Alligator: A Deductive Approach for the Integration of Industry 4.0 Standards

Irlán Grangel-González[1,2(✉)], Diego Collarana[1,2], Lavdim Halilaj[1,2], Steffen Lohmann[2], Christoph Lange[1,2], María-Esther Vidal[1,2,3], and Sören Auer[1,2]

[1] Enterprise Information Systems (EIS), Computer Science, University of Bonn, Bonn, Germany
{grangel,collaran,halilaj,vidal,auer}@cs.uni-bonn.de
[2] Fraunhofer Institute for Intelligent Analysis and Information Systems (IAIS), Sankt Augustin, Germany
steffen.lohmann@iais.fraunhofer.de
[3] Universidad Simón Bolívar, Caracas, Venezuela

Abstract. Industry 4.0 standards, such as AutomationML, are used to specify properties of mechatronic elements in terms of views, such as electrical and mechanical views of a motor engine. These views have to be integrated in order to obtain a complete model of the artifact. Currently, the integration requires user knowledge to manually identify elements in the views that refer to the same element in the integrated model. Existing approaches are not able to scale up to large models where a potentially large number of conflicts may exist across the different views of an element. To overcome this limitation, we developed Alligator, a deductive rule-based system able to identify conflicts between AutomationML documents. We define a Datalog-based representation of the AutomationML input documents, and a set of rules for identifying conflicts. A deductive engine is used to resolve the conflicts, to merge the input documents and produce an integrated AutomationML document. Our empirical evaluation of the quality of Alligator against a benchmark of AutomationML documents suggest that Alligator accurately identifies various types of conflicts between AutomationML documents, and thus helps increasing the scalability, efficiency, and coherence of models for Industry 4.0 manufacturing environments.

Keywords: AutomationML · Semantic data integration · Industry 4.0

1 Introduction

In the engineering and manufacturing domain, there is an atmosphere of departure to a new era of digitized production, where traditional industrial engineering methods are synergistically combined with IT and internet technologies, such as cyber-physical systems, sensor networks, big data analytics, and semantic data integration. In different regions, initiatives in these directions are known

E. Blomqvist et al. (Eds.): EKAW 2016, LNAI 10024, pp. 272–287, 2016.
DOI: 10.1007/978-3-319-49004-5_18

under different names, such as *industrie du futur* in France, *industrial internet* in the US or *Industrie 4.0* in Germany. A core vision of these initiatives is to make manufacturing and production more flexible, efficient, and less error-prone by shifting more 'intelligence' to the edge. This shall be achieved by enabling sensors, devices, machines, and storage and transport equipments to directly communicate with each other. To realize this *Industry 4.0* vision, a vast variety of areas related to manufacturing, security, and machine communication need to interoperate by aligning their information models using domain-specific standards.

The *Automation Markup Language* (AutomationML or AML) for exchanging plant engineering information as specified by IEC 62714 [4,9,17,21] is one of the core standards of Industry 4.0. AutomationML can describe plant components and their sub-components from different perspectives, e.g., mechanical or electrical. A key challenge in such settings is *intra-standard interoperability*, i.e., the consistent integration of multiple pieces of information described in AutomationML. To overcome this challenge, we present Alligator, a deductive approach to integrate AutomationML specifications, and potentially similar document types.

We define an RDF-based representation of AutomationML input documents, aiming to resolve structural semantic inconsistencies, such as granularity of representations, schematic differences, and groupings and aggregations. Based on this semantic representation, we define a set of Datalog rules for identifying conflicts that generate structural semantic inconsistencies. A deductive engine is used to compute the conflicts from the Datalog representations. Conflict resolution is utilized to merge the input documents and produce an integrated AutomationML document.

By automatizing a crucial part of the engineering and modeling processes, Alligator addresses a key pillar of the Industry 4.0 vision. To the best of our knowledge, Alligator is the first comprehensive approach for automatically resolving the semantic ambiguity of AutomationML. As a result, the Alligator approach enhances scalability, efficiency, and coherence of models for Industry 4.0 manufacturing environments. Although our initial implementation and evaluation of the approach focuses on AutomationML, the approach is easily transferable to other Industry 4.0 standardization initiatives. We empirically evaluated the quality of Alligator against a benchmark of AutomationML documents. The evaluation results suggest that Alligator accurately identifies various types of conflicts between AutomationML documents.

In summary, this work makes the following contributions:

1. Alligator, a deductive approach, that combines Deductive Database and Semantic Web technologies for the integration of Industry 4.0 Standards.
2. A set of Datalog rules to characterize semantic heterogeneity types among AutomationML documents.
3. An empirical evaluation that reveals the effectiveness of Alligator during the integration of AutomationML documents.

The remainder of this paper is structured as follows. Section 2 motivates the problem with a concrete example. Section 3 gives an overview on the background

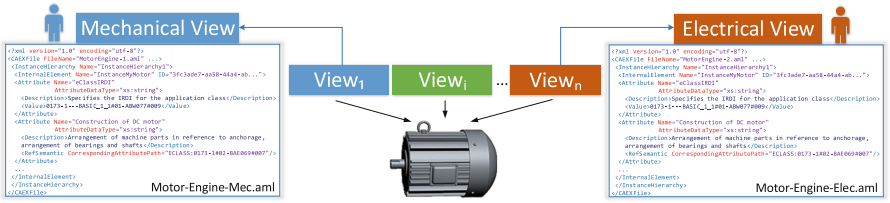


Fig. 1. Motivating example. Results of an engineering process where a motor engine is modeled from different views: a mechanical and an electrical view. Identical elements of the motor engine are defined as different elements in the views, resulting in conflicts between the views.

and introduces the terminology relevant to our approach. Section 4 presents the ALLIGATOR approach, which is evaluated in Sect. 6. Section 7 reviews related work. Section 8 concludes and gives an outlook to future work.

