Enabling Sustainable Interoperability for Enterprise Applications with Knowledge Links

Artur Felic[1,2], Felix Herrmann[1], Christian Hogrefe[1], Michael Klein[1] and Birgitta König-Ries[2]
[1]*CAS Software AG, Karlsruhe, Germany*
[2]*Friedrich-Schiller-Universität Jena, Institut für Informatik, Jena, Germany*

Keywords: Semantic Web, Knowledge Links, Semantic Interoperability, Sustainable Interoperability, Enterprise Interoperability.

Abstract: In complex and collaborative ecosystems like business networks, different partners need to share their respective expert knowledge in order to be successful. Due to the diversity of business applications and highly customized application suites used by business partners, knowledge exchange and the establishment of interoperability between enterprise applications is extremely difficult. Different representations and meanings of enterprise data lead to incomprehensibility between partners inside collaborative environments. This paper presents an model-driven approach to support sustainable interoperability for enterprise applications in collaborative environments. Based on an event-driven architecture, Knowledge Links enable dynamic modelling of knowledge transformations between knowledge domains. They keep background consistency between the connected domains, thus making enterprise applications interoperable. Knowledge Links can be created and modified at any time, which enables sustainable interoperability. Business partners are able to rely on their enterprise applications and don't need to switch to another system. Sensitive data stays covert due to the nature of modular ontologies. The presented approach is exemplified in the context of the OSMOSE Project and will be evaluated by the proof-of-concept scenarios in this Project.

1 INTRODUCTION

Enterprise Applications can be described as highly customized applications or application suites that serve individual enterprise needs. They often comprise a set of expert tools for specialist capabilities. According to Fowler (Fowler, 2003) "Enterprise applications are about the display, manipulation, and storage of large amounts of often complex data and the support or automation of business processes with that data". Enterprises usually compose their set of applications by applications from different competing vendors, which do not want to reveal sensitive data in order to be integrated. Since the paradigm shift from individual businesses to Complex and collaborative ecosystems like business networks with many business partners, the situation became even more complex. Competing and globally spread enterprises, having many different systems installed, have to collaborate in order to create business value. This circumstance leads to a problem from which a whole branch of science called Enterprise Interoperability emerged.

According to the IEEE Standard Computer Dictionary (IEEE, 1991) Interoperability can be defined as the "ability of systems or components to exchange and use information". Complementary, the ISO 1610 standard defines interoperability as the "ability to share and exchange information using common syntax and semantics to meet an application-specific functional relationship through the use of a common interface". Enterprise Interoperability concerns data, services, processes and internal business. Thus, different data models, various services and processes as well as different levels of organization are aimed to be interoperable. Many emerging Enterprise Interoperability Frameworks like EIF (IDABC et al., 2004), FEI (Chen and Daclin, 2006), C4IF (Peristeras and Tarabanis, 2006) and AIF (ATHENA Project, 2007) aim at enabling Enterprise Interoperability. Although the implementation of Enterprise Interoperability in a collaborative ecosystem is desirable, the dynamics inside the ecosystem also need to be taken into account to be successful. The implementation is not a one-off activity, but a continuous process that needs to be sustainable in order to allow the entrance of new business partners or changes inside the existing

Felic A., Herrmann F., Hogrefe C., Klein M. and König-Ries B..
Enabling Sustainable Interoperability for Enterprise Applications with Knowledge Links.
DOI: 10.5220/0005369605970607
In *Proceedings of the 3rd International Conference on Model-Driven Engineering and Software Development* (MDE4SI-2015), pages 597-607
ISBN: 978-989-758-083-3

ecosystem.

Sustainability and interoperability are two inherently linked and inseparable aspects that need to be addressed together (Dassisti et al., 2013). Sustainable Interoperability can be described as ”novel strategies, methods and tools to maintain and sustain the interoperability of enterprise systems in networked environments as they evolve with their environments” (Jardim-Goncalves et al., 2012). Detection of new enterprise systems, learning capabilities, adaptability, analysis of the internal behaviour as well as communication inside the network is addressed by this research area. Interoperable collaborative ecosystems with many different business partners running many different enterprise applications must be seamlessly configurable.

The Object Management Group (OMG) (Object Management Group, 2014b) proposed Model-Driven Architecture (MDA) (Object Management Group, 2014a) as a strategy towards interoperability of heterogeneous systems (Singh and Sood, 2009). The separation of computation independent models (CIMs), platform independent models (PIMs) and platform specific models (PSMs) enables interoperability between enterprise applications by allowing transformations between CIMs, PIMs and PSMs. Long term flexibility, incorporating models from different viewpoints, technology independence, multi platform model support and easier integration and interoperability are some of the major benefits of MDA. Cost for the development life cycle is reduced whereas software quality can be improved.

MDA can be extended by Semantic Web technologies, i.e. ontologies, to create unambiguous domain vocabularies (W3C, 2006). Gruber defines ontologies as ”an explicit specification of a conceptualization” (Gruber, 1993). Knowledge sharing is fundamental in collaborative ecosystems. Ontologies can be used to structure knowledge and knowledge properties in order to make knowledge understandable for machines and humans. There are many ontologies defined for different purposes to create shared knowledge structures and to set up a common knowledge base.

