A Formal Process for Community-Based Reference Model Evolution for Smart Manufacturing Systems

Farhad Ameri[1(✉)], Boonserm Kulvatunyou[2], and Nenad Ivezic[2]

[1] Engineering Informatics Lab, Texas State University, San Marcos, USA
ameri@txstate.edu
[2] Engineering Laboratory, National Institute of Standards and Technology (NIST), Gaithersburg, USA
{boonserm.kulvatunyou,nenad.ivezic}@nist.gov

Abstract. Service-oriented manufacturing systems need to be supported by formal reference models for effective service description, discovery, and composition. A reference model should evolve continuously throughout its lifecycle to respond to changing requirements. The objective of this work is to propose a formal process for collaborative and community-based reference-model evolution and identify the computational components required for effective model evolution. The main steps of the proposed process include service registration, vocabulary extraction, evolution triggering, change evaluation and approval, and change implementation. An important feature of the proposed process is the use of a Simple Knowledge Organization System (SKOS) thesaurus for the initial stages of knowledge elicitation and organization.

Keywords: Ontology evolution · Service-oriented manufacturing · Smart manufacturing · Crowdsourcing reference models · Cloud manufacturing

1 Introduction

Distributed intelligence is one of the key distinguishing features of the already emerging, next-generation manufacturing systems. The new factories of the future are supported by *service-oriented, data-driven*, and *knowledge-based* systems [1]. Such factories will be operated and controlled by networks of interconnected devices and processes with embedded intelligence. These embedded intelligent agents will provide and consume a wide range of services such as process selection, production planning and control, inventory management, quality control, and maintenance planning. A service-oriented manufacturing enterprise comprises a network of loosely-coupled processes that can be configured and reconfigured in a timely manner in response to changing needs. This enhances the overall responsiveness, agility, and resilience of the system.

Although the idea of Service-oriented Manufacturing (SOM) has been around for more than a decade, there are still multiple infrastructural issues that need to be addressed before all the benefits of SOM can be realized in practice [2–4]. One of those issues is the development of shared reference models that are used for representation of the services. In the presence of shared and formal reference models, a human user, or an intelligent agent

S. Umeda et al. (Eds.): APMS 2015, Part II, IFIP AICT 460, pp. 30–38, 2015.
DOI: 10.1007/978-3-319-22759-7_4

embedded in the smart manufacturing system, can (1) offer and consume services, (2) perform inference and reasoning about service capabilities, and (3) participate in automated or semi-automated services composition and system configuration. Several types of reference models, such as information, functional, and resource models, are needed to describe different types of engineering services including engineering information services, cyber-physical system services, and hardware services. Without useful and effective reference models, ambiguity and imprecision ensues in communications among collaborators, stifling their efforts to enable smart manufacturing when using those manufacturing services.

A reference model, like any other ontology, is a living entity that evolves over time due to change in the conceptualization of the domain, business strategy evolution, or new user's needs. A reference model in a large manufacturing domain needs to be collaboratively and evolutionally developed [5]. For managing the evolution of ontologies that are maintained by a single organization and used by homogenous and collocated users, there are established processes based on data-schema evolution that have been successfully adopted and implemented [6]. However, the socio-technical aspects of the community-based, ontology-evolution process have not been adequately investigated and understood [7]. In this paper, we discuss a general framework for evolving a reference model from a service-oriented perspective. We collectively refer to all the necessary reference models as a single service reference model, or simply a reference model (RM).

There are multiple requirements for the model-evolution process in a collaborative and distributed setting. Most importantly, the evolution process should enable detection and resolution of semantic conflicts and ambiguities. Also, it should be a social process driven by community goals and interests. The evolution process should follow a structured methodology that (1) ensures long-term validity and consistency of the model and (2) involves all stakeholders including service providers, service users, and reference-model administrators and developers. The process should be transparent, formal, and timely.

Traditionally, the evolution of a reference model adopted for a data exchange standard is a batch process. The model or standard custodian either updates the model after a certain time period or waits until there are significant new requirements for change. In some cases, the models can evolve sooner, even though the updates will have no impact on existing usages. Conversely, some updates may cause large impact on existing usages. Formalizing the process means that automation could be applied as much as possible. Automation is preferred since manual handling of the evolution can be time-consuming and error-prone – especially as the size and complexity of the reference model increases.