2 Motivating Example

A typical scenario in the mechatronic domain is data exchange between engineering tools during the modeling process. Engineering tools are utilized in different disciplines, such as mechanical and electrical engineering, or systems control. Figure 1 illustrates the results of an engineering modeling process where a *motor engine* is modeled from mechanical and electrical viewpoints. Mechanical engineers design the *motor engine* from the mechanical point of view, whereas electrical engineers model the electrical wiring topology inside the *motor engine*. *AutomationML* is utilized in both views to semantically describe the engine. However, because physical structures in these views are modeled with different properties, *conflicts* might arise when integrating these designs, thus inducing structural semantic inconsistencies.

Figure 2 details the mechanical and electrical views of the motor engine given in Fig. 1. The motor engine is identified as `0173-1#01-AKE162#012` **DC Engine** according to the eCl@ss product classification standard[1]. This reference enables the semantic description of the mechatronic component by pointing to the standard definition of a motor engine in eCl@ss. The AML document `Motor-Engine-Mec.aml` (cf. Fig. 2a) specifies the motor in terms of its construction form as a **DC Engine** (mechanical view). The AML element `RoleClassLib` (lines 2–23) comprises two AML elements `RoleClass`. The first `RoleClass` (lines 4–14) contains AML attributes with references to eCl@ss that semantically describe the engine according to the standard definition of version, classification in the eCl@ss catalog, and the International Registration Data Identifier (IRDI). The second `RoleClass` (lines 15–20) is composed of an AML attribute that

[1] http://www.eclasscontent.com/index.php?id=27022501&version=9_1&language=en&action=det.

defines the construction form of the **DC Engine**; `RefSemantic` (line 18) refers to the eCl@ss standard definition of this AML attribute (`0173-1#02-BAE069#007`).

Figure 2b depicts an AML document that aims at defining the same engine from the electrical viewpoint. As in the mechanical view, the first `RoleClass` (lines 4–14) semantically describes the engine using eCl@ss, while the second `RoleClass` (lines 15–20) defines not the engine as a whole, but a data cable *in* the engine. The `Attribute` in line 16 specifies the data cable and includes the semantic reference to eCl@ss (line 18).

Albeit the structural definition in these views of the **DC Engine** differs in the AML documents, the specification of AutomationML and its eCl@ss integration [19] imply that both descriptions are semantically equivalent. On one hand, the references to eCl@ss indicate that the AML elements between lines 4 and 14 in the two views correspond to the same element in the real world. For example, the specification of AutomationML states that two `RoleClass` elements are semantically equivalent whenever they share the same eCl@ss references for the AML attributes `eClassVersion`, `eClassClassification`, and `eClassIRDI` [19]. However, these views describe different real-world objects, and they should not be defined using `RoleClass` elements in the mechanical and electrical views which are considered semantically identical according to AML. Therefore, these elements are in *conflict*. Accordingly, there are five pairs of conflicting AML elements in this simplified example; each pair of these needs to be merged into one AML element in case the two views are integrated.

Currently, this integration is performed manually by experts, negatively affecting engineering processes. We present Alligator, a deductive framework that exploits the features of logic programming and the RDF data model for representing AML documents, as well as for detecting conflicts whenever AML documents are integrated.

3 Background

AutomationML. AutomationML (Automation Markup Language, IEC 62714) is a standard to exchange information about engineering tools, such as mechanical plant engineering, electrical design, or robot control. AutomationML provides an XML Schema, incorporating three different standards for describing real plant components [20]. At the top level there is the CAEX (IEC 62424) format for plant topology, storing hierarchical object information, properties, and libraries [8]. Secondly, the geometry (mechanical drawings) and kinematics (physical properties, such as force, speed, or torsion) are implemented with COLLADA [3]. Finally, the logic (sequencing, behavior, and control information) is implemented with PLCopen XML (IEC 61131).

AutomationML is built upon four main CAEX concepts: *RoleClassLibrary*, *SystemUnitClassLibrary*, *InterfaceClassLibrary*, and *InstanceHierarchy*. *RoleClassLibrary* specifies vendor independent requirements for the specification of system equipment objects; a *RoleClassLibrary* may comprise several *RoleClass*es, which provide role descriptions of a given class. Such descriptions aim

```
<CAEXFile FileName="Motor-Engine-Mec.aml" ...>
 <RoleClassLib Name="RoleClassLibDCEngine">
  <Version>1.0.0</Version>
  <RoleClass Name="eClassClassSpecification">
   <Attribute Name="eClassVersion">
    <Value>9.0</Value>
   </Attribute>
   <Attribute Name="eClassClassificationClass">
     <Value>27022501</Value>
   </Attribute>
   <Attribute Name="eClassIRDI">
    <Value>0173-1#01-AKE162#012</Value>
   </Attribute>
  </RoleClass>
  <RoleClass Name="BASIC_27-02-25-01 DC engine
   (IEC)">
   <Attribute Name="Construction form of DC motor">
    <Description>Arrangement of machine parts in
      reference to anchorage, arrangement of
      bearings  and shafts.</Description>
    <RefSemantic CorrespondingAttributePath=
     "ECLASS:0173-1#02-BAE069#007"/>
   </Attribute>
  </RoleClass>
 </RoleClassLib>
</CAEXFile>
```