In highly dynamic collaborative ecosystems, experience has shown that it is unrealistic to assume commitment from every current and future business partner to an integrated ontology covering all relevant aspects of all business applications. A new business partner entering the collaborative ecosystem or the substitution of an enterprise application could lead to the necessity to change the whole knowledge base. Upper ontologies like SUMO (Niles and Pease, 2001) are purposed to structure only very general concepts in order to support semantic interoperability. Following the idea of modular ontologies (Ben Abbs et al., 2012) large monolithic ontologies can be divided into smaller domain modules in terms of interchangeability. As the name suggests, domain ontologies are often not comprehensible outside the domain, but the knowledge structured inside is of importance for other domains. Therefore, links between those ontology modules are necessary in order to provide different perspectives of knowledge to the business partners that require it.

In this paper, we propose a model-driven approach using Knowledge Links (Felic et al., 2014). Knowledge Links are used to connect the knowledge domains and transform knowledge from one to another. Their specification constitute a Platform Independent Model (PIM). Depending on the underlying Platform Model (PM), i.e. the platform, enterprise applications or database of business partners involved in the Knowledge Link, the modelled Knowledge Link is further transformed into events and business processes by model transformations into a Platform Specific Model (PSM).

By connecting different knowledge domains in a black box manner, business partners of a collaborative environment can rely on their knowledge structures. We present an event-driven and service oriented architecture approach in which each enterprise application constitute a separated knowledge domain with its own ontology. Furthermore, a tool is presented with which knowledge links can be created and maintained at any time during system runtime and thus enable sustainable interoperability. During system runtime, the complex event processor of the middleware is aware of keeping background consistency by analysing events that are associated with Knowledge Links and reacting according to the modelled behaviour.

In the OSMOSE Project (OSMOSE, 2014), the aim is to interconnect the real, digital and virtual world for sensing liquid enterprises. In the real world physical devices, objects and actors are present, whereas their virtual models and hypothetical scenarios are contained in the virtual world. The digital world includes data sets about objects and their models as well as metadata and multimedia content. The sensing liquid enterprise is composed of two enterprise paradigms: the sensing enterprise and the liquid enterprise. The sensing enterprise is a cooperative nervous system where virtual and physical smart objects, equipment and infrastructure form an active network. In the liquid enterprise, boundaries are fuzzy and it is hard to distinguish the inside from outside in terms of human resources, markets, products and

processes due to the prevailing cloudiness. An event-driven and service-oriented middleware automatically keeps background consistency between these worlds, i.e. the shadow ambassadors of things in each of the world. Each world has there own applications and data structures, while the middleware has to be aware of the meaning of data in order to decide which data has to cross world borders to transform and deliver data to the other worlds. The situation is quite similar to business networks and enterprise applications. The three worlds together with the middleware can be seen as an collaborative environment, whereas the worlds themselves are business partners with their installed enterprise applications.

After this introduction, in Section 2, we will explore related work to this paper. In Section 3, the modular and hierarchical knowledge base concept is introduced. The proposed Event-Driven Architecture is described in Section 4. The two latter sections constitute the foundation of Section 5, in which the sustainable interoperability approach is illustrated. In Section 6 the paper is concluded and summarized and future research directions are specified.

2 RELATED WORK

Enterprise Interoperability has received growing attention in the last decade. (Lampathaki et al., 2012) identified the lack of scientific foundations and formulated several scientific areas of enterprise interoperability based on technological trends and enterprise interoperability background knowledge. The main scientific areas the present paper addresses are data, software, knowledge and ecosystems interoperability.

Many sustainable interoperability approaches in literature arise from the scientific fields of Semantic Web, Model-Driven Architecture (MDA) or Event-Driven Architecture (EDA). While one of the main objectives of semantic web technologies is to induce a common meaning about knowledge for human beings and machines, MDA and EDA are concerned with platform independence and heterogeneity of systems besides other things. The following scientific findings and publications give an overview about related work in these scientific areas.

(Chen et al., 2008) analysed enterprise interoperability architectures from 1985 to 2000 and states that most effort has been put in framework approaches rather architectures of enterprise systems. Furthermore, the authors identified the lack of an enterprise architecture ontology, the absence of scientific methodologies and interoperability between existing architectures as well as weak impact in industry. The authors identified conceptual barriers as one of the main category of interoperability barriers that are ”concerned with the syntactic and semantic differences of information to be exchanged”, which highly motivates our work. Moreover, they describe the interoperability of data as a category of interoperability concerns that ”deals with finding and sharing information from heterogeneous data sources”. According to the categorization of interoperability approaches, we present a federated approach, where partners do not impose own models on the enterprise architecture. In our approach, we allow that each partner can rely on its own knowledge structured in an ontology, although partners may share knowledge in a common ontology. Semantic mapping is handled with Knowledge Links by interconnecting the ontologies appropriately.

The authors of (Chungoora et al., 2013) utilize a model driven architecture approach and ontologies to enable knowledge sharing for manufacturing systems interoperability. The Manufacturing Core Ontology constitutes the heart of the Platform Independent Model (PIM), whereas other domains extend the Manufacturing Core Ontology to specify domain specific models. To enable knowledge sharing of the further transformed Platform Specific Model (PSM), knowledge verification mechanisms are specified at PIM level. However, the approach is limited to the specification of constraints on PIM level and does not allow appropriate knowledge propagation with transformations in order to provide partners with new knowledge build from their knowledge base.

Kadar et Al. (Kadar et al., 2013) propose a multi-agent system to support sustainable interoperability for networked organizations. Rule based negotiation at business level by agents in a distributed environment enable the re-establishment of interoperability between partners. Although the work of Kadar et Al. is inspiring and knowledge negotiation is demanding, the need to transform knowledge between different knowledge domains and the creation of complex knowledge relationships among different partners in terms of sustainable interoperability is not covered. Knowledge Links could therefore complement this work.

