Automating Integration of Manufacturing Systems and Services: A Semantic Web Services Approach

Zhonghua Yang1, Robert Gay1, Chunyan Miao2, Jing-Bing Zhang3, Zhiqi Shen1 and Liqun Zhuang3, and Hui Mien Lee1
1School of Electrical and Electronics Engineering
Nanyang Technological University, Singapore 639798
2School of Computer Engineering
Nanyang Technological University, Singapore 639798
3Singapore Institute of Manufacturing Technology (SIMTech), Singapore 638075

eZhYang@ntu.edu.sg

Abstract— In this paper, we present a Semantic Web Services approach to automated integration of manufacturing systems and services. A service-oriented integration architecture, enhanced with semantics, is presented. We construct a service-oriented environment via virtualization by taking a WS-Resource Framework approach. All Web services (newly developed manufacturing services or those from the virtualization) are semantically enhanced using service upper ontology OWL-S and manufacturing domain ontologies which are written in standard Web Ontology Language (OWL). Within this semantic rich environment, integration is achieved by Web services composition, and automation is accomplished via the combination of Semantic Web and software agents.

I. INTRODUCTION

Enterprise integration refers to “the vertical and horizontal alignment of plans, business processes, and information systems across organizations and functional boundaries to provide competitive advantage” [14]. Typically, it is the process of integrating multiple IT-related services and facilities that were independently developed, may use incompatible technology, and remain independently managed, including the applications, processes, various data sources (including databases or files), and legacy systems involved in an organization’s critical business processes. For decades, enterprise integration for manufacturing has been notoriously complex and challenging. There are three conventional approaches to enterprise integration: “Data centric approach”, “process-oriented approach”, and “message-oriented approach”.

Generally, all these approaches and attempts have relied on proprietary APIs and library-call-like interactions, requiring a high degree of coordination. They do not enable open interoperable solutions.

The emergence of Web services and Semantic Web services opens the new opportunities for enterprise integration and provides great potentials for automated integration. Many of the concepts for Web services come from a conceptual architecture called service-oriented architecture (SOA). SOA is an architectural style for building software applications that use services available in a network such as the Web. Service-oriented architecture is not a new concept, and not necessarily built using Web services, but SOA with Web services becomes the default in which the concept of services refers to as network-enabled entity delivered over the web using XML-based technologies.

SOA provides a great level of flexibility and higher level of abstraction in the following ways:

• Services are network-enabled components with well-defined interfaces that are implementation-independent.
• Such services are consumed by message exchange and clients that are not concerned with how these services will execute their requests.
• Services are self-contained to offer functionality as defined in interfaces.
• Services are loosely coupled, and thus more independent.
• Services can be dynamically discovered
• Composite services can be built from aggregates of other services

Web service-oriented approach provides a standard-based higher level of abstraction than the conventional message oriented approach. It standardizes the various aspects of message exchange (including message format, message exchange pattern, and message transport protocol). The service-oriented architecture (SOA) and Web services technologies are rapidly emerging as the most practical, and cost-effective approach for integrating a wide array of manufacturing resources in the manufacturing services environment. By adopting this standard-based service-oriented model, this approach makes all components of the environment virtual. As described late, Within this paradigm, there appears a standard framework for virtualization.

In this paper, we explore the service-oriented paradigm for achieving enterprise integration based on Web service standards. As an industry analysis shows, the 95% of difficulties in integration requires to resolve semantic issues which surfaces in the many aspects on integration. We explicitly address the
semantic issue in the integration using the Semantic Web technologies. We attempt to achieve automated integration using software agents. A service-oriented integration architecture is proposed in a manufacturing setting. In this architecture, the service-orientation is realized via virtualization taking a WS-resource Framework approach; the service-oriented manufacturing environment thus built is further enhanced with semantics using Semantic Web technology. In other words, these Web services become Semantic Web services which can be composed for integration as required by the business processes. In this process, automation can be achieved via software agents mediation.

II. SERVICE-ORIENTED INTEGRATION ARCHITECTURE

The overall semantic-enhanced service-oriented integration architecture is shown in Figure 1. Fundamental to the service orientation is the virtualization layer. Via a virtualization, all manufacturing systems to be integrated are exposed as Web services, and they can offer their services and functionality in a standard Web service environment. The emerging WS Resource Framework provides a standardized way for the virtualization [4], [1]. Using WS-RF, manufacturing resources will be modeled as a WS-Resource which is the combination of the Web Service and the stateful resource (Figure 3). The ultimate objective of virtualization is the construction of a service-oriented manufacturing environment, and all services are hosted in a standard Web service platform. The basic Web service infrastructure (SOAP-WSDL-UDDI) is augmented with Business Process Execution Service to support automated business process execution. This is the Web services infrastructure layer.

