Bridging Semantic Gaps Between Stakeholders in the Production Automation Domain with Ontology Areas

Thomas Moser, Wikan Danar Sunindyo, Stefan Biffl

Institute of Software Technology and Interactive Systems, Vienna University of Technology
Favoritenstrasse 9-11/188, Vienna, Austria
{thomas.moser, wikan.sunindyo, stefan.biffl}@tuwien.ac.at

Abstract—**Stakeholders from several domains with local terminologies have to work together to develop and operate software-intensive systems, like production automation systems. Ontologies support the translation between local terminologies via common domain concepts. Unfortunately, the ontology models can become large and complex if they include several aspects on a domain and some parts of the data model are volatile. In this paper, we propose a data modeling approach to support ontology users based on ontology building blocks, so-called "Ontology Areas" (OAs), which allow solving tasks with smaller parts of the overall ontology. We evaluate the proposed approach with use cases from the production automation domain: translation between stakeholder roles to support design-time and run-time decision making. Major result in the study context is that OAs improved the efficiency of data collection for decision making.**

I. Introduction

The integration of business processes and IT systems in homogeneous environments (i.e., consistent data formats and terminology) is supported by well-established approaches like data integration using Scheer's ARIS for CIM [20]. However, in more heterogeneous environments with a range of data formats and local terminologies like the production automation domain, typically stakeholders from several areas (e.g., business experts, software engineers and electrical engineers) work together to develop and operate software-intensive systems. A homogenization of these environments is often not achievable, if the stakeholders come from different organizational backgrounds or organizations change over time due to mergers and acquisitions. The precondition for successful semantic integration is a common understanding on the relevant concepts in the problem domain of the project.

An example for a collection of common problem domain concepts is the *Enterprise-Control System Integration*[1] (ECSI) standard [1] for developing automated interfaces between enterprise and control systems. The objectives of ECSI are to provide a) a consistent terminology as foundation for supplier and manufacturer communications, b) consistent information models, and c) consistent operations (process) models, which are the basis for clarifying application functionality and how information shall be used.

However, a standard like ECSI can only cover parts of the problem domain without getting too complex and hard to use. Further, many key players in the production automation domain currently do not follow this standard, which often hinders the cooperation of stakeholders in projects, since transformations between stakeholder terminologies to overcome semantic gaps between the stakeholders need to be conducted by scarce experts or carefully hand-crafted.

Ontologies are flexible open-world data models for knowledge representation, which store information in machine-understandable notation [10]. Therefore, ontologies can help to bridge semantic gaps between partial data models by providing mappings between them via common domain concepts. Ontologies usually capture problem-domain-specific information which can be reused later. Due to their concurrent development ontologies need to be checked for inconsistencies to stay useful. However, ontologies in practice usually have to combine several view points and thus get large and complex, particularly, if the ontology contains volatile domain elements, such as run-time data.

In this paper, we propose a data modelling approach that helps structure ontologies with ontology building blocks, so-called *"Ontology Areas"* (OAs). An OA is a meaningful part of an ontology for a stakeholder, which helps ontology users managing a complex ontology. The combination of all needed OAs represents the overall ontology for supporting the original engineering process.

We evaluate the proposed OA approach with use cases in the production automation domain: 1. Translation between local stakeholder terminologies; 2. Provision of design context for run-time data interpretation; and 3. Run-time measurement representation for design model improvements. The use cases are based on the data model of the "Simulator for Assembly Workshops" (SAW) [14] and compare the performance of an ontology with and without OAs. The evaluation showed that OAs made the data collection in the ontology for decision support more efficient in the study context, since the OAs result in a smaller ontology for the tasks in the use cases.

The remainder of this paper is structured as follows: Section 2 summarizes related work on system integration and ontologies. Section 3 describes the industry use case and Section 4 derives research issues. Section 5 introduces the OA approach, while Section 6 evaluates the approach and discusses the results. Finally, Section 7 concludes the paper and identifies further work.

II. Related Work

This section summarizes related work on system integration and ontologies for semantic integration to reconcile different views of stakeholders on system data.

[1] http://www.isa-95.com

A. Integration of Heterogeneous Systems

System integration is the task to combine a range of smaller systems to appear as one big system. There are several levels at which system integration could be performed [3], but there is so far no standardized integration process that explains how to integrate systems in general.

Typical integration solutions focus either on technical heterogeneity (how to connect systems that use different platforms or protocols) or on semantic heterogeneity (how to translate data in messages between systems that use different data formats or terminologies). In order to cope with technical heterogeneity on service level middleware technology [9] supports syntactical transformation between services, while the semantic heterogeneity of services can be addressed with a common data schema [12]. Limitations of these integration approaches are: 1. The need for a common data schema [12], which is hard and time-consuming to negotiate, sometimes impossible if stakeholders continue to disagree. 2. The need for integration over heterogeneous middleware technologies (with different APIs or network architecture styles) implies the development of static and therefore inflexible wrappers between each combination of middleware technologies, and thus increases the complexity of communication.

