, the problem domain can be described based on the definition of an artefact at the beginning of this article: it must contain the community, its rules and division of labour, and information about the individual members of the community. Based on this description, the set of individuals would, for example, be a subset of all elements in the problem domain. In a hypothetical system based on workflow ontologies, an author name can be added to the problem domain, usually represented by  $\Delta^I$ . The author name then becomes an *individual* of the problem domain, denoted  $a_i \in \Delta^I$  where  $a_i$  represents the added author. The concept *author* can now be defined as all authors that are part of a subset of the problem domain,  $A \subseteq \Delta^I$ . The concept definitions defined over sets are used in combination with operators, also called *constructors*, in order to construct more complex concepts<sup>8</sup>. The constructors construct the various description logic languages by adding more constructors used to define concepts from a basic description logic language containing only a few constructors. Such a

language is  $AL^2$ , which has the following syntax:

$$\begin{array}{l}
 C, D \rightarrow \quad A \\
 \quad \quad \quad | \quad \top \\
 \quad \quad \quad | \quad \perp \\
 \quad \quad \quad | \quad \neg A \\
 \quad \quad \quad | \quad C \sqcap D \\
 \quad \quad \quad | \quad \forall R.C \\
 \quad \quad \quad | \quad \exists R.\top
 \end{array}$$

The semantics of this syntax is stated in terms of sets. The intersection operator  $C^I \sqcap D^I$  has the semantics  $C^I \cap D^I$ , where  $C$  and  $D$  are concepts, which themselves forms subsets of the problem domain. A more expressive description logic language can be created by adding, for example, general concept negation in order to produce the language  $ALC^8$ . This adds  $C \rightarrow \neg C$  to the syntax of  $AL$  with the semantics  $\Delta^I \setminus C^I$  allowing for concept complement<sup>27</sup>.  $AL$  can be used to describe the definition of an artefact in the following manner:

$$artefact \doteq caption \sqcap image$$

meaning that an image with a caption is an artefact.  $ALC$  can be used to define what is not an artefact:

$$author \doteq \neg artefact$$

which would not be possible with  $AL$ . This is because the syntax and semantics of  $AL$  does not allow for general concept negation. The additional expressiveness comes at a price: some description logic variants become computationally unfeasible when extra language constructors are added, as can be seen in the definition of the ontology web language<sup>f</sup>, which takes this into account<sup>14</sup>. Except for language constructors, a description logic system also has a terminology box, or TBox, and an assertion box, or ABox<sup>1</sup>. The TBox stores definitions such as those of artefacts and authors given above that define the domain ontology. The ABox stores assertions based on the TBox definitions. An example of this, using the above TBox, would be<sup>g</sup>:

$$author(a_1); image(horse.pang); caption(\text{“This is a picture of a horse”})$$

<sup>f</sup>Discussed in section 4.

<sup>g</sup>Using non-standard notation

In assigning individuals to concept names, an interpretation is constructed of the ontology described by the description logic system. This interpretation is denoted as  $I = (\Delta^I, \cdot^I)$ , and makes the unique names assumption<sup>30</sup>. The unique names assumption states that the individual values assigned to concept names are unique in the application domain  $\Delta$ . In order to provide the reasoning services required by a ontology-enabled system, reasoning is done with a *tableaux* algorithm<sup>h</sup>. Other methods for reasoning with description logic include<sup>27</sup> rewriting the concept to be reasoned with into an equivalent logic based formalism and using known algorithms to carry out reasoning, structural analysis and tree based automata.

### 3.2. Comparing and evaluating formalisms

In evaluating the frame systems, semantic networks and description logic, the following conclusions are drawn with regard to the requirements stated at the beginning of this section:

- Description logic shows potential for a workflow ontology formalism due to its ability to be extended with new language constructors. Some of the challenges that would be faced with this approach is incorporating temporal reasoning and recursion (i.e. reasoning about workflows in workflows) into a system utilising this formalism.
- The semantic net formalism is attractive because it is easy to implement. Mini-workflows can be embedded in larger workflows at the nodes of the network, and existing graph algorithms can be used to process workflow information. One of the problems that could arise in implementing a workflow system based on the semantic net formalism is that the semantics of nodes and edges has not been formalised to the extent that it has in description logic, which could lead to implementation errors or ambiguities<sup>i</sup>.
- The frame based formalism has the same problems and advantages as the semantic net formalism: it is easy to implement but not as formally well defined as description logic.

None of the above formalisms completely meets the requirements stated at the beginning of section 3. One way to solve this problem may be to invent a special formalism that supports all stated requirements.