The objectives of this work are (1) to propose a formal process for community-based reference model evolution and (2) to identify the computational components required for effective model evolution. The remainder of this paper is organized as follows. The next section lays out the necessary assumptions and definitions pertaining to the proposed process. The model evolution process is discussed next, followed by an example scenario. The paper ends with some closing remarks.

2 Assumptions and Definitions

Assumptions:

- The reference model contains the classes, properties, and axioms required for describing different types of manufacturing services.
- The proposed evolution process only applies to extensions of the reference model. The evolution of the core of the reference model may call for a different set of procedures and is out of scope.
- The reference model uses Ontology Web Language Description Logic (OWL DL) syntax and semantics.
- Services need to demonstrate Minimal Ontological Agreement (MOA) with the reference model before they can be registered for discovery. In other words, the service description that uses the reference model to formally describe the service must at least agree with the core of the reference model.
- There are multiple sub-problems related to ontology evolution including backward compatibility, change propagation, and consistency maintenance [8]. In this paper, we discuss only the overall process of ontology evolution.

Definitions:

- Service: A service is a complete, self-contained, unit of functionality with well-defined interfaces and possibly quality measures.
- Service Scheme: Service scheme is an ontology fragment describing conceptual-level classes, properties, and axioms used in defining a service. Ontology design patterns may also be part of the service scheme.
- Service Provider: An organization responsible for description and provision of a service.
- Reference Model Evolution: Reference model evolution is defined as a timely enrichment, extension, or adaptation of the model to the arisen changes through subtractions and additions of the classes, properties, and axioms while maintaining the semantic consistency of the model. Propagation of the changes to the depending artifacts (i.e., service descriptions) is also a part of the evolution process.
- RM Admin: RM admin is in charge of overseeing the evolution process and directing the workflow.
- RM Engineer: RM engineer is an expert in ontology development and knowledge modeling.
- RM Community: RM community comprises domain experts in those technical fields related to the concepts in the reference model. Community members may play multiple roles including the service provider and RM engineer.
- Service Registration: Service registration refers to a formal request to add a service to the service registry. Upon registration, the service description is accessible to the service registry user/agent. A service registration event may or may not trigger an evolution.
- Optimistic Service Registration: The service is accessible right away for discovery after registration despite any possible misalignment between the service scheme and

the reference mode (note however that the service scheme still has to align with the core of the RM, i.e., satisfying the MOA). This strategy allows the service to still be used while it may not be easily discovered and composed.
- Pessimistic Service Registration: The service is available for discovery only after its internal service scheme is aligned with the reference model.
- Primitive change: A change that affects only a single OWL primitive.
- Complex Change: A change that affects more than one OWL primitive.

3 Reference Model Evolution Process

The reference model evolution process proposed in this paper is based on the optimistic service registration scenario. Under this scenario, service registration and RM evolution are decoupled, yet related, processes and are modeled in the same flow. In the proposed RM evolution process, the reference model is supported by a SKOS (Simple Knowledge Organization System) [9] thesaurus tool, which provides linguistic and semantic analysis of the concepts and terms used in the model. The SKOS model also makes the link between the lexical (terms for concepts) model and conceptual model explicit. Such features help in harmonizing jargons in a distributed, multi-field environment. The main steps of the service-registration and model-evolution processes are described below.

Service Registration: The registration process starts with a *service provider* submitting a *service registration package* containing multiple components including the service scheme, corresponding description, and, possibly, installation files. By comparing the service scheme with the core RM, the MOA prerequisite is verified and if the service meets this prerequisite, it is queued for registration. This prerequisite ensures that the service scheme is in agreement with the core conceptual model of the RM. The service is then registered and stored in a temporary space of the service registry and is made available for discovery and use by service users. Expiration may be applied to services in the temporary registry. The workflow of the service registration process is shown in Fig. 1.

Vocabulary Extraction: The service scheme included in the service registration package contains the vocabulary of the service - classes and properties. The vocabulary is extracted automatically from the service scheme and saved as a SKOS model under the category of *Candidate Vocabulary*.

Change Process Triggering: The RM change process can be triggered automatically either after a predetermined time has elapsed or after a certain event takes place. Examples of the events that can trigger the change process include (1) a threshold is reached on the number of candidate concepts, (2) a semantic conflict between the candidate concepts and the RM occurs, (3) semantic gap in the RM is identified, or (4) a change in the usage behavior or goal of the system users is observed. While events in #1 can be detected totally automatically, the rest will be detected by a combination of human and computational algorithms. In particular, bottom-up (inductive) change recognition algorithms are frequently used [7].

