Ontological Analysis for Generating Baseline Architectural Descriptions

Arvind W. Kiwelekar and Rushikesh K. Joshi

Department of Computer Science and Engineering
Indian Institute of Technology Bombay
Powai, Mumbai-400076, India
{awk,rkj}@cse.iitb.ac.in

Abstract. Mapping elements from an application domain to architectural abstractions is a significant architecture description activity from the point of view of seamlessness in descriptions. For establishing such a mapping of domain elements to architectural abstractions, an approach based on *ontological analysis* is presented. The central idea of the approach is to establish the mapping through a uniform framework of understanding that is applicable over the problem domain as well as the solution domain. The reference ontology used is an adaptation of Bunge-Wand-Weber (BWW) ontology. Typically, an element from an application domain is mapped with an architectural abstraction when both represent the same phenomena from BWW ontology. The approach is realized as a model-driven transformation process.

1 Introduction

Domain understanding is one of the concerns that needs to be addressed while deriving architectural descriptions from requirements [1]. Some of the challenges faced while understanding application domains are- (i) Different models are used to represent outcomes of the activities of requirement analysis and architecture design. (ii) Domain requirements are elaborated through domain terminology. (iii) Correspondence among requirements and architectural viewpoints is undefined.

In this paper, an *ontological analysis technique* to guide the transition from requirements to architecture is presented. The technique aims to understand application domains and architectural abstractions through a universal language formalism. The BWW ontology [2,3] is used to interpret domain elements and architectural abstractions. The reason for selecting this ontology was that it has been applied to evaluate the expressive power of various modeling languages such as UML [4], and ebXML [5], and it is a generic ontology capturing diverse phenomena. The outcomes of the application of the analysis technique include identification of architecturally significant domain elements specific to Component-Connector (C&C) viewpoints, and an initial architectural configuration aligned with interactions among domain elements. The technique assumes the availability of domain descriptions in natural language text or scenario descriptions.

M. Ali Babar and I. Gorton (Eds.): ECSA 2010, LNCS 6285, pp. 417–424, 2010.

Earlier approaches that derive architectural descriptions from requirements vary in terms of analysis models used. Analysis models such as goal oriented analysis [6,7], global analysis [8,9], and quality attribute based analysis [7] are some of the techniques used earlier. In this paper, it is proposed to perform ontological analysis of application domains for describing domains in terms of ontological categories. Some of the earlier approaches such as the collaborative approach called CBSP [10] target the component-connector (C&C) view. The CBSP approach analyzes requirements in terms of components, buses, systems and properties. A limitation of this approach is that a given requirement may be mapped to a component or a connector by different analysts. The conflicts on the choice of abstraction is resolved through a voting process that lacks analytical reasoning. The approach presented in this paper also targets C&C views. An element from an application domain is mapped with an architectural abstraction only when both represent the same phenomena from BWW ontology. The approach presented in this paper also attempt to address the problem of vagueness [11] in semantics of components and connectors. The problem is handled by performing ontological analysis of components and connectors. The distinctness of these abstractions is brought out by showing that components are the representations of *Things* and connectors represent *Coupled Events*. *Things* and *Coupled Events* are two different phenomena from the reality.

The next section describes the reference model based on BWW ontology. Section 3 presents an overview of the process with various artifacts involved. Ontological interpretations of architectural abstractions and also of domain elements are presented in Section 4. Section 5 discusses transformation rules that transform the output ontological analysis to components and connector specifications.

2 BWW Reference Model

Bunge's original ontology [2] is considered as a general system theory. Later, Wand and Weber [12] subsequently adapted it to model information systems. The adapted version is referred to BWW ontology. An *ontological category* capturing a real world phenomenon is a high-level generic notion defined in an ontology. Ontological categories are valuable conceptual abstractions to analyze a particular domain because they can be used to identify the roles played by domain abstractions and relationships among them. BWW ontology includes a comprehensive set of ontological categories that model static and dynamic features of reality. Earlier we developed a classification scheme for ontological categories and a meta-model for BWW ontology that have been reported in [13]. The ontological categories and the relationships among them are represented through a UML meta-model is summarized in Figure 1. The meta-model has been formalized through type-theoretic notations. By specifying the meta-model through type theoretic notations, we learnt that the ontological categories are generic types and they can play the roles of *types* to describe elements from application domains.