## 4. Workflow ontology mark-up

In creating a unified approach to workflow, one would wish workflow systems, and workflows themselves to be integratable and nestable. The problem of interoperability between human and computer, and computer and computer agents does not

<sup>h</sup>Many of the cited references refer to Schmidt-SchauB<sup>31</sup> as the paper that started the tableaux method

<sup>i</sup>Two developers could for example use the same part of the semantic network differently, resulting in inconsistencies in the application.

only take the form of semantics associated with an ontology, but with the syntax of information exchange as well. The semantics and the syntax need to be interoperable in order to enable applications to utilise the layered structure presented in section 3.

#### 4.1. *Mark-up languages for ontology information*

The following standards are available for expressing ontological information:

- *RDF*. RDF<sup>j</sup> is a standard of the W3C<sup>k</sup> for describing online resources. In this regard it is ideally suited for serving as a mark-up language for the artefacts in a workflow system. The W3C RDF primer provides an introduction to RDF<sup>l</sup>. The *subject - predicate - object* triple forms the underlying theoretical model called the RDF graph, which is clearly influenced by the frame based paradigm and the semantic networks paradigms from the field of knowledge representation<sup>m</sup>. The subjects take the roles of frames and the objects take the role of slots. The predicates take the role of edges in a semantic network describing the role between the subject and the object<sup>5</sup>. A parallel between the semantic web and the frame-based representation is drawn in Lassila<sup>17</sup>.
- *DAML+OIL*. DAML+OIL<sup>n</sup> is an extension of RDF Schema, and therefore an RDF vocabulary. This allows DAML+OIL to be build on existing web technologies, which improves its chances of success in actual applications<sup>10</sup>. The aim of DAML+OIL is to serve as an ontology mark-up language, where ontologies are defined in terms of classes. These are not classes in the sense of object oriented programming languages, though there are some parallels through the frame-based paradigm. DAML+OIL is related to frame based knowledge representation through the definition of properties for classes. But the classes in DAML+OIL are better understood as sets of individuals, in the sense of description logic interpretations, which assigns individuals to concepts.
- *OWL*. OWL<sup>o</sup> is based on the DAML+OIL language<sup>p</sup> and also serves as an ontology mark-up language. One of the key differences between OWL and DAML+OIL lies in the consideration of implementation: OWL does not just specify an ontology mark-up language, but also takes into account whether it is possible to implement reasoning services for the ontology language.

<sup>j</sup>Acronym for Resource Description Framework

<sup>k</sup>Acronym for World Wide Web Consortium

<sup>l</sup><http://www.w3.org/TR/rdf-primer/>

<sup>m</sup>See *section 2.3.1, The Theoretical Foundations*.

<sup>n</sup>DAML is an acronym for DARPA Agent Mark-up Language<sup>10</sup>

<sup>o</sup>The ontology web language

<sup>p</sup>OWL language overview, <http://www.w3.org/TR/owl-features/>

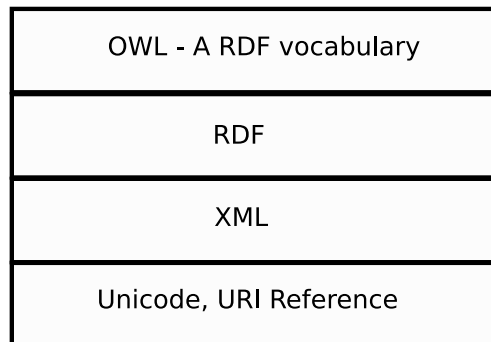


Fig. 4. Layering of semantic web standards

#### 4.2. Interoperability between standards

It is conceivable that a situation could arise where cross-operation between ontology based workflow systems are required. This is dependant on the semantic web architecture: The semantic web standards are layered on top of one another, with each layer defining new concepts based on the previous layers. At the bottom of the architecture are UNICODE and URI references. Based on these concepts, XML and XML Schema are defined. Taking only the standards so far discussed in this article into account, RDF is based on XML, and DAML+OIL and OWL is based on RDF. The layering of the W3C standards solves syntactic problems associated with parsing the mark-up of ontological information, but leads to semantic inconsistencies between the layers. The inconsistencies between OWL and RDF are discussed in Horrocks<sup>13</sup>, and relates to their underlying theoretical models. The languages that form part of current semantic web standards cannot be translated into each other.

#### 5. Constructing a hypothetical system

In constructing a hypothetical ontology based workflow system with agents, the opportunity exists to create a flexible software architecture which itself is based on the workflow ontology. This software architecture differs from what could be considered as a normal information storage and retrieval system for the following reasons:

- (1) The use of agents impacts on the structuring of the system.
- (2) The system makes it's underlying ontology explicit, and connects its execution with this ontology.
- (3) Interoperability can be easily achieved.

The first point is motivated by the argument that agent based systems violate the clear distinction between components found in the Model-View-Controller (MVC) architectural pattern<sup>18</sup>. In the MVC pattern, the user interface is separated from

the underlying application model, and the application logic is further separated from the other two components. Each agent can have its own user interface and model of understanding the workflow system while at all agents are subjected to the same global control mechanism in the form of the ontology based workflow. The *model* and the *control* components are merged in the ontology based workflow model: the data used to route information and the data used for control flow in the system is the same.

