Universität Stuttgart

A Service-oriented Integration Platform for Flexible Information Provisioning in the Real-time Factory

Von der Graduiertenschule GSaME Graduate School of Excellence advanced Manufacturing Engineering der Universität Stuttgart zur Erlangung der Würde eines Doktor-Ingenieurs (Dr.-Ing.) genehmigte Abhandlung

Vorgelegt von

Jorge Mínguez

aus Madrid (Spanien)

Hauptberichter:	Prof. Dr. Bernhard Mitschang
Mitberichter:	Prof. Dr. Engelbert Westkämper
Tag der mündlichen Prüfung:	15. März 2012

Institut für Parallele und Verteilte Systeme (IPVS),
Abteilung Anwendersoftware

2012

Acknowledgement

First of all, I would like to express my most sincere thankful thoughts to my doctoral supervisor and mentor Prof. Mitschang at the Institute for Parallel and Distributed Systems for giving me the opportunity to do my research in his group. I am specially thankful for his continuous support and extraordinary dedication during the last four years of my career. Thanks to his valuable advice and experience, I have been able to successfully conduct my scientific research and to grow as a researcher and as a person.

I also want to thank Prof. Westkämper for creating an outstanding academic environment for doctoral researchers in engineering, computer science, management and economics. Through his vision and dedication, the Graduate School of Excellence advanced Manufacturing Engineering (GSaME) was born in 2007, where I, as well as over 60 other engineers and scientists, have had a unique opportunity to do a doctor's degree up to now. I would also like to thank him for acting as an active member in the Thesis Committee that supervised my work. I do not want to forget all the staff of the Institute for Industrial Manufacturing and Management at the University of Stuttgart that have made GSaME possible how it is today. Thank you Vera Hummel, Lars Aldinger, Andreas Dietrich, Heiko Herrmann, Petra Langbein and Eva-Marie Ill. Specially, I would like to sincerely thank Prof. Rohr for her uninterrupted dedication to the success of GSaME.

I would like to thank some of my current and former collegues for many interesting discussions and for their valuable comments and suggestions. I would specially like to thank Mihály Jakob and Thorsten Scheibler for their effort in the supervision of this Thesis, Peter Reimann for his tremendous dedication in the correction of my written English, Fabian Kaiser, Tobias Kraft, Nazario Cipriani and Oliver Schiller for all the fun in the coffee breaks, Sylvia Radeschútz, Marko Vhrovnik and Manfred Rasch for the wonderful tabletop soccer games. I would also like to thank Stefan Silcher for his cooperation and his effort to always improve the quality of our research. I also want to express my gratitude to Frank Ruthardt and Alexander Braunstein for their dedication and contribution to the results of this Thesis.

During my time at GSaME, I have also met extremely smart colleagues and I have made great and wonderful friendships. I would like to thank David Baureis, Markus Hartkopf, Donald Neumann, David Schindhelm for all the fun in our nights out, I will always remember the fun at the Stuttgart Beer Festivals. Thanks to Sema Zor, Michelangelo Masini and Michael Wörner for their support and for all the good moments we lived together at conferences and in the University. Also thanks to Miriam Floristán for the support and good times that we shared in Spanish in the multiple lunch times and coffee breaks.

Finally, I would like to thank the closest persons that made it all worth it. First, an honest, sincere and deep Thank you goes to my parents Olga and Serafín, who have unconditionally supported me along the path during school, university, abroad and during my doctoral thesis. Last, but not least, I would like to thank Lena for her support and love. I would like to thank her sincerely for her encouragement and patience in the long hours of work in the weekends and at night. Thank you Lena, without you, I would not be where I stand today.

Contents

Acronyms

B2MML Business to Manufacturing Markup Language.

BPM Business Process Management.

BPMN Business Process Management Notation.

CAD Computer-Aided Design.

CBR Content-Based Router.

CEP Complex Event Processing.

CIM Computer-integrated Manufacturing.

CRM Customer Relationship Management.

CSV Comma-Separated Values.

DPWS Devices Profile for Web Services.