Semantic integration is defined as the solving of problems originating from the intent to share data across disparate and semantically heterogeneous data [12]. These problems include the matching of ontologies or schemas, the detection of duplicate entries, the reconciliation of inconsistencies, and the modelling of complex relations in different sources [19]. Over the last years, semantic integration became increasingly crucial to a variety of information-processing applications and has received much attention in the web, database, data-mining and AI communities [6]. One of the most important and most actively studied problems in semantic integration is establishing semantic correspondences (also called mappings) between vocabularies of different data sources [7].

B. Ontologies for Semantic Integration

An ontology is a representation vocabulary for a specific domain or subject matter, like production automation. More precisely, it is not the vocabulary as such that qualifies as an ontology, but the (domain-specific) concepts that the terms in the vocabulary are intended to capture [5]. Goh [11] identified three main categories of semantic heterogeneities in the context of data integration that can appear: confounding conflicts (e.g., equating concepts are actually different), scaling conflicts (e.g., using different units for the same concept), and naming conflicts (e.g., synonyms and homonyms).

Noy [18] identified three major dimensions of the application of ontologies for supporting semantic integration: the task of finding mappings (semi-)automatically, the declarative formal representation of these mappings, and reasoning using these mappings. There exist two major architectures for mapping discovery between ontologies: 1. It is possible to create a general upper ontology which is agreed upon by developers of different applications. Two examples for ontologies that are built specifically with the purpose of being formal top-level ontologies are the *Suggested Upper Merged Ontology* (SUMO) [17] and DOLCE [10]. 2. There are approaches comprising heuristics-based or machine learning techniques that use various characteristics of ontologies (e.g., structure, concepts, instances) to find mappings. These approaches are similar to approaches for mapping XML schemas or other structured data [4, 6]. The declarative formal representation of mappings is facilitated by the higher expressive power of ontology languages which provide the opportunity to represent mappings themselves in more expressive terms.

Uschold and Gruninger [22] identified four main categories of ontology application to provide a shared and common understanding of a domain that can be communicated between people and application systems [8]: Given the vast number of non-interoperable tools and formats, a given company or organization can benefit greatly by developing their own neutral ontology for authoring, and then developing translators from this ontology to the terminology required by the various target systems. While it is safe to assume there will not be global ontologies and formats agreed by all possible stakeholders, it is nevertheless possible to create an ontology to be used as a neutral interchange format for translating among various formats. There is a growing interest in the idea of "Ontology-Driven Software Engineering" in which an ontology of a given domain is created and used as a basis for specification and development of some software [19]. The benefits of ontology-based specification are best seen if there is a formal link between the ontology and the software. To facilitate search, an ontology is used as a structuring device for an information repository (e.g., documents, web pages, names of experts); this supports the organization and classification of repositories of information at a higher level of abstraction than is commonly used today.

As alternative approach for semantic integration of system models the infrastructure of Model-Driven Architecture (MDA) [15] provides architecture for creating models and meta-models, defining transformations between these models, and managing meta-data. Although the semantics of a model is structurally defined by its meta-model, the mechanisms to describe the semantics of the domain are rather limited compared to machine-understandable representations using, e.g., knowledge representation languages like RDF[2] or OWL[3]. In addition, MDA-based languages do not have a knowledge-based foundation to enable reasoning (e.g., for supporting quality assurance), which ontologies provide [2]. Beyond traditional data models, like UML class diagrams or entity relationship diagrams, ontologies provide methods for integrating fragmented data models into a common model without losing the notation and style of the individual models [13].

Seidenberg and Rector [21] proposed web ontology segmentation to counter decreasing ontology performance when ontology size increases. The algorithm to make ontology segmentation is similar to our approach, but we extend the usage of ontology areas for more stakeholders and volatilities.

[2] Resource Description Framework: http://www.w3.org/RDF/
[3] Web Ontology Language: http://www.w3.org/2007/OWL