(a) Mechanical View

```
<CAEXFile FileName="Motor-Engine-Elec.aml" ...>
 <RoleClassLib Name="RoleClassLibDCEngine">
  <Version>1.0.0</Version>
  <RoleClass Name="eClassClassSpecification">
   <Attribute Name="eClassVersion">
    <Value>9.0</Value>
   </Attribute>
   <Attribute Name="eClassClassificationClass">
     <Value>27022501</Value>
   </Attribute>
   <Attribute Name="eClassIRDI">
    <Value>0173-1#01-AKE162#012</Value>
   </Attribute>
  </RoleClass>
  <RoleClass Name="27-06 Cable, wire">
   <Attribute Name="Data cable">
    <Description>Type of cable for transmission
     of electrical or optical signals on the
     basis for the use in data transmission.
    </Description>
    <RefSemantic CorrespondingAttributePath=
     "ECLASS:0173-1#02-AKE197#013"/>
   </Attribute>
  </RoleClass>
 </RoleClassLib>
</CAEXFile>
```

(b) Electrical View

Fig. 2. Example of AML Documents. A motor engine is semantically described in terms of the eCl@ss standard. Role classes (highlighted in red) model the engine in terms of (a) a construction form and (b) a data cable. Elements of the same type (highlighted in yellow) correspond to conflicts between the views. (Color figure online)

at representing a physical or logical object, e.g., a motor or a robot. The *InterfaceClassLibrary* defines a set of interfaces to describe a plant model. First, it can define relations between the objects of a plant topology. Secondly, it can reference external information, e.g., a 3D description of a motor. The *InstanceHierarchy* describes the plant topology, and defines specific equipment for actual projects. Further, *Attributes* are used to define properties, e.g., length or size, of AML objects, e.g., RoleClasses or Internal Elements. In this paper, we focus on modeling topology information by means of the CAEX format.

AutomationML. Biffl et al. [4] and Kovalenko and Euzenat [11] have characterized mappings to deal with semantic heterogeneity in the engineering domain, and specifically in AutomationML. The authors have identified the following types of semantic heterogeneity: **(M1)** Value processing same properties are not modeled equally, e.g., using different datatypes; **(M2)** Granularity same objects are modeled at different levels of detail; **(M3)** Schematic differences differences in the way how semantics is represented for the same object; **(M4)** Conditional mappings relations between entities exist only if certain conditions occur; **(M5)** Bidirectional mappings relations between entities have to be defined bidirectionally; **(M6)** Grouping and aggregation different semantic modeling criteria are applied to group elements for the same object; and **(M7)** Restrictions on values mandatory values for properties in the object that have to be handled in the mapping process. As a proof of concept, we focus on semantic heterogeneity types, such as granularity (M2), schematic differences (M3), and grouping and aggregation (M6). We selected these types because they present major semantic

structural differences to describe similar objects. Additionally, they characterize semantic mappings between two AML elements that can be performed in two ways:

1. *Direct identification* considers two elements to refer to the same entity if the same identifier is used.
2. *Indirect identification* considers two elements to refer to the same entity if both refer to the same identity-providing elements from an external catalog, e.g., *RoleClass* or *Attributes*. For more complex structures as RoleClasses, it is assumed that if the combination of the eCl@ss IRDI, classification level, and version are equal, then the RoleClasses are considered to be the same.

AutomationML Vocabulary. Several approaches exist for adding semantics to the AutomationML language by means of ontologies [1,2,5,6,12,15]. With the exception of the AutomationML ontology[2], designed for the AutomationML Analyzer [16], none of the aforementioned ontologies covers all concepts given in the AutomationML schema. Additionally, they are not available on the web for consulting or querying. Crucial information for ALLIGATOR, such as the mapping with eCl@ass concepts, are not included in the AutomationML Analyzer vocabulary. Therefore, we have developed an RDFS vocabulary describing the main concepts of the AutomationML language.[3] Also, we have included concepts related to the integration with the eCl@ss standard.

4 Our Approach: ALLIGATOR

In this section, we present a formalization of AML documents, as well as the integration problems and proposed solution addressed by the ALLIGATOR approach. Finally, the architecture of ALLIGATOR is described in detail.

4.1 ALLIGATOR Representation of AML Documents

Definition 1 (Alligator Document). *An* ALLIGATOR *document is a tuple* $\Gamma = \langle \theta, V, F \rangle$ *such that* θ *is a set of URIs that identify AML elements,* V *is a set of properties in the AML vocabulary and* F *is an RDF graph composed of triples in* $\theta \times V \times (\theta \cup L)$ *where* L *is a set of literals.*

An ALLIGATOR document $\Gamma = \langle \theta, V, F \rangle$ can represent information from one or several AML documents D_i, where θ is the set of URIs that identify the AML elements in D_i, and the RDF graph F describes the relationships between the AML elements in D_i. In general, V can refer to different vocabularies, e.g., for other standards than AML such as OPC UA, but in this work, we focus on the AML vocabulary.

[2] http://data.ifs.tuwien.ac.at/aml/ontology#.
[3] https://w3id.org/i40/aml/.

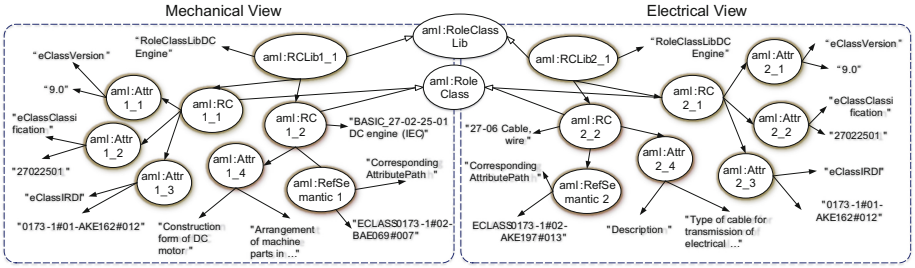
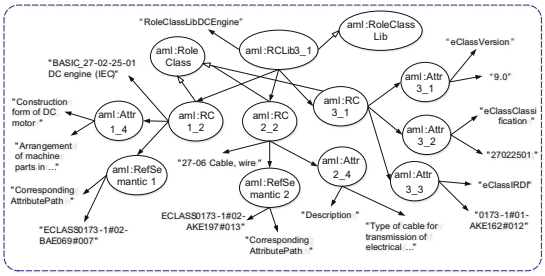
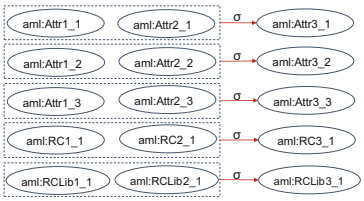