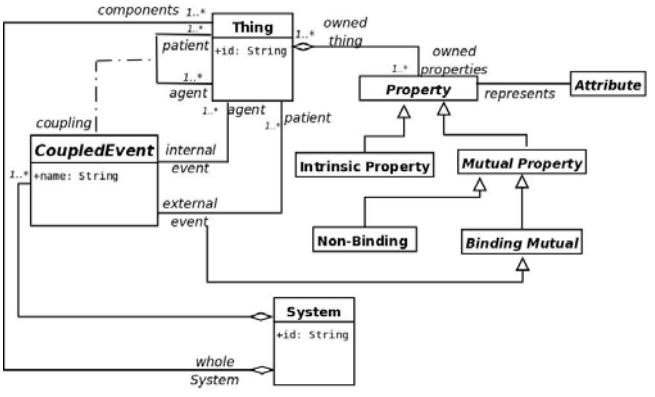

Fig. 1. An Object Oriented meta-model for BWW Ontology

3 The Process and Its Artifacts

Two different types of modeling artifacts are needed. Firstly, a *domain ontology model* describes an application domain through BWW ontological categories. This model is manually developed after performing ontological analysis of requirements. The second model is automatically generated and it is referred to as a *baseline architectural description*. It consists of a set of architecturally significant elements from application domain. Baseline architectural abstractions i.e., abstractions present in baseline architectural descriptions can be further refined into full-fledged architectural abstractions by adding features found in software modeling languages. For example, in *Meeting Scheduler* domain [14], *getting a preferred date for meeting* is an element of the domain that can be considered as a baseline connector. This baseline connector can be further refined to a full-fledged software connector by adding non-functional features such as distribution, concurrency and security around the baseline connector. Earlier, we have used the concept of the *baseline connector* to extract C&C views from the UML models of existing systems [15]. A baseline connector becomes a *full-fledged* connector by adding non-functional properties to it.

4 Interpreting Architectural Abstractions and Domain Elements through BWW Ontology

The existing practices and taxonomies [11,16] provide guidelines to represent solution-domain specific entities through architectural abstractions. For example, computational processes or database entities can be represented through components. The interaction entities such as pipes, semaphores and communication protocols can be represented through connectors. These guidelines are convenient to describe architectures of many software systems. But they offer little assistance to derive architectural abstractions from domain descriptions.

The interpretation mappings are developed to provide an assistance by relating the architectural abstractions with the generic types of domain elements. In this context, the ontological categories play the roles of generic types. The architectural abstractions from ACME ADL [17] are considered as the reference model for architectural abstractions in the process of developing interpretation mappings.

As shown in Table 1, two different types of interpretation mappings are defined. The *abstraction mapping* relates the ACME abstractions with the BWW ontological categories. The prevalent usages of the architectural abstractions and their intended meanings are the criteria used to establish the correspondence. The architectural abstractions and the relationships among them are represented through a UML meta-model as shown in Figure 2. The *domain mapping* assigns exactly one category to a domain element because each domain element represents a single phenomenon from the reality. A noun, or a noun phrase, a verb or a verb phrase or a single sentence describing domain requirements is considered as a domain element. The domain mapping is also referred to as *domain*

Table 1. Interpretation Mappings

(a) Interpretation Mapping for Architectural Abstractions

Architectural Abstractions	**BWW Category**	**Architectural Abstractions**	**BWW Category**
Component	Thing	Port	Attributes representing Binding Mutual property
Connector	Coupled Process	Roles	Mutual Binding Property
System	System	Representation	System Composition
Property	BWW Property	Attachments	Representation relation between mutual property and attributes

(b) Interpretation Mapping for Example Domain Elements from Meeting Scheduler Problem

Sr. No.	**Domain Element**	**BWW Category**	**Sr. No.**	**Domain Element**	**BWW Category**
1.	Meeting Initiator	Thing	2.	Meeting Attendee	Thing
3.	To Ask all for exclusion set	Coupled-event	4	To ask all for preference set	Coupled-event
5.	Exclusion Set	Property	6.	Preference set	Property
7.	Meeting Date	Property	8.	To ask for special Equipment	Coupled Event
9.	To ask for preferred location	Coupled Event	10.	The proposed meeting date should belong to the stated date range and to none of the exclusion sets	Law

