Leveraging Semantic Web technologies for business component specification

Axel Korthaus *, Michael Schwind, Stefan Seedorf

Lehrstuhl für Wirtschaftsinformatik III, University of Mannheim, D-68131 Mannheim, Germany

Abstract
Although component-based development (CBD) is among the predominant software engineering paradigms today, numerous issues still remain preventing business component marketplaces from taking off. For example, it is state of the practice to apply different notations and modeling languages for the specification of different aspects of a business component. Besides the fact that there is a lack of standards for holistic approaches to multi-faceted business component specification, the individual specification techniques very often are not powerful enough to allow for the reliable and efficient discovery and retrieval of matching components or the automatic deduction of statements about the semantic and syntactic compatibility of components for application composition. In this article, we argue that CBD processes can greatly benefit from the use of Semantic Web technologies for business component specification. The Resource Description Framework (RDF), for example, can provide a means to integrate existing specification approaches and add new value by superimposing a common knowledge representation layer on all specification artifacts, thus enabling semantic queries and reasoning about the properties of business components.

Keywords: Semantic Web-enabled software engineering; Business component reuse; Semantic components; Component ontology

1. Introduction

The development of large-scale distributed enterprise software applications is becoming increasingly complex due to global markets imposing constantly changing requirements and enterprises relying more and more on information systems. In the last century, most manufacturing industries have faced similar challenges and have acted accordingly by systematically reusing standardized parts, often from different sources, to build more and more complex products efficiently. The identification of the need for the systematic reuse of software artifacts has led to different reuse paradigms, such as component-based software development or, in recent years, model-driven software development. When developing software from reusable building blocks, a number of questions arise, such as how suitable components can be identified or wired together in an advantageous way.

In the context of CBD, a great number of component specification methodologies and notations can be found on the market, but today none of these approaches appears to be comprehensive enough to cover all relevant component specification aspects or to provide satisfying answers to the questions or problems mentioned above. Applying a combination of different notations and modeling languages to provide a complete description of a component in order to promote its reuse across isolated software projects or organizational boundaries potentially leads to an unmanageable set of specification artifacts. Thus, a mechanism is required that helps to integrate those different artifacts and further supports typical CBD tasks such as retrieving components for reuse or determining the match of different components meant to cooperate.

In this article, we argue that the facilities offered by the Semantic Web can be applied to achieve a higher degree of integration and uniformity of specification approaches and support of various CBD activities. We demonstrate the suitability of today’s Semantic Web technologies to integrate heterogeneous component specifications and semantically enhance existing notations like the UML. It is furthermore pointed out how various component aspects can be represented using Semantic Web technologies.

The article is structured as follows: in Section 2, we provide a short overview of the fundamentals of business component
specification, describe the challenges faced by current specification approaches and consider the potential of Semantic Web technologies to address these issues. Section 3 reviews existing applications of Semantic Web technologies for similar specification tasks in order to illustrate their benefits in that context. In Section 4, we propose a new way of integrating business component specification artifacts with Semantic Web technologies. The article concludes with a brief summary and critical discussion of the approach in Section 5.

2. Component-based software development

The component-based approach to software artifact reuse is a way of coping with complexity by decomposing a system into less complex parts with standardized interfaces that can interoperate to serve a purpose in the business domain and can be composed in different combinations to be suitable for solving various kinds of problems.

2.1. Fundamentals of business component specification

It is commonly agreed that a standardized method for the specification of business components plays a key role in facilitating systematic reuse. Moreover, the semantic description a business component provides to its peers and its environment seems to be a crucial factor for achieving this objective [1,2]. The need for a concise description of the interfaces a component provides to and requires from its environment has led to a number of methodologies and standardized notations that cover various aspects of business component specification. Approaches to component specification are part of CBD methods such as Catalysis [3], UML Components [4] or KobrA [5]. The goal of these approaches is to support various CBD-specific tasks, such as component identification, component retrieval and component composition. For a comprehensive example of multi-faceted component specification we refer the reader to the “Memorandum for the Specification of Business Components” proposed by a working group of the German Informatics Society (Gesellschaft für Informatik e.V.) [6]. Even though this proposal is not an industry standard, from our point of view it nicely describes the crucial aspects of component specification, emphasizing the aspect of component marketability. The memorandum contains comprehensive recommendations for the specification of functional as well as non-functional component aspects. The authors of the memorandum propose seven specification layers that describe the external view on a business component on various levels of abstraction with each level being targeted at a different aspect (cf. Fig. 1). A component specification conforming to this proposal contains a detailed description of the services the component provides to its peers and the prerequisites for the integration of the component into a particular environment with regard to technical and non-technical component properties.