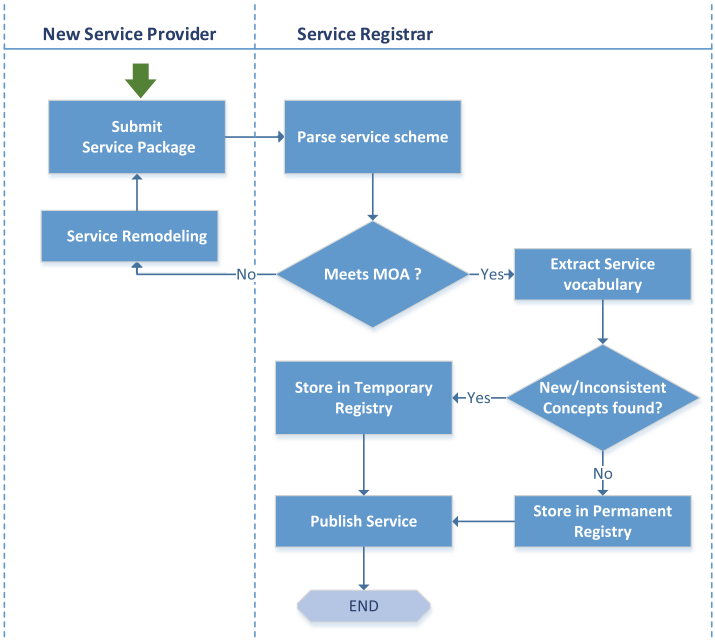

Fig. 1. Service registration process

The RM engineer can also trigger the change process manually to adapt the ontology to new requirements or enhancements. Enhancements refer to the usability of the ontology, which is based on feedback from the end users. In either case, the RM engineer creates the change request centrally and initiates the evolution process. The RM engineer also conducts an impact analysis to identify the consequences of the change. The formal change request together with the impact analysis result is sent to the RM admin for approval. Figure 2 shows the workflow of the change approval process regardless of whether or not the change is triggered automatically or manually.

Change Evaluation and Approval: The proposed change should be first approved by the RM community and then by the service owners who will be affected by the change. The change approval process is administered and managed by the RM admin. The admin first notifies the RM community of the details of the requested change and its nature and scope. The change can be as simple as renaming a class or attribute (*primitive change*) or it can involve adding new ontology constructs composed of multiple classes, properties, and axioms (*complex change*).

The RM community members, who have vested interest in the impacted artifacts, evaluate the requested change with respect to technical and terminological validity and viability. These interested members may come from subscriptions to particular concepts or based on prior services for which they had registered. Once the RM community approves the proposed change, the RM admin identifies the registered services, which will be affected as a result of the change, and submits the proposed changes to providers of those services.