(Ni and Fan, 2008) present an ontology based approach. Domain ontologies are connected by special collaboration ontologies, which are semi-automatically created. These collaboration ontologies are described by OWL-S (W3C, 2004) in order to ”enables automatic service discovery, invocation, composition, interoperation, and execution monitoring”. The OWL-S descriptions from these collaboration ontologies are further transformed to BPEL

and WSDL to allow automated execution of ontology mappings. Thus, an business process execution engine is able to react appropriately to requests according to data changes. Although the approach is very closely related to the knowledge link approach, it is strongly dependent on the mechanisms behind the semi-automatic creation process of the collaboration ontologies. Additionally, complex links with operations and transformations cannot be created by hand.

The authors of (Agostinho et al., 2011) propose to enhance information systems by traceability functionalities to enable sustainable interoperability. Using model morphisms that allow altering and non-altering model mapping, the relationship of models from different knowledge domains can be formalized. These mappings are translated in executable code in order to exchange mapped or transformed data between business partners. The work of Agostinho et Al. is closely related to the Knowledge Links due to the support of morphisms between different knowledge domains. Mappings are created on ontology level, therefore Knowledge Links could complement this work by allowing end users to model the executable connections with given tools for end users. By using Knowledge Links, the presented approach could be enhanced by reusability of mappings. The result of a mapping could further be used to define new mappings.

The approach presented in this paper combines ontology-driven, event-driven and model-driven approaches and is based on three pillars: A modular and hierarchical knowledge base, an event-driven architecture for collaborative environments and a model-driven knowledge link approach with a business process execution engine. In the following sections, we will take a look at these in turn.

3 KNOWLEDGE BASE STRUCTURE

The knowledge base structure is following the work of (Ben Abbs et al., 2012) about modular ontologies. Major advantages of the modularization approach that are of particular importance are knowledge reuse across application domains, distributed ontology engineering over different knowledge domains and effective management of ontology modules. Business partners in collaborative environments often have different applications running in their system. Business networks in which business partners have to comply to a centralized knowledge base and specific enterprise applications in order to collaborate are hard to establish and nearly impossible to maintain. Furthermore, they build barriers and obstacles for new business partners, which is fatal for highly dynamic environments. Therefore, modular ontologies allow business partners to rely on their own knowledge structures and maintain their own knowledge base while connecting their knowledge structure to a shared common denominator. Additionally, the modular approach allows business partners to structure their inner knowledge base similarly.

The knowledge base structures can be categorized in different hierarchy levels. The highest level is constituted by a common ontology structuring necessary knowledge about the business network. At the second hierarchy level the business partners' ontologies are located. Each business partner structures its knowledge separately. The third and lowest hierarchy level comprise the enterprise applications' ontologies. Figure 1 illustrates the hierarchy levels.

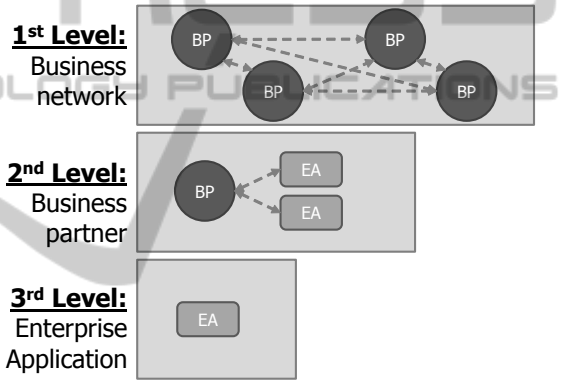

Figure 1: Knowledge Base Hierarchy.

Ontologies of lower levels inherit the ontologies of the next higher level and enrich the higher level concepts with new specific concepts or add specific concepts underneath higher level concepts. According to (Ben Abbs et al., 2012), this pattern of imports establishes an unidirectional correspondence between the lower level ontologies and the higher level ontology. More precisely, the highest level ontology constitutes an upper ontology for the underlying ontologies. According to (Healy et al., 2010), an upper ontology is a high-level, domain independent ontology whose concepts generic and basic and "from which more specific ontologies can be derived". Mid-level ontologies between the lowest and highest level of the ontology hierarchy ease the mapping between these levels, whereas the domain ontologies on the lowest level reuse the concepts from the levels above and extend them with domain specific concepts.

Upper ontologies are helpful in terms of interoperability. They are often used to describe generic, platform independent concepts in order to generalise the

meaning of domain specific concepts at a higher level. There are many initiatives that aim to define standard upper ontologies, for instance, the Suggested Upper Merged Ontology (Niles and Pease, 2001). Domain ontologies that are compliant with the upper ontologies are able to interoperate by shared terms and definitions and mapping to these terms.

In order to describe resources like documents, websites or parts of them in a computer-understandable manner, resources of enterprise applications need to be semantically annotated. In (Oren et al., 2006), annotations of various domains are analysed and a unified formal model for semantic annotations is presented. Using such an annotation model allows enterprise application resources to be structured and processable by our presented event driven knowledge link approach.

In our approach, we use Apache Jena (Foundation, 2014) as storage and interface for ontologies. Jena is a semantic web framework consisting of three modules. An OWL (W3C, 2012) and inference API allows interaction with ontologies and reasoning over data. The built-in triple store enables persistence and allows querying over SPARQL (W3C, 2008) and http. An RDF (W3C, 2014) API allows users to create, read and manipulate triples and handles serialization.