![Fig. 1. Automated Integration Architecture based on Semantic Web Services Approach](image)

All the Web services, newly developed manufacturing services or those from the virtualization, will be enhanced with domain specific semantic information. The emerging Semantic Web standards provide the Web services infrastructure with the semantic interoperability that integration needs. It provides formal languages and ontologies, most notably OWL and OWL-S, to reason about service description and capability [8]. A Semantic Web service is a Web Service whose description is in a language that has well-defined semantics [17]. The description is therefore unambiguously computer interpretable and facilitates maximal automation and dynamism in Web service discovery, selection, composition, negotiation, and invocation. This feature provides the sound technical foundation for achieving automated integration. Semantic Web transforms the Web service infrastructure into repository of computer manipulable data with rich semantics. Architecturally, this is achieved in the Semantic-enhanced Web services layer. The significance of this architectural design is that we obtain a semantic rich Web services environment and the semantics marked up in service upper ontology OWL-S and manufacturing domain ontologies are readily processable by autonomous agents [9].

As indicated above, semantic issues surface in the arena of enterprise integration as a consequence of coherent heterogeneity. In order to solve the semantic-level integration, adaptation and mediation is necessary. The Adaptation layer is designed to provide strong mediation. Generally, strong mediation enable integration and interaction in a scalable manner.

The automation with Agents and Semantic Web layer finally realizes the automated integration. The issues addressed at this layer are two folds: business process integration and dynamic service integration.

The requirements of enterprise integration are first modeled as business process integration. The essence of business process modeling is to create a virtualized enterprise (VE) [13] which represents a coherent, integrated (and interoperated) business process for achieving enterprise’s ultimate strategic business goals. There are many approaches and tools available for business process modeling (such as ARIS) [6]. The integrated business process is enabled and realized using IT technology and infrastructure. Using the service oriented paradigm, the problem of business process integration is transformed into service processes. In effect, this transformation turns business processes into the service-oriented requirements which is represented in OWL-S (Figure 2) [8].

![Fig. 2. Service-oriented approach to business process integration](image)

For the effective decision-making on integration issues, it is crucial to establish a community of trust among parties involved. In this context, trust is a entirely different aspect from security. Even if messages are secured and non-repudiated does not imply that endpoints trust each other. A supplier might promise to deliver goods at a specific time and in a specific quality, but it is not guaranteed that this happens every single time. Therefore, trust management is fundamental for discovering and selecting Web services for composition. The related issues are service negotiation and agreement. It is noted that the enterprise integration must address the dynamics of both the processes and services that realize the processes. The negotiation and agreement-based approach appears to be the promising one in a dynamic, service-oriented environments. As seen from the architecture (Figure 1), these two issues are orthogonal to other integration issues addressed in the
architecture.

In a service-oriented approach to enterprise integration, Web services become an integral part of the enterprise IT landscape, and, as such, are vital resources to enterprises. Managing such a service-oriented environment is critical for enterprises that use Web services to automate and integrate various internal functions, and deal with partners and clients electronically. Our proposed approach is based on OASIS standard on Web service management and has two aspects: (1) Management Using Web Services and (2) management of the Web services resources via the former [15].

In the following sections, we focus on the three key elements of the proposed Semantic Web services approach to integration: (1) building a service-oriented environment via virtualization, (2) integration via Semantic Web services composition, and (3) Automated integration via software agents and Semantic Web.

III. SERVICE-ORIENTATION VIA VIRTUALIZATION

As emphasized previously, fundamental to realizing service-oriented enterprise integration is virtualization, creating a service-oriented environment. Every thing in a considered environment (machine, devices, equipments, data sources, processes and applications) is required to be virtualized as a Web service. In other words, via virtualization, all manufacturing systems are exposed as Web services, and they can offer their services and functionality in a standard Web service environment.

The benefits and advantages of service virtualization are well-documented [3]. Virtualization enables consistent resource access across multiple manufacturing systems and environments. Virtualization also enables mapping of multiple logical resource instances onto the same physical resource and facilitates management of resources within/across enterprises based on composition from lower-level resources. Further, virtualization lets us compose basic services to form more sophisticated services-without regard for how these services are implemented. Virtualizing services also underpins the ability to map common service semantic behavior seamlessly onto native systems and facilities.

Web services are generally stateless. It is argued that although Web service’s behavior is defined solely by the message exchanges through the interface description supported by the service, the external observable behavior implies the existence of a stateful resources that are used and manipulated in the processing of a Web service request message. Particularly in manufacturing setting, many resources (machine and equipment, inventory state, process state) have a strong notion of state that resources are in. It is important to identify and standardize the patterns by which state is represented and manipulated, so as to facilitate the construction and use of interoperable services. Web Service Resource Framework (WS-RF), currently under standardization by OASIS, provides a standard based approach for creation, addressing, inspection, and lifetime management of stateful resources. The framework provides the means to express state as stateful resources and codifies the relationship. In WS-RF, a WS-Resource is the combination of a Web service and a stateful resource on which it acts (Figure 3). A WS-Resource has various properties. The values of these properties define the state of the resource. The change of a property value, the state changes.

As an illustration of virtualization based on WS-RF, we have virtualized a physical AGV (Automatic Guided Vehicle) system [7]. An Automated Guided Vehicle system is a material handling system in which driverless, battery-powered carts are moved by means of Laser, Optical or Electronic guidance. The vehicles themselves may be towing vehicles, fork lifts, pallet trucks, or unit load carriers.

When an AGV is virtualized as Web services, they offer different views of the AGV, such as Physical, Operational, Performance, and Financial view. The view represents an abstraction of the physical system. In WS-RF, different views of an AGV are defined in WS-ResourceProperty [5]. The resource properties document type associated with a Web service’s WSDL 1.1 portType definition provides the declaration of the exposed resource properties of the WS-Resource[5]. It represents a particular composed structural view or projection of the resource properties of the WS-Resource, essentially exposing the stateful resource component within the WS-Resource composition.

Once a stateful resource is virtualized as a WS-Resource, we can manipulate the resource using the Web services associated with the resource. Very often, WS-Notification is used to obtain notifications of various events (e.g., state changes). WS-Notification is a family of specifications that define a standard way for Web service clients to subscribe to a particular topic and receive notification of various events.

It has to be pointed out that WSRF actually refers to several different specifications

- **WS-ResourceProperties (WSRF-RP)** specifies the form in which ResourceProperties are defined in a WSDL file. It also specifies the form of messages that request and receive the values of these properties, and explains how to change, add, and remove properties from a WS-Resource.
- **WS-ResourceLifetime (WSRF-RL)** specifies the lifecycle. For example, a WS-Resource needs to expire or be explicitly destroyed when it’s no longer needed.
- **WS-ServiceGroup (WSRF-SG)** defines a way to create a collection of Web services, such as a registry of available services.
- **WS-Base Faults (WSRF-BF)** defines a standard way of indicating errors in a WSRF-based application.
IV. INTEGRATION VIA SERVICE COMPOSITION

Integration of an enterprise is motivated by some business value proposition to achieve some business goals which are captured by business process models. In order to use IT, especially Web service oriented solutions to realize the business process models, it is necessary to transform the business process models into service-oriented process model, resulting in a service-oriented requirement (or goals). This service-oriented process is a high level description of what to be done, and can be realized via service composition (Figure 2).

Fig. 4. Semantic Web service composition

Generally, the problem of composing Web services to fulfill a high level goal (objective) can be reduced to the following fundamental problems (Figure 4):

- **Specification** of service compositions which is derived from business process model. We have developed a mapper which takes a business model in ARIS as input and returns an OWL-S file as output.
- **Discovery** (matchmaking): this involves locating Web services available and match the requirements. The conventional UDDI based matchmaking is not sufficient as it does not consider the behavioral (semantic) aspects of Web services.
- **Selection**: from all matching Web services, select the most appropriate services. The criteria vary. It can be based on QoS, Cost, or other policy-related and trust-related factors.
- **generation of composition**: A sequence of Web services is generated which collectively fulfills the goals as specified in “specification”.
- **Execution**: Typically the constituent services are described in WSDL, binding to the specific URL address.

The composed Web services are readily executable.

These problems impose a set of challenges for service-oriented integration. In the following, we restrict ourselves to a service process model in OWL-S to illustrate the service composition. In OWL-S, services can be atomic or composite services. Atomic services are the basic units of implementation where a single Web service (i.e., Web-accessible computer program, sensor, or device) is invoked by a request message, performs its task and perhaps produces a single response to the requester. With atomic services there is no ongoing interaction between the user and the service. For example, a service that logs a user in, or returns a postal code or the longitude and latitude when given an address would be in this category. An atomic service is a “black box” representation in a sense that no description is given as to how it works apart from input, output, preconditions and effects. In contrast, composite services are complex and composed of multiple more atomic services, and may require an extended interaction or conversation between the requester and the set of services that are being utilized.

OWL-S is an upper-ontology for Web services written in OWL (Web Ontology Language) [10], and offers a standard-based service ontology to describe Web services. It describes a Web service by providing three essential types of knowledge about the service [8]:

1) what the service provides for prospective clients (captured in profile),
2) how the service is used (captured by the ServiceModel or process model), and
3) how to interact with the service (given in the grounding which provides the needed details about transport protocols).

With these Semantically described information, OWL-S and supporting tools enable the location, selection, composition, and inter-operation of Web services to perform some complex task.

For the illustration purpose, consider a popular example for Air Travel Reservation which accompanies [8]. The overall objective of Air Ticket Reservation is achieved by a composite process. It is composed of a sequence whose components are two atomic processes, GetDesiredFlightDetails and SelectAvailableFlight, and a composite process, BookFlight. The process BookFlight is again a composite process. It is composed of a sequence whose components are two atomic processes, LogIn and ConfirmReservation, as shown in Figure 5.

Fig. 5. An example service process: Air Ticket Reservation

In other words, the Air Ticket Reservation process is realized as the execution of a sequence of Web services GetDesiredFlightDetails, SelectAvailableFlight and BookFlight, and BookFlight involves the execution of a sequence of Web services LogIn and ConfirmReservation (Figure 6).

Fig. 6. Realizing the Air Ticket Reservation as a service sequence

OWL-S include a rich set of control constructs for describing complex, composite processes, including...
Sequence, Split, Split + Join, Choice, Any-Order, Condition, If-Then-Else, Iterate, Repeat-While, and Repeat-Until with their obvious meaning implied by the names. A process described in OWL-S can be viewed at different levels of granularity, either as a primitive, undecomposable process or as a composite process. These are sometimes referred to as “black box” and “glass box” views, respectively [8].

The process description written in OWL-S highlights the service composition of the process. An example description for the air ticket reservation process (service) is as follows. For the complete description, including the input, output, precondition, and effects of each service (hereafter IOPEs), the reader is referred to [11].

```
<process:CompositeProcess
  rdf:ID="AirTicketReservation_Process">
  ...
  <process:composedOf>
    <process:Sequence>
      <process:ControlConstructList>
        <list:first>
          <process:Perform
            rdf:ID="PerformGetDesiredFlightDetails">
            <process:process
              rdf:resource="#GetDesiredFlightDetails" />
        </list:first>
        ...
        <list:rest>
          <process:ControlConstructList>
            <list:rest>
              <process:Perform
                rdf:ID="PerformSelectAvailableFlight">
                <process:process
                  rdf:resource="#SelectAvailableFlight" />
            </list:first>
            ...
          </list:rest>
        </list:rest>
      </process:ControlConstructList>
    </process:Sequence>
  </process:composedOf>
</process:CompositeProcess>
```

Our approach to service composition is first to derive the service process from mapping the business process, the service process is then described in OWL-S with IOPEs. In OWL-S, IOPEs are the representation of process goals and constraints, and Goals are characterized by a PostCondition and Effects. The implementation involves service location (matchmaking) based on IOPEs, selection based on service composeability [18] and match ranking of services (exact, fail, subsumes, and plug in) [12], and finally composition of Web services that realize the service process.

Figure 7 shows the architecture of the Semantic Web service composer (SWSC). The inputs to SWSC are the requested service capability from the Service Consumer in terms of OWL-S IOPEs and all advertised services imported from the OWL-S Repository. The requested service capability is OWL-S based service specification. The OWL-S Repository stores the services advertised by Service Providers. To import the services from OWL-S Repository into SWSC the OWL-S API library is used to read the OWL-S specifications of advertised services. OWL-S API is developed and maintained by Evren Sirin of University of Maryland [16]. The output from SWSC is the optimal composition sequence which consists of one or more component services. The component services execute in order to realize the requested capability.