Fig. 3. RDF graph of an ALLIGATOR document. An RDF graph representing AML elements in the union of the mechanical and electrical views in Fig. 2



(a) Ideal RDF graph



(b) Homomorphism σ: from nodes in Figure 3 to ideal RDF graph nodes

Fig. 4. Ideal conflict-free ALLIGATOR document. (a) An RDF graph where there is only one RDF resource for the conflicting resources in the mechanical and electrical views of Fig. 6. (b) A homomorphism σ maps conflicting resources in the RDF graph in Fig. 3 to the same resource in the ideal RDF graph.

Example 1. Consider the RDF graph F_1 in Fig. 3. This graph comprises RDF resources representing the AutomationML elements in the mechanical and electrical views shown in Fig. 2; the AutomationML RDF vocabulary is used to describe these resources. An ALLIGATOR document $\Gamma_1 = \langle \theta_1, V, F_1 \rangle$ formally describes this RDF representation of the two views, where θ_1 is the set of the resources in F_1, and V is the AutomationML RDF vocabulary.

Definition 2 (Ideal Alligator Document). *Given an* ALLIGATOR *document* $\Gamma = \langle \theta, V, F \rangle$*, there is an ideal* ALLIGATOR *document* $\Gamma^* = \langle \theta^*, V, F^* \rangle$ *such that* Γ^* *comprises only conflict-free AML elements. Additionally, there is a homomorphism* $\sigma : \theta \rightarrow \theta^*$*. The RDF ideal graph* F^* *is defined as follows:*

$$F^* = \{(\sigma(s),\ p,\ \sigma(o)) \mid (s,\ p,\ o) \in F\}$$

Example 2. Consider the RDF graph in Fig. 4a. The ALLIGATOR document $\Gamma^* = \langle \theta^*, V, F^* \rangle$ describes this RDF graph, where θ is the set of RDF resources in the

graph, V is the AutomationML RDF vocabulary, and F^* is this RDF graph. Γ^* represents the ideal *conflict-free* ALLIGATOR document of Γ_1. Figure 4b shows a homomorphism σ that maps two conflicting resources in the RDF graph in Fig. 3 to the same resource in Fig. 4a.

Definition 3. *Consider an* ALLIGATOR *document* $\Gamma = \langle \theta, V, F \rangle$*, an ideal conflict-free* ALLIGATOR *document* $\Gamma^* = \langle \theta^*, V, F^* \rangle$*, and a homomorphism* $\sigma : \theta \rightarrow \theta^*$*. A set of conflicts in* Γ *with respect to* Γ^* *and* σ*, conflicts*$(\Gamma \mid \Gamma^*, \sigma)$*, corresponds to the set of AML element pairs* (E_i, E_j) *in* $\theta \times \theta$ *such that* E_i *and* E_j *are different but that* σ *maps to the same target AML element in* θ^**:*

$$conflicts(\Gamma \mid \Gamma^*, \sigma) = \{(E_i, E_j) \mid E_i, E_j \in \theta \text{ and } E_i \neq E_j \text{ and } \sigma(E_i) = \sigma(E_j)\}$$

Example 3. Given ALLIGATOR documents Γ_1 and Γ^* from Examples 1 and 2, and the homomorphism σ in Fig. 4b. The set of conflicts$(\Gamma_1 \mid \Gamma^*, \sigma)$ corresponds to the set of pairs of RDF resources in the RDF graph of Fig. 3 that σ maps to the same resource in the ideal RDF graph (Fig. 4b).

4.2 Problem Definition and Proposed Solution

Given an ALLIGATOR document $\Gamma = \langle \theta, V, F \rangle$, the *AML Conflict Identification* problem determines if a pair (E_k, E_l) of AML elements in θ is conflicting.

Definition 4. *Consider an* ALLIGATOR *document* $\Gamma = \langle \theta, V, F \rangle$*, an ideal conflict-free* ALLIGATOR *document* $\Gamma^* = \langle \theta^*, V, F^* \rangle$*, and a homomorphism* $\sigma : \theta \rightarrow \theta^*$*. The* AML Conflict Identification *problem corresponds to the problem of deciding if* $(E_k, E_l) \in \theta \times \theta$ *belongs to conflicts*$(\Gamma \mid \Gamma^*, \sigma)$*.*

Solving the *AML Conflict Identification* problem requires the existence of the ideal conflict-free AML document Γ^* and the homomorphism σ. However, in practice neither Γ^* and σ is known, and ALLIGATOR computes an approximation of the problem. We use $SC(\Gamma)$ to refer to the set of pairs (E_k, E_l) that correspond to the solutions of this problem. Once a set $SC(\Gamma)$ of conflicting AML elements in F is identified as the solution of the AML *Conflict Identification problem*, the problem of *AML Conflict Resolution* corresponds to the problem of creating an ALLIGATOR document where conflicts in $SC(\Gamma)$ are solved.