DSDL Data Service Description Language.

EAI Enterprise Application Integration.

ECA	Event-Condition-Action.
EDA	Event-Driven Architecture.
EPC	Event-driven Process Chain.
ERP	Enterprise Resource Planning.
ESB	Enterprise Service Bus.
ETO	Engineering-To-Order.
FTP	File Transport Protocol.
HTTP	Hypertext Transport Protocol.
ICT	Information and Communication Technologies.
IIS	Internet Information Services.
IPS2	Industrial Product Service Systems.
JBI	Java Business Integration.
JMS	Java Message Service.
KPI	Key Performance Indicators.
MAPE	Monitor, Analyze, Plan and Execute.
MDA	Model-Driven Architecture.
MES	Manufacturing Execution System.
MM	Mechatronic Modules.
MOM	Message Oriented Middleware.

MQ Message Queuing.

MSB Manufacturing Service Bus.

MSB-PDL MSB Process Description Language.

MSMQ Microsoft Message Queuing.

NEGM Nexus Execution Graph Model.

NMR Normalized Message Router.

NPGM Nexus Plan Graph Model.

OEM Original Equipment Manufacturer.

OPC-DA Object Linking and Embedding for Process Control - Data Access.

OPC-UA OPC Unified Architecture.

OWL Web Ontology Language.

PDA Personal Digital Assistant.

PKB Process Knowledge Base.

PLC Programmable Logic Controllers.

PLM Product Lifecycle Management.

PSS Product-Service Systems.

QoS Quality of Service.

RDF Resource Description Framework.

RDFS RDF Schema.

RMI Remote Method Invocation.

SAWSDL Semantic Annotations for WSDL and XML Schema.

SBPM Semantic Business Process Management.

SCA Service-Component Architectures.

SCADA Supervisory Control and Data Acquisition.

SCM Supply Chain Management.

SDLC Service Development Life Cycle.

SKB Service Knowledge Base.

SMTP Simple Mail Transport Protocol.

SOA Service Oriented Architecture.

SOAP Simple Object Access Protocol.

SOC Service-Oriented Computing.

SPPaaS Stream Processing Platform as a Service.

SPQL Service Provenance Query Language.

STEP Standard for the Exchange of Product Model Data.

UDDI Universal Description Discovery and Integration.

UI User Interface.

URI Universal Resource Identifier.

WCF Windows Communication Foundation.

WfMS Workflow Management System.

WS-BPEL Web Service Business Process Execution Language.

WS-CDL Web Services Choreography Description Language.

WSDL Web Service Description Language.

XRL eXchangeable Routing Language.

XSL Extensible Stylesheet Language.

XSLT Extensible Stylesheet Language Transformations.

Zusammenfassung

In heutigen turbulenten Szenarien müssen produzierende Unternehmen sowohl die Gestaltung von ihren technischen Prozessen und Ressourcen als auch die herzustellenden Produkte an die sich ständig verändernden Geschäftsbedingungen anpassen. In diesem Umfeld gewinnt der Einsatz digitaler Werkzeuge zunehmend an Bedeutung. Um diese Anpassung schnell und effektiv realisieren zu können, ist eine Wissensmanagement-Strategie notwendig, deren Umsetzung fähig ist, eine Ist-Analyse der laufenden Produktion durchzuführen. Die Auswertung von Produktionsdaten ermöglicht dann Wissen aus der realen Fabrik zu extrahieren, um später, durch einen kontinuierlichen Simulationsprozess, einen optimalen Ablauf der Produktion zu erreichen. Hierfür spielt die intelligente Speicherung von Produktionsereignissen eine entscheidende Rolle, da der Zusammenhang zwischen verschiedenen Ereignissen den Kontext in einem Produktionsumfeld zu einem bestimmten Zeitpunkt abbildet. Um relevante Ereignisse für die Produktion zu erkennen, ist eine Informationsbeschaffung in Echtzeit durch die Steuerung relevanter Ereignisse erforderlich.

