Perspectives

Perspectives of an Enterprise Integration Plug-in System Architecture for Networked Manufacturing Systems

Adel N. Ghannam
Faculty of Management Sciences
MSA University
Cairo, Egypt
mghanam@msa.eun.eg

Hesham Mansour
Faculty of Computer Science
MSA University
Cairo, Egypt
hhmansour@msa.eun.eg

Ali El-Bastawissy
Computer Science
MSA University
Cairo, Egypt
aelbastawissy@msa.eun.eg

Mohamed Hamed
Development
Dirac Systems
Cairo, Egypt
mhamed3476@gmail.com

Abstract—**This paper is a part of a research project funded by the Science and Technology Development Fund (STDF), to create a product serving the digital transformation of Egyptian manufacturing companies. It allows a manufacturing company to be a member of a distributed manufacturing network. The resulting system can be plugged into any ERP system. In this work, the limitation of a centralized integration entity to satisfy loosely coupling of distributed systems is overcome. The SOA framework and the remote method invocation (RMI) are applied using SOAP-XML technology. Enterprise integration patterns (EIP) were used in the architecture design.**

Keywords-enterprise integration; ERP; SOA; web service; enterprise integration pattern; P2P

I. INTRODUCTION

Distributed manufacturing [1] consists of enterprise networks supported by computer networks. It dynamically combines core resources and skills of different entities to flexibly respond to business opportunities. This new organizational structure, used in worldwide manufacturing systems, requires the integration of distributed production planning and control systems known as enterprise resource planning (ERP) systems of the individual partners. With the modern distributed manufacturing networks, global coordination among partners is thus required to integrate the ERP systems of those partners to synchronize production planning and control. In other words, a function of an application may be distributed to several companies requiring the cooperation of different parts to work properly. This feature is known as processes integration.

This paper is a part of a research project funded by the STDF [2-4] to create a software system that can be plugged-in any ERP system of a manufacturing company to form a distributed manufacturing system. It is a plug-in component, which means that a core feature of the developed software is loosely coupling. The following features are mandatory for loosely coupling:

- Services are invoked remotely independently of their technology, any third part centralized entity, and location.
- The plug-in component does not impose any restrictions on the hosting ERP system.
- The software is easily configurable to fit the member platform easily.

Traditionally the process of integration is done between the member systems, with the architecture shown in Figure 1: In the middle, the integration broker, is a centralized hub [5]. It uses a central logic that can receive messages from multiple destinations, determine the correct destination and route the message to the correct channel. However, this architecture violates the loosely coupled feature of distributed systems, because having a single hub may make the system easy to manage but scalability becomes a problem. At some point, as the number of members increases, the messages increase, scalability gets bigger, and central integration management becomes more sophisticated

Corresponding author Adel N. Ghannam

Fig. 1. Traditional central integration

II. THE SOLUTION FRAMEWORK

In this work, we overcome the centralized limitation by developing a system that can be plugged- in the ERP system of the network members based on:

- SOA framework.
- Point-to-Point messaging architecture.
- The concept of enterprise service bus (ESB).

SOA is an architectural pattern that involves decomposing applications into managed networks of inter-operating services. Those services could be implemented in a variety of technologies. SOA, the web services description language (WSDL) and web services solutions [6, 7] support two key roles: a service requester (client) and a service provider, which communicate via service requests. Service requests are messages formatted according to the simple object access protocol (SOAP). SOAP is a light-weight protocol allowing remote procedure calls (RPCs) over the Internet using a variety of transport protocols. In principle, SOAP provides a distributed processing model that assumes a SOAP message originates at an initial SOAP sender and is sent to an ultimate SOAP receiver. The SOAP specification establishes a standard message format that consists of an XML document. The SOAP request is received by a runtime service (an SOAP "listener") that accepts the SOAP message, extracts the XML message body, and delegates the request to the actual function or business process within an enterprise. After processing the request, the provider typically sends a response to the client in the form of an SOAP envelope carrying an XML message. WSDL is a W3C specification providing the foremost language for the description of web service definitions. Web services need to be defined in a consistent manner so that they can be discovered by and interfaced with other services and applications. A WSDL service description, describes the point of contact for a service, also known as the service endpoint or endpoint. It establishes the physical location of the service and provides a formal definition of the endpoint interface so that programs wishing to communicate with the service know exactly how to structure the required request messages. WSDL is often used in combination with SOAP and an XML schema to provide web services over the Internet. A client program connecting to a web service can read the WSDL file to determine what operations are available on the server. The client can then use SOAP to actually call one of the operations listed in the WSDL file using for example XML over HTTP

Point-to-Point (P2P) messaging architecture [8] is adopted (Figure 2). The architecture is developed using the EIPs [8], to benefit from the architects expertise encapsulated in the EIP design patterns.

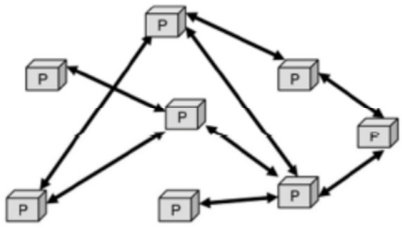

Fig. 2. Scheme of P2P messaging

The main issues to notice in P2P are:

- Components directly interact as peers by exchanging services, which is matching the business case of a distributed manufacturing network, where each member is communicating with other identified members.
- A system with completely decentralized self-organization and resource usage.
- Each peer provides and consumes similar services.
- Peers have the same capabilities (ability to act in any role, can act as "clients" and "servers" at the same time).
- High scalability (the system can grow to a very large number of peers).

The third member employed in our base concepts is the ESB [9] (Figure 3). SOA, EDA, and web services are combined to implement the integration media. The ESB provides messaging, routing and transformation capabilities that enable services to be easily integrated at development time or runtime. The ESB acts as an intermediary layer to enable communication between different application processes. A service deployed onto an ESB can be triggered by a consumer or an event. We have used the ESB concept, but without having a centralized reference entity, to configure the plug-in component at installation time on a partner ERP server.

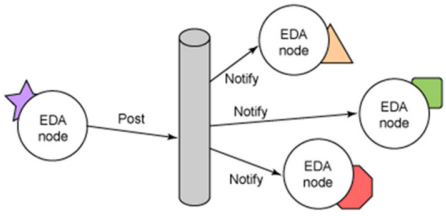

Fig. 3. Illustration of ESB

III. THE PROPOSED ARCHITECTURE WITH EIP

The system should be able to respond to the following cases:

- A company wants to join a particular manufacturing network.
- A company creates a network for its end product.
- A prime partner (the end product owner) wants delegating production orders to specific members.

- WIP monitoring at different partners locations.

The metamodel of the adopted architecture is given in Figure 4. Each member of the network is having a plug-in integration unit known as the end point logic.

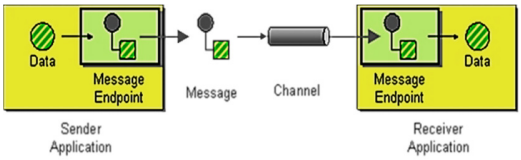

Fig. 4. The role of the endpoint

It is the messaging endpoint code that takes remote service call-command or data, turns them into a message, and sends it on a particular messaging channel using HTTP. It is the endpoint that receives a message, extracts the contents, and gives them to the application in a meaningful way.

The endpoint main function is orchestration of how services interact at the message level, including the business logic and execution order of interactions under control of a single endpoint. The detailed architecture of the endpoint logic, using EIP components, is given in Figure 5.

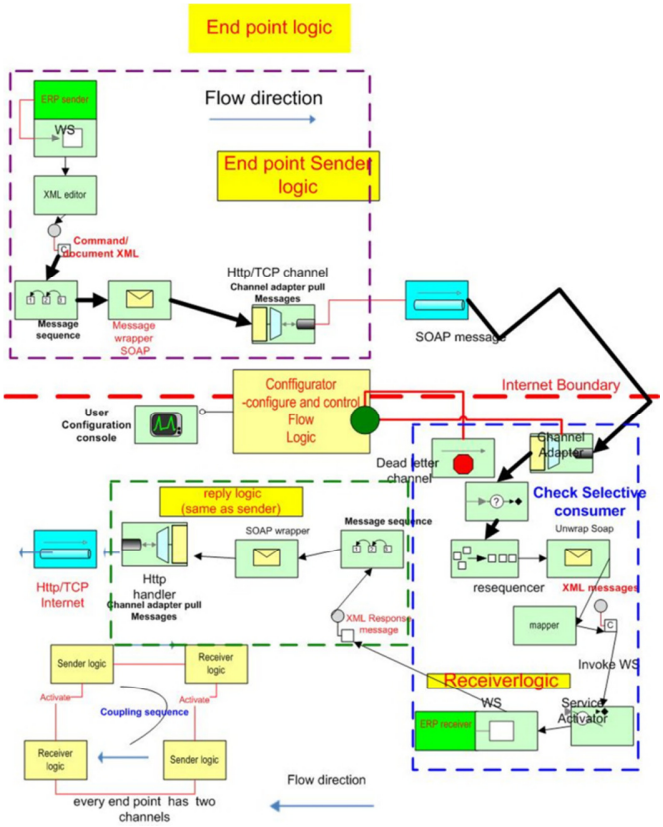

Fig. 5. Endpoint architecture using EIP components

Each endpoint contains two processors, the sender logic and the receiver logic. At any instance, every endpoint is managing two processing channels.

A. On the Sender Side

The owner identifies the partner, the end-product code and the component code to be assigned to the partner, using the configuration console. The configurator request, from the WSDL object, the XML of the end-product BOM and the XML of the component are constructed. The WSDL object will then invoke the XSD editor to construct the XSD of both the end-product and the component of target partner. For a large XSD document we may need to divide the document into a sequence of segments, which is done by the message sequence object. Segmentation is required when the data packet is larger than the maximum transmission unit supported by the network or when the network is unreliable and it is desirable to divide the information into smaller segments to maximize the probability that each one of them can be delivered correctly to the destination. A set of identifiers are used to designate the components of the segment, as illustrated in Figure 6.

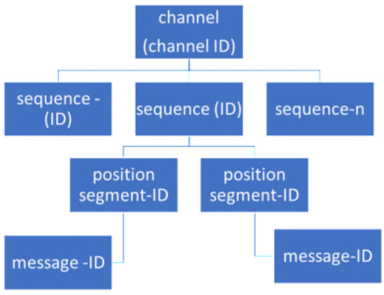

Fig. 6. Hierarchy of the sequence identifiers

This hierarchy indicates that each sequence is divided into segments using the sequencer in the sender logic. These segments are collected on the receiving logic. Each item in the hierarchy has an ID. It is the role of the message sequencer to generate the sequence. Then the resulting sequence is passed to the SOAP wrapper and the wrapped messages are passed to the channel adapter which will assign a specific channel and then generate the HTTP request for the remote partner.

B. On the Receiver Side:

The channel adapter captures the relevant HTPP request to the receiver URL, using the logic of "check selective consumer" component. After checking the receiver address, the sequence is passed to the resequencer. The resequencer secures reconstructing the send sequence into a correct ordered series using the identifiers of the sequence, the segment position, and the message ID. Each message coming out of the resequencer is passed to the "Unwrap SOAP" component, to produce a plain XML/XSD message. The resulting message invokes the "service activator" interfacing with the WS hosted by the receiving ERP. For each message, a reply message is generated. The reply message is carrying a "correlation identifier", which is the sender message ID. When the caller receives this correlation identifier, the caller can thus link the reply to the sender.

IV. CONCLUSION

In this paper we presented the developed architecture of a plug-in integration unit with the following attributes:

- It integrates manufacturing planning processes of members in a distributed manufacturing network.
- The unit is plug-in and easily configurable, working in P2P network and not needing a centralized integration system. Therefore, it eliminates the difficulty of scalability with centralized integration logic.
- It satisfies the mandatory loosely coupling of distributed systems. It works in the SOA framework through web services WS.
- It applies the concept of RMI using SOAP-XML messaging technology.
- It uses EIP to build the architecture based on the logical UML models developed prior to the presented architecture.

The architecture is implemented using the SPRING development framework [10].

REFERENCES

[1] H. Kuhnle, Distributed Manufacturing, Paradigm, Concepts, Solutions and Examples, Springer-Verlag, 2010

[2] A. Ghannam, A. El-Bastawisy, N. S. Abdel Nour, H. Mansour, M. Hamed, Developing A National Technology Toolkit, Integrating Technologies of BPM, Web Services, and RFID to Consolidate Egyptian Manufacturing into Global Value Chain, Technical Report, 2017

[3] A. Ghannam, A. El-Bastawisy, N. Sobhi, H. Mansour, M. Hamed, Developing A National Technology Toolkit, Integrating Technologies of BPM, Web Services, and RFID to Consolidate Egyptian Manufacturing into Global Value Chain, Technical Report 2, 2017

[4] A. Ghannam, A. El-Bastawisy, N. Sobhi, H. Mansour, M. Hamed, Developing A National Technology Toolkit, Integrating Technologies of BPM, Web Services, and RFID to Consolidate Egyptian Manufacturing into Global Value Chain, Technical Report 3, 2018

[5] H. L. Lee, S. Whang, E-Business and Supply Chain Integration, Stanford University, 2001

[6] D. Georgakopoulos, M. P. Papazoglou, Service-Oriented Computing. MIT Press, 2009

[7] K. J. Clark, Integration architecture: Comparing web APIs with service-oriented architecture and enterprise application integration, available at: https://www.ibm.com/developerworks/websphere/library/techarticles/1503_clark/1305_clark.html, 2015

[8] G. Hophe, B. Woolf, Enterprise Integration Patterns, Addison-Wesley, 2012

[9] D. A. Chappell, Enterprise Service Bus, O'Reily Media, 2004

[10] M. Fisher, M. Bogoevici, I. Fuld, J. Partner, O. Zhurakousky, G. Russell, D. Syer, J. Long, D. Turanski, G. Hillert, A. Bilan, A. Nayak, J. Bryant, Spring Integration, available at: https://docs.spring.io/spring-integration/reference/html/index.html