2.2. Challenges in the field of business component specification

Generally, a software component can result from various kinds of software development efforts. Ambler distinguishes a top-down and a bottom-up approach [8]. While the former is usually applied when building a new component-based system from scratch, the latter can be useful when some components already exist as parts of a system. Regardless of whether a component has been developed as part of a greater development project or with the intention of being marketed on a software component marketplace, for it to be commercially successful, it needs to be found and deemed suitable by potential buyers. Therefore, a detailed description of the services the component provides with respect to the customer’s business domain is the fundamental prerequisite for marketability. As a consequence, a component specification technique must provide the means to
support component discovery and retrieval by providing tools and methods to integrate domain knowledge into the component specification (cf. [9]). Once a component has been identified to support a number of use cases in the domain under observation, more technical questions start to arise. The service provider responsible for the integration of the component into an existing environment must be able to determine whether the component can interoperate with other components deployed in the same environment. This aspect includes the question of syntactic compatibility of component interfaces and of data representation standards or dependencies with other components that may be needed for a component to function as expected. Therefore a component specification technique must include a syntactic description of a component’s provided and required interfaces.

Before a component can be integrated into an environment, assertions about the resulting system may be needed. Even if a component fulfills all functional requirements and is technically compatible with its peers, it may violate non-functional aspects, such as security, scalability or performance requirements. Again a component specification is not complete without the means to facilitate the deduction of facts regarding the non-functional properties of composed systems [10]. The challenges described here are not a comprehensive summary of all aspects of CBD, but rather are intended to serve as examples of problems that can be tackled by providing software developers with tool support to produce and process component specification artifacts. To facilitate tool support for tasks such as component discovery or the deduction of facts about the composed system, a number of key requirements must be met. In order to be machine-processible, component specification data must be presented in a formal, uniform and coherent way. On the other hand, many of the resources used for the specification of business components have to be read and modified by humans. Thus, a desired feature is the ability to link specification elements with secondary information resources that do not constitute an integral part of a component specification but rather reflect background knowledge on the underlying design decisions. This feature helps to increase human-understandability of component specifications [10]. In other words, an all-embracing business component specification framework needs to integrate two basic interests, namely machine-processibility and human-understandability.

To allow for the integration of components from different vendors, a mutual agreement on the specification elements and representation formats is required. Ideally, CBD applications can cope with variations on these elements and the introduction of new termini without compromising machine-processibility. Thus, adaptability and extensibility are important aspects with regard to the integration of new specification aspects as well as new domain knowledge.

2.3. Semantic Web technologies as solution?

In recent years, the emerging vision of the Semantic Web [11] has led to a series of standardization activities under the umbrella of the W3C Consortium [12]. The main objective of the Semantic Web is to provide machine-processible Web content. The Semantic Web is described as “an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation” [13].

The Semantic Web activities controlled by the W3C have produced two key technologies, namely the Resource Description Framework (RDF) [14] and the Web Ontology Language (OWL) [15]. Both standards are intended to set up a framework which allows the sharing and reuse of data on the Web. RDF is used to represent information and to exchange knowledge between Web applications. OWL is used to publish and exchange ontologies which are formal sets of concepts, relations and constraints. The technologies developed for the Semantic Web target a broad range of application areas such as advanced Web search, software agents, knowledge management and enterprise application integration. Accordingly, these technologies are potentially capable of describing and processing component specification-related entities for various purposes. Moreover, they provide the means for publishing and reusing domain-specific concepts.

In the subsequent sections we will argue that the requirements stated in the last subsection can be best fulfilled when relying on Semantic Web technologies. They can be used to solve two main problems:

- The semantic linking among specification resources and
- The enhancement of specification elements with explicit semantics which is a prerequisite for software component retrieval and composition.

Some aspects of a component specification can easily be represented using widely accepted notations, e.g. the syntactical specification of interfaces using the CORBA Interface Definition Language (IDL) [16] or the formal description of the behavior a component service provides by specifying pre- and postconditions in the Object Constraint Language (OCL) [17]. While a graphical notation such as the UML is well suited for describing various aspects of a business component, it is limited with respect to the integration of heterogeneous, distributed resources and the sharing of component specifications between different applications. Furthermore, many other aspects like component quality criteria have not been standardized yet. It is crucial that a comprehensive component specification needs to represent both functional and non-functional component aspects in a way that is understandable by all parties concerned.