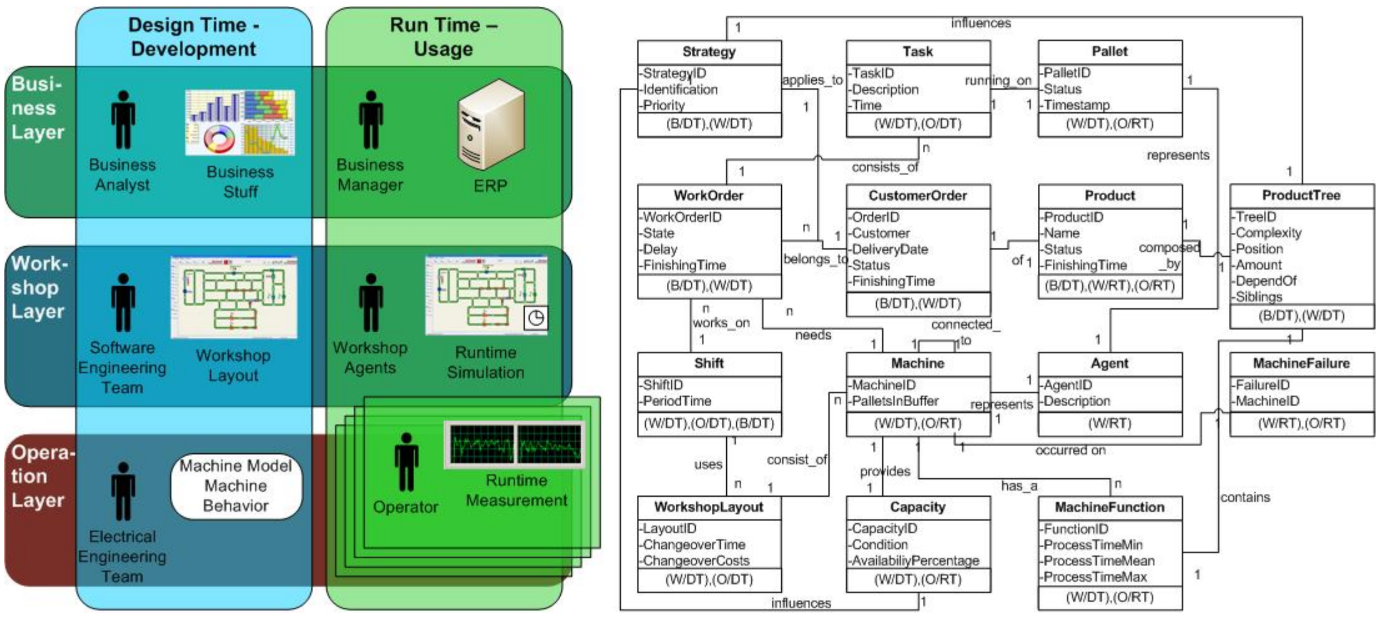

Figure 1: Sources of semantic gaps between stakeholders: domain layers, design-/run-time views; the data model contains common domain concepts to bridge semantic gaps.

III. Industry Use Case

In cooperation with industry partners in the production automation domain we conducted the project “Simulator for Assembly Workshops” (SAW) [14], which simulates complex reconfigurable production automation systems by scheduling sequences of transport and machine tasks over 100 times faster than the actual hardware[4]. The SAW simulator has been validated with real hardware components to ensure simulation validity for real-world production automation systems. In the SAW context stakeholders from different backgrounds work together and could benefit from better automated access to each others data models which is currently only possible via the stakeholders themselves as the data models are not well integrated.

Figure 1 illustrates sources of semantic gaps between stakeholders: stakeholder domain layers with different local terminologies; and design-/run-time views which are semantically not well connected. The data model, in our case an ontology model (the Engineering Knowledge Base (EKB) [16]), contains common domain concepts to bridge the semantic gaps between stakeholder terminologies and design-/run-time views.

The three stakeholder layers in Figure 1 are: a) the *business layer (B)* for production planning to fulfil customer orders by assigning optimal work orders to the workshop; b) the *workshop layer (W)* for coordinating the complex system of transport elements and machines to assemble smaller basic products into larger more comprehensive products according to the work orders; and c) the *operation layer (O)* for monitoring the individual transport system elements and machines to ensure their contributions to the workshop tasks. Those three layers are divided into two parts based on the time those layers worked on, namely design time (development) and run time (usage).

Figure 1 (right hand side) illustrates part of the data model that represents common domain concepts for the uses cases in UML-class-diagram style notation. The bottom box of each data element shows which stakeholder layer (B, W, and O) needs this data element to conduct their tasks and when: at Design Time (DT) or Run Time (RT).

From the SAW project we derived the following use cases that illustrate semantic gaps between stakeholders and how to overcome these gaps using ontology-based approaches.

UC-1. Translation between local stakeholder terminologies. The business manager on the business layer receives customer orders and schedules work tasks to the coordinator in the workshop layer. While they have a defined interface for exchanging work task information, they use local terminologies for concepts that are only occasionally needed to resolve scheduling issues, e.g., reference to specific customer orders if limited workshop capacity does not allow to fulfil all work tasks in a shift and negotiation on which tasks have higher priority are necessary to determine which customer orders will be fulfilled. Because the stakeholders use different terminologies, translations are necessary to automate references to customer orders between stakeholders in business and workshop layers.

UC-2. Design context for run-time data interpretation. The workshop operator at run time needs to resolve the meaning of multiple warnings from systems to determine his best next actions. Systems may be connected by design relationships like process, communication or energy networks, which could be evaluated automatically, if the run-time warnings were related to design-time knowledge, which is currently available from the operation experts but not in machine-understandable form like an engineering knowledge base.

[4] Automation & Control Institute; http://www.acin.tuwien.ac.at

UC-3. Run-time measurement data representation and analysis for design model improvements. If an engineering knowledge base is available to support run-time decisions with design knowledge, it is easy to also provide all kinds of run-time measurements linked to design elements, e.g., actual capacity of infrastructure, to iteratively improve the accuracy of design estimates with feedback from run time.