The knowledge base structure of the OSMOSE Platform follows the schema described above. Like depicted in Figure 2, the OSMOSE Upper-Ontology constitutes a common upper ontology for the whole platform at the highest hierarchical level. It contains generic concepts about events, entities, services and processes as well as platform specific concepts like 'OSMOSE World' or 'OSMOSE Process'. This common knowledge base is imported by each of the three worlds allowing the worlds to extend these concepts. The Intra-World Ontologies are ontologies specific for each of the world, acting as a mid-level ontology. Inner world applications use own ontologies by importing the Intra-World Ontology and adding specific concepts. Thus, they define the lowest level of ontologies, the domain ontologies. By semantic annotations, the inner-world applications can map their data to their ontologies.

To give a brief example, let us assume that the OSMOSE Upper-Ontology defines the concept of a generic entity. Inside the real world, the concept 'human actor' extends the generic 'entity' concept. For a physical machine having its own ontology, 'human actor' is extended by concepts for administrators and users.

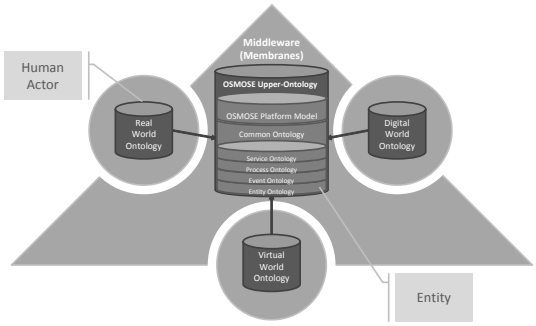

Figure 2: OSMOSE Knowledge Base Hierarchy.

4 EVENT-DRIVEN ARCHITECTURE APPROACH

An Event-Driven Architecture is a style of Service-Oriented Architecture, where the occurrence of an event can trigger the invocation of services, which in turn can generate new events. (Michelson, 2006) Event-driven systems are designed to process events as they occur, allowing the system to observe, react dynamically to and issue personalized data depending on the recipient and situation.(Etzion and Niblett, 2010). Event-driven architectures are deployed in many areas and have proofed as a competitive advantage for a lot of organizations. Thus, they are becoming increasingly popular. (Grabs and Lu, 2012)

Events can be seen as kind of messages that are generated by resources in domains and recognised by information systems. Events can occur and be converted from anywhere in an environment (Stojanovic et al., 2011), indicating hat an occurrence has been observed in a system (Grabs and Lu, 2012). Luckham defines events as objects that are records of activities in a system and can have a relation to other events with a timestamps as properties (Luckham, 2002). According to McGovern, "an event is a change in the state of a resource or request for processing" (McGovern et al., 2006).

Complex Event Processing (CEP) is a specialized field of event processing and part of event-driven architectures dealing with complex events. Complex events are combinations of multiple events consisting of all the activities that the aggregated events signify (Luckham, 2002) (Robins, 2010). Complex Event Processing deals with the identification and handling of interdependent events across organizations, analysing their impact and taking subsequent action in real time into account. It is an emerging technology which allows to find real time relationships between different events using elements such as timing, causality, and membership in a stream of data in order to extract relevant information (Perro-

chon et al., 2001). Luckham states that ”Complex Event Processing (CEP) is a defined as a set of tools and techniques for analysing and controlling the complex series of interrelated events that drive modern distributed information systems” (Luckham, 2008).

According to (Vernadat, 2007), loose coupling among services and applications is the key to building interoperable enterprise systems. Message-Oriented Middleware (MOM) or their extension, the Enterprise Service Bus (ESB), enable loose coupling using message queuing techniques. They implement the Event-Driven Architecture pattern and provide neutral message formats with which applications are able to exchange messages independently and asynchronously using simple transport protocols. Furthermore, it enables communication between different business units with heterogeneous platforms and environments.

Our approach utilizes the WSO2 Enterprise Service Bus (WSO2, 2014) together with the RabbitMQ Message Broker (Pivotal Software, 2014) in order to take advantage of concepts explained before. The WSO2 Enterprise Service Bus is characterized by WSO2 as the highest performance, lowest footprint, and most interoperable SOA and integrated middleware today. High connectivity and compliance to standards are additional key benefits. Additionally, it provides features for routing, mediation and transformation of messages as well as monitoring, security and service management features. The high-performance and lightweight ESB is advertised as highly available, scalable and stable. ESBs support distributed environments. Each business partner in a business network may have its own ESB running or even multiple ESBs that are connected to a main ESB of the business partner. The ESBs of each business partner are in turn connected to the business networks' ESB. RabbitMQ is capable of propagating messages sent between business partners. RabbitMQ is described as robust, highly available, easy to use and platform independent message broker. It supports a variety of protocols and allows flexible routing and clustering. In order to utilize Complex Event Processing, Esper (EsperTech, 2014) is used. Esper supports management of Data Windows for fine-grained event expiry, event series analysis, and different event representation types. Additionally, it is based on SQL standards, scalable and offers full API control. Additionally, it provides its own Domain Specific Language (DSL) called Event Processing Language (EPL). Additionally, we've added access to the knowledge base to combine the power of semantic web technologies with complex event processing. Following the work of (Schaaf et al., 2012), events and their relations to each other can be enriched by semantics in order to utilize knowledge inference about simple and complex events as well as everything that is related to this event, for instance the event publisher. Figure 3 illustrates the components of the middleware. Events that are sent have to proceed the middleware, i.e. the event channel. During this process, events and entities related to them are analysed by the Context Manager and the complex event processor, i.e. the Esper CEP Engine. The Context Manager encompass the Contect Store and the Reasoner component, implemented by Apache Jena. Additionally, the knowledge link engine, which will be explained in more detail, handles activation of modelled knowledge links.

