Interoperability through integrating Semantic Web Technology, Web Services, and Workflow Modeling

John Krogstie, Csaba Veres, Guttorm Sindre

Department of Computer and Information Science
Norwegian University of Science and Technology (NTNU)
{krogstie,veres,guttors}@idi.ntnu.no

Abstract. A number of technologies are mentioned under the rubric of "The Semantic Web", but good overviews of these technologies with an eye toward practical applications are scarce. Moreover, much of the early focus in this field has been on the development of representation languages for static conceptual information, while there has been less emphasis on how to make semantic web applications practically useful in the context of knowledge work. To achieve this, a better coupling is needed between ontology, service descriptions and workflow modeling. This paper reviews all the basic technologies involved in this, and outlines what can be achieved by merging them in the context of real world workflow descriptions.

1. Introduction

"The Semantic Web" [1] is seen as the next generation of web systems, providing better information retrieval, better services, and enhanced interoperability between different information systems. The Semantic Web initiative is currently overseen in the semantic web activity of the W3C[1], and includes a number of core technologies. Some core technologies that will be relevant to this overview are XML, RDF, RDF/S, OWL, and Web Services (SOAP, WSDL, UDDI). While these technologies are promising, it can still be argued that alone, they are not sufficient to achieve interoperability in the business domain, allowing for a smooth integration between different information systems within and between organizations. For this to be accomplished, it is not enough to describe ontological metadata about the information and services available – one also needs to know the work context in which the different types of information and services are requested. Hence there is a need to integrate ontologies and service descriptions with models of workflows and business processes. The purpose of this paper is as follows:

1http://www.w3.org/2001/sw/

a) to provide a survey of the relevant technologies (ontology, service models, workflow models). While there is much literature about each of them, there is no one unified survey.

b) To show how these technologies fit together, both in theory (presented as something called “The interoperability pyramid”) and in practice (illustrated by an example of how the technologies could be used).

The rest of this paper is structured as follows: Sections 2-4 survey ontologies, service models, and workflow models, respectively. Section 5 then presents an integrated approach to enterprise and IS development, where interoperability among the various systems (and enterprises) would be a major focus. Finally, section 6 provides some concluding remarks.

2. Ontology

The lower level technologies XML, RDF, and RDF/S are less relevant to us due to their shortcomings for semantic annotation. Also, we assume that they are well known (or otherwise easily investigated by the reader looking at other sources), so we jump directly to the level of ontology languages.

A good starting point for understanding what an ontology entails, is to consider figure 1, adopted from [9], which places a number of knowledge models on a continuum. As you go from the lower left corner to the upper right, the richness of the expressible semantics increases. This is shown on the right side of the arrow with some typical expressions that have some sort of defined semantics for the particular model. The names for the knowledge models are given on the left of the arrow. It is important to note that all of the terms on the left hand side have been called an “ontology” by at least some authors, which is part of the source for confusion about the word.

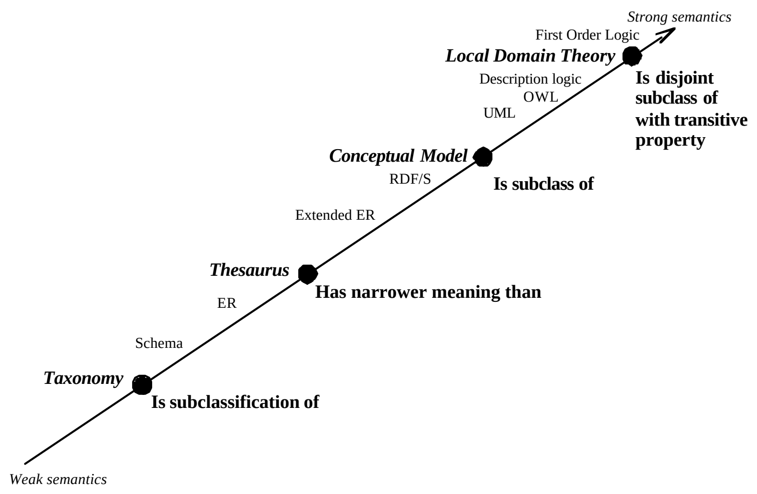


Figure 1. The ontology spectrum

Representational models on the various points along the ontology spectrum have different uses [14]. In the simplest case, a group of users can agree to use a controlled vocabulary for their domain. This of course does not guarantee that they will use the terms in the same way all the time, but if all the users including database designers chose their terms from an accepted set, then the chances of mutual understanding are greatly enhanced.