IV. Research Issues

The general idea of Ontology Areas (OAs) is to structure a comprehensive ontology into smaller building blocks with the following benefits for the designer and user of the ontology:

- A *smaller ontology* based on OAs that contains the minimal necessary knowledge for a specific task can be selected from a comprehensive ontology to facilitate more efficient use and change.
- We expect a smaller ontology (consisting of selected OAs) to exhibit *lower cognitive complexity* for designers who work with ontologies to make tools that support the automation of stakeholder tasks.
- Specific OAs can contain the more volatile ontology elements and thus make the design of the overall ontology *more stable against changes*.

As measurement criteria for evaluation we use the size of an ontology (and an OA) by counting the number of facts and relationships. In our study context the comprehensive ontology consists of: a) the production automation domain concepts (i.e., data model in Fig. 1) for design-time and run-time elements; and b) stakeholder extensions to the data model, such as local terminologies and mappings, for all stakeholders.

We used the following guidelines to design the OAs: a) concepts that a particular stakeholder (in business, workshop, or operation layer) needs to fulfil his typical tasks in order to achieve cohesiveness of the OAs; b) discern between common domain concepts and local add-ons of a stakeholder (such as terminology), which may change in different project contexts; c) keeping apart more stable design-time concepts from more volatile run-time concepts; and d) structuring volatile run-time data by manageable time intervals depending on the frequency of data elements' change. According to these guidelines examples for concrete OAs are: the design-time concepts of a business stakeholder and the run-time terminology of a workshop stakeholder.

From the use cases we derive the following research issues (RIs) to investigate the benefits of an ontology structured with OAs compared to an ontology without OAs.

UC-1. Translation between local stakeholder terminologies. The ontology supports each role by allowing to use their local terminology to communicate with other stakeholders. For this task sufficient OAs need to contain for the communicating stakeholders: the common domain concepts in their universe of discourse (see also in Fig. 1 the data elements and their link to associated stakeholders), local terminologies, mappings between local terminology elements and common domain concepts (on class level).

RI-1a: Compare the *complexity* (size) of the minimal ontology with OAs to the complexity of the overall ontology in the study context.

RI-1b: Compare the *efficiency* of the minimal ontology with OAs to the efficiency of the overall ontology in the study context to conduct the translation task.

The other use cases address benefits from making links between design-time and run-time data elements available at run time.

UC-2. Provision of design context for run-time data interpretation. While typical run-time applications do not have access to comprehensive design models on system level, there are benefits for tasks like planning and defect finding if in addition to the signals from individual machines there are also relationships from design time models available, such as information which machines are linked as they may be part of a common process, transport system neighbourhood, or information network branch.

RI-2: Determine the *minimal (cognitive) complexity of OAs* to support a specific task more efficiently, such as filtering failure warnings from machines that are connected by design model relationships in a non-obvious way. Compare the result with OAs to the (cognitive) complexity of using a comprehensive ontology.

UC-3. Run-time measurement data representation and analysis for design model improvements. In the study context the collection of run-time data points, e.g., on process characteristics and quality of service of the infrastructure, helps to provide data for future design improvements, e.g., for more realistic planning and more efficient system configurations. The designers and quality management personnel, who conduct the data analysis procedures, often do not know in advance precisely which analysis functions they will need. Thus, a considerable amount of raw data would be beneficial to store in the ontology for querying design-time relationships and run-time data together. Unfortunately, even moderate data collection (10 data points) at reasonable frequency (e.g., one measurement every second) leads over a shift of 8 hours to a number of run-time data elements that easily exceeds the size of the design-time data elements in the ontology.

OAs that are designed to hold all measurement instances of a data element in a certain time interval (e.g, one minute) allow to keep the complexity of the ontology needed for analysis manageable: Only the OAs that contain relevant run-time measurements for a given analysis need to be considered.

RI-3a: Determine the *minimal complexity of OAs* to support a specific data analysis task more efficiently, such as calculating process characteristics. Compare the result with OAs to the (cognitive) complexity of using a whole ontology.

RI-3b: Compare the *efficiency* of the minimal ontology with OAs to the efficiency of the overall ontology in the study context to conduct the data analysis task

V. Ontology Areas For Bridging Semantic Gaps

In this Section we explain in more detail how to address the use cases with an ontology that uses OAs as basis for the evaluation of the RIs in Section 6.

An ontology area is a subset of ontology as a building block that can solve a certain task. The ontology can be broken into ontology areas based on several aspects, for example by the time, volatility, layer and roles. Figure 1 shows the break down of ontology into several ontology areas based on the stakeholder layers (business, workshop, operation) and time when models are mostly used (design time and run time). Some parts of the data mode are much more volatile than others, e.g., run-time process measurements compared to design-time workshop layout. For example, each data point measured once a second in a shift that takes 8 hours produces around 30,000 data point instances, which need to be reduced by statistical methods or will take considerably storage space.