Definition 5. *Consider an* ALLIGATOR *document* $\Gamma = \langle \theta, V, F \rangle$ *and a set* $SC(\Gamma)$ *of pairs of conflicting AML elements in* F*. The problem of* AML Conflict Resolution *corresponds to the problem of creating an* ALLIGATOR *document* $\Gamma' = \langle \theta', V, F' \rangle$ *and a homomorphism* $\sigma' : \theta \rightarrow \theta'$*, such that:*

- *For each* (E_i, E_j) *in* $SC(\Gamma)$*, there is an AML element* E_m *in* θ' *such that* $\sigma'(E_i) = \sigma'(E_j) = E_m$*.*
- $F' = \{(\sigma'(s),\ p,\ \sigma'(o)) \mid (s,\ p,\ o) \in F\}$*.*

Γ' *represents the* ALLIGATOR *document where pairs of AML elements in* $SC(\Gamma)$ *are represented as one RDF AML element.*

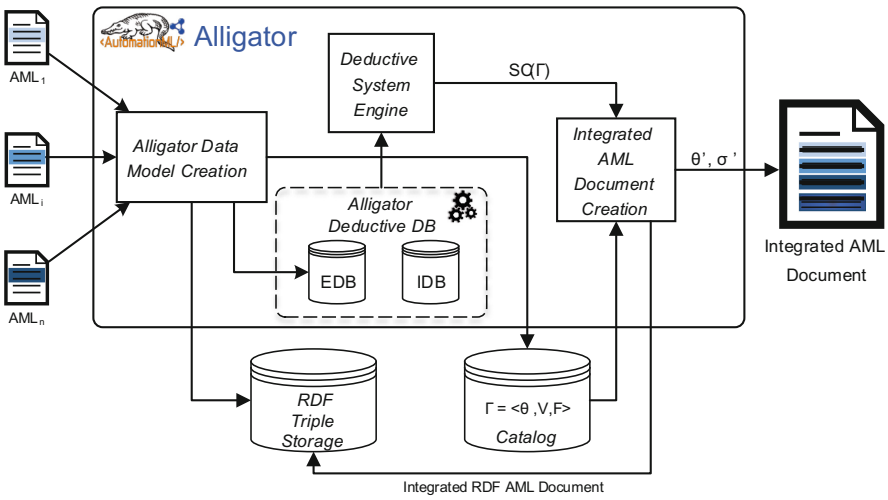


Fig. 5. The ALLIGATOR Architecture. ALLIGATOR receives AML documents and creates an integrated AML document. AML documents are represented as RDF graphs and Datalog predicates (EDB); Datalog intentional rules (IDB) characterize semantic heterogeneity types. A bottom-up evaluation of the Datalog program identifies conflicts between AML documents

We developed ALLIGATOR, an integration tool that relies on deductive database techniques for solving the problems of *AML Conflict Identification* and *AML Conflict Resolution*. Figure 5 depicts the architectural components of ALLIGATOR. Given a set of AML documents, the *Alligator Data Model Creation* component generates an ALLIGATOR document $\Gamma = \langle \theta, V, F \rangle$ that formally describes the union of these input AML documents. Additionally, a set of Datalog extensional facts (EDB) representing the triples in the RDF document F is created. The *Deductive System Engine* relies on the set of Datalog intentional rules (IDB) to compute the set $SC(\Gamma)$ from the Datalog representation of Γ. The set of Datalog intentional rules (IDB) defines different types of semantic heterogeneity that can occur among AML documents that correspond to views of the same mechatronic object definition. $SC(\Gamma)$ is computed as the least minimal fixpoint of the Datalog rules in IDB and the facts in EDB. Further, $SC(\Gamma)$ is utilized by the *Integrated AML Document Creation* component to solve the *AML Conflict Resolution* problem, and to produce an integrated AML document where RDF AML elements in $SC(\Gamma)$ are integrated as one AML element.

4.3 ALLIGATOR Data Model and Deductive System Engine

ALLIGATOR represents AML documents as RDF graphs. AML documents are translated to RDF using Krextor [13], an XSLT-based framework for converting XML to RDF. The RDF AML vocabulary is used to describe AML elements and relations. Further, AML documents are modeled as facts in an extensional

database (EDB) of a Datalog program P; for each type of AML element in the AutomationML standard exists an extensional Datalog predicate in P. Rules in the intensional database (IDB) of the Datalog program P characterize types of semantic heterogeneity. Intensional Datalog predicates represent conflicts that can exist between the different AML elements according to the types of semantic heterogeneity. The ALLIGATOR *Deductive System Engine* performs a bottom-up evaluation of P following a semi-naïve algorithm that stops when the least fixed-point is reached [7]. The intensional predicates inferred in the evaluation of P correspond to the pairs of conflicts in the set $SC(\Gamma)$.

5 ALLIGATOR rule-based representation of AutomationML Semantic Heterogeneity

One of the key innovations of ALLIGATOR revolves on the use of a Datalog-rule approach to effectively solve types of semantic heterogeneity. We have developed a set of rules covering the main characteristics of AML. Regarding the attributes, it is possible to determine that, if two attributes refer to the same eCl@ss value, i.e., eCl@ss IRDI, it can be assumed that their semantic meaning is the same. In detail, the AML element `refSemantic` refers to the eCl@ss IRDI using `CorrespondingAttributePath` (cf. Fig. 2 line 18). Thereby, even if two attributes are defined with different names, e.g., *Length* and *StrictLength*, they can still be semantically equivalent whenever they are linked to the same IRDI reference. It is important to remark that these rules have been defined taking into account the AML vocabulary properties. Based on this, the rule in Listing 1 states when two attributes are semantically equivalent.

```
1  sameAttribute(X,Y)    :- hasRefSemantic(X,T) & hasRefSemantic(Y,Z) &
2                           sameRefSemantic(T,Z).
3  sameRefSemantic(X,Y) :- hasCorrespondingAttributePath(X,Z) &
4                           hasCorrespondingAttributePath(Y,Z).
```