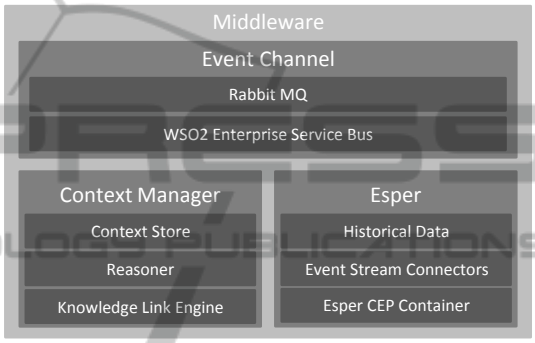

Figure 3: Middleware components.

WSO2 ESB, RabbitMQ and Esper are used together in the OSMOSE Project in order to allow communication and analysis about events between the real, digital and virtual world. Autonomous components deploy services in a WSO2 ESB. Thereby, they can react to events, call services or negotiate with other autonomous components. The communication is handled by RabbitMQ and analysed by the Esper CEP Engine.

5 KNOWLEDGE LINKS

In the previous two main sections, we defined the foundation of our approach for sustainable interoperability. We have defined a knowledge base concept with which structuring of knowledge from different knowledge domains of business partners is enabled. Thus, business partners can rely on their data structures that are connected to upper ontologies in terms of compliance to the business networks' generic structures. The event driven architecture approach enables business partners to connect their enterprise applications to an ESB in their environment, while those environments of all business partners can be connected together to a business network environment by utilizing ESB and message broker technologies.

In this section, the application of knowledge links is explained as a linkage of knowledge concepts between knowledge domains that runs on the event driven architecture. Due to the fact that knowledge links are used to link structured knowledge between enterprise applications and business partners, they are suitable to address interoperability concerns. Furthermore, they can be defined and maintained during runtime of the business network. Thus, interoperability can be sustained whenever business partners enter the environment or change enterprise applications.

Knowledge links are connections between knowledge concepts of different knowledge domains in a black box manner. Inside the black box, operators and transformations are applied to convert knowledge from one or more source domains appropriately to the target domain. They constitute a morphism of how knowledge in one domain can be constructed by using knowledge from other domains.

Knowledge Links are intended to be modelled by end users or domain experts. Modelling should be as easy as possible to reduce the effort to share knowledge to a minimum. Therefore, we've chosen a graphical way. In order to model knowledge links graphically, we implemented the Knowledge Link Configurator. This graphical modelling tool allows users to import knowledge structures from different knowledge domains and create knowledge links between them. Creating a new knowledge link or loading an existing one opens a graphical editor that allows dragging and dropping of imported knowledge concepts as well as operators from a palette. By drawing connections between dropped knowledge concepts and operators, knowledge links can be defined. Currently, we have defined a set of numeric (Sum, Subtraction, Average...), text (Concat...) and boolean (AND, OR, XOR...) operators. After the user finished with modelling, knowledge links can be stored and uploaded to the server. The uploaded knowledge links are further translated and pushed to the Knowledge Link Engine in the Context Store of the middleware.

In Figure 3, modelling of knowledge links with the Knowledge Link Configurator is illustrated. On the top right corner of the screenshot, the different knowledge base structures are imported. On the lower right side the knowledge link definitions can be found. Creating a new knowledge link definition will create a rule sheet on which the graphical modelling is done (squared area in the middle). The operators can be found on the left side. In the knowledge link on the screenshot, the set point of a machine part is subtracted from it's measurement in order to calculate the deviation of the machine part. The measurement and set point is taken from the real world machine and machine part, more precisely, from the real world ontology. The deviation is a digital world concept and thus described in the digital world ontology. The description of the knowledge link can be interpreted as follows:

1. First two nodes, 'measurement' and 'set point':
   Retrieve 'measurement' and 'set point'.
2. Subtraction node, 'SUB':
   Subtract the set point from the measurement.
3. Equals node, '==':
   Store the result ...
4. Last node, 'deviation':
   ...as 'deviation' in the ontology

The translation of knowledge links is done in two steps. First, the knowledge concepts of the source domain are identified that participate in the knowledge link. For all of those knowledge concepts, a listener in the CEP Engine is registered that listens to changes of the data behind that knowledge concept. It is assumed that for all changes to data that are relevant for the knowledge base semantic annotations are used and a change of data results in an event that is sent to the middleware having the knowledge concept as subject. Thus, when knowledge concepts of the source domain are recognized in events that represent changes of the data behind, the knowledge link will be executed in order to keep background consistency. After registering the listeners, an event automata is created that handles requests on middleware level. When requesting knowledge that is defined by a knowledge link, events for gathering knowledge from each domain and for calculation need to be executed.

For the latter case, we use business processes to perform actions modelled with palette operators. During the second step of translation, the modelled behaviour is translated into process-files that are associated with the knowledge link. If a listener recognises an event that is relevant for a knowledge link, the data from knowledge bases that refers to the knowledge link is requested by events and passed to the process. The process is built from predefined building blocks that correspond to the palette operators. For instance, the 'Subtraction' palette operator has a predefined building block for business processes. When used, the two values that should be subtracted are loaded by event requests from the appropriate knowledge bases and passed to the business process for calculation. After the business process has finished, the result is passed back to the event automata in order to proceed with another business process or to push the data in the appropriate knowledge base by using events. The translated process file knows about the execution order and the input parameters that are nec-

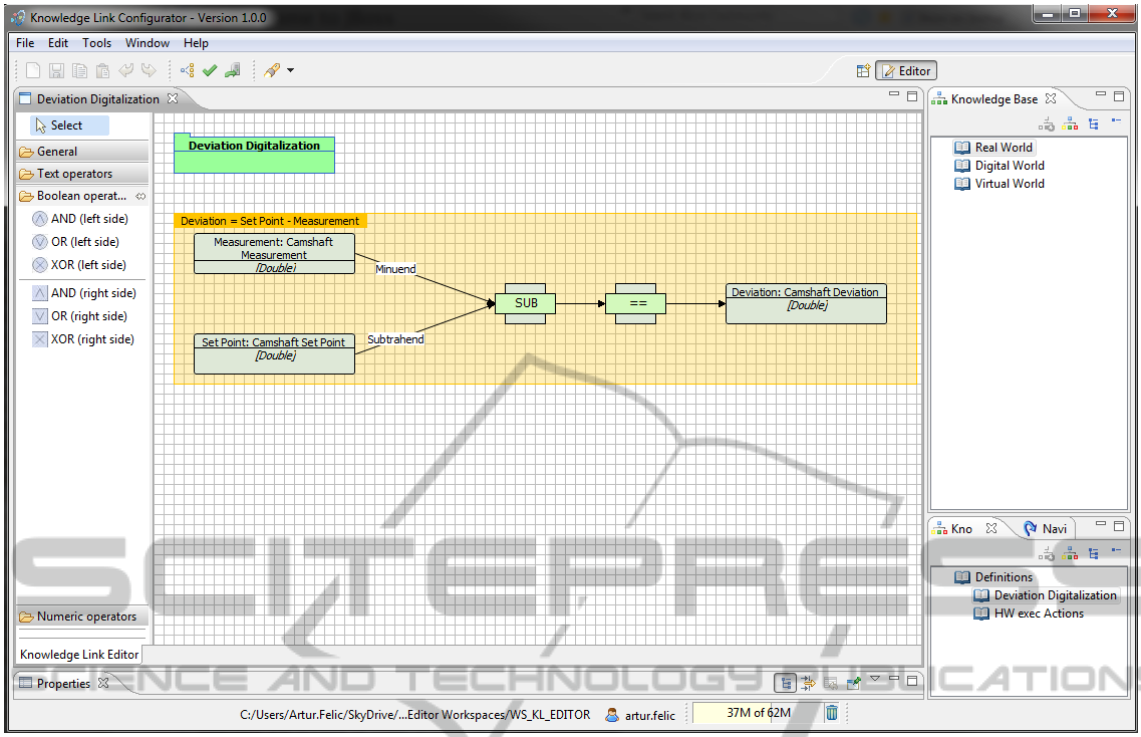

Figure 4: Knowledge Link Configurator.

essary. Because of the flexibility of the predefined building blocks for processes, sequences of them can be reused when similar parts of knowledge links are modelled.

5.1 Model-Driven Architecture Approach

Knowledge Links follow the Model-Driven Architecture (MDA) (Object Management Group, 2014a) approach by separating business and application logic from platform-specific software of dynamic business networks. According to (Truyen, 2006), MDA distinguishes between three different MDA Models. The Computation Independent Model (CIM) is also called business model and describes system expectations, i.e. the output of the system, independently from implementation. It bridges the gap between domain experts and information technologists. The Platform Independent Model (PIM) abstracts out technical details by defining appropriate services and exhibiting a sufficient degree of independence to enable mapping to platforms. Contrary, the Platform Specific Model (PSM) combines specifications of the PIM with platform specific details.

From their demand to their specification, modelling and translation, knowledge links undergo several model transformations between MDA Models. Figure 5 illustrates the different MDA Models and the transformations that are explained below. Vertical arrows represent CIM to PIM or PIM to PSM transformations respectively. Horizontal arrows represent CIM to CIM or rather PIM to PIM transformations. In some cases, one-to-one transformations may not be suitable. Instead, many-to-one, one-to-many or even many-to-many transformations will be applied. This is illustrated by multiple consecutively arranged boxes as input or output for the arrows in Figure 5.

5.1.1 Computation Independent Model (CIM)

Models of the CIM are business requirements that are expressed in natural language. They describe the knowledge demand that business partners have. Further, they can be transformed or consolidated to other, higher-level CIM models to express the knowledge demand inside the business network. These descriptions are computation independent.

For instance, a quality manager of a business partner inside a business network would like to know about the deviation of a measurement from a product component, which is manufactured by another business partner, from its set point. The access to the knowledge base could be restricted or the value that is demanded is not comprehensible. Therefore, the quality manager could express his or her demand in natural language. Other business partners could in

turn read the description and provide a solution for the demand and further rewrite the description.

Up to now, there is no appropriate automated methodology described to transform business requirements to PIM models. Thus, the transformation from CIM to PIM has to be done manually, i.e. with the Knowledge Link Configurator by knowledge link creation.

5.1.2 Platform Independent Model (PIM)

Knowledge links themselves are models of the PIM. Their description is platform independent and can be graphically modelled with the Knowledge Link Configurator. Furthermore, they are translated into other PIM models, the event automata and business processes (PIM to PIM transformation), after storing them on the platform. The transformation result is independent from enterprise applications dthat are implemented by business partners. Business processes and event descriptions are also platform independent and could be used by various engines or middleware components in different integrated systems.