Perhaps the most publicly visible use for simple ontologies is the taxonomies used for site organization on the World Wide Web. This allows designers to structure information and users to browse and search. Yahoo is one such hierarchy. In addition the DMOZ (www.dmoz.com) project attempts to establish a taxonomy of resources that are administered by its very large number of volunteer editors. Taxonomies can also help with sense disambiguation since the context of a term is given by the more general terms in the taxonomy.

Structured ontologies provide more sophisticated usage scenarios. For instance, they can provide simple consistency and completeness checks. If all *products* must have a *price* then web sites can automatically be checked for missing or conflicting information. Such ontologies can also provide completion where partially specified information can be expanded automatically by reference to the terms in the ontology. This expanded information could also be used for refining search, for instance.

Ontologies can facilitate interoperability, by aligning different terms that might be used in different applications. For example an ontology in one application might define a `StanfordEmployee` as a `Person` whose `employer` property is filled with the individual `StanfordUniversity`. If another application does not understand `StanfordEmployee` or `employee` but does understand `Person`, `employer` and

`StanfordUniversity`, then it is possible to make the two applications talk to each other if the second application can intelligently interpret the ontology of the first [14].

Now we are in a position to see why the ontologies on the most formal end of the spectrum are often taken as the default interpretation in the context of the semantic web, providing the conceptual underpinning for " ... making the semantics of metadata machine interpretable" [15]. But for the semantics of a domain model to be machine interpretable in any interesting way, it must be in a format that allows automated reasoning in a flexible manner. Obviously, taxonomies can specify little in this sense. Database schemas are more powerful, but limit the interpretation to a single model. The only automated reasoning that can be performed is what is allowed by the relational model, and the semantics can only be understood through complex inferences supplied by humans (e.g., the database designer). Formal logic based ontologies provide multiple possible models allowing machine based inferences, but still limit the set of formal models to the set of intended meanings. They are at the same time more formally constrained and more semantically flexible than database schemas. Ontologies based on different logical models can support different kinds of inference, but a minimal set of services should include reasoning about class membership, class equivalence, consistency, and classification [13].

The ontology representation language adopted by the Web Ontology Working Group of the W3C[2] is the Web Ontology Language (OWL). OWL is a response to a number of requirements [16] including the need for a language with formal semantics that enables automated reasoning, and to address the inherent limitations of RDF/S.

2.1 OWL

According to the original design goal, OWL was to be a straightforward extension of RDF/S, guaranteeing downward compatibility such that an OWL aware processor could also understand RDF/S documents without modification. Unfortunately this did not succeed because the generality of some RDF/S elements (e.g. the semantics of *class* as "*the class of all classes"*) does not make RDF/S expressions tractable in the general case. In order to maintain computational tractability, OWL processors include restrictions that prevent the interpretation of some RDF/S expressions. OWL comes in three flavors: OWL Full, OWL DL, and OWL Lite, and OWL Full is upward and downward compatible with RDF but OWL DL and OWL Lite are not.

The names of the three sub languages of OWL describe the expressiveness of the languages, keeping in mind a fundamental tradeoff between expressiveness, efficiency of reasoning, and support for human understanding. OWL Full has constructs that make the language undecidable. Developers should therefore only use OWL Full if the other two sub languages are inadequate for modeling the relevant domain. Similarly, OWL DL should be used if OWL Lite is not sufficient. Details of the syntax and semantics can easily be obtained from the technical documentation web site of the W3C, http://www.w3.org/TR/

In order for ontology based technologies to be useful, it is necessary to have development tools and platforms. There are currently a large number of tools

2http://www.w3.org/2001/sw/WebOnt/

available for authoring, managing, visualizing, and merging ontologies. The ontology editor survey[3] is a valuable resource comparing most of the available tools on a number of dimensions. One popular tool (with almost 20000 registered users) is Protégé[4], maintained at Stanford University. It is a freely available, Java based platform that has a rich feature set that is easily expandable through a plug-in based architecture. In addition, the Protégé web site contains a large number of links to ontology resources, and an interesting guide to building ontologies[5].

3. Web Services

There is a great deal of interest about web services and service oriented architectures in general. A useful definition can be found in [9]: "Web services are software applications that can be *discovered, described*, and *accessed* based on XML and standard Web protocols over intranets, extranets, and the Internet." This definition exposes the main technical aspects of web services, to do with discovery and description, as well as the role of WWW technologies for data exchange and communication. These core concepts along with the associated technologies are shown in figure 2 below.