To make an OA from the whole ontology, we can follow this basic algorithm. First, define a task that is needed to be solved by the stakeholder. Second, find related classes for doing the task. Third, find classes that linked to the classes in step two. Fourth, drop other classes that are not needed and save as a new ontology. Also, we can reconstruct the whole ontology from the ontology areas, by merging them together into one ontology by using ontology tool like Protégé.

We illustrate in three use cases (UC-1 to UC-3) how OAs help reduce the complexity of the ontology for bridging semantic gaps in production automation systems.

UC-1. Translation between local stakeholder terminologies. The stakeholders of the production automation systems need to work together to achieve their goal. A common data schema is not possible because the stakeholders usually use different data formats, local terminologies and tools to access the data from the system. The ontology (EKB – Engineering Knowledge Base) plays a role as a common domain concept, where the local terminologies from the stakeholders will be mapped to. By mapping each local terminology to the ontology, the system can translate the local terminologies from one stakeholder to the other stakeholders. The translation could be the name of function, some names in the argument of the function, different data format, or the meaning of some parameters. However, the complexity of the ontology may increase when the number of the terminologies and the stakeholders is also increases, since the ontology should store all terminologies, the mappings and the common concepts.

By using the ontology areas, the stakeholder can take a small part of the ontology that he really cares and solving his task with the same results but less complexity than by using the full ontology. The example is illustrated on figure 2.

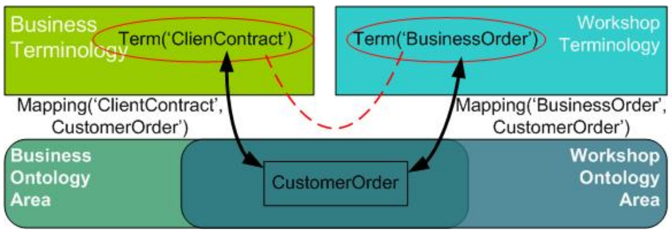

Figure 2. Translation between Business Terminology and Workshop Terminology.

The business stakeholder has a local terminology "`ClientContract`", while the workshop stakeholder has a local terminology "`BusinessOrder`". Both have a common concept to class `CustomerOrder` in the Ontology Areas. Then, both terminologies will be mapped to the class CustomerOrder as mention in Listing 1a.

Listing 1a. Mapping terminologies to the common concept.

```
mapping('ClientContract','CustomerOrder').
mapping('BusinessOrder','CustomerOrder').
```

From the mappings above, we can have a translation between two local terminologies by using rule on Listing 1b. The query and result can be seen on Listing 1c.

Listing 1b. Simple translation rules.

```
translate(Term1,Term2) :-
 mapping(Term1,CommonConcept),
 mapping(Term2,CommonConcept),
 not(Term1 = Term2).
```

Listing 1c. Translation result.

```
translate(X,Y).
X = 'ClientContract'
Y = 'BusinessOrder'
```

The translation is just one example for translations in general. OAs for this use case would just consider the parts of the ontologies for the stakeholders involved (see Figure 2): stakeholder concepts, their local terminologies and mappings, which can more easily be added to and removed from an ontology as stakeholders change in a particular context. The evaluation for this use case will be explained on section 6.

UC-2. Provision of design context for run-time data interpretation. The current models of the production automation systems primarily are used at design time and do no support run-time activities (e.g. GUI and reports) with a semantic description of exchanged data. The information at run time is usually local to a system part such as a machine and consists of efficient codes that are not easily understandable for non-experts and not connected to the information in other system parts or processes, which hinders the automatic analysis of local messages coming from several machines. For example the operator receives information on machine warnings and failures but can not automatically connect these messages with design-time information on the machine and the other machines and processes the machine is part of. Thus, the design time information can enrich the information at run-time to compare the outcomes of decision alternatives, e.g., which machine warning to address first.

By using an ontology, we can make connections between run-time information and design-time models automatically. OAs reduce the complexity of the ontology by taking a subset of the ontology, which is related to the stakeholder task. For example, the workshop stakeholder gets several machine failure messages from run-time and he wants to know whether there is any relationship between those failure messages. With design knowledge, the system can filter out redundant machine failure messages that would distract the workshop stakeholder. From the design knowledge, the workshop person knows how the machines are connected so it is possible to detect the original source of several machine failure messages.

The illustration is as follows. From the run-time information, the workshop person gets at least three machine failure messages as follows.

Listing 2a. Run-time machine failure messages.

```
MachineFailure(Failure/warning code, machine instance).
MachineFailure('FAIL041','MAC500').
MachineFailure('FAIL053','MAC610').
MachineFailure('FAIL057','MAC620').
```

To know the meaning and the relationship between the machine failure messages, he should check with the design time information, which contains such data like these.

Listing 2b. Machine instances.

```
Machine(machine instance, list of pallets in buffer).
Machine('MAC500',[P110, P130]).
Machine('MAC610',[P120]).
Machine('MAC620',[P140]).

ConnectedTo(machine instance, machine instance).
ConnectedTo('MAC500','MAC610').
ConnectedTo('MAC500','MAC620').
```

By using the rules below, he can find the original source of the machine failures. One machine is predecessor for another machine if it is connected to the other machine or a predecessor of the other machine, recurrently. In our example, we can identify the machine "MAC500' as the original source of the machine failure.