Listing 1. Rule 1: Semantic equivalence of two AML attributes

To determine that two `RoleClasses` are semantically equivalent according to their reference to eCl@ss, they have to contain the same version, classification, and IRDI. Based on these three conditions, **Rule 2** (cf. Listing 2) defines two semantically equivalent `RoleClasses`.

```
1  sameRoleClass(X,Y) :- type(X,roleClass) & type(Y,roleClass) & sameEClassIRDI(A,B)&
2                        sameEClassClassification(C,D) & sameEClassVersion(E,F)&
3                        hasAttribute(X,A) & hasAttribute(X,C) & hasAttribute(X,E)&
4                        hasAttribute(Y,B) & hasAttribute(Y,D) & hasAttribute(Y,F).
```

Listing 2. Rule 2: Semantic equivalence of two RoleClasses

Rule 2 relies on simpler rules such as **Rule 3** (cf. Listing 3), which defines the equivalence of two `eClassIRDI` attributes. Similarly, we have defined rules to decide if two values of `eClassVersion` and `eClassClassification` are the same.

```
1  sameEClassIRDI(X,Y) :- hasAttributeName(X,'eClassIRDI') &
2                         hasAttributeName(Y,'eClassIRDI') &
3                         hasAttributeValue(X,Z) & hasAttributeValue(Y,Z).
```

Listing 3. Rule 3: Semantic equivalence of two eClassIRDI AML attributes

These three rules are only examples of the type of rules implemented in ALLIGATOR; the complete set of rules is given on GitHub[4].

6 Empirical Evaluation

We studied the effectiveness of ALLIGATOR in the solution of the problems of *AML Conflict Identification* and *AML Conflict Resolution*. In particular, we assessed the following research questions: (RQ1) Is ALLIGATOR able to identify pairs of conflicting AML elements in AML documents?; (RQ2) Does ALLIGATOR exhibit *equal behavior* whenever different types of semantic heterogeneity occur during the integration of AML documents? The experimental configuration to evaluate these research questions was as follows:

Testbeds. Testbeds were based on the semantic mapping types M2 (granularity), M3 (schematic differences), and M6 (grouping and aggregation), with ten testbeds for each of them, respectively. First, a seed (AML document) was manually created for each testbed according to the type of semantic mapping. Next, we automatically generated two AML documents derived from this seed containing a random number of conflicting AML elements[5]. The generation was performed following a uniform distribution. Testbeds corresponded to pairs of AML documents, and thirty testbeds were evaluated in the study[6].

Gold Standard. To compile a Gold Standard, we relied on the generated testbeds. Formally, the Gold Standard corresponds to an ideal conflict-free ALLIGATOR document $\Gamma^* = \langle \theta^*, V, F^* \rangle$, for each pair of the AML documents in the testbeds. The creation of the conflict-free document as well as the computation of the conflicting elements and different elements was performed manually.

Metrics. We measured the *behavior* of ALLIGATOR in terms of the following metrics:

(a) **Precision** is the fraction of the conflicts identified by ALLIGATOR (i.e., $SC(\Gamma)$) that are conflicts in an AML document (i.e., conflicts$(\Gamma \mid \Gamma^*, \sigma)$).

$$Precision = \frac{|SC(\Gamma) \cap conflicts(\Gamma \mid \Gamma^*, \sigma)|}{|SC(\Gamma)|}$$

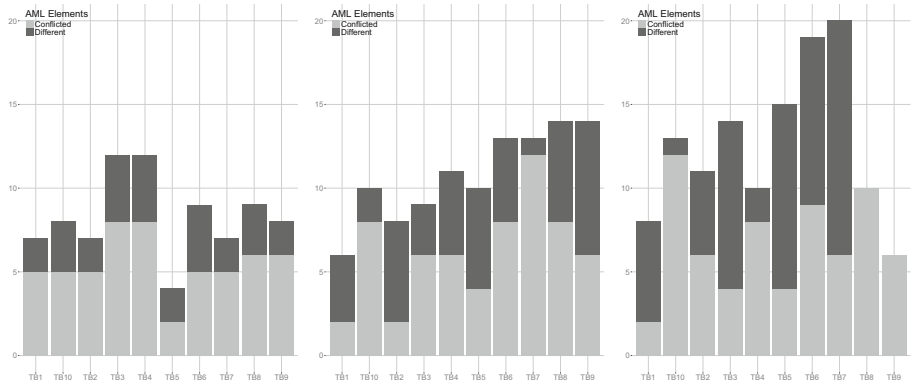
[4] https://github.com/i40-Tools/AlligatorRules.
[5] https://github.com/i40-Tools/AMLGoldStandardGenerator.
[6] https://github.com/i40-Tools/HeterogeneityExampleData.

(b) **Recall** is the fraction of the conflicts in an AML document (i.e., conflicts($\Gamma \mid \Gamma^*, \sigma$) that are identified by ALLIGATOR (i.e., $SC(\Gamma)$).

$$Recall = \frac{|SC(\Gamma) \cap conflicts(\Gamma \mid \Gamma^*, \sigma)|}{|conflicts(\Gamma \mid \Gamma^*, \sigma)|}$$

(c) **F-measure** is the harmonic mean of *Precision and Recall.*

Implementation. Experiments were run on a Windows 8 machine with an Intel I7-4710HQ 2.5 GHz CPU and 8 GB 1333 MHz DDR3 RAM. We implemented the *Deductive System Engine* as a meta-interpreter in Prolog that follows the semi-naïve bottom-up evaluation of Datalog programs [7]; we utilized SWI-Prolog version 7.2.3 and the Prolog Development Tool (PDT[7]). An AML extraction module was developed as a part of Krextor to transform AML documents into RDF graphs. This module comprised a set of mapping rules[8] that are executed in Krextor to create RDF graphs using the AML vocabulary. Further, the transformation of the RDF files into Datalog extensional predicates was implemented in Java 1.8. The ALLIGATOR framework, the testbed generator, and the testbeds evaluated in this experiment are publicly available on GitHub[9].



(a) Granularity (M2) (b) Schematic (M3) (c) Grouping (M6)

Fig. 6. Size of Integrated AML Documents. Per type of semantic heterogeneity: Granularity (M2), Schematic (M3), and Grouping (M6), the size of the integrated AML documents was reported in terms of the number of conflicts solved (light grey bars), and the different AML elements in the document (dark grey bars). In all the evaluated testbeds, the solved conflicts comprised at least 25 % of the total number of AML elements in the AML document, showing the heterogeneity of the evaluated testbeds

[7] https://sewiki.iai.uni-bonn.de/research/pdt/docs/start.
[8] https://raw.githubusercontent.com/EIS-Bonn/krextor/master/src/xslt/extract/aml.xsl.
[9] https://github.com/i40-Tools/.

Size of the Integrated AML Documents. The goal of this evaluation was to analyze the size of the integrated AML documents with respect to conflicting and different elements. For each type of semantic heterogeneity and testbed of that type, we computed the number of conflicts solved by ALLIGATOR. Further, the number of different AML elements was measured; a different AML element corresponded to an element that appeared in one of the AML documents in the testbed, and was not conflicting with any other AML element. For example, the AML elements in line 15 of the two views in Figs. 2a and 2b are different elements. In consequence, both should be included in the integrated AML document. On the other hand, the AML elements in lines 2, 4, 6, 9, and 12 in both views are pair-wise conflicted AML elements, and each pair should be integrated into only one AML element. Figure 2 reports on the number of *conflicted* and *different* AML elements. We observed that a large number of AML elements in the integrated AML documents result from solving the *Conflict Resolution* problem; being the number of these AML elements at least 25 % of the total elements in the integrated documents. These results illustrated the complexity of the evaluated testbeds, and clearly showed the enhancement assessed by ALLIGATOR during the integration of AML documents.

Effectiveness of ALLIGATOR. The goal of this experiment was to answer our research questions **RQ1** and **RQ2**. ALLIGATOR was run on each of the 30 testbeds to create $SC(\Gamma)$, and precision, recall, and F-measure were computed according to the Gold Standard ($conflicts(\Gamma \mid \Gamma^*, \sigma)$). Table 1 reports on the values of these metrics for each type of semantic heterogeneity, i.e., M2, M3, and M6. We observed that for these semantic heterogeneity types, the value for precision is 1.0, i.e., ALLIGATOR correctly detected all the conflicting elements in $conflicts(\Gamma \mid \Gamma^*, \sigma)$. Further, recall and F-measure are also 1.0 in the testbeds of semantic heterogeneity M2. These results suggest that ALLIGATOR rules capture the knowledge required to *accurately* solve the *AML Conflict Identification* problem. For the semantic heterogeneity types M3 and M6, ALLIGATOR rules are not completely covering all possible conflicts generated between *nested structures* composed of conflicting AML elements. Thus, ALLIGATOR could not identify at most two conflicts in five out of 20 testbeds of type M3 and M6. These results allowed us to positively answer research questions **RQ1** and **RQ2**.

7 Related Work

In the literature, many different approaches are proposed for integrating CAEX documents. In [18], a tool to map two CAEX files is presented. It allows to integrate the AutomationML documents, their respective descriptions, and the modified parts of one file into the other. Further, a mapping algorithm for CAEX files is presented. Nevertheless, the process of mapping is performed manually. Himmler [10] presents a framework to create standardized application interfaces in plant engineering based on AutomationML. The work provides a function-based based standardization framework for the plant engineering domain.

Table 1. Effectiveness of ALLIGATOR. Per semantic heterogeneity type, the effectiveness of ALLIGATOR is reported. In all the testbeds, precision is 1.0. ALLIGATOR exhibits the highest performance in the testbeds of type M2 (F-measure is always 1.0), while in M3 and M6, the F-measure values are at least 0.8

Granularity (M2)										
	TB1	TB2	TB3	TB4	TB5	TB6	TB7	TB8	TB9	TB10
Precision	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Recall	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
F-Measure	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Schematic (M3)										
	TB1	TB2	TB3	TB4	TB5	TB6	TB7	TB8	TB9	TB10
Precision	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Recall	1.0	1.0	1.0	1.0	1.0	1.0	0.83	1.0	0.88	0.75
F-Measure	1.0	1.0	1.0	1.0	1.0	1.0	0.90	1.0	0.94	0.85
Grouping (M6)										
	TB1	TB2	TB3	TB4	TB5	TB6	TB7	TB8	TB9	TB10
Precision	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Recall	1.0	1.0	1.0	0.66	1.0	1.0	1.0	1.0	1.0	0.83
F-Measure	1.0	1.0	1.0	0.80	1.0	1.0	1.0	1.0	1.0	0.90

Persson et al. [14] utilize an RDF-based approach to integrate robotized production information modeled with AutomationML. Kovalenko et al. [12] explore how AutomationML can be represented by means of Model-Driven Engineering and the Semantic Web. A small part of an AutomationML ontology is developed, based on the main concepts of the language. Also, the use of rules for consistency checking is proposed, using the Semantic Web Rule Language (SWRL), but no explicit definition of the role of Semantic Web technologies on the integration problem is presented. The *AutomationML Analyzer* [16] is an online tool to browse, query and analyse different AML data by means of Semantic Web technologies; a conceptual design to overcome integration problems in AML is described. All these approaches have the potential to solve specific integration problems for AML. However, they solve rather isolated problems, and a general method capable to automatically integrate AML information from different perspectives is not provided. Contrary, ALLIGATOR combines deductive databases and Semantic Web technologies to effectively integrate documents specified using Industry 4.0 Standards like AML.