This article elaborates on two issues that are closely connected to the standardization of business component specification.

First, we have given reasons for the necessity to present the different aspects of business component specification in a uniform and coherent way in order to promote the marketability of business components. Furthermore, a repository, i.e. a storage infrastructure for component specifications should be able to cope with the particularities of different notations. When an integration layer is imposed on distributed information resources, the automated exchange of specifications between different applications becomes possible.

The second issue being discussed in this article concerns the augmentation of specification elements with explicit semantics regarding their domain-specific meaning. All specification
layers rely on a common notion of the concepts of the business domain under observation. Domain-specific references are introduced by describing a component aspect in unequivocal terms. In the memorandum mentioned in Section 2.1, the layer for describing functional terms is referred to as terminology layer. The authors of the memorandum suggest the use of the Resource Description Framework (RDF) as one option for specifying the required terminology, but also encourage the use of natural language descriptions as the primary notation. While both approaches have advantages and disadvantages, the latter suggestion inhibits the automatic detection of domain violations and the deduction of assertions about the expected properties of the composed system. Natural language descriptions are appealing, as they can be read and understood by domain experts lacking a strong technical background, but it is also very difficult to ensure their unambiguity. Therefore, this article presents a basic approach for applying RDF to the terminology and interface levels as proposed and supports the parallel use of formal as well as informal notations in order to leverage the benefits of both approaches.

3. Related applications of Semantic Web technologies

In some respects, Semantic Web languages could be regarded as software specification languages in line with UML + OCL, Alloy and Z (cf. [18]). From that perspective, Semantic Web languages like OWL-DL represent a smaller subset of First-Order-Logic (FOL), however providing distinct features regarding automatic deduction and interoperability. Before we present our approach to the use of Semantic Web technologies for business component specification in Section 4, we shall provide some background with respect to related work by giving an overview of existing or emerging application areas for Semantic Web technologies in the field of software specification. In the following, we will single out exemplary problems specifically addressed by Semantic Web technologies in the areas of web services, middleware and software modeling techniques.

3.1. Semantic Web services

Web services are pieces of functionality that are accessible through Web-based technology. Platform-independent W3C standards like WSDL, SOAP and XML are used to describe interfaces and message-exchange patterns, hence promoting the reuse of distributed software components and increasing interoperability between application platforms. The process of “wiring” simple services together to constitute complex workflows can be partially automated by industry-driven service orchestration languages such as the Business Process Execution Language (BPEL). However, the development of business applications from internally and externally accessible Web services is still a manual task. So far, searching, selecting and composing services to workflows requires the intervention of human developers.

The limitations of Web services in terms of ad hoc discovery, composition and execution gave rise to the notion of “Semantic Web services” [19]. The purpose of Semantic Web services is to make service definitions machine-understandable by describing their capabilities, inputs and outputs, constraints as well as service choreography and orchestration in formal language. Several competing approaches are currently under development; a comprehensive comparison can be found in [20]. Two main directions can be identified:

- The definition of a formal conceptual model that serves as the basis for ad hoc service matching and composition as represented by OWL-S [21] and WSMO [22] and
- The extension of syntactical description formats by defining an annotation mechanism. The latter approach is represented by WSDL-S [23].

OWL-S consists of a top-level service ontology together with three complementary ontology modules: a service profile for describing the service’s capabilities, a service model for describing processes, and a service grounding to link the semantic description with the syntactic description level. Although the OWL-S ontology defines the basic constructs to describe a service, it does not specify formal semantics. The formal verification of process models, e.g. based on situation calculus or petri nets, is not included in the OWL-S 1.0 specification, but is addressed in [24]. A different approach to Semantic Web services is proposed by WSDL-S [23], which focuses on semantic extensions to existing infrastructures and tools. It can be used to provide semantic annotations that can be embedded into WSDL documents. This is achieved by introducing model references from operations, input and output elements, preconditions, and effects to a domain model. The semantic annotation mechanism is agnostic to a semantics representation language, since the domain model in turn may be specified using OWL, RDF Schema or any other ontology language. WSDL-S provides an incremental approach to service discovery and composition, since it enhances syntactic specification formats with semantic features that allow for semantic search, semi-automatic service composition and service mediation. Currently, the standardization of semantic annotations in WSDL is targeted by the SAWSDL working group [25]. In contrast to the fully fledged Semantic Web service frameworks, WSDL-S takes a more evolutionary approach by addressing the semantic enhancement of existing specification resources. Due to its nature, an explicit conceptual model is missing in WSDL-S.