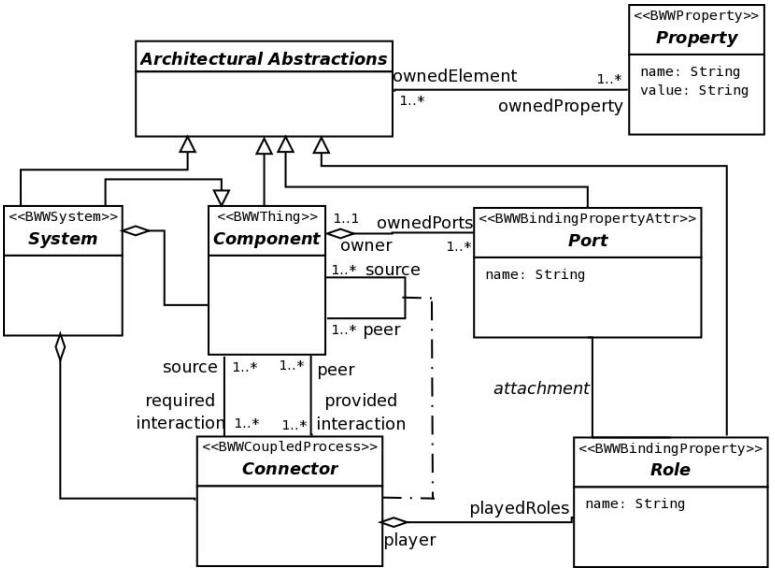

Fig. 2. Meta-model for Architectural Abstractions

ontology model and it is described as an XMI document. The BWW ontological categories are the tags in XMI document. The XMI document describing application domain follows the schema presented in the meta-model of the reference BWW ontology shown in Figure 1. The mapping from domain elements to BWW ontology is depended on an application domain and it is defined every time for each domain. The mapping from architectural abstractions to BWW ontology is independent of application domains and it is defined only once as a reusable process. The metamodel represented in Figure 2 is used as schema for the output description.

5 Transformation Rules

Transformation rules are defined to create architectural abstractions from a domain ontology model. A C&C view described in ACME is the output model that the transformation rules generate. Transformation rules are invoked based on the ontological types of domain elements. Things, Properties, Coupled Events and BWW System are some types of domain elements. They are transformed to ACME Components Properties, Connectors and System respectively. The type of ACME abstraction to be created is guided by the interpretation mapping defined in Table 1(a). Transformation rules are realized in ATLAS Transformation Language (ATL). One example of a transformation rule for transforming coupled event to connector is included below.

An instance of a connector is created when the ontological type of a domain element is *Coupled Event*. Steps involved in creating new instance of connectors

```
1rule ce2con{
2  from
3   ce:BWWOntology!CoupledEvent
4  to
5 --Constructing new connector.
6  con:C2ADL!Connector(
7   name<- ce.name,
8   ownedRoles<- Sequence{}),
9  -- Constructing Properties
10 p: C2ADL!Property(
11  name <- 'Parameters',
12  value<- ce.parameter,
13  ownedElement<- Sequence{}
14      ->append(ce)),
15 -- Constructing Roles
16 role1: C2ADL!Role(
17  name<-ce.name+'Requester'),
18 role2: C2ADL!Role(
19  name<-ce.name + 'Replier'),
20    -- Constructing Ports
21 port1: C2ADL!Port(
22   name<-ce.name + 'RepPort',
23   owner<- ce.agent),
24 port2: C2ADL!Port(
25   name<-ce.name + 'ReqPort',
26   owner<- ce.patient),
27 binding1 : C2ADL!Attachment(
28   port<- port2,
29   role<- role2),
30 binding2 : C2ADL!Attachment(
31   port<- port1,
32   role<- role1)
33 do{
34 con.ownedRoles<- con.
35 ownedRoles->append(role1);
36 con.ownedRoles<- con.
37 ownedRoles->append(role2);
38 }}
```

are (i) defining properties of connectors, (ii) constructing roles of connectors, (iii) creating ports in the components playing the roles of connectors, and (iv) binding ports to roles through attachments. Connectors, Ports, Roles and Attachments are created as a result of processing of instances of *Coupled Events*. In some cases, though the number of connectors, ports, roles and attachments can be minimized by merging all coupled events that share their participant things, this may be undesirable from the point of view of losing flexibility. For example, in Figure 3, the instances of coupled events *askAllPrefSet* and *askAllExSet* are not merged to form a single connector instance. It can be noted that non-functional properties of interactions may depend on the semantics of interactions among things and not on the participant types. In interaction such as *withdraw meeting*, an acknowledgment or reply may not be expected, while in the interaction *askAllPrefSet* an acknowledgment and reply are expected. Coupled events that need an acknowledgment may use a *reliable* communication protocol. In this context, reliability is a non-functional property.

The transformation rules generate a XMI document that conforms to the XMI schema representing meta-model of architectural abstractions. The baseline architectural configuration for *Meeting Scheduler Problem* as generated by the approach is shown in Figure 3. An earlier C&C-based architectural model of *Meeting Scheduler problem* can be found in [18]. The appropriateness of the generated architectural description is checked by comparing it with the existing architectural description.

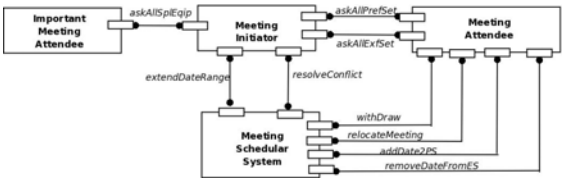

Fig. 3. Generated Architectural Configuration for Meeting Scheduler Problem

6 Conclusion

A process of mapping domain elements to produce *C&C* based architectural descriptions was discussed in this paper. The outcomes of the process include identification of architecturally significant domain elements and an architectural configuration aligned to interactions among things present in the domain. Usefulness of the approach comes from the definition of transformation rules to create an architectural configuration. The transformation rules maintain traceability links between domain requirements and architectural abstractions. By introducing the concept of *baseline architectural descriptions* the approach achieves a separation of application domain specific concerns from software specific concerns. Currently, the task of interpreting domain elements in terms of BWW ontological categories is manually performed by mapping the decisions depend on the judgment of a domain expert. The approach can be explored further-(i) by considering non-functional properties, (ii) validating the approach through non-trivial software systems and (iii) evaluating the approach from scalability point of view and by (iv) developing knowledge-based automated techniques to assist the process.

References

1. Ferrari, R.N., Madhavji, N.H.: Architecting-problems rooted in requirements. Inf. Softw. Technol. 50(1-2), 53–66 (2008)
2. Bunge, M.: Treatise on Basic Philosophy, 1st edn. Ontology I: The Furniture of the World, vol. 3. D. Reidel Publishing Compant (1977)
3. Yair, W., Weber, R.: An ontological model of an information system. IEEE Transactions on Software Engineering 16(11), 1282–1292 (1990)
4. Opdahl, A., Henderson-Sellers, B.: Ontological evaluation of the uml using the bunge-wand-weber model. Software and Systems Modeling J. 1(1), 43–67 (2002)
5. Green, P.F., Rosemann, M., Indulska, M.: Ontological evaluation of enterprise systems interoperability using ebxml. IEEE Transactions on Knowledge and data Engineering 17(5), 713–724 (2005)
6. Liu, W., Easterbrook, S.: From requirements to architectural designs using goals and scenarios. In: STRAW 2001 located at ICSE 2001 (2003)
7. Kim, J., Park, S., Sugumaran, V.: Drama: A framework for domain requirements analysis and modeling architectures in software product lines. J. Syst. Softw. 81(1), 37–55 (2008)

8. Schwanke, R.W.: Architectural requirements engineering: Theory vs. practice. In: STRAW, pp. 1–8 (2003)
9. Hofmeister, C., Kruchten, P., Nord, R., Obbink, H., Ran, A., America, P.: Generalizing a model of software architecture design from five industrial approaches. In: Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture, WICSA 2005 (2005)
10. Grunbacher, P., Egyed, A., Medvidovic, N.: Reconciling software requirements and architectures with intermediate models. Journal on Software and System Modeling (December 2003)
11. Mehta, N.R.: Towards a taxonomy of software connectors. In: 22nd International Conference on Software Engineering (June 2000)
12. Yair, W., Weber, R.: On the ontological expressiveness of information system analysis and design grammars. Journal of Information Systems (3), 217–237 (1993)
13. Kiwelekar, A.W., Joshi, R.K.: An object oriented metamodel for bunge-wand-weber ontology. In: Proc. of SWeCKa 2007, Workshop on Semantic Web for Collaborative Knowledge Acquisition at IJCAI 2007 (January 2007)
14. Shaw, M., Garlan, D., Allen, R., Klein, D., Ockerbloom, J., Scott, C., Schumacher, M.: Candidate model problems in software architecture (January 1995)
15. Kiwelekar, A.W., Joshi, R.K.: Extracting high-level component-connector view from detailed uml models: A case study. COMPSAC (2), 527–534 (2007)
16. Liu, W., Easterbrook, S.: Eliciting architectural decisions from requirements using a rule based framework. In: STRAW 2003 located at ICSE (2003)
17. Garlan, D.: Acme: An architecture description interchange language. In: Proceedings of CASCON 1997 (November 1997)
18. Medvidovic, N.: Modeling software architectures in unified modeling language. ACM Transactions on Software Engineering and Methodology 11(1), 2–57 (2002)