The second point is derived from the unified approach to workflow: not only is the system interoperable with the user in the sense that artefacts that gets routed in user workflow uses the same ontology as computer agent interactions, but the system can interact with itself by using the workflow system for internal procedure calls. This can be done by using the *Perceive - Reason - Act* cycle that forms a basic part of agent architectures<sup>15</sup>. In practical terms this would mean that the agents that forms the sub-components of the system could, for example, based on the underlying formalism in the system make assertions about workflow in the agent environment, where the assertions refers to concepts defined in the workflow ontology. In a description logic based system, for example, the ontology would be described in a TBox, and the assertions would be made in a ABox, with both the TBox and the ABox being visible in the agent environment.

The third point is that utilising standards from section 4 and separating the underlying control flow in the system from other system components creates simplified systems interoperation opportunities: When another system is to be plugged into this system, or *vice versa*, the workflow could be loaded from the marked-up ontology representation of the system, and modified to integrate the other system with this system, effectively changing information flow inside the system.

## 6. Conclusion

In order to create a unified approach to workflow based on ontologies, the product of workflow - the artefact - was considered. From the concept of an artefact the concept of a community was derived, which forms a context in which workflow can take place. Collaborative work in a community leads to the identification of workflow elements that must be part of the ontology for work flows. On the more technical side, various knowledge representation formalisms were considered as a foundation for a system embodying the concepts in this article. These formalisms were subject to the requirements of workflow systems, including the capability to represent partial orderings of tasks, temporal reasoning capabilities and the ability to recursively reason about workflows in order to support task decomposition. Utilising the mark-up standards associated with the underlying formalism of ontologies opens the way towards more interoperable and flexible systems.

## 7. Future work

The integration of agent based systems and user labour processes creates various opportunities for further research<sup>4</sup>:

- *Workflow and security.* In terms of security, the workflow ontology can be used to manage security: the underlying formalism can be formulated in such a way that attacks on an organisation utilising an overall ontology automatically leads to contradiction in the underlying formalism, rendering the system useless to the attacker.
- *Flexible organisational structures.* In the globalising economy, an organisation must be flexible enough to adapt quickly to changes in its environment. Traditional ERP systems require redesign to cater for changes, leading to businesses missing opportunities in the market place. With the unified approach presented in this article, only the organisational workflow ontology needs to be modified, with the agents utilising the workflow adapting to the changes.
- *Self-optimising organisational structures.* Taking the previous point one step further, and considering that everything becomes a cost centre, it should be possible to optimise the organisational structure for speed, cost saving, production output or any other factors deemed necessary, as long as the optimisation still satisfies the organisational requirements, based on its ontologies.

## 8. Acknowledgements

We would like to thank the NRF<sup>r</sup> and Microsoft Research<sup>s</sup> for making this research possible with grant money. We would like to thank our colleagues for their helpful comments.

## References

1. F. Baader. Logic-Based Knowledge Representation. *Artificial Intelligence Today: Recent Trends and Developments*, Jun 2003.
2. F. Baader and W. Nutt. Basic Description Logics. In *The Description Logic Handbook: Theory, Implementation and Application*. Cambridge University Press, 2003.
3. W.J. Clancy. The Epistemology of a Rule-Based Expert System—a Framework for Explanantion. *Artificial Intelligence*, 1983.
4. M.M. Cluts. The Evolution of Artifacts in Cooperative Work: Constructing Meaning Through Activity. In *GROUP '03: Proceedings of the 2003 international ACM SIG-GROUP conference on Supporting group work*, pages 144–152, New York, NY, USA, 2003. ACM Press.
5. B. Coppin. *Artificial Intelligence Illuminated*. Jones and Bartlett Publishers, 2004.
6. S. Cranefield, S. Haustein and M. Purvis. Uml-based ontology modelling for software agents. In *Workshop on Ontologies in Agent Systems*, 2001.

<sup>4</sup>Note that this section is mostly speculative, and is not supported by rigorous argumentation as the subject matter falls beyond the scope of this article.