While OWL-S is a heavyweight approach requiring a large investment during development-time since high quality domain ontologies have to be developed and agreed upon, WSDL-S can be considered more lightweight. Project decisions with respect to which approach is to be used largely depend on the trade-off between the required modeling effort and the expected productivity gains.

Although both directions mentioned above are closely related to the problems addressed in this article, there are major differences in their orientation towards the development process. For example, OWL-S is constructed towards the discovery and composition of abstract service descriptions at runtime. In contrast, our understanding of business component specification is geared towards a development-oriented perspective. Further, service
specification can be viewed as a special case of component specification, since it raises the level of abstraction by disregarding platform-specific concerns such as object-oriented technology.

3.2. Semantic middleware

Middleware plays a key role in the development and deployment of component-based systems since it conceals much of the complexity inherent to distributed software components. Web service standards and XML-based configuration have helped to increase flexibility in the configuration and management of middleware and application servers. However, the conceptual model underlying various description formats remains implicit, which suggests that management tasks are not supported as efficiently as they could be [26].

Oberle [26] therefore proposes a semantic approach to middleware management. The purpose is to support both developers and administrators in managing server applications, e.g. by making knowledge about library dependencies explicit. Potential use cases include the semantic description of licensing information, dependencies and versioning, access rights, security, transactional settings, policy handling, detecting of inconsistencies and monitoring of changes. Oberle developed several foundational software ontologies to describe core concepts of component-based and service-oriented development. The ontologies provide a precise, formal definition of sometimes ambiguous concepts (e.g. "component" or "service") and support the modeling of middleware knowledge as mentioned above.

Thus, ontologies and Semantic Web technologies are used to create an information space for the management of components both at runtime and development time. Part of the required information is already captured in various artifacts (e.g. deployment descriptors) and has to be associated with other modeled information. Reasoning and rules can then be applied to realize the use cases mentioned above. The approach is implemented as semantic extension of an application server by employing the KAON ontology infrastructure [27]. Oberle’s work shows that semantic integration of distributed information enhances the efficiency in a wide range of component-based development and management activities.

3.3. Software modeling techniques

Most CBD development processes today, such as UML Components, make use of the predominant industry standard for modeling languages, OMG’s Unified Modeling Language (UML), which is formally based on the Meta Object Facility (MOF). However, software modeling languages with MOF-based metamodels, such as the UML, lack mechanisms for automatic inference and information integration, two features increasingly sought after in distributed business application development processes. Although the semantics of a model are structurally defined by its metamodel, the mechanisms to describe the semantics of the domain are rather limited compared to knowledge representation languages [28]. MOF-based languages do not have a knowledge-based foundation to enable reasoning. Other possible shortcomings include validation and automated consistency checking, which are however partially addressed by the Object Constraint Language (OCL) [17].

For this reason, the integration of software modeling languages with Semantic Web-based ontology languages is subject to ongoing discussions (cf. [29–31]) and of great importance for business component specification. Several proposals aiming at the integration of software modeling languages and ontology languages including RDF Schema and OWL claim that this integration helps to reduce language ambiguity and further enables validation and automated consistency checking [28]. Ontology languages provide better support for logical inference, integration and interoperability. Since ontologies promote the notion of meaning and identity much more rigorous than MOF-based languages, the sharing and mediation of domain models can be simplified.

There are several alternatives for integrating MOF-based information representation and ontology languages, which are exemplified in [30]. While some use the UML as ontology representation language by defining direct mappings between OWL and UML language constructs [32], others extend the UML by introducing a UML profile to represent ontology concepts as new modeling elements [33]. In most cases, MOF-based languages and RDF/OWL are regarded as two distinct technological spaces sharing a “semantic overlap” where synergies can be realized by defining bridges between them [29]. The Ontology Definition Metamodel (ODM) [31] is an effort by the OMG to standardize the mappings between knowledge representation and conceptual modeling languages. It specifies a set of MOF metamodels, informative mappings between those languages, and profiles for a UML-based notation. The ODM includes metamodels for RDF (Schema) and OWL, Common Logic, Topic Maps and ER.