Die Datenverwaltung in der Fabrik setzt auf Informationsflüsse, die auf verschiedene Systeme zugreifen. Diese Informationsflüsse werden durch Datenbearbeitungsprozesse gesteuert, die die unterschiedlichen Informationssysteme verknüpfen. Das Problem in vielen Unternehmen liegt darin, dass die meisten digitalen Werkzeuge sehr heterogene Insellösungen sind. Dies stellt für die Verknüpfung von Systemen eine gewaltige Herausforderung dar, da mangelnde Schnittstellenkompatibilität den Datenaustausch zwischen Einzelanwendungen erschwert und den Aufwand der Integration neuer Systeme mit bereits vorhandenen erhöht. Häufig werden Einzellösungen durch individuelle Anpassungen für die Kopplung digitaler Werkzeuge eingesetzt. Solche Ansätze basieren auf einer starren Integration, die kurzfristige Lösungen schafft. Allerdings, lassen sie sich meist nur aufwendig erweitern und ändern. Dies führt zu hohen Entwicklungs- und Wartungskosten bedingt durch die Komplexität und die Unzuverlässigkeit einer starren Vernetzung. Aus diesen Gründen ist bei der Integration von unterschiedlichen Informationssystemen ein bestimmter Grad an Flexibilität notwendig. Nur mit der Unterstützung von der richtigen Infrastruktur können die Informationsflüsse so gesteuert werden, dass diese Flexibilität gewährleistet ist. Damit ist eine effektive Anpassung der systemübergreifenden informationstechnischen Prozesse möglich.

Im Rahmen dieser Arbeit wird ein Integrationsansatz vorgestellt, der, auf dem Konzept der Serviceorientierten Architektur (SOA) basierend, die Problematik der Umsetzung einer flexiblen Integration adressiert. Die Verbindung zwischen IT-Systemen und der Integrationsplattform wird über servicebasierte standardisierte Schnittstellen hergestellt. Diese Plattform wird als Manufacturing Service Bus (MSB) bezeichnet. Die funktionalen Eigenschaften und Vorteile dieser Plattform werden im Rahmen der Integration von Produktionssystemen in der Lernfabrik für advanced Industrial Engineering (aIE) am Institut für Industrielle Fertigung und Fabrikbetrieb (IFF) der Universität Stuttgart aufgezeigt.

Die bereits erzielten Ergebnisse in den Forschungsbereichen zur agilen und skalierbaren Turbulenzbehandlung zeigen, dass Kommunikations- und Integrationstechnologien dazu beitragen, die Wandlungsfähigkeit der produzierenden Unternehmen zu unterstützen. Jedoch muss ein flexibler Integrationsansatz die schnelle Anpassung von Integrationsprozessen ermöglichen, um die gewünschte Wandlungsfähigkeit der Produktion gewährleisten zu können. Dabei stellt sich die Frage, durch welche Methoden die in der Fabrik laufenden, systemübergreifenden Informationsflüsse effektiv und effizient angepasst werden können.

Diese Arbeit befasst sich mit der Kernfrage der Umsetzung von Wandlungsfähigkeit auf informationstechnischen Ressourcen der realen Fabrik. Diese und weitere Fragen zum optimalen Ablauf von Datenverarbeitungs- und Anpassungsprozessen werden in der Dissertation erörtert. Nachfolgend werden die Prinzipien der Anwendungsintegration und der Serviceorientierung vorgestellt, die als Leitfaden einer erfolgreichen Umsetzung wandlungsfähiger informationstechnischen Strukturen dienen. Anschließend wird der Manufacturing Service Bus als Lösungsansatz vorgestellt, der die angesprochenen Herausforderungen in einem heterogenen und sich ständig verändernden Produktionsumfeld adressiert. Diese Zusammenfassung schließt mit einem kurzen Fazit und einem Ausblick auf mögliche Folgearbeiten.