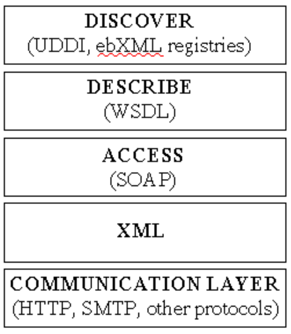


Figure 2. The basic layers of Web services [9]

The basic steps involved in composing a web service to meet a given application scenario are as follows:

1. the client application discovers information about Web Service A in a UDDI registry
2. the client application gets the WSDL for Web Service A from the UDDI registry to determine Web Service A's API.
3. the client application sends a request to Web Service A using SOAP
4. the client application receives a SOAP response from Web Service A.

3http://www.xml.com/2004/07/14/examples/Ontology_Editor_Survey_2004_Table_-_Michael_Denny.pdf
4http://protege.stanford.edu/
5http://protege.stanford.edu/publications/ontology_development/ontology101.html

It is important to situate the role of Web services in the real world. Daconta, Obrst and Smith [9] argue that the most important factor for determining the future of a new technology is not "... how well it works or how "cool" it is ..." but on business adoption. Along this line they see a bright future for Web services which is being promoted by Microsoft, IBM, Sun, as well as the open source community. But why such widespread support? One reason is the promise of interoperable systems. Once businesses adopt standardized web service descriptions, the possibility of exchanging data and sharing the cost of services increases. In addition, the open standards prevent monopolization of applications, preventing the dreaded "vendor lock-in" associated with proprietary solutions. Finally, a widespread adoption of Web service protocols means that existing applications can be leveraged by turning them into Web services. As an example, it is even possible for .NET clients and servers to talk to J2EE servers using SOAP.

The point of all this is that Web services enable interoperability at the level of business processes without having to worry about interoperating between different applications, data formats, communication protocols, and so on. We will see in the next section that this has profound implications for the way workflows and knowledge based work processes are modeled and instantiated in particular work environments.

Details of the various technologies can be found in many sources, and will not be covered here. The purpose of the technologies is to provide some means of discovering services that can be *orchestrated* into an application that accomplishes some need. The Web services technologies combined with Ontologies can provide the infrastructure, protocols, and semantics to get the job done. The final remaining problem is in defining the job that needs to get done. This is where workflow technology must be considered.

4. Workflow and enterprise process modeling

The unprecedented flexibility of web services provides a particular challenge for how to integrate their use in enterprise work practices. On the one hand demand based service provision promises to be a blessing for facilitating problem solving; on the other hand, the instance based variability provided through the relatively free range of solutions offered in service composition could result in a serious challenge to established workflow modeling paradigms.

Process modeling implicates a family of techniques used to document and explicate a set of business and work processes in a way that allows their analysis at various levels, and for various purposes. Our specific interest in the current project is to use workflow modeling to analyze the work contexts that are likely to be involved in the day to day activities of an enterprise, with the aim of improving the timely delivery of appropriate information related resources and services. The purpose is to integrate workflow modeling with the potential of web services, to capture the likely usage scenarios under which the services will need to operate and to model this integrated use. The aim is that the model of work practices will allow better specification of actual information needs, which will in turn allow for richer requirements for the service descriptions expected from a web service, which will facilitate service composition and interoperability. The research problems therefore complement one another: workflow modeling helps web service design, but the availability of these services in turn improves workflow modeling techniques.

The challenge for us is to construct modeling approaches that maintain sufficient expressive power for the required points of view as well as to allow flexibility for changing situations. Jørgensen [20] argues that static workflow models cannot handle the changing demands of real world situations, and that adaptive models, while providing greater flexibility, still cannot adequately handle the instance based user driven modifications needed in many situations. He argues for interactive models that can be dynamically configured by users.

4.1 Workflow and Process Modeling Languages

Workflow modeling has been used to learn about, guide and support practice in a number of different areas including software process improvement [21], [22]; enterprise modeling [23], [24]; process centric software engineering [25]; and workflow systems [26]. The process modeling languages employed in these areas have been usefully categorized into one of the following types: transformational, conversational, role-oriented, constraint-based, and systemic [27]. In addition, [20] argues that UML based approaches need to be considered as a sixth category as they are becoming increasingly widely used. A summary of each type is given in [20] where they are considered for their suitability as interactive modeling paradigms.