On the one hand, the integration with Semantic Web languages creates new potentials for sharing and reusing UML-based business component specifications, thereby lowering the barriers for subsequent application composition. On the other hand, Semantic Web technologies can benefit from software specification languages like Z and Alloy, since they provide model checking techniques currently not supported by Semantic Web reasoners. Those might be applied to reveal model inconsistencies, e.g. in OWL-S process models, which cannot be detected by Description Logic reasoners [18].

4. Semantic Web technologies for business component specification

In this section, we investigate the application of Semantic Web technologies for the representation of various component specification aspects. When developing software components for reuse, additional effort both in terms of providing generic functionality and responding to the additional specification requirements has to be considered. To promote the concise specification of components, appropriate specification activities should be integrated seamlessly with the component-based development practices already in place. We believe that maintaining the established methodologies, notations and formats is an important factor for the overall acceptance of component reuse strategies.
We therefore propose a general architecture for the semantic integration of business component specifications. This architecture supports incremental improvement of CBD processes by allowing heterogeneous specification artifacts to be shared and described with explicit semantics. The two-tier integration architecture is illustrated by giving an example specification which is represented in RDF Schema. Finally, we consider special issues concerning the semantic annotation and mapping of business component specifications from UML to a Semantic Web-based representation.

4.1. Towards an integration architecture

It is possible to identify several alternatives in which Semantic Web technologies may be utilized for the specification of business components. We first derive the fundamental advantages of Semantic Web technologies, state the identified requirements and then propose a two-tier architecture for the semantic integration of business component specifications. At this stage we focus on RDF Schema as ontology representation language. It is nevertheless possible to apply all the arguments stated below to OWL.

The analysis of related application areas in the previous section already suggests that Semantic Web technologies are well suited to support a diverse range of CBD-related activities. During the analysis and design phase, software modeling languages such as the UML serve as the starting point to describe the domain model and to decompose the desired system into components. The integration of software modeling with ontology languages (Section 3.3) enables the transformation of domain models into an ontology representation format, thus helping to improve consistency checking and validation (cf. [28]). In the following phases, semantic modeling of the relations between components implementing the specifications can be used to create a knowledge base and automate the administrative tasks regarding the management of components in an application server (Section 3.2). In the case of Semantic Web services (Section 3.1), a formal specification of the service is provided to enable ad hoc discovery and invocation of services mainly at runtime.

The chosen application areas not only pinpoint some of the advantages of Semantic Web technologies but also share an overlap with business component specification issues. However, there also remain some subtle differences. For example, Semantic Web services represent an abstraction from arbitrary component models. The underlying component technology or the deployment usually is not part of a service specification. Furthermore, the development of shared domain ontologies is seen as an indispensable prerequisite to Semantic Web services. Since the development and sharing of domain ontologies are not yet established in the CBD domain we follow an approach where semantics can be added as considered necessary. Accordingly, our approach in this article is clearly focused on providing support for the development process instead of runtime automation. Therefore, it recognizes the inherent heterogeneity of specification artifacts of both formal and informal character.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Specification task</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metadata representation</td>
<td>Description of component metadata and annotation with secondary resources that provide background knowledge</td>
</tr>
<tr>
<td>Resource integration</td>
<td>Integration of distributed, heterogeneous resources that constitute a logical coherent component specification</td>
</tr>
<tr>
<td>Domain referencing</td>
<td>Unequivocal referencing of domain-specific concepts that are represented as Web resource</td>
</tr>
<tr>
<td>Formal specification</td>
<td>Formal specification of concepts and axioms, reasoning on component specification data</td>
</tr>
</tbody>
</table>

The application areas mentioned before help to reveal some fundamental advantages of Semantic Web technologies in CBD. These are outlined in Table 1 and described in more detail in the following paragraphs: metadata representation, resource integration, referencing of domain-specific concepts, and formal specification.

Many aspects of a business component specification, e.g. commercial and licensing information, are specified using metadata. Since the main rationale behind RDF is metadata representation, it offers adequate support for describing the various component aspects. At the same time the graph structure of RDF provides higher flexibility compared to a plain XML format.