Anwendungsintegration und Serviceorientierung

Serviceorientierte Architektur (engl. Service-oriented Architecture, SOA) ist ein Systemarchitekturkonzept für die Strukturierung und Bereitstellung von Diensten. Dienste sind gekapselte Software-Komponenten mit einer wohldefinierten Schnittstelle, die plattformunabhängig sind, die in einem Verzeichnis registriert sind und bei Bedarf benutzt werden können. Diese Eigenschaften zeichnen SOA mit zwei wichtigen Prinzipien aus: lose Kopplung und Wiederverwendbarkeit. Auch wegen diesen Prinzipien wird das SOA-Paradigma immer häufiger auf Geschäftsebene diskutiert und eingesetzt.

Dienste können interagieren ohne Anforderung an spezifische Vorkenntnisse über die jeweilige Implementierung. Somit hat eine Änderung nur eine lokale Auswirkung. Bei einer engen Kopplung, erfordert die Änderung einer Komponente zusätzlich die Anpassung gekoppelter Komponenten. Die Komplexität eines solchen Ansatzes im Produktionsumfeld endet letztendlich in langen Entwicklungszeiten und gewaltigen Integrationsaufwänden. Bei SOA handelt es sich um eine Struktur, welche die Komplexität der einzelnen Anwendungen hinter den standardisierten Schnittstellen verbirgt. Das Potenzial liegt in der Ermöglichung der Umsetzung einer flexiblen Unternehmensanwendungsintegration.

Implementierte Dienste mit einer bestimmten Funktionalität werden in einer sogenannten Serviceregistry von Serviceanbietern registriert. Dort können Dienste von Servicekonsumenten gefunden werden, die anschließend den Dienst benutzen und ausführen. Dieser Mechanismus ist als SOA-Dreieck bekannt und trägt dazu bei, bestehende Dienste einfach wiederverwenden zu können. Das Potenzial für produzierende Unternehmen ist hierbei die langfristige Kostensenkung in der Entwicklung von Punkt-zu-Punkt Verbindungen sowie das Erreichen einer höheren Flexibilität der Geschäftsprozesse durch Wiederverwendung bestehender Services, was für Unternehmen im heutigen Geschäftsumfeld von großer Bedeutung ist.

Webservices werden bei der Implementierung von SOA am häufigsten eingesetzt, da sie einen sehr hohen Standardisierungsgrad aufweisen. Webservices basieren auf XML-Standards, die die Identifizierung und Registrierung von Diensten ermöglichen. Darüber hinaus, wird die Interaktion mit anderen Diensten unter der Verwendung von XML-Nachrichten über Internet-Protokolle als Kommunikationskanal unterstützt.

Die Problematik der Schnittstellenkompatibilität von den verschiedenen IT-Systemen in einem Produktionsumfeld stellt sich als größte Herausforderung für die Umsetzung eines ganzheitlichen Lösungsansatzes dar. Die SOA-Prinzipien Lose-Kopplung und Wiederverwendbarkeit können nur in einer Architektur umgesetzt werden, wenn alle notwendigen Systemfunktionen in gekapselten Diensten zu Verfügung stehen [MJH+09]. Das Ziel der Integration gekapselter Services in einer SOA ist, eine höhere Flexibilität der Geschäftsprozesse durch die Wiederverwendung bestehender Services zu schaffen. Um diese Flexibilität umzusetzen ist eine Integrationsplattform notwendig, die die Kommunikation zwischen den eingebundenen Diensten ermöglicht.

Manufacturing Service Bus

Um die Problematik der Schnittstellenkompatibilität zu lösen, wird hier eine Integrationsplattform vorgestellt, die basierend auf den SOA-Prinzipien, eine flexible Verknüpfung von verschiedenen Produktionsanwendungen ermöglicht. Die Verbindungen werden über den sogenannten 'Manufacturing Service Bus (MSB)' realisiert [MLJ+10]. Der MSB erweitert das Konzept des 'Enterprise Service Bus (ESB)' für Produktionsumgebungen. Der ESB bildet eine Kommunikationsinfrastruktur, über die Nachrichten zwischen Dienstanbieter und Dienstkonsumenten ausgetauscht werden können. Verschiedene Kommunikationsprotokolle werden von solch einem Kommunikationsbus unterstützt, