A Trading-Based Knowledge Representation Metamodel for Management Information System Development

José-Andrés Asensio[1], Luis Iribarne[1], Nicolás Padilla[1], and Cristina Vicente-Chicote[2]

[1] Applied Computing Group, Department of Languages and Computing
University of Almeria, Spain
{jacortes,luis.iribarne,npadilla}@ual.es
[2] Department of Information and Communication Technologies
Technical University of Cartagena, Spain
cristina.vicente@upct.es

Abstract. Design patterns are useful in Software Engineering to abstract common implementations regardless of the scope of application. This paper presents a metamodel for Trading-based Knowledge Representation (TKR), which embeds useful design patters for modeling Management Information Systems (MIS). The paper presents a case study in which this TKR metamodel is used to specify an Environmental Management Information System (EMIS). It also presents a GMF graphical model editor aimed to ease TKR model specification and validation.

Keywords: Trading-Based Knowledge Representation (TKR), Model Transformation, MDE, GMF.

1 Introduction

Management Information Systems (MIS) facilitate information retrieval and decision-making, allowing cooperative work. The design of these systems requires the use of standard methods and techniques that provide both a common vocabulary for the representation of the system knowledge and a real-time interaction with it. In order to achieve them, recently, some MIS for web (WMIS) have been developed under open and distributed paradigms [14]. Semantic web, data-mining and information querying components appear in most WMIS [12].

Environmental Management Information Systems (EMIS) are a clear example of WMIS [2]. These systems are commonly shared by a wide variety of users (e.g., technicians, politicians, administrators, etc.), who cooperate with each other and interact with the system for decision-making and problem resolution. One of the main features of the EMIS is that they usually manage large knowledge bases, normally distributed in different places. This requires the use of common vocabularies (ontologies) for knowledge representation sharing and understanding among the different actors in the system.

In this context, in addition to the above elements, we propose the use of traders to improve the interoperability of the web components included in these open and distributed systems [5], [10], [6]. Traders represent another solution for such systems [13], which extend the *Object-Request Broker* (ORB) mechanism for the *Object Management Architecture* (OMA). A **trader** is an object which acts as intermediary between objects providing certain capacities (services) and objects that require a dynamic use of these capacities. Although their traditional use is for object interoperability, traders (together with the use of ontologies) also allow us to improve querying and information retrieval in these systems. The SOLERES system [1], [11] is a web-based EMIS with spatio-temporal management capabilities, based on neural-networks, ontologies, agents, and software trading components [7]. More precisely, this system has a knowledge representation subsystem called SOLERES-KRS, which uses the "concept" of *ontological trader*.

This paper presents a metamodel for describing *Trading-based Knowledge Representation* (TKR) systems. This metamodel aims to ease the design of these systems regardless of the application scope. Its use is illustrated using the SOLERES-KRS system as a case study. The rest of the paper is organized as follows: Section 2 describes the proposed metamodel; Section 3 describes the GMF tool developed to ease the design of TKR systems and presents the case study; finally, Section 4 presents some conclusions and future work.

2 Trading-Based Knowledge Representation Metamodel

The components of the TKR system architecture, proposed in this paper, can be distributed on different nodes. These nodes can be composed of some modules (related to each other) according to the wished configuration. As a base configuration, each node must contain, at least, a *Service Module* (SM), a *Management Module* (MM) and a *Querying Module* (QM). Additionally, it can also contain zero or more *Trading Modules* (TM) and *Processing Modules* (PM). The system must have, at least, a TM module and a PM module in some of its nodes. The functionality of these modules is described next.

SM provides a complete set of services shared by the other modules, including services for module and component registration, status checking, etc. MM acts as a link between the user interface and the other modules, enabling the configuration of the former and being responsible for managing the user demands. QMs are exclusively concerned with the information queries demanded by the users. TMs enable the search and location of information in the system, establishing a filter based on the query parameters. Finally, PMs are responsible for managing the knowledge bases (insertion, modification and removal of information). More than one instance of QM, TM and PM may be needed in order to accomplish the requirements of the system under design.

Figure 1 represents the proposed TKR system metamodel, which includes the concepts and restrictions previously described. It is worth noting that this metamodel enables the description of models of TKR systems. It should also be

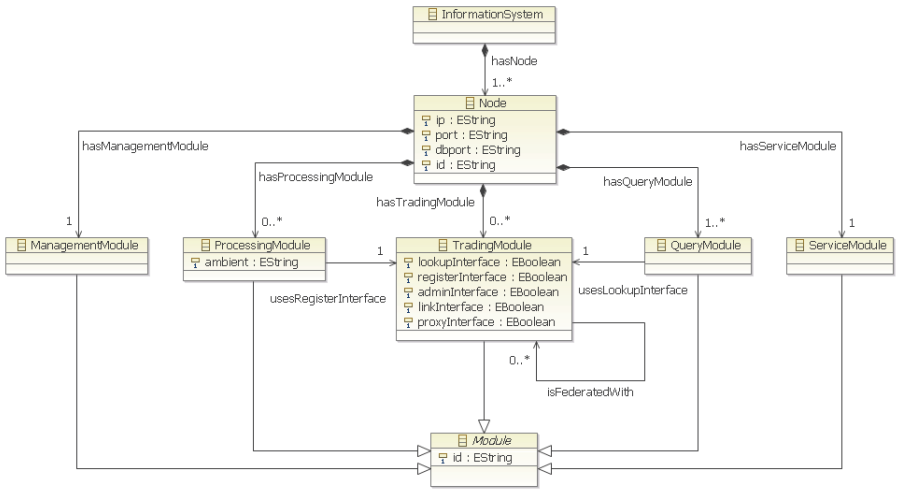

Fig. 1. TKR system metamodel.

noted the inclusion of the abstract *Module* metaclass, from which the different types of modules, previously described, inherit. Although this metaclass only includes the identifier (*id*) attribute, its aim is to gather in the future all the common attributes shared by the different modules. The rest of the metaclasses already include the necessary properties (i.e., their attributes and relations with other metaclasses). For instance, the following attributes need to be set for each *Node*: an identifier (*id*), an IP address (*ip*), a communication port (*port*) and a secondary port to set up the repository (*dbport*). Similarly, for each *TradingModule*, five *Boolean* attributes need to be set, each one indicating whether the trader implements or not the following interfaces: lookup (*lookupInterface*), register (*registerInterface*), admin (*adminInterface*), link (*linkInterface*) and proxy (*proxyInterface*). It is worth noting that both the lookup and the register interfaces are required because each trading module is inherently related to a metadata repository used for improving information retrieval [13]. The metamodel also enables the creation of TM federations (by means of the *isFederatedWith* relation) and the binding of each PM and QM (by means of the *usesRegisterInterface* and the *usesLookupInterface* relations, respectively) with the TM responsible of storing and querying the information in the metadata repository, respectively. However, the metamodel does not allow us to define certain constraints, such as the one that states that "a TKR system must have, at least, a TM and a PM in some of its nodes", or the one that states that "if there exists a federation between two TMs, then it is required that the link interfaces of both modules are enabled". In order to cope with this kind of syntactic restrictions that cannot be described in the metamodel, we have defined a set of OCL rules, further detailed in the following section. The proposed metamodel has been defined using the *Eclipse Modeling Framework* (EMF) [3], which pro-

vides the most widely used implementation of a subset of the OMG Standard *Meta-Object Facility* (MOF) [8], known as *Essential MOF* (EMOF).

3 A GMF Tool to Deploy TKR Models

In order to ease the creation of TKR system models, conforming to the metamodel described in Section 2, we have implemented a graphical model editor[3] using the Eclipse *Graphical Modelling Framework* (GMF) [4]. In order to implement GMF editors, the following process needs to be followed: (1) define a graphical representation for each metamodel concept, (2) define a toolbar that enables the creation of instances of each metamodel concept, and (3) define a mapping among each metamodel concept, its graphical representation, and its creation tool. GMF also allowed us to extend the syntactic rules defined in the metamodel with additional constrains, defined using the standard *Object Constraint Language* (OCL) [9]. Table 1 gathers three of these constraints: the first rule indicates that each TKR system must contain, at least, a TM and a PM in some of its nodes; the second rule makes sure that TMs are not federated with themselves; and the third rule makes sure that when two TMs are federated, their link interfaces are enabled.

Table 1. Some OCL constraints of the GMF tool.

```
rule #1:   context InformationSystem inv:
              self.hasNode -> exists(n | n.hasTradingModule -> size() > 0) and
              self.hasNode -> exists(n | n.hasProcessingModule -> size() > 0)
```

```
rule #2:   context TradingModule inv:
              self.isFederatedWith -> forAll(tm | (tm <> self))
```

```
rule #3:   context TradingModule inv:
              self.isFederatedWith -> notEmpty() implies (self.linkInterface = true) and
              self.isFederatedWith -> forAll(tm | (tm.linkInterface = true))
```

Figure 2 shows an example model created using the GMF graphical model editor developed as part of this work. The metamodel concepts appear in the toolbar shown on the right side of the editor window. The toolbar is divided into three types of instance creation tools, namely: a) those for creating nodes, b) those for adding modules inside each node, and c) those for creating relationships between some modules. Federation relationships are depicted using dashed arrows, while the relationships between PMs or QMs and their corresponding TMs are depicted using solid arrows. Some model element attributes are shown in the diagram as labels (e.g., IP address, ports, ambient, interfaces, etc.), while

[3] This tool is available at `http://www.ual.es/acg/soleres/tkr`

others are not for the sake of readability. Regardless of whether the attributes are graphically depicted or not, they can all be inspected and modified using the Eclipse properties tab.

The process to design a TKR system model using this GMF editor involves the following steps: (1) create as many *Nodes* as required —for each *Node*, its compulsory *Modules* (i.e., SM, MM, and QM) are automatically added; (2) for each *Node*, specify its *ip*, *port*, *dbport* and *id* attributes; (3) for each *Node*, if appropriate, add the corresponding PMs and TMs, taking into account that, at least, there should be one of each type in the entire system; (4) for each TM, select the interfaces it implements by setting the corresponding attributes — the *lookupInterface* and the *registerInterface* are automatically set to true, as described in Section 2; (5) connect the federated TMs (if any); and (6) link each PM and QM with a TM.

The model depicted in Figure 2 corresponds to the TKR subsystem of the SOLERES WMIS environment, called *Knowledge Representation Subsystem* (S-KRS[4]). This subsystem is comprised of three nodes, all of them containing, in turn, the compulsory modules SM, MM and QM. In addition, the nodes with *id* "Node_1" and "Node_2" contain a PM and a TM, respectively, fulfilling the minimum system configuration requirements. In addition, the system contains a federation (dashed arrow) between the TMs included in "Node_1" and "Node_2", representing that the first module may delegate its information queries to the second one. Finally, the required links (solid arrows) between all PMs and QMs and the corresponding TMs (belonging the same or to other Nodes) have been also defined in the model.

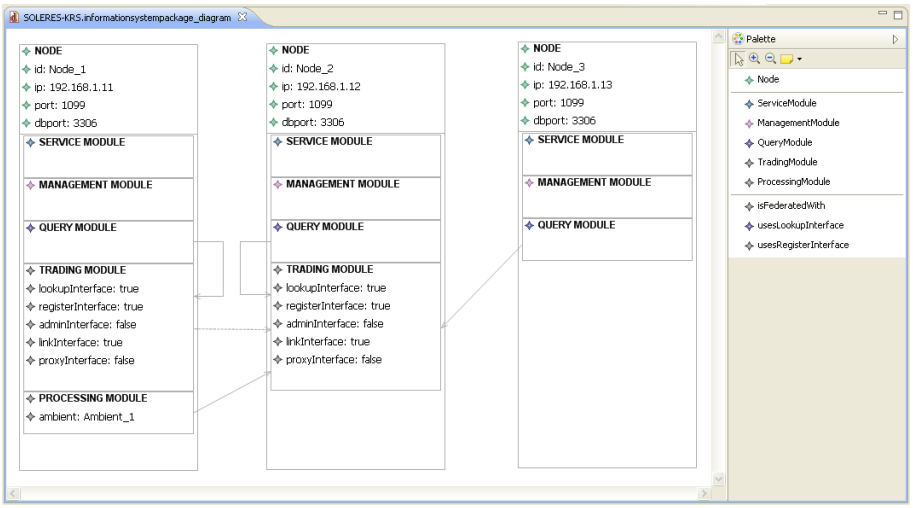

Fig. 2. GMF tool for designing TKR system models.

[4] SOLERES project is available at `http://www.ual.es/acg/soleres`

4 Conclusions and future work

In this work we have presented a MDE approach to TKR system design in the context of WMIS. We have presented a metamodel and a GMF model editor developed from it that aims to ease the design of this kind of systems in a fast and simple way through a graphical interface. A model of the SOLERES-KRS subsystem has been presented that illustrates the proposed design process using the developed GMF model editor. As future work, we plan to create an implementation repository for storing the binary components developed for these systems. Besides, we also plan to develop a configuration definition language that allows us to map (1) the design-time components, defined using the GMF model editor presented in this paper, and (2) the reusable binary components stored in the repository. The final goal of this research is to automatically obtain the implementation of the whole system from these configuration models.

Acknowledgment. This work has been supported by the EU (FEDER) and the Spanish Ministry MICINN under grant of the TIN2010-15588 and TRA2009-0309 projects, and the JUNTA de ANDALUCÍA (proyecto de excelencia) under grant TIC-6114 project, http://www.ual.es/acg/soleres.

References

1. J.A. Asensio, L. Iribarne, N. Padilla, and R. Ayala. Implementing Trading Agents for Adaptable and Evolutive COTS Components Architectures. *Int. Conf. on e-Business, Porto, Portugal*, pages 259–262, 2008.
2. O. El-Gayar and B. Fritz. Environmental Management Information Systems (EMIS) for Sustainable Development: a Conceptual Overview. *Communications of the Assoc. for Inf. Systems*, 17(1):34, 2006.
3. EMF. Eclipse Modeling Framework. `http://www.eclipse.org/modeling/emf`
4. GMF. Graphical Modeling Framework. `http://www.eclipse.org/modeling/gmf`
5. M. Huang. A New Method to Formal Description of Spatial Ontology. *Information Technology and Environmental System Sciences*, 3:417–421, 2008.
6. L. Iribarne, J.M. Troya and A. Vallecillo. A Trading Service for COTS Components. *The Computer Journal*, 47(3):342-357, 2004.
7. L. Iribarne *et al.* The SOLERES R&D Project: A Spatio-Temporal Environmental Management Information System based on Neural-Networks, Agents and Software Components. Applied Computing Group, University of Almeria, Spain, 2008.
8. OMG. Meta-Object Facility. Tech. Report, `http://www.omg.org/mof/`.
9. OMG. Object Constraint Language Specification, version 2.0. `http://www.omg.org/technology/documents/formal/ocl.htm`.
10. OMG. Trading Object Service Specification. `http://www.omg.org.`, 2001.
11. N. Padilla, L. Iribarne, J.A. Asensio, F. Muñoz, and R. Ayala. Modelling an Environmental Knowledge-Representation System. *WSKS'2008*, pages 70–78, 2008.
12. D. Taniar and J. Rahayu. *Web Information Systems*. IGI Global, 2004.
13. I. Trader. ISO/IEC DIS 13235-1: IT–Open Distributed Processing–ODP Trading Function–Part 1: Specification, 1996.
14. M. Xiao-feng, X. Bao-wen, L. Qing, Y. Ge, S. Jun-yi, L. Zheng-ding, and H. Yan-xiang. A Survey of Web Information Technology and Application. *Wuhan University Journal of Natural Sciences*, 11(1):1–5, 2006.