Another useful feature provided by a Semantic Web-based representation is resource integration, facilitating the description of physically existing specification resources, such as files containing interface specifications in WSDL or a component’s domain model in UML/XMI format, for example. Furthermore, it is possible to state which specification aspects are covered by a particular artifact.

One of the main characteristics shared by all Semantic Web standards is the Uniform Resource Identifier (URI) concept. By identifying both RDF properties and resources using URIs, unambiguous references to domain-specific concepts can be introduced. The use of URIs facilitates the augmentation of specifications with functional terms in the component’s business domain (found in the terminology layer). For example, assuming a generic business component for monitoring the credit transactions in a bank, one could explicitly specify that the attribute “accountID” is conforming to the “IBAN” format. The referencing of domain-specific concepts provides the means to execute queries on structured knowledge, which is a feature required for retrieving business components by their functional or non-functional characteristics and also for testing their semantic interoperability. The advantages of Semantic Web technologies for providing an unambiguous representation of domain terminologies and for enhancing common software engineering approaches, e.g. in model-driven development, are discussed further in [28].

Finally, formal specification refers to the ability to specify rich axiomatizations when using an ontology language. This feature is a prerequisite for automated CBD tasks such as automated checking of semantic interoperability. Here, we do not discuss the formal foundations of RDF Schema or OWL (cf. [34,35]).
The actual choice of a concrete representation format depends on several constraints such as the expressiveness of logical constraints and decidability. As described in the previous sections, component specification turns out to be a multi-faceted issue and so are the possible application scenarios for Semantic Web technologies.

Recapitulating the requirements described in the first part, a framework for the semantic integration of business component specifications should:

- Support component retrieval and composition during development-time.
- Strive for full coverage of the identified specification layers.
- Enable the integration of different kinds of specification artifacts in common specification notations and formats.
- Improve machine-interpretablity and understandability for human developers.

Given the above assertions, the general architecture of such a framework is described in the following paragraphs.

As depicted in Fig. 2, a number of information resources required for component specification is already produced as result of typical development activities. This comprises formal (e.g. code artifacts), semi-formal (e.g. UML models) and informal elements (e.g. natural-language documentation). On one hand, there are formal specification elements such as WSDL documents or a J2EE Session Façade for specifying the external behavior. On the other hand, a component provider will most likely choose a text or XML-based format to describe the component’s commercial aspects, e.g. licensing, pricing, and others.

On the given heterogeneous specification resources, a semantic specification layer is imposed in two phases. In the first phase, the specification artifacts are described according to their format and the specification aspects covered. This can be seen as the “glue” for interconnecting heterogeneous specification resources and augmenting them with a well-defined meaning. A component ontology allows the assignment of specification artifacts to a business component specification. The basic classes required in this scenario are described in Fig. 3(a). Here, RDF Schema is used to define the classes, subclass relationships and properties. Domains and ranges of a property are omitted for brevity. Key concepts in this lightweight ontology are Aspect and Format. A specification artifact may cover one or more aspects—these roughly correspond to the specification levels identified in Section 2.1. Each specification artifact should be expressed in a pre-defined format. Furthermore, it is possible to refine aspects through subclassing and to establish a compatibility relationship between aspects and formats. For example, the aspect Interface would be compatible to the formats WSDL and CORBA IDL.

The first phase achieves a structural integration of heterogeneous specification resources but does not increase their machine-interpretablity. In the second phase, the content of a specification artifact should be extracted and inserted in a semantic specification layer. This layer is based on a component ontology as partially depicted in Fig. 3(b). It provides unambiguous definitions of well-known concepts from the world of CBD. Given a one-way mapping from each specification format onto the component ontology the desired RDF-based representation could be automatically obtained. Therefore, a transformation needs to be defined for every format responding to a particular

![Fig. 2. Representation of component specifications in a semantic layer.](image-url)
specification aspect. For example, the provided interface specification may be derived from a WSDL document (cf. [36]) or an XMI-encoded specification, which is based on a particular UML profile as part of a component development methodology. Since the two cases are possibly based on different conceptualizations, the unifying elements should be codified by a common component ontology. The ontology fragment shown in Fig. 3 only contains the interface and terminological level but it can be extended to cover other specification aspects as well. As a side effect, a comprehensive component ontology may help to establish a shared understanding of business components and allow to integrate different types of component models.