Listing 2c. Rules to find original source failures.

```
predecessor(X,Y) :- ConnectedTo(X,Y).
predecessor(X,Y) :- ConnectedTo(X,Z),
                    predecessor(Z,Y).
```

This is a simplified version of many real-world use cases that illustrates how the analysis of design time can support run-time decision making. In general, the OAs can help filtering certain run-time information that needed by the user and then connect to the related design time information. OAs in this case would provide the parts of the ontologies that are relevant for the machines and their relationships that the operator needs for his tasks. In Figure 1 the annotation of the data elements (bottom box for each data element) shows which stakeholder will need this data elements, which allows selecting the relevant data elements for the OAs of each stakeholder. The evaluation will be explained on section 6.

UC-3. Run-time measurement data representation and analysis for design model improvements. Run-time measurement information can be used to make design time information more accurate. Volatile information like run-time measurement can produce large amounts of data which would make a single ontology unnecessary large and slow down the performance of the ontology. The need for storing a high volume of run-time measurement data in the ontology occurs if the concrete future statistical analysis procedures are not known at the time of measurement.

Partitioning of the ontology in areas of similar volatility allows building partial ontologies for the task or query at hand. Run-time measurement at the frequency of 1 data point per second provides 30,000 data points of shift of 8 hours. If this is too much information for the ontology to hold, it is possible to define OAs for smaller time windows, which allow including the data for a certain time frame to be loaded into the ontology for data analysis as needed without exceeding the capacity of the ontology.

Semantic gaps between run-time measurement and design-time information occur when we have data elements from the interface of the machine at run time, but there is no machine-understandable documentation for the design of the interface. To solve this problem, we first give meaning to run-time data that are needed to be stored in the ontology and then provide a link from run-time to design-time semantics.

For example, to find out the maximum process time of certain machine functions, we can measure the process duration of that machine function in one shift, so we collect sufficient and still manageable data. The measurement result is an event named "process" that consists of the id, the batch number, status and timestamp of machine function. Listing 3a shows several measurement results that can be obtained by filtering run time data. The real data themselves is a very long list.

Listing 3a. Run-time event data with semantic annotation.

```
% process(machine function id, batch number, status, timestamp)
process('MF1','B-100','start',2009-02-03 T 10:01:06.01)
process('MF1','B-100','stop',2009-02-03 T 10:01:06.11)
process('MF2','A-200','start',2009-02-03 T 10:01:06.12)
process('MF1','B-101','start',2009-02-03 T 10:01:06.13)
process('MF1','B-101','stop',2009-02-03 T 10:01:06.21)
process('MF2','A-200','stop',2009-02-03 T 10:01:06.24)
```

To calculate the maximum process time of certain machine function, first we should calculate each process time by using predicate "process_time" to find the difference between the timestamp of "stop" status and the related timestamp of "start" status from the same machine function and batch number, and the keep it in the list using "list_of_process_time" predicate. Then with using the predicate "maxprocess" we will find the maximum value of process time of certain machine function (MFun) from the list of process time.

Listing 3b. Example analysis rule on run-time data.

```
max(X,Y,X) :- X >= Y.
max(X,Y,Y) :- X < Y.
maxlist([X],X).
maxlist([X,Y|Tail],Max) :-
maxlist([Y|Tail],MaxTail),max(X,MaxTail,Max).
process_time(MF,SN,T) :-
  process(MF,SN,start,X),
  process(MF,SN,stop,Y),
  T is Y - X.
list_of_process_time(List,MFun) :-
findall(T,(process_time(MF,SN,T),MF =
MFun),List).
maxprocess(MFun,T) :-
  list_of_process_time(List,MFun),
  maxlist(List,T).
```

For query, for example we want to know the maximum process time of 'MF1'. The result can be seen on Listing 3c.

Listing 3c. Result of data analysis.

```
maxprocess('MF1',T).
T = 0.1
```

The machine function entity in design time consists of the id and process time attributes. Usually the values of process time attributes come from estimation, but by using run-time measurement on process time, we can compare the previous design-time estimates to actual run-time data analysis for research on design improvements.

The illustrating example above is simple enough to conduct statistical analysis at run time, but for more complex statistical analyses, a solution for storing large amounts of data in an ontology may be necessary, which would inflate ontology size and decrease the ontology reasoning performance. OAs allow to manage stacks of run-time data elements and keep the size of ontology within well-performing capacity ranges.

VI. Evaluation and Discussion

We have implemented the OAs from the SAW ontology using Protégé 3.3.1. The SAW ontology consists of 24 classes and 3,000 instances from the simulation of production automation system. The evaluation will compare the measurement of the whole ontology and the ontology areas for three different use cases explained in section 5, as follows.