By way of example, the quality manager in the scenario described above uses the Knowledge Link Configurator and creates a knowledge link between the concept of 'deviation' in the quality managers' knowledge domain and the 'measurement' and 'set point' concepts of a business partners' knowledge domain. After uploading this knowledge link, the translation algorithm generates an event automata that listens to changes of 'measurement' and 'set point' values and triggers business processes that are generated from building blocks if changes to these values happen or knowledge about the 'deviation' is requested.

5.1.3 Platform Specific Model (PSM)

During knowledge link translation, the PIM models are transformed to PSM models taking semantic annotations into account. Semantic annotations map the knowledge concepts of ontologies to real data like documents in different formats. Thus, knowledge gathering depends on the underlying platform and enterprise application that business partners are using. Additionally, code fragments like event listeners and event triggers are generated in order to control the knowledge and event flow of the knowledge link at the middleware level. Business processes are translated and stored into specific process files that can be executed by the business process execution engine.

In case of the quality manager that created and uploaded the 'deviation' knowledge link to the middleware, the business processes of the knowledge link is provided with input from the semantically annotated files that are maintained by the other business partners' enterprise applications. the event listeners and triggers are created according to the middleware technology that is implemented.

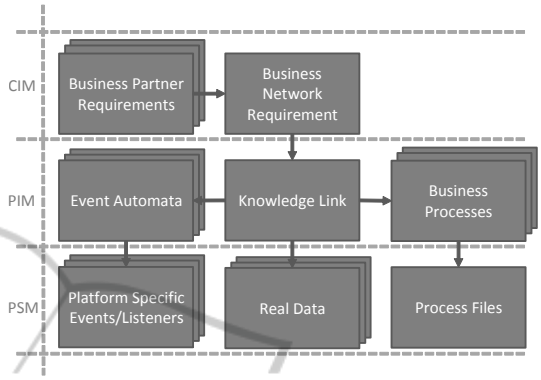

Figure 5: Model Transformations of Knowledge Links.

6 SUMMARY AND CONCLUSION

We combined semantic web technologies, event-driven and model-driven architecture approaches with knowledge links to propose a solution for sustainable interoperability in collaborative environments. Our approach is based on three pillars: (1) a hierarchical and modular knowledge base concept utilizing upper and modular ontologies, (2) an event-driven and message-message oriented middleware based on enterprise service busses that can be connected and (3) knowledge links, a model-driven approach with which domain ontologies can be linked in order to achieve enterprise interoperability. Sustainable interoperability is enabled due to the possibility to create and maintain knowledge links during runtime allowing to react to changes in collaborative environments by end users that are affected by these changes as they appear. Middleware technologies and a tool to create knowledge links have been presented. Consistency between knowledge bases of business partners is warranted automatically in the background. Obstacles are reduced due to the ability of business partners to define their own knowledge structures.

We exemplified our approach with use cases of the OSMOSE Project and will further evaluate our solution in this Project. The software components of the architecture presented in this paper are currently prototypical and will be evaluated after integration with the proof-of-concept scenarios of the OSMOSE Project.

ACKNOWLEDGEMENTS

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n°610905, the OSMOSE project.

REFERENCES

Agostinho, C., Sarraipa, J., Goncalves, D., and Jardim-Goncalves, R. (2011). Tuple-based semantic and structural mapping for a sustainable interoperability. In Camarinha-Matos, L. M., editor, *Technological Innovation for Sustainability*, volume 349 of *IFIP Advances in Information and Communication Technology*, pages 45–56. Springer Berlin Heidelberg, Berlin, Heidelberg.

ATHENA Project (2007). Da8.2: Guidelines and best practices for applying the athena interoperability framework to support sme participation in digital ecosystems. *http://www.asd-ssg.org/html/ATHENA/Deliverables/Deliverables*, Accessed 17.11.2014.

Ben Abbs, S., Scheuermann, A., Meilender, T., and D'Aquin, M. (2012). Characterizing modular ontologies. In *7th International Conference on Formal Ontologies in Information Systems - FOIS 2012*, pages 13–25, Graz, Austria.

Chen, D. and Daclin, N. (2006). Framework for enterprise interoperability. *Interoperability for Enterprise Software and Applications: Proceedings of the Workshops and the Doctorial Symposium of the Second IFAC/IFIP I-ESA International Conference: EI2N, WSI, IS-TSPQ 2006*, pages 77–88.

Chen, D., Doumeingts, G., and Vernadat, F. (2008). Architectures for enterprise integration and interoperability: Past, present and future. *Computers in Industry*, 59(7):647–659.

Chungoora, N., Young, R. I., Gunendran, G., Palmer, C., Usman, Z., Anjum, N. A., Cutting-Decelle, A.-F., Harding, J. A., and Case, K. (2013). A model-driven ontology approach for manufacturing system interoperability and knowledge sharing. *Computers in Industry*, 64(4):392–401.

Dassisti, M., Jardim-Goncalves, R., Molina, A., Noran, O., Panetto, H., and Zdravković, M. M. (2013). Sustainability and interoperability: Two facets of the same gold medal. In *On the Move to Meaningful Internet Systems: OTM 2013 Workshops*, volume 8186 of *Lecture Notes in Computer Science*, pages 250–261. Springer Berlin Heidelberg, Berlin, Heidelberg.

EsperTech (2014). Esper - complex event processing. *http://esper.codehaus.org/*, Accessed 30.11.2014.

Etzion, O. and Niblett, P. (2010). *Event Processing in Action*. Manning Publications Co, Greenwich, CT, USA, 1st edition.

Felic, A., König-Ries, B., and Klein, M. (2014). Process-oriented semantic knowledge management in product lifecycle management. *Procedia CIRP*, 25(0):361 – 368. 8th International Conference on Digital Enterprise Technology - DET 2014 Disruptive Innovation in Manufacturing Engineering towards the 4th Industrial Revolution.

Foundation, A. S. (2014). Apache jena - home. *https://jena.apache.org/*, Accessed 30.11.2014.

Fowler, M. (2003). *Patterns of enterprise application architecture*. The Addison-Wesley signature series. Addison-Wesley, Boston.

Grabs, T. and Lu, M. (2012). Measuring performance of complex event processing systems. In *Topics in Performance Evaluation, Measurement and Characterization*, pages 83–96. Springer.

Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220.

Healy, M., Poli, R., and Kameas, A. (2010). *Theory and Applications of Ontology: Computer Applications*. Springer.

IDABC, Echipa, and D. G. Industry (2004). European interoperability framework for pan-european egovernment services: Draft document as basis for eif 2.0. *European Communities*.

IEEE (1991). Ieee standard computer dictionary: A compilation of ieee standard computer glossaries. *IEEE Std 610*, pages 1–217.

Jardim-Goncalves, R., Popplewell, K., and Grilo, A. (2012). Sustainable interoperability: The future of internet based industrial enterprises. *Computers in Industry*, 63(8):731–738.

Kadar, M., Muntean, M., Cretan, A., and Jardim-Goncalves, R. (2013). A multi-agent based negotiation system for re-establishing enterprise interoperability in collaborative networked environments. In *2013 UKSim 15th International Conference on Computer Modelling and Simulation (UKSim)*, pages 190–195.

Lampathaki, F., Koussouris, S., Agostinho, C., Jardim-Goncalves, R., Charalabidis, Y., and Psarras, J. (2012). Infusing scientific foundations into enterprise interoperability. *Computers in Industry*, 63(8):858–866.

Luckham, D. (2002). *The power of events*, volume 204. Addison-Wesley Reading.

Luckham, D. (2008). *The power of events: An introduction to complex event processing in distributed enterprise systems*. Springer.

McGovern, J., Sims, O., Jain, A., and Little, M. (2006). Event-driven architecture. *Enterprise Service Oriented Architectures: Concepts, Challenges, Recommendations*, pages 317–355.

Michelson, B. M. (2006). Event-driven architecture overview. *Patricia Seybold Group*, 2.

Ni, Y. and Fan, Y. (2008). Ontology based cross-domain enterprises integration and interoperability. In *IEEE Congress on Services Part II (SERVICES-2)*, pages 133–140.

Niles, I. and Pease, A. (2001). Towards a standard upper ontology. In Guarino, N., Smith, B., and Welty, C., editors, *the international conference*, pages 2–9.

Object Management Group (2014a). Model driven architecture (mda). *http://www.omg.org/mda*, Accessed 17.11.2014.

Object Management Group (2014b). Object management group. *http://www.omg.org/*, Accessed 17.11.2014.

Oren, E., Möller, K. H., Scerri, S., Handschuh, S., and Sintek, M. (2006). What are semantic annotations? *http://www.siegfried-handschuh.net/pub/2006/whatissemannot2006.pdf*, Accessed 29.11.2014.

OSMOSE (2014). Osmose project - homepage. *http://www.osmose-project.eu/*, Accessed 29.11.2014.

Peristeras, V. and Tarabanis, K. A. (2006). The connection, communication, consolidation, collaboration interoperability framework (c4if) for information systems interoperability. *Ibis*, 1(1):61–72.

Perrochon, L., Kasriel, S., and Luckham, D. C. (2001). *Managing event processing networks*. Stanford University.

Pivotal Software, I. (2014). Rabbitmq - messaging that just works. *http://www.rabbitmq.com/*, Accessed 30.11.2014.

Robins, D. (2010). Complex event processing. In *Second International Workshop on Education Technology and Computer Science. Wuhan*.

Schaaf, M., Grivas, S. G., Ackermann, D., Diekmann, A., Koschel, A., and Astrova, I. (2012). Semantic complex event processing. *Recent Researches in Applied Information Science*, pages 38–43.

Singh, Y. and Sood, M. (2009). Model driven architecture: A perspective. In *2009 IEEE International Advance Computing Conference (IACC 2009)*, pages 1644–1652.

Stojanovic, N., Stojanovic, L., Anicic, D., Ma, J., Sen, S., and Stühmer, R. (2011). *Semantic complex event reasoning—beyond complex event processing*. Springer.

Truyen, F. (2006). The fast guide to model driven architecture. *Cephas Consulting Corp.*

Vernadat, F. B. (2007). Interoperable enterprise systems: Principles, concepts, and methods. *Annual Reviews in Control*, 31(1):137–145.

W3C (2004). Owl-s: Semantic markup for web services. *http://www.w3.org/Submission/OWL-S/*, Accessed 13.12.2014.

W3C (2006). Ontology driven architectures and potential uses of the semantic web in systems and software engineering. *http://www.w3.org/2001/sw/BestPractices/SE/ODA/*, Accessed 17.11.2014.

W3C (2008). Sparql query language for rdf. *http://www.w3.org/TR/rdf-sparql-query/*, Accessed 3.12.2014.

W3C (2012). Owl 2 web ontology language document overview (second edition). *http://www.w3.org/TR/owl2-overview/*, Accessed 3.12.2014.

W3C (2014). Rdf - semantic web standards. *http://www.w3.org/RDF/*, Accessed 3.12.2014.

WSO2 (2014). Wso2 enterprise service bus. *http://wso2.com/products/enterprise-service-bus/*, Accessed 30.11.2014.