After the second step is completed and the specification is fully represented in an RDF graph, deductive queries can be executed on the content of a business component specification using an RDF query language. Moreover, this RDF graph enables the interoperability testing of the component’s parameters on the semantic level. Another application scenario for the semantic layer is the retrieval of business components according to their domain-specific functional meaning. Logical inference turns out to be a major advantage compared to traditional specification and modeling approaches based on the Unified Modeling Language and its supporting standards.

In this section, we have shown that Semantic Web technologies are suitable for both the integration and representation of specification resources in a comprehensive business component specification approach.

4.2. Representation with RDF Schema

In this section, we use RDF Schema as knowledge representation format in order to present some basic examples for the proposed semantic layer. Let us consider the example of a banking component AccountStatementTeller which returns the transactions a customer’s account has been involved in. Its specification covers commercial metadata, the component’s provided interface and domain-specific meaning of data types.

In the first example, the commercial aspects of this component are described in RDF. Fig. 4 depicts an exemplary RDF graph which describes standard metadata of our business component. The example uses the RDF Schema classes BusinessComponent and ComponentProvider from the component ontology defined in Section 4.1. An actual business component specification is represented as a resource using a qualified name within a predefined namespace. The parameter values are given as literals and resources, respectively. Standard component metadata typically involves a component publisher, address data, version info and functional descriptions among other things. In a “commercial off-the-shelf” (COTS) application scenario, such an RDF model can be applied to publish component descriptions on a component market.

As proposed in the last section, specification elements contained in various artifacts can be extracted and transformed into an RDF graph. Fig. 5 shows a simplified example which shows the representation of a provided interface specification. If appropriate mappings to the RDF model are defined, it does not matter whether the input is a serialized UML model in XMI or a WSDL description. Thus, heterogeneous specification resources can easily be integrated using RDF Schema. The presented example is simplified in order to improve understandability.

A business component specification to be represented in the form of an RDF graph usually includes both syntactic and semantic descriptions. As mentioned before, RDF Schema is highly suitable for referencing domain-specific functional terms that have been published in a domain vocabulary or ontology. Explicit semantics are introduced if a domain-specific term is
provided as web resource. Since there is a conceptual correspondence between the interface level and higher levels of abstraction [6], such as the terminology level, RDF statements can be used to semantically enhance specification elements. As an example, the relation between a data type and a functional term is demonstrated in Fig. 6: the output parameter `sourceAccountID` of type `String` is semantically described by concept `IBANAccount` found in a domain ontology. It should be pointed out that the interpretation of functional terms is unambiguous only within the given namespace. The same method for augmenting component specifications with domain-specific knowledge can be applied to other specification aspects on different degrees of granularity, for example the terminology description of a component service [10].

4.3. Integration of UML-based specifications

As mentioned in Section 3.3, most component specification methodologies use the UML as primary modeling language. It is rather unlikely that the UML will be substituted by an ontology-based approach which does not provide an adequate graphical notation in the near future. However, as we have seen in the previous sections, a Semantic Web-based representation provides several advantages with respect to machine-interpretability and linking between different specification layers. For this reason we will now analyze how existing UML-based business component specifications can be semantically annotated and represented in RDF Schema or OWL.

Fig. 4. Specifying commercial component metadata with RDF.

Fig. 5. Example of a component specification represented as RDF graph.
In Section 3.3, we analyzed the relationship between software modeling languages and the ontology languages of the Semantic Web. Although both offer several distinct features, it is possible to define bridges between those languages. As an example, the Ontology Definition Metamodel defines MOF-based metamodels and mappings, which permits the interchange of conceptual models among different modeling languages. Since component specification methodologies such as UML Components [4] use proprietary UML profiles and more than one diagram type, this mechanism is not applicable to our case. Instead, individual mappings to the component ontology have to be specified.

In the example shown in Fig. 7, a business component specification for retrieving a customer’s account statement is depicted. It is applied to a proprietary UML Profile which uses stereotypes and stereotyped attributes (tagged values) as extension mechanism. The interrelation between the interface definitions in the component view and the provided interface specification is established by the dependency stereotype “provides”. Similar approaches to component specification are illustrated in the UML Components methodology [4] and in the BOOSTER*Process [37].

Ideally, this component specification should be propagated as a coherent unit with the possibility to define semantic annotations between the different specification layers, e.g. the interface layer and the domain terminology layer. First, an individual mapping of the UML-based component specification onto a component ontology will have to be specified. Second, it will be useful to augment model elements such as data types with external model reference to a domain ontology. If UML is maintained as primary notation for specification, a mechanism for semantically annotating UML model elements is thus desirable.

