Advanced Engineering Informatics 29 (2015) 1041–1054

Contents lists available at ScienceDirect

Advanced Engineering Informatics

journal homepage: www.elsevier.com/locate/aei

Planning-based semantic web service composition in factory automation

Juha Puttonen [a,*], Andrei Lobov [a], Maria A. Cavia Soto [b], José L. Martinez Lastra [a]

[a] Tampere University of Technology, Fast-Lab., P.O. Box 600, FIN-33101 Tampere, Finland
[b] Universidad de Cantabria, Department of Electrical and Energy Engineering, Avda. De los Castros s/n, 39005 Santander, Spain

ARTICLE INFO

Article history:
Received 18 February 2015
Received in revised form 25 August 2015
Accepted 30 August 2015
Available online 26 September 2015

Keywords:
Semantic web services
Web service composition
Factory automation

ABSTRACT

The Service Oriented Architecture (SOA) paradigm enables production systems to be composed of web services. In an SOA-based production system, the individual production devices provide web service interfaces that encapsulate the behavior of the devices and abstract the implementation details. Such a service-oriented approach makes it possible to apply web service orchestration technologies in the development of production workflow descriptions. While manual formulation of production workflows tends to require considerable effort from domain experts, semantic web service descriptions enable computer algorithms to automatically generate the appropriate web service orchestrations. Such algorithms realize AI planning and employ semantic web service descriptions in determining the workflows required to achieve the production goals desired. In addition, the algorithms can automatically adapt the workflows to unexpected changes in the goals pursued and the production devices available.

1. Introduction

The rapid development of microprocessors has dramatically increased the computational power embedded into miniature devices [1]. Consequently, the constituent devices of a modern manufacturing system are controlled by embedded computers. The increased use of computers has rendered manufacturing systems software-intensive. However, software developed by various vendors tends to be incompatible, and the adoption of new architectures may cause system integration problems in factory automation [2] as well.

The SOA paradigm enables system integration problems to be relaxed by encapsulating each device as a web service. When a device controller exposes a web service interface providing access to the device functionality, the implementation details, such as any vendor-specific programming languages, become less relevant [2]. Furthermore, production systems consisting of web services are highly reconfigurable [3]. For example, the workflow for producing a certain type of product using the production system devices can be formulated as an executable Web Services Business Process Execution Language (WS-BPEL) [4] process. BPEL processes extensively employ the syntactic web services descriptions formulated in the Web Services Description Language (WSDL) [5].

While WSDL defines the communication syntax of a web service, it conveys no information on the service semantics, which would be essential for the dynamic discovery of services suitable for a task [6]. Nevertheless, when semantic information is included, it facilitates the automated discovery, invocation, and composition [7] of the resulting semantic web services. Thus, encapsulation of production devices as semantic web services facilitates automatic composition of production workflows [3,8].

Successful integration of diverse production devices requires the semantic information to be explicitly formulated in a machine-interpretable language [9]. OWL-S (OWL for Services) [10] is currently one of the most popular languages for semantic web service descriptions. It is essentially an ontology formulated using the Web Ontology Language (OWL) [11]. OWL is a vocabulary extension for the Resource Description Framework (RDF) [12].

This paper is organized as follows. Section 2 presents earlier research on which the methodology proposed in Section 3 is based. The methodology is then applied to a manufacturing system in Section 4. Finally, Section 5 presents the conclusions on the application results and identifies points of further work.

2. Related research

Hatzi et al. [13] have developed a framework that translates OWL-S processes describing web services into Problem Domain Definition Language (PDDL) [14] descriptions, which the framework submits to external AI planning components. Based on the solution plans acquired, the framework generates composite

* Corresponding author.
E-mail addresses: juha.puttonen@tut.fi (J. Puttonen), andrei.lobov@tut.fi (A. Lobov), maria.cavia@unican.es (M.A. Cavia Soto), jose.lastra@tut.fi (J.L. Martinez Lastra).

http://dx.doi.org/10.1016/j.aei.2015.08.002
1474-0346/

OWL-S processes. Hatzi et al. compare the efficiency of different AI planners in computing the solution plans, and they intend to further extend the system with the deployment of the composite processes using external OWL-S facilities.

Semantic service descriptions, such as OWL-S processes describe the meaning of the services in terms of ontologies. While the manual formulation of such service descriptions requires considerable effort from domain experts, Paolucci et al. [15] point out that OWL-S descriptions can be automatically derived based on SAWSDL annotations. However, such automatically derived OWL-S descriptions include no information on the service pre- and post-conditions [15].