Transformational languages represent the majority of process modeling languages in use, adopting an input-process-output approach. Some well known languages adopting this approach are Data Flow Diagrams (DFD), Activity diagrams, Event-driven Process Chains (EPC), and Petri nets. While there are clear differences between the formalisms, it is possible to generalize in terms of their basic expressive commitments and therefore suitability for modeling dynamic, flexible workflows [28], [29], [30], [31]. The standards defined by the Workflow Management Coalition (WfMC) [26], the Internet Engineering Task Force (IETF) [32], and the Object Management Group (OMG) [33] are all predicated on a common perspective. We consider a few languages from this perspective.

The WfMC standards for process definition interchange between systems [34] include a large portion of the primitives involved in transformational languages. Processes are modeled with hierarchical decomposition, control flow structures for sequences, iteration, AND and XOR branching. Activities can be associated with organizational roles and actors, tools and applications. The core terminology of the WfMC is shown in figure 3 below. Note the distinction between process definition (idealized process) and instance (actual work),

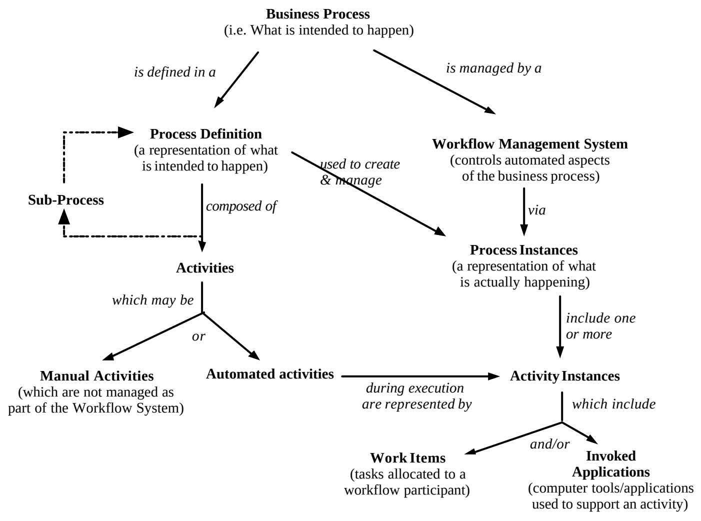

Business Process
(i.e. What is intended to happen)
is defined in a
is managed by a
Process Definition
(a representation of what is intended to happen)
Workflow Management System
(controls automated aspects of the business process)
used to create & manage
Sub-Process
composed of
via
Process Instances
(a representation of what is actually happening)
Activities
which may be
or
include one or more
Manual Activities
(which are not managed as part of the Workflow System)
Automated activities
during execution are represented by
Activity Instances
which include
and/or
Work Items
(tasks allocated to a workflow participant)
Invoked Applications
(computer tools/applications used to support an activity)

Figure 3. WfMC core terminology [26]

The Business Process Modeling Language (BPML) [35] defines a *web service* interface description language, which presents obvious promise concerning the present requirements. BPML emphasizes low-level execution and contains several control flow primitives for loops (foreach, while, until), branching (manual choice or rule based switch, join), decomposition (all, sequential, choice), instantiation (call, spawn), properties (assign), tools (action), exceptions (fault), and transactions (compensate). The ability to define manual as well as rule based branching is promising for use in flexible systems. Unfortunately the promise is only partially realized since different primitives are used for the two cases, implying that the automation boundary must be defined during process design. Additionally, BPML has weak support for local change and unforeseen exceptions. A visual notation, for BPML, BPMN has been developed lately and is getting increasing attention.

Event-driven Process Chains (EPC) is a process language used in industrial systems, being part of the SAP ERP system and ARIS modeling tool [36]. EPC models units of work as *functions* enabled by pre-events and causing post-events. Relationships between events and functions are modeled by arcs, AND, XOR, and OR connectors. Some attempts have been made to map EPC to Petri nets [37] and UML activity diagrams [38]. One point of difference is that OR-connectors are not well supported in the Petri net notation because they result in a degree of uncertainty in execution. This gives EPCs some degree of freedom to represent alternative events, though the alternatives once again need to be determined at design time.