<sup>r</sup><http://www.nrf.ac.za>

<sup>s</sup><http://research.microsoft.com>

7. V. Devedzić. Understanding ontological engineering. *Commun. ACM*, 45(4):136–144, 2002.
8. F.M. Donini, M. Lenzerini, D. Nardi and A. Schaerf. Reasoning in description logics. In *Principles of knowledge representation*, pages 191–236. Center for the Study of Language and Information, Stanford, CA, USA, 1996.
9. C.A. Ellis, S.J. Gibbs and G. Rein. Groupware: some issues and experiences. *Communications of the ACM*, 34(1):39 – 58, Jan 1991.
10. D. Fensel, F. van Harmelen, I. Horrocks, D. L. McGuinness and P. F. Patel-Schneider. OIL: An Ontology Infrastructure for the Semantic Web. *IEEE Intelligent Systems*, pages 38–45, 2001.
11. R. Fikes and T. Kehler. The role of frame-based representation in reasoning. *Communications of the ACM*, 28(9):904–920, Sep 1985.
12. C. Heinlein, K. Kuhn and P. Dadam. Representation of Medical Guidelines Using a Classification-Based System. In *Proceedings of the third international conference on Information and knowledge management*, pages 415 – 422. ACM Press, 1994.
13. I. Horrocks and P.F. Patel-Schneider. Three Theses of Representation in the Semantic Web. In *Proceedings of the 12th international conference on World Wide Web*, pages 39 – 47. ACM Press, 2003.
14. I. Horrocks, P.F. Patel-Schneider and F. van Harmelen. From SHIQ and RDF to OWL: The Making of a Web Ontology Language. *Journal of Web Semantics*, 2003.
15. N.R. Jennings. Agent-Based Computing: Promise and Perils. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 1429 – 1436. Morgan Kaufmann Publishers Inc., 1999.
16. N.R. Jennings. On agent-based software engineering. *Artificial Intelligence*, 117(2):277–296, Mar 2000.
17. O. Lassila and D. McGuinness. The Role of Frame-Based Representation on the Semantic Web. Technical Report Laboratory Technical Report KSL Tech Report Number KSL-01-02, Stanford University, 2001.
18. T.C. Lethbridge and R. Laganière. *Object-Oriented Software Engineering*. McGraw-Hill Education, 2001.
19. T. Ahmed and A. Tripathi. Security policies in distributed csw and workflow systems. 2002.
20. D.F.C. Brewer and M.J. Nash. The chinese wall security policy. In *1989 IEEE Symposium on Security and Privacy*, pages 206 – 214, May 1989.
21. P.R. Chesnais. Canard: A framework for community messaging. In *ISWC '97: Proceedings of the 1st IEEE International Symposium on Wearable Computers*, page 108, Washington, DC, USA, 1997. IEEE Computer Society.
22. D.F. Ferraiolo and R. Kuhn. Role-based access control. In *Proceedings of the 15th NIST-NSA National Computer Security Conference*, pages 554–563, Oct 1992.
23. M. Hornick D. Georgakopoulos and A. Sheth. An overview of workflow management: from process modeling to workflow automation infrastructure. *Distrib. Parallel Databases*, 3(2):119–153, 1995.
24. Y. Lei and M. P. Singh. A comparison of workflow metamodels. In *Proceedings of the ER-97 Workshop on Behavioral Modeling and Design Transformations: Issues and Opportunities in Conceptual Modeling*, Los Angeles, 1997.
25. L. Palen and P. Dourish. Unpacking "privacy" for a networked world. In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 129–136, New York, NY, USA, 2003. ACM Press.
26. J. Rama and J. Biship. Towards a mobile agent framework for nomad using .net. Technical report, Polelo Research Group, University of Pretoria.

20 *Wikus Coetser, Judith Bishop*

27. S. Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, LuFG Theoretical Computer Science, RWTH-Aachen, Germany, 2001.
28. T. Tsoaeli and J. Bishop. Enhancing adaptability of distributed groupware applications. In *Proceedings of SAICSIT*, pages pp. 260–267, September 2005.
29. W.M.P. Van der Aalst. Generic workflow models: How to handle dynamic change and capture management information? In *COOPIS '99: Proceedings of the Fourth IECIS International Conference on Cooperative Information Systems*, page 115, Washington, DC, USA, 1999. IEEE Computer Society.
30. U. Sattler and F. Baader. An Overview of Tableau Algorithms for Description Logics. In *Tableaux 2000*. Kluwer Academic Publishers, May 2000.
31. M. Schmidt-SchauB and G. Smolka. Attributive concept descriptions with complements. *Artif. Intell.*, 48(1):1–26, 1991.
32. B. Smith and C. Welty. Ontology: Towards a New Synthesis. In *FOIS '01: Proceedings of the international conference on Formal Ontology in Information Systems*, pages .3–.9, New York, NY, USA, 2001. ACM Press.
33. M. Wooldridge and N.R. Jennings. Agent theories, architectures, and languages: a survey. In *ECAI-94: Proceedings of the workshop on agent theories, architectures, and languages on Intelligent agents*, pages 1–39, New York, NY, USA, 1995. Springer-Verlag New York, Inc.
34. G.L. Zúñiga. Ontology: its transformation from philosophy to information systems. In *FOIS '01: Proceedings of the international conference on Formal Ontology in Information Systems*, pages 187–197, New York, NY, USA, 2001. ACM Press.