A simple solution to this requirement has been investigated in [38] where tagged values are used to enrich UML 1.3 and 1.4 models with domain-specific knowledge. Tagged values are name-value pairs whose interpretation lies outside the scope of the UML metamodel. They are similar to an RDF statement which is represented as a triple consisting of subject, predicate and object. To be processible within an RDF or OWL-based representation, tagged values must be provided as URIs or an unambiguous mapping must be specified.

Providing either of these facilitates the addition of a semantic meaning to the individual model elements, e.g. functional terms and process task descriptions defined in an external domain ontology. In the given example (Fig. 7), we use a UML 2 profile for modeling, which contains the definition of the new stereotype business object and an associated attribute describedBy. In our model, this stereotype is applied to define the Payment business object. Since the stereotype prescribes the attribute describedBy, Payment is annotated with a corresponding tagged value. This

Fig. 6. Augmentation of data types with domain-specific meaning.

Fig. 7. Example business component specification in UML.
tagged value provides identity to the Payment business object by specifying a Web resource, which is defined in a domain ontology. The next step, i.e. the mapping of this model information to an RDF-based representation, is omitted from Fig. 7, however the result looks similar to Fig. 6.

The semantic extension of UML-based business component specification methodologies serves two purposes: improving the machine-interpretablity of component specifications and achieving a more precise understanding for human developers at the same time.

5. Conclusion

In this article, we have proposed a two-step approach to the application of Semantic Web technologies in the context of CBD, thereby providing a means for the integration of heterogeneous component specification artifacts, the formal representation of domain-specific knowledge and the support for reasoning and deductive querying. The first step is limited to achieving the integration goal, thus providing a single point of access to the set of heterogeneous specification resources in different formats describing a component. In the second step, relevant information captured in the specification artifacts is extracted and transformed into an RDF representation in order to facilitate the deduction of facts about the specified component and its characteristics in conjunction with other components. To show the practical applicability of this approach, we have presented a general idea of how to map a profile-based UML interface specification into an RDF representation. For the purpose of illustration, RDF Schema has been used for the examples in this article; however, they can be analogously transferred to other knowledge representation languages such as OWL.

It is important to note that the two steps towards semantic integration pursue complementary goals with respect to component specification: while the first step aims at providing the means to retrieve all specification assets relevant for a given functional or non-functional component property, the second one targets the aspect of using inference to support the process of retrieval and composition. This approach enables us to incorporate almost arbitrary resources and gradually expand the tools to create RDF representations from different types of specification assets without rendering existing resources useless. A big advantage of this approach is that the two steps can be introduced incrementally. Although the full potential of the approach cannot be realized until both steps are in place, realizing only the first step will still bring forward the benefits of integrating heterogeneous specification artifacts.

A potential drawback of this proposal is not unique to our approach but applies to all ontology-based approaches: the required extra effort for ontology development and a suitable tool infrastructure. The trade-off between this additional overhead and the expected productivity gains has to be thoroughly considered. However, by describing an incremental path to Semantic Web-based business component specification, we manage to considerably mitigate those concerns. For example, the definition of individual mappings from conventional specification formats to a Semantic Web-based representation can be postponed if considered too resource-intensive. However, once a mapping for a particular format has been defined, it can be applied over and over again.

The research we carry out in this context is performed within the CollaBaWue project, which aims at facilitating the creation of software supply chains to strengthen local software suppliers by providing the means to efficiently market business software components. The potential role of Semantic Web technologies as an enabling factor for the retrieval of business components and the automated deduction of statements about composed systems will be subject to further research within this project.

The next steps will be the prototypical implementation of a semantic integration layer as described here and the design and implementation of a prototype for the extraction of RDF graphs from heterogeneous specification assets such as serialized UML models.

While the introduction of semantically enriched and uniform component specifications is certainly not the last problem to be solved on the way to business component markets, it definitely promotes the vision of industrial software development by supplying the different stakeholders with concise, uniform and machine-processible component specification artifacts. Although the approach discussed here promises to be beneficial in the context outlined, it is important to note that standardization efforts for the specification of business components are still essential and should be pursued more actively to better promote the idea of software component marketability.

Acknowledgement

This work was supported by a grant from the Landesstiftung Baden-Wuerttemberg foundation to Michael Schwind and Stefan Seedorf.

References