8 Conclusions and Future Work

This paper presented ALLIGATOR, a deductive framework for the integration of AML documents. ALLIGATOR relies on Datalog and RDF to *accurately* repre-

sent the knowledge that characterizes different types of semantic heterogeneity in AML documents. The results of the empirical evaluation indicate that ALLIGATOR is able to effectively solve the problems of *AML Conflict Identification* and *AML Conflict Resolution*, and exhibits similar behavior for the three studied semantic heterogeneity types, i.e., granularity (M2), schematic (M3), and grouping (M6). In the future, we will empower the ALLIGATOR Deductive System Engine with the expressiveness of Datalog with negation and built-in predicates. Thus, ALLIGATOR will be able to represent other types of semantic heterogeneity in AML, e.g., value processing (M1) and conditional mappings (M4). Further, we plan to extend ALLIGATOR to integrate documents of other Industry 4.0 Standards, such as the OPC-UA machine-to-machine communication protocol.

Acknowledgements. This work has been supported by the German Federal Ministry of Education and Research (BMBF) in the context of the projects *LUCID* (grant no. 01IS14019C), *SDI-X* (no. 01IS15035C), and *Industrial Data Space* (no. 01IS15054).

References

1. Abele, L., Kleinsteuber, M., Hansen, T.: Resource monitoring in industrial production with knowledge-based models and rules. In: PIKM@CIKM, pp. 35–42. ACM (2011)
2. Abele, L., Legat, C., Grimm, S., Muller, A.W.: Ontology-based validation of plant models. In: INDIN, pp. 236–241. IEEE (2013)
3. Barnes, M., Finch, E.L.: COLLADA-Digital Asset Schema Release 1.5.0 specification. Khronos Group, Sony Computer Entertainment Inc (2008)
4. Biffl, S., Kovalenko, O., Lüder, A., Schmidt, N., Rosendahl, R.: Semantic mapping support in AutomationML. In: ETFA, pp. 1–4. IEEE (2014)
5. Björkelund, A., Bruyninckx, H., Malec, J., Nilsson, K., Nugues, P.: Knowledge for intelligent industrial robots. In: AAAI Spring Symposium: Designing Intelligent Robots, vol. SS-12-02. AAAI (2012)
6. Björkelund, A., Malec, J., Nilsson, K., Nugues, P.: Knowledge and skill representations for robotized production. In: Proceedings of the 18th IFAC Congress, Milan (2011)
7. Ceri, S., Gottlob, G., Tanca, L.: What you always wanted to know about datalog (and never dared to ask). IEEE Trans. Knowl. Data Eng. **1**(1), 146–166 (1989)
8. Fedai, M., Epple, U., Drath, R., Fay, D.: A metamodel for generic data exchange between various CAE systems. In: 4th Mathmod Conference, vol. 24, pp. 1247–1256 (2003)
9. Henßen, R., Schleipen, M.: Interoperability between OPC UA and AutomationML. In: Procedia CIRP 25 8th International Conference on Digital Enterprise Technology DET (2014)
10. Himmler, F.: Function based engineering with automationml - towards better standardization and seamless process integration in plant engineering. In: 12 Int. Tagung Wirtschaftsinformatik, WI (2015)
11. Kovalenko, O., Euzenat, J.: Semantic matching of engineering data structures. In: Bill, S., Sabou, M. (eds.) Semantic Web for Intelligent Engineering Applications, Springer (2016)

12. Kovalenko, O., Wimmer, M., Sabou, M., Lüder, A., Ekaputra, F.J., Biffl, S.: Modeling automationml: semantic web technologies vs. model-driven engineering. In: 20th IEEE Conference on Emerging Technologies and Factory Automation, ETFA, pp. 1–4 (2015)
13. Lange, C.: Krextor - an extensible XML→RDF extraction framework. In: Scripting and Development for the Semantic Web (SFSW), CEUR Workshop Proceedings, Aachen, vol. 449, May 2009
14. Persson, J., Gallois, A., Björkelund, A., Hafdell, L., Haage, M., Malec, J., Nilsson, K., Nugues, P.: A knowledge integration framework for robotics. In: 41st International Symposium on Robotics and ROBOTIK 2010 (2010)
15. Runde, S., Dibowski, H., Fay, A., Kabitzsch, K.: A semantic requirement ontology for the engineering of building automation systems by means of OWL. In: ETFA. IEEE (2009)
16. Sabou, M., Ekaputra, F., Kovalenko, O., Biffl, S.: Supporting the engineering of cyberphysical production systems with the automationml analyzer. In: 2016 1st International Workshop on Cyber-Physical Production Systems (CPPS), pp. 1–8. IEEE (2016)
17. Schleipen, M., Gutting, D., Sauerwein, F.: Domain dependant matching of MES knowledge and domain independent mapping of automationml models. In: 2012 IEEE 17th Conference on Emerging Technologies and Factory Automation (ETFA), pp. 1–7. IEEE (2012)
18. Schleipen, M., Okon, M.: The CAEX tool suite - user assistance for the use of standardized plant engineering data exchange. In: 15th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA (2010)
19. Schmidt, N., Lüder, A.: AutomationML and eCl@ss integration (2015)
20. Schmidt, N., Lüder, A.: White paper: automation ML in a nutshell. Technical report (2015)
21. Schmidt, N., Lüder, A., Rosendahl, R., Ryashentseva, D., Foehr, M., Vollmar, J.: Surveying integration approaches for relevance in cyber physical production systems. In: ETFA, pp. 1–8. IEEE (2015)