The SWSC system works as follows: The Composition Sequences Generator, one component of the Composition Engine, uses the backward or forward chaining composition algorithm to generate all valid composition sequences given the requested service capability and available registered services. During the process of sequences generation the OWL-S Matcher is consulted to match the inputs, outputs, outputs-inputs of services based on OWL-S description of services. The OWL-S Matcher realizes the matching through interaction with OWL inference engine (reasoner). The matcher sends the semantically described parameter types which needed to be matched to OWL inference engine. The OWL inference engine makes an inference about the relationship of the two parameter types, and sends the inference result back to Matcher. The backward chaining composition algorithm is outlined as follows:

```
BackwardChaining(input, output) {
  for (each service) {
    if (service’s output == output) {
      CompositionGraph = CompositionGraph +
      BackwardChaining(input, service’s input);
    }
  }
  return CompositionGraph;
}
```

The forward chaining algorithm is similar except for the chaining in a reverse order. The chaining algorithm takes advantage of the interdependency between services to link services together and form one or more valid composition sequences.

After all valid composition sequences are generated the Optimal Composition Sequence Selector, another component of Composition Engine, selects the best one and returns it to Service Consumer as the final composition result. The composition result is a sequence of Web services.

V. AUTOMATION VIA AGENTS AND SEMANTIC WEB
A key in a service-oriented integration for formulating a virtual enterprise is a set of standards and facilities that enable
to automatically discover/locate services and then integrate (compose) them without prior agreement. Such automated integration is essential for scaling in a sense that companies are able to leverage globally available services or infrastructure that more effectively serve the business goals and more cost-effectively respond to dynamically changing conditions, even possibly forming a VE of multiple companies for specific business objectives. However, to be automatic, there are two essential elements: (1) services must be marked with machine-processable semantics with standard ontologies; (2) software entities that are capable to process and manipulate these semantic information for some purposes. The emerging Semantic Web and software agents offer the opportunities for achieving automated integration. Especially the crucial components of Semantic Web infrastructure, a standardized Web ontology language (OWL) and Semantic Markup for Web Services (OWL-S), and supporting tools are available.

The automation dimension of (semantic Web) service-oriented approach to integration involves the following tasks [8], [9]:

- Automatic Web service discovery for finding and locating Web services that can provide a particular class of service capabilities, while adhering to some client-specified constraints in terms of IOPEs (Inputs, Outputs, Preconditions, and Effects). With OWL-S markup of services, service properties and capabilities can be declaratively advertised in a service registry or ontology-enhanced search engine could be used to locate the services automatically.

- Automatic Web service composition and interoperation. This is to integrate a set of Web services to formulate an executable service process. In other words, given a high-level description of an objective, this task involves the automatic selection, composition, and interoperation of Web services to perform some complex task whose final effects are the objective of the process.

- Automatic Web service invocation: Once Web services are enhanced with semantics, it requires an execution model that preserves the semantics. In addition, an implementing execution environment must enable dynamic run time Semantic Web services discovery, selection, composition and interoperation.

Since machine-processable semantics for Web services are available, it is natural to build a multiagent system to achieve automation as described above. It should be noted that as compared to Web technology, there is much less (if not no) identifiable, off the shelf agent technologies that can be used to build applications [2]. This remains a challenging task to combine agents with Semantic Web services.

VI. CONCLUSION

The main theme of this paper is the Semantic Web services-based architecture that enables automated integration. The key to the Semantic Web services-based approach to enterprise integration is that the business process modeling transforms an integrated enterprise into a VE, which is then dynamically and automatically constructed out of available semantic-enhanced Web services as needed at runtime using intelligent software agents. In a nutshell, the service-orientation is achieved via virtualization in a WS-Resource Framework approach, the integration is via services composition, the service semantics enhancement uses the OWL and OWL-S, and automation is achieved by software agents which manipulate the semantics of Web services in a service-oriented environment.

Acknowledgments

This work is partially supported by the IMSS Programme of A*Star of Singapore. We acknowledge the contributions of Jiao Tao and Rajaram Shivram Kumar to the prototypings of the Semantic Web Services Composer and AGV virtualization, respectively.

REFERENCES