UC-1: Translation between local stakeholder terminologies. We compare the complexity (size) of the minimal ontology with OAs to the complexity of the overall ontology in the study context. For the minimal ontology with OAs, the business and workshop stakeholders have local terminologies of 300 and 400 words, respectively. Both need 100 words to communicate with each other. There are 200 to 700 data elements representing common knowledge, and 200 words for mapping from both local terminologies to the common concepts. Totally 1,100 to 1,600 entities are needed for the OAs.

Meanwhile, the comprehensive ontology for 6 stakeholders consists of around 1,800 words for local terminologies and around 300 words to communicate with each other. There are 1,600 words of common knowledge, and 600 to 1,800 words for mapping of all local terminologies to common concepts. In total, the comprehensive ontology consists of 4,200 to 5,400 words. In this case, OAs can reduce the ontology size to 20 to 30 % of the comprehensive ontology.

We can compare the efficiency of the minimal ontology with OAs to the efficiency of the whole ontology in conducting the translation task as follows. To produce 100 words of translation results from 200 words of mapping, the OAs needs 3 operators of query applying to those mapping.

The comprehensive ontology can produce more translations (300 words) with 3 operators of query as well. But the query should be applied to more mapping (600 to 1,800 words). With OAs we can reduce the size of mapping and make the operation faster.

UC-2: Design time context for run-time decision making. For evaluation we determine the minimal (cognitive) complexity of OAs to support a specific task more efficiently, such as filtering failure warnings from machines that are connected by design model relationships in a non-obvious way. After that, we will compare the result with OAs to the (cognitive) complexity of using a comprehensive complexity.

The cognitive complexity in this case is the number of data elements an operator needs to check in order to address his task. By implementing the OA to the comprehensive ontology, the operator can get only 3 classes to check, from 16 classes of the whole ontology. The size of ontology is also reduced, from 533 KB to 90 KB by applying the OA. So the operator will have smaller number of data elements to be connected in this case, by applying the OA.

UC-3: Run-time measurement and analysis for design improvement. For evaluation we will determine the minimal complexity of OAs to support a specific data analysis task more efficiently, such as calculating process characteristics. Then we will compare the result with OAs to the (cognitive) complexity using a comprehensive complexity.

In the OAs of the specific task, for 1 volatile entity the run-time measurement consists of 30,000 data points per shift. In the overall ontology, there may be many more, e.g., 300,000, data points in one shift. By using the OAs, the user can focus only on entity that he needs, and thus reduce the complexity of data handling considerably.

The efficiency of the minimal ontology with OAs is compared to the efficiency of the overall ontology in the case to conduct the data analysis task as follows. In the OA, to obtain 5 data points analysis, it needed to run 3 operators of query over 30,000 data points at one shift. Hence 18,000 operations on data points are needed to obtain one of the measurements.

In the whole ontology, to obtain 20 data points analysis, it needed to run 3 operators of query over 300,000 data points at one shift. Hence 45,000 operations on data points are needed to obtain one of the measurements. OA is notably more efficient than overall ontology.

Lesson learned. From the experiences with these use cases, we can learn the following lessons.

Building a smaller ontology for a task. As OAs allow focusing on the content of interest for a stakeholder task, we could show that the resulting ontology is considerable smaller. A smaller ontology is often also more efficient to handle and allows tackling tasks that use a particularly large number of data elements (e.g., run-time measurements in UC-3).

Focus stakeholders on relevant data elements. The combination of OAs, design-time, and run-time data elements allowed filtering relevant data elements for stakeholders, which would not be possible without the combination. Thus the OA approach helped lower the cognitive complexity for stakeholders by providing just the relevant subset of the comprehensive ontology.

Version management for ontology areas. With the OA concept we can flexibly build task-oriented ontologies based on different criteria (like volatileness, layers, roles). It is even possible to compare different versions of the same OA (e.g., production automation system designed with different parameter settings) to compare the run-time reactions to from changing design parameters. However, this ability also raises the need for better version management for OAs to ensure the building of consistent ontologies for specific tasks.

VII. Conclusion And Further Work

Ontologies support the translation between stakeholder local terminologies via common domain concepts, in our case production automation concepts. Typically, the ontology models become very large and complex compared to the basic data model (such as used in a data base to automate run-time processes) if they include several aspects on a domain and some parts of the data model are volatile. In this paper, we proposed a data modeling approach based on ontology building blocks, so-called “Ontology Areas” (OAs), which allow solving tasks with smaller parts of the overall ontology. We evaluated the proposed approach with use cases from the production automation domain. Major result in the study context is that OAs improved the efficiency of data collection task for decision making by lowering the cognitive complexity for designers and users of the ontology.

Further work. We see further research in the following directions. *Effort for OA design and use.* While OAs make a comprehensive ontology, which stores and uses engineering knowledge both at design time and run time, more manageable, their application needs the effort of designers for structuring the overall ontology and for building task-specific smaller ontologies. Thus we will conduct empirical studies on the effort needed to design and use ontologies with OAs.