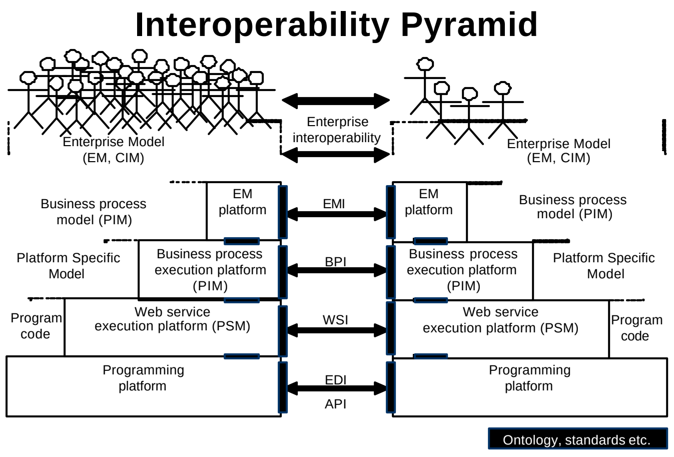
There is some recognition for the need to separate design components from run-time components for increased flexibility. This is realized in the WfMC's XML Process Definition Language (XPDL) [39]. But even here, the separation is focused mainly for facilitating the reuse of design components across different workflow engines and design tools. There is little support for user driven interaction at run-time.

It appears that current approaches are not designed with the flexibility required to accommodate the adaptive workflows that are enabled by Web Services technologies. One approach worth pursuing is the interactive models of Jørgensen [20].

5. Integrating enterprise and IS development

The different approaches outlined above can be combined with more traditional model-driven system development to support a number of problem situations from very static, to very dynamic, even emergent processes. The different process types decide the extent to which the underlying technology can be based on hard-coded, predefined, evolving or implicit process models. This gives a number of development approaches. On one extreme; systems are manually coded on top of a traditional runtime environment, and on the other enterprise process models are used directly to generate solutions. In between these, we have the approaches typically described in MDA, namely the development of Platform Independent Models (PIMs) for code-generation (e.g. on top of a UML Virtual Machine), or for Platform Specific Models (PSMs) for more traditional code-generation.

In figure 4, we outline the different types of interoperability across these different levels, being illustrated as a pyramid of different interoperating levels [40]. Whereas traditional systems use special APIs and approaches such as EDI for interchange of data, on the next level (PSM), we can identify Web Services Interfaces. Above this level, there is a lot of work on specific business process execution platforms, with a possibility to exchange directly using a BPI. Finally, projects such as EXTERNAL(http://www.external-ist.org) and UEML (http://www.ueml.org) have provided solutions for how to interoperate on the enterprise model level, using different modeling languages and different tools in the process. Standards and ontologies can be used across all levels, and also between


Interoperability Pyramid
Enterprise interoperability
Enterprise Model (EM, CIM)
Enterprise Model (EM, CIM)
Business process model (PIM)
EM platform
EMI
EM platform
Business process model (PIM)
Platform Specific Model
Business process execution platform (PIM)
BPI
Business process execution platform (PIM)
Platform Specific Model
Program code
Web service execution platform (PSM)
WSI
Web service execution platform (PSM)
Program code
Programming platform
EDI
API
Programming platform
Ontology, standards etc.

levels to make the interoperation happen more smoothly.

Figure 4. Interoperability between different platforms

We will try to clarify this picture with an example which originally had focus on the enterprise level and cross-organizational business processes to support dynamically networked organizations. The infrastructure to support networked organizations originally developed in the EXTERNAL IST project (which is currently further developed in the ATHENA (http://www.athena-ip.org), MONESA and WISEMOD projects) can be described as consisting of three layers. These layers are identified as:

- Layer 1, the *information and communication technology* (ICT) layer: – defining and describing the execution platform, software architectures, tools, software components, connectivity and communication.
- Layer 2, the *knowledge representation* layer: - defining and describing constructs and mechanisms for modeling, including ways of annotating models and meta-models with content from ontologies.
- Layer 3, the *work performance and management* layer; - modeling and implementing customer solutions, generating work environments as personalized and context-sensitive user interfaces available through web-portals.

5.1 The ICT Layer

The ICT-infrastructure is an integration of the enterprise and process modeling tools, such as:

- METIS [41], a general purpose enterprise modeling and visualization tool,
- XCHIPS [42], a cooperative hypermedia tool integrated with process support and synchronous collaboration,
- SimVision [43], a project simulator used to analyze resource allocation, highlighting potential sources of delay and backlogs.
- WORKWARE [44],[20] a web-based emergent workflow management system with to-do-lists, document sharing, process enactment and awareness mechanisms.
- FrameSolutions [45], a commercially available framework for building automated workflow applications.

The architecture has 3-tiers, clients, application servers, and data servers. The implementation is web-based, utilizing HTTP both for control and data integration, and XML for data exchange. Three types of integration mechanisms are provided :

1. Data-centered integration: based on a common EXTERNAL XML DTD, XML importing/exporting utilities are implemented in each of the enterprise modeling tools for data exchange between the tools or with an XML repository.
2. Control-centered integration: uses the APIs provided by the tools and the repository to be integrated. With the APIs, the tools can call each other and access the shared repository. Some of the APIs may have parameters for denoting content objects, whose implementation requires data-centered integration capabilities.
3. Worktop-based integration: this is a service-oriented integration at the user-interface level, utilizing both data-centered integration and control-centered integration to access shared models, information objects, and tools.

New components can be developed on a need-driven basis, using the approaches of service oriented architecture (SOA) or model-driven architecture (MDA).

5.2 The Knowledge Representation Layer

The knowledge representation layer defines how models, meta-models and meta-data are represented, used and managed. A version of Action Port Modeling (APM) [46], [20] constitutes the core of the modeling language (EEML). The kernel concepts are shown in Figure 5 as a simplified logical meta-model of EEML. The process logic is mainly expressed through nested structures of *tasks* and *decision points*. The sequencing of the tasks is expressed by the *flow* relation. *Roles* are used to connect resources of various kinds (people, organizations, information, and tools) to the tasks. Hence, modeling in EEML captures an extensive set of relationships between the goals, organizations, people, processes and resources. This is particularly useful considering the dynamic nature of networked organizations. For new partners joining the network, the rich enterprise models provide a valuable source of knowledge on how to “behave” in the network. We plan to further enable the annotation of such models with OWL-ontologies, improving model matching and model interoperability across organizations.

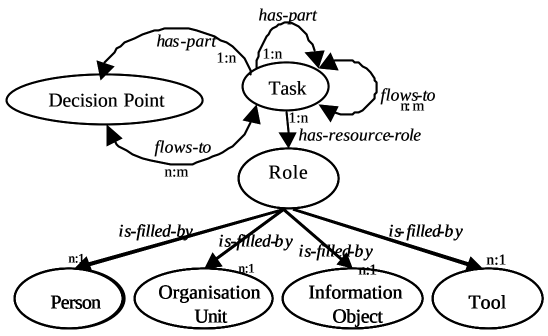


Figure 5. Simplified meta-model of EEML

Moreover, the interactive nature of the models, meaning that the users are free to refine them during execution, increases their potential as sources of experience and knowledge. As such they *document* details on how the work was actually done, not only how it was once planned. EEML can be extended to link into for formal process modeling (E.g. BPMN) as well as languages used in the SOA and MDA approaches, particularly UML, through the meta-modeling features of METIS, which is the main implementation platform for EEML.

From a knowledge management perspective, process models are carriers of process knowledge; knowledge of how to do things. But through the possibility in EEML of attaching information resources to the tasks at any level, such a model also imposes a structure upon the set of information resources relevant for the work described by the process model. That way, the process models themselves form the basis for information management. Further extensions with semantic annotations based on

OWL would enhance this even further, combining process ontologies with more traditional structural ontologies.

5.3. The Work Performance and Management Layer

Users access their solutions through project portals. A project portal for a networked organization must have support for methodology adaptation, communication, co-ordination and collaboration. Project management, reporting and other services must be offered, and finally project work must be performed with possibilities for repetition, providing security and privacy for knowledge workers.

In our infrastructure, the web-based portal registers and qualifies users, and invokes other tools through WORKWARE, an example of what we term a model-generated workplace (MGWP). The modeled tasks are also executed through the invocation of tools and applications from the web based user environment comprised of the portal and WORKWARE. WORKWARE sets up the context for each task, giving access to the knowledge and resources needed to perform the task. The actual work performance is done by invoking appropriate web services. The task performers may access desktop tools, organizational information systems, web services, or automated processes (in FrameSolutions) through this user environment.

User environments are generated dynamically based on the definition of tasks using EEML. Forms and components for interacting with different model objects are selected and composed based on user interface policies.

The dynamically generated work management interface includes services for work performance, but also for process modeling and meta-modeling. The *worktop* is the main component in this interface. In addition to the services for performing and managing the task, it contains links to all knowledge in the process models that is relevant for the task. Since the worktop is dynamically generated, subject to personal preferences, the skill levels of task performers can be taken into account, e.g. to provide more detailed guidelines for people who have not previously worked on such tasks. Similarly, customized worktops for project management can support the project management team. The contents may include an overview of the project, adopted management principles, applicable methodologies, project work-break-down structure, results, plans and tasks, technologies and resources, status reporting and calculations. Further details on this approach can be found in [20], [47].

6. Conclusion

This paper has provided a unified survey of relevant technologies for achieving semantic interoperability in the context of enterprise information systems, namely ontologies, service descriptions, and workflow models including both automated and interactive tasks. The Interoperability Pyramid, together with the example explanations of this, illustrates how the combination of these technologies can provide more advanced interoperability that with current systems. We suggest that “interoperability” in the abstract may be an untenable goal, at least in the immediate future. But interoperability in the context of dynamic and interactive workflows, as the next best thing, is very much within our reach.

Acknowledgements

Csaba Veres is funded by the Norwegian Research Council through the WISEMOD project (NFR IKT-2010). John Krogstie has been funded by the projects ATHENA and MONESA. The ideas of this paper are pursued in the context of the EU NoE project INTEROP, and we thank other partners of INTEROP for valuable inspiration.

References

1: Berners-Lee, T., Hendler, J., and Lassila, O. The Semantic Web., *Scientific American*, 2001, May,
2: Manola, F., Miller, E. *RDF Primer, W3C Recommendation 10 February 2004*, 2004, http://www.w3.org/TR/rdf-primer/,
3: Broekstra, J., Kampman, A., van Harmelen, F. Sesame, in *Spinning the Semantic Web*, Fensel, D., Hendler, J., Lieberman, H., Eds. , MIT Press, 2003, Cambridge, MA
4: Powers, Shelly *Practical RDF*, O'Reilly, 2003
5:*Orbis Latinus, Linguistic Terms*, 2004, http://www.orbilat.com/General_References/Linguist,
6: Butler, H.M. *Barriers to real world adoption of semantic web technologies*, Technical Report, 2002, Hewlett Packard, Filton Road Bristol BS34 8QZ UK
7: Hayes, P. *RDF Semantics*, , http://www.w3.org/TR/rdf-mt/,
8: *The RDF Validation Service*, 2004, http://www.w3.org/RDF/Validator/,
9: Daconta, M.C., Orbst, L.J., Smith, K.T *The Semantic Web*, Wiley, 2003
10: Brickley, D. *RDF: Understanding the Striped RDF/XML Syntax*, 2001, http://www.w3.org/2001/10/stripes/,
11: Miles, A. RDFMolecules:Evaluating Semantic Web Technology in a Scientific Application, 2003
12: Klein, M., Broekstra, D., Fensel, F., van Harmelen, F., Horrocks, I Ontologies and Schema Languages on the Web, in *Spinning the Semantic Web*, Fensel, D., Hendler, J., Lieberman, H., Eds., MIT Press, 2003, Cambridge, MA
13: Grigoris, A. & v. Harmelen, F. Web Ontology Language: OWL, in *Handbook On Ontologies*, , Eds. International Handbooks on Information Systems, Springer, 2004, Berlin, Germany
14: McGuiness, D. L. Ontologies Come of Age, in *Spinning the Semantic Web*, Fensel, D., Hendler, J., Lieberman, H. Wahlster, W, Eds., MIT Press, 2003, Cambridge, MA.
15: Staab, S. & Studer , R. , *Handbook On Ontologies*, International Handbooks on Information Systems, Springer, Berlin, Germany, 2004
16: *OWL Web Ontology LanguageUse Cases and Requirements*, 2004, http://www.w3.org/TR/webont-req/, Accessed Aug. 23, 2004
17: *OWL Web Ontology LanguageSemantics and Abstract SyntaxSection 2. Abstract S*, 2004, http://www.w3.org/TR/owl-semantics/syntax.html,
18: *OWL Web Ontology LanguageGuide*, 2004, http://www.w3.org/TR/owl-guide/, Accessed Aug. 23, 2004
19: OASIS/ebXML Registry Technical Committee *OASIS/ebXML Registry Information Model v2.6*, 2004, http://www.oasis-open.org/committees/download.php/,

20: Jørgensen, H. D., 2004, *Interactive Process Models*, PhD-thesis, NTNU, Trondheim, Norway, ISBN 82-471-6203-2.

21: Bandinelli, S., Fuggetta, A., Lavazza, L., Loi, M., and Picco, G.P. *Modeling and Improving an Industrial Software Process,* IEEE Transactions on Software Engineering. vol. 21, no. 5, 1995.