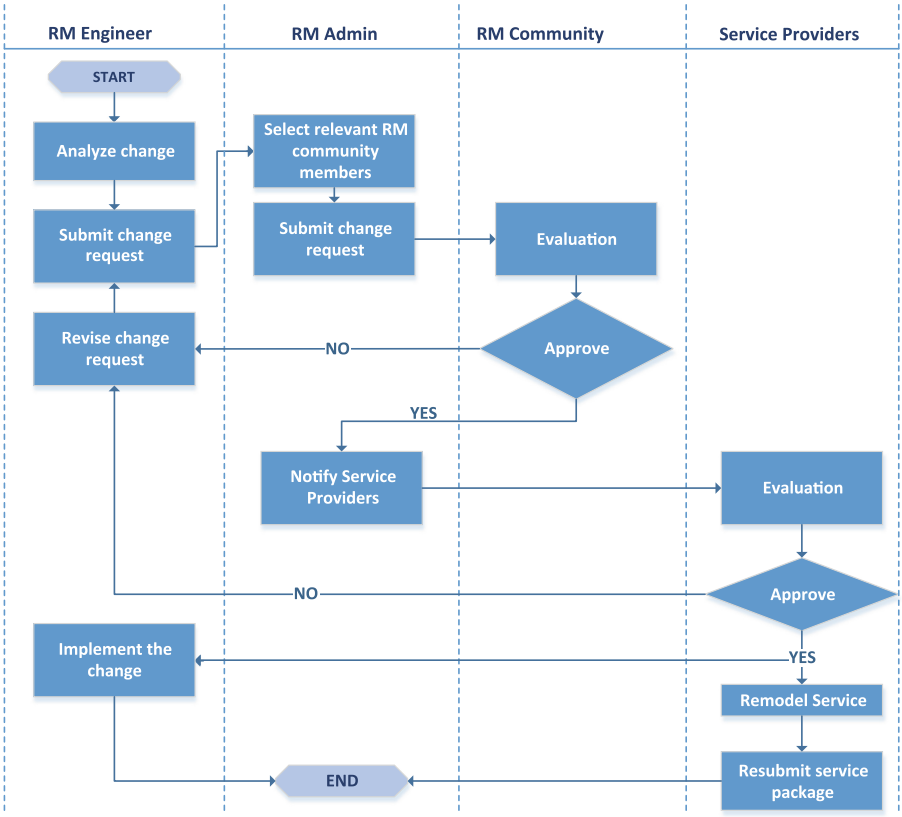

Fig. 2. RM change process

If a majority of the service providers approve the proposed changes, the change is formally approved and implemented in the reference model by the RM engineer. The affected services are moved to the temporary registry space. Service providers then restructure those services based on the proposed change and resubmit a new service registration package, which contains the new service scheme, for registration (Fig. 2). If the proposed change is rejected by a majority of the affected service providers, the RM admin requests the RM engineer to redefine the change based on the feedback received and submit a new request or discard the change request.

Change Implementation: If a change request is approved, the RM needs to be updated accordingly. The RM engineer is in charge of applying the change to the thesaurus and the ontology. The concepts and properties that are approved are moved to the *Approved Vocabulary* scheme of the SKOS thesaurus. Then the corresponding changes are applied to the ontology itself. This involves both required change (directly modifying the ontological entities mentioned in the request) and derived change (discovering new changes based on the explicit changes).

4 Example

In this section, we describe the proposed service registration and model evolution process through an example scenario. The service used in this example receives operational data from an MTConnect [10] agent of Computer Numerical Control (CNC) machines through Hyper Text Transfer Protocol (HTTP) requests and calculates the energy consumption for the given set of machining instructions written in G-code. This service is named Machining Energy Consumption Calculator (Mecca) by the service provider. At the time of registration, the service scheme is extracted from the submitted service registration package as shown in Table 1.

Table 1. The extracted vocabulary for Mecca

Classes	Vertical Milling Machine, Material, Part, Energy, Power, Cutting Tool, Spindle, Machining Operation
Object props	hasSpindle, acceptsMaterial, hasCuttingTool
Data-type props	Feed rate, spindle speed, depth of cut, specific energy, machinability rating, cut length in X, cut length in Y, cutting energy

The extracted vocabulary is then exported to the SKOS tool as shown in Fig. 3. The service scheme is also recreated in the SKOS tool to maintain the relationships between Mecca model entities. Through a comparison between the Mecca service scheme and the reference model, the following discrepancies are detected:

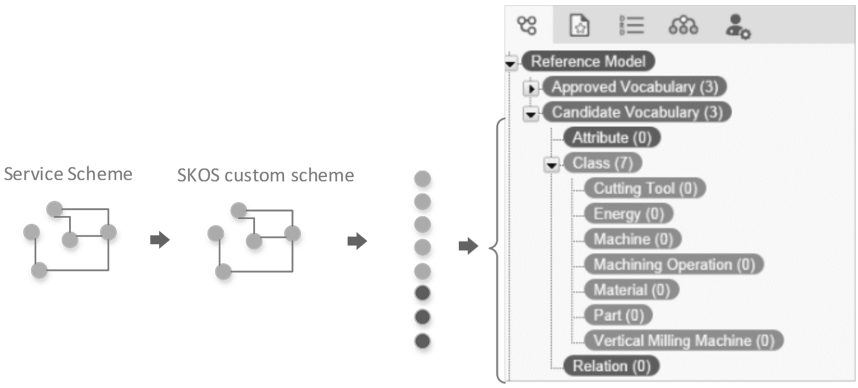

Fig. 3. Vocabulary extraction

- D1: *machinability rating* is an attribute of the Material class in Mecca, but the Material class doesn't have such attribute in the RM.
- D2: *spindle speed* is an attribute of the Vertical Milling Machine class in Mecca, but *spindle speed* is an attribute of the Spindle class in the RM.

In this example scenario, we assumed that the change process is triggered when the accumulated number of candidate entities (class, property, relationship) reaches a threshold. Once the change process is triggered, the RM engineer creates a formal change request and submits it to the RM admin. The RM admin then routes the change workflow to the appropriate RM community subgroups for evaluation and approval. The RM community approves the addition of *machinability rating* as a new attribute of the Material class. A notification is then sent to all providers who use the Material class in their service scheme. Since no conflict arises on the service providers' side, the change receives final approval and the RM engineer updates the RM accordingly.

The RM community, however, maintains that it is more appropriate to keep the spindle speed as a property of the Spindle class rather than a property of the Machine class (note that the Machine in turn owns a Spindle).

The RM community, however, maintains that it is more appropriate to keep the spindle speed as a property of the Spindle class rather than a property of the Machine class (note that the Machine in turn owns a Spindle in the reference model). Therefore, the RM admin notifies Mecca service provider about the need for service description revision. The provider agrees to revise and resubmit the service description. To assure an effective revision, the RM engineer points the service provider to a library of modeling patterns that have been approved for use in the RM development and evolution process.

5 Closing Remarks

In this paper, we proposed a methodical process for community-based reference model evolution and identified the necessary computational requirements. The main steps of the proposed framework include service registration, vocabulary extraction, change request submission, change evaluation and approval, and change implementation. An important feature in the process is the use of a SKOS thesaurus for the initial stages of knowledge elicitation and organization. The thesaurus separates meaning and labels; thus, simplifying the resolution between model developers and service providers. Also its simple semantics facilitates active involvement of community members in the evolution process.

There are multiple issues that need to be addressed before the proposed framework can be fully utilized. Currently, protocols that guide service providers in creating services that are conceptually in agreement with the reference model do not exist. Also, the components of the service registration package should be formally defined. Semantic-and-linguistic analysis and mapping of the incoming service schemes are technical challenges that need to be addressed. The events that can trigger the evolution process

also need to be defined more precisely. Developing the machine-learning algorithms and data-mining techniques that support change discovery, based on ontology instances and usage patterns, is another future task. Other areas that need further investigation include impact analysis, change propagation, and change validation.

Acknowledgement. The work described in this paper was funded in part by NIST cooperative agreement with Texas State University No. 70NANB14H255.

References

1. EFFRA, European Factoris of the Future Association. Factories of the Future: Multi-annual Roadmap for the Contractual PPP Under Horizon 2020, Publications office of the European Union, Luxembourg (2013)
2. Ivezic, N., Kulvatunyou, B.S., Srinivasan, V.: On architecting and composing through-life engineering information services to enable smart manufacturing. In: Procedia CIRP (2014)
3. German Academy of Science and Engineering. Recommendations for implementing the strategic initiative INDUSTRIE 4.0, Securing the future of German manufacturing industry, Germany (2013)
4. Karnouskos, S., Colombo, A., Bangemann, T., Manninen, K., Camp, R., Marcel, T.: A SOA-based architecture for empowering future collaborative cloud-based industrial automation. In: Annual Conference on IEEE Industrial Electronics Society (2012)
5. Kulvatunyou, B.S., Ivezic, N., Lee, Y.: On enhancing communication of the manufacturing service capability information using reference ontology. Int. J. Comput. Integr. Manuf. **27**(12), 1105–1135 (2014)
6. Maedche, A., Motik, B., Stojanovic, L.: Managing multiple and distributed ontologies in the semantic web. Int. J. Very Large Data Bases **12**(4), 286–302 (2003)
7. Aufaure, M.A., Djedidi, R.: Ontology evolution: state of the art and future directions. In: Ontology Theory, Management and Design: Advanced Tools and Models, pp. 179–207 (2010)
8. Plessers, P., Troyer, O.D., Casteleyn, S.: Understanding ontology evolution: a change detection approach. Web Semant. Sci. Serv. Agents World Wide Web **5**(1), 39–49 (2007)
9. Simple Knowledge Organization System (SKOS) Reference. http://www.w3.org/2004/02/skos/
10. Vijayaraghavan, A., Sobel, W., Armando, F., Dornfeld, D., Warndorf, P.: Improving machine tool interoperability using standardized interface protocols: MT connect. In: 2008 Symposium on Flexible Automation, Atlanta, GA (2008)
11. Gruber, T.R.: Toward principles for the design of ontologies used for knowledge sharing. Int. J. Hum. Comput. Stud. **43**, 907–928 (1993)