Guidelines for the OA approach. While we found OAs useful to manage a large and complex ontology, we see the need for guidelines for the application the OA approach when designing a new ontology as well as for structuring already established ontologies with OAs to improve their performance.

Maintenance effort. Particularly for ontologies which should be changed by many users concurrently, we see a potential advantage of the concept of OAs, as areas with different rates of change can be easily separated, simplifying the checking of models for consistency etc. In the context of our case study this could be measuring the effort for typical changes, such as a new workshop layout, new machines, or new connections between machines.

Acknowledgment

We want to thank our colleagues at ACIN and TU Prague, for their feedback and inspiring discussions; and the SAW team at TU Wien for providing the application environment for the research use case.

References

[1] American National Standard, "Enterprise-Control System Integration," in *Part 1: Models and Terminology*. vol. ANSI/ISA-95.00.01-2000 North Carolina, USA: ISA (the Instrumentation, Systems, and Automation Society), 2000.

[2] K. Baclawski, M. K. Kokar, P. A. Kogut, L. Hart, J. Smith, J. Letkowski, and P. Emery, "Extending the Unified Modeling Language for Ontology Development," *International Journal of Software and Systems Modeling (SoSyM),* vol. 1, 2002.

[3] K. Balasubramanian, A. Gokhale, G. Karsai, J. Sztipanovits, and S. Neema, "Developing Applications Using Model-Driven Design Environments," *COMPUTER,* pp. 33-40, 2006.

[4] S. Bergamaschi, S. Castano, and M. Vincini, "Semantic integration of semistructured and structured data sources," *SIGMOD Rec.,* vol. 28, pp. 54-59, 1999.

[5] B. Chandrasekaran, J. R. Josephson, and V. R. Benjamins, "What are ontologies, and why do we need them?," *Intelligent Systems and Their Applications, IEEE [see also IEEE Intelligent Systems],* vol. 14, pp. 20-26, 1999.

[6] I. R. Cruz, X. Huiyong, and H. Feihong, "An ontology-based framework for XML semantic integration," in *International Database Engineering and Applications Symposium (IDEAS '04)*, 2004, pp. 217-226.

[7] A. Doan, N. F. Noy, and A. Y. Halevy, "Introduction to the special issue on semantic integration," *SIGMOD Rec.,* vol. 33, pp. 11-13, 2004.

[8] D. Fensel, *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce*: Springer, 2003.

[9] E. H. Gail, L. David, C. Jeromy, re, N. Fred, C. John, and N. Martin, "Application servers: one size fits all ... not?," in *Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, Anaheim, CA, USA, 2003.

[10] A. Gangemi, N. Guarino, C. Masolo, and A. Oltramari, "Sweetening WordNet with DOLCE," *AI Magazine,* 2003.

[11] C. H. Goh, "Representing and Reasoning about Semantic Conflicts in Heterogeneous Information Systems." MIT, 1996.

[12] A. Halevy, "Why your data won't mix," *Queue,* vol. 3, 2005.

[13] M. Hepp, P. De Leenheer, A. De Moor, and Y. Sure, *Ontology Management: Semantic Web, Semantic Web Services, and Business Applications*: Springer-Verlag, 2007.

[14] M. Merdan, T. Moser, D. Wahyudin, and S. Biffl, "Performance evaluation of workflow scheduling strategies considering transportation times and conveyor failures," in *International Conference on Industrial Engineering and Engineering Management (IEEM 2008)*, 2008, pp. 389-394.

[15] J. Miller and J. Mukerji, "Model Driven Architecture (MDA)," *Object Management Group, Draft Specification ormsc/2001-07-01, July,* vol. 9, 2001.

[16] T. Moser, A. Schatten, W. D. Sunindyo, and S. Biffl, "A Run-Time Engineering Knowledge Base for Reconfigurable Systems," Institute for Software Technology and Interactive Systems, Vienna University of Technology, Austria, Vienna 2009 http://www.isis.tuwien.ac.at/files/u290/A_Run-Time_EKB_-_techrep_VUT_2009.pdf.

[17] I. Niles and A. Pease, "Towards a standard upper ontology," in *2nd International Conference on Formal Ontology in Information Systems*, 2001, pp. 2-9.

[18] N. F. Noy, "Semantic integration: a survey of ontology-based approaches," *SIGMOD Rec.,* vol. 33, pp. 65-70, 2004.

[19] N. F. Noy, A. H. Doan, and A. Y. Halevy, "Semantic Integration," *AI Magazine,* vol. 26, pp. 7-10, 2005.

[20] A. W. Scheer, *Computer-Integrated Manufacturing*, 4th ed.: Springer, 1989.

[21] J. Seidenberg and A. Rector, "Web Ontology Segmentation: Analysis, Classification and Use," in *International World Wide Web Conference (WWW 2006)* Edinburgh, Scotland: ACM, 2006, p. 10.

[22] M. Uschold and M. Gruninger, "Ontologies and semantics for seamless connectivity," *SIGMOD Rec.,* vol. 33, pp. 58-64, 2004.