22: Derniame, J. C. *Software Process: Principles, Methodology and Technology,* , LNCS 1500, Springer, Berlin, Germany, 1998

23: Brathaug, T. A., and Evjen, T. A. *Enterprise Modeling*, , STF 38 A96302 ,1996, , Trondheim, Norway

24: Fox, M. S. and Gruninger, M. *Enterprise Modeling*, AI Magazine, 2000

25: Ambriola, V., Conradi, R., and Fuggetta, A. *Assessing Process-Centered Software Engineering Environments*, ACM Transactions on Software Engineering and Metho, 6, 3, 1997

26: *WfMC Workflow Handbook 2001*, , Workflow Management Coalition, Lighthouse Point, Florida, USA, 2000

27: Carlsen, S *Conceptual Modeling and Composition of Flexible Workflow Models*, Norwegian University of Science and Technology, 1997

28: Conradi, R. and Jaccheri, M. L. Process Modelling Languages, in *Software Process: Principles, Methodology and Tech,* , Eds. Lecture Notes in Computer Science, Springer LNCS 1500, 1998,

29: Curtis, B., Kellner, M. I., and Over, J. *Process Modeling*, Communications of the ACM, 35, 9, 1992.

30: Green, P. and Rosemann, M. *Integrated Process Modeling: An Ontological Evaluation,* Information Systems, 25, 3, 2000

31: Lei, Y. and Singh, M. P. A. A Comparison of Workflow Metamodels, in *ER Workshop on Behaivoral Modeling,* , Eds. , Springer LNCS 1565, 1997,

32: Bolcer, G. and Kaiser, G. *SWAP: Leveraging the Web to Manage Workflow*, IEEE Internet Computing, 3, 1, 1999, ,

33: *OMG Workflow Management Facility v. 1.2*, , Object Management Group, , 2000

34: *WfMC Workflow Management Coalition, Interface 1: Process Definition Interch*, Draft 5.0, Workflow Management Coalition, , 1996

35: Arkin, A. *Business Process Modeling Language - BPML 1.0 Working Draft*, BPML.org, 2002, ,

36: Scheer , A. W. and Nuttgens, M. ARIS Architecture and Reference Models for Business Process Management, in *Business Process Management*, W. v. d. Aalst, J. Desel, and A. Oberweis, Eds. , Springer LNCS 1806, 2000, Berlin, Germany

37: Aalst, W. M. P. v. d. *Formalization and Verification of Event-driven Process Chains,* Information and Software Technology, 41, 10, 1999, ,

38: Loos, P. and Allweyer, T. *Process Orientation and Object Orientation - An Approach for Integrating UM,* , , University of Saarland, Germany, 1998

39: *Workflow Process Definition Interface -- XML Process Definition Language*, 2002, http://www.wfmc.org/standards/docs/TC-1025_10_xpdl, Accessed Aug. 19, 2004

40: Krogstie, J. Integrating Enterprise and IS-development Using a Model-Driven Approach. in 13th International Conference on Information Systems Development. 2004. Vilnius, Lithuania: Kluwer.

41: Lillehagen, F. , 1999, Visual extended enterprise engineering embedding knowledge management, systems engineering and work execution, *Proceedings of IEMC '99, IFIP International Enterprise Modelling Conference,* Verdal, Norway.

42: Haake, J. M., and Wang, W., 1997, Flexible support for business processes: Extending cooperative hypermedia with process support, *Proceedings of GROUP '97,* Phoenix, Arizona USA

43: Kuntz, J. C., Christiansen, T. R., Cohen, G. P., Jin, Y., and Levitt, R. E., 1998, The virtual design team: A computational simulation model of project organizations, *Communications of the ACM*, vol. 41, no. 11.

44: Jørgensen, H. D., 2001, Interaction as a framework for flexible workflow modeling. *Proc International ACM SIGGROUP Conference on Supporting Group Work*, Boulder CO, September.

45: Kallåk, B. H., Pettersen, T. B., and Ressem, J. E. , 1998, Object-oriented workflow management: Intelligence, flexibility, and real support for business processes, *Proceedings of OOPSLA Workshop on Implementation and Application of Object-Oriented Workflow Management Systems*, Vancouver, Canada.

46: Carlsen, S., 1998, Action port model: A mixed paradigm conceptual workflow modeling language, *Proceedings of Third IFCIS Conference on Cooperative Information Systems (CoopIS'98)*, New York.

47: Krogstie, J., and Jørgensen, H. D., 2004, Interactive models for supporting networked organizations. *Proceedings of CAiSE'2004*, June 9-11 Latvia, Riga.