The approach presented in this article differs from most contemporary service composition approaches in the method of acquiring the OWL-S descriptions of the constituent services. Furthermore, the service composition framework presented in this paper includes a built-in planner component while retaining the option to attach external planners through a plugin mechanism. In addition, the explicit conversion of service descriptions to PDDL is omitted, unless the planning problems are submitted to external planners. Since the framework itself consists of a set of interacting web services, the services in the actual production systems are termed domain services.

The service composition framework proposed in this article includes a planner that automatically composes the domain web services to achieve production goals. An earlier version of the framework [16] required the service orchestration instructions to be specified through externally provided BPEL processes.

In the proposed methodology, a domain ontology is required for decision-making and planning. However, ontologies have been applied in various information processing tasks. For example, Rijgersberg et al. [17] have developed an ontology of measurement units that facilitates the exchange of quantitative information.

The framework proposed in this article employs semantic information on web service capabilities to determine the appropriate composition of services to achieve a production goal. Schubert et al. [18] present a framework in which semantic information somewhat analogously describes the participants of virtual organizations and the services they provide.

The production goals considered in this article differ in detail to those considered by Shea et al. [19]. This article focuses on assembly goals specifying the part types to be attached to a product template, whereas Shea et al. [19] consider fabrication goals specifying more qualitative properties of the products, such as geometry.

Lobov et al. [20] outline the basic principles for the proposed service composition framework. However, in the new framework discussed in this article, some implementation technologies, such as multi-agent systems are omitted, and the *Orchestration Engine* component is considerably less essential.

Ramis et al. [21] have proposed a knowledge-based framework for modeling production system statuses. The central framework component corresponds to the *Ontology Service* component discussed in the next section. However, Ramis et al. [21] suggest hosting the service on cloud resources to enable external agents to access the data.

Lastra et al. [22] have presented a framework for composing composite services in agent-based production systems. The semantic web service composition framework presented in this article is somewhat analogous, although it operates on a higher level where physical modeling is less critical.

Huckaby et al. [23] have applied PDDL in describing robotic production systems and the desired production goals. They have shown that the required workflows can be automatically acquired through AI planning, thus reducing the effort required from the end user. In addition to automated planning, this article will consider automated solution plan deployment, which is possible when the production devices are encapsulated as semantic web services.

Keddis et al. [24] have applied AI planning in scheduling the operations of production plans. They propose an algorithm that considers the temporal dependencies between individual operations to produce schedules with valid material flow between machines. The consideration of material flow enables manufacturing systems to automatically adapt work schedules to equipment changes. In addition, the algorithm is able to generate schedules involving the least idle time [24].

Jian and Ai-Ping apply genetic algorithms to manufacturing cell layout optimization problems [25]. Genetic algorithms start from a randomly selected initial population of solutions, which the algorithms iteratively refine until obtaining optimal solutions [26]. While genetic algorithms appear a promising approach to solving planning problems without exhaustive state space search, their application in AI planning is omitted in this paper.

While PDDL is widely applied in automated planning, Anis et al. [27] point out that the selection of modeling language affects the amount of modeling effort required. Furthermore, Anis et al. [27] compare PDDL, Prolog, and Timed Automata in terms of, for example, usability and performance.

3. AI planning based on semantic web service descriptions

This Section presents an approach to deriving OWL-S descriptions of web services and applying the descriptions in service composition.

3.1. Service composition framework overview

Decentralized web service orchestration can be achieved through the use of a web service-based framework, such as the Orchestration Tools proposed in [28]. The framework consists of three web services, *Ontology Service*, *Service Monitor*, and *Orchestration Engine*, each of which fulfills a different task in service orchestration. Detailed descriptions of all the framework components are included in [29].

Ontology Service hosts a semantic model of the production system. It receives update requests from an event listener service, *Service Monitor*, as state changes manifest in the production devices. Thus, the OWL model remains synchronized with the current system status [30].

The primary responsibility of *Service Monitor* is, however, to compose and deploy workflows that achieve production goals. The goals are formulated as SPARQL queries and submitted to *Service Monitor* by invoking the *StartGoal* operation. As indicated in the sequence diagram of Fig. 1, *Service Monitor* sends a *GoalStatusChanged* event notification, for example, when a new goal is registered or achieved.

Service Monitor creates a separate goal achievement process for each goal received. A goal achievement process is initially in the *pending* state and proceeds to the *planning* state when it is submitted to a planner. Once the planner has produced a solution plan, the process continues to the *executing* state. Once the plan has successfully been carried out, the process will remain in the *completed* state.

3.2. Extracting OWL-S descriptions from web services

This Section presents a set of conventions on the application of SAWSDL annotations that facilitate automatic derivation of the preconditions and effects of OWL-S processes. Thus, the conventions overcome one of the main deficiencies in the derivation of OWL-S processes identified by Paolucci et al. [15].

An OWL-S model can be extracted from any WSDL document containing sufficient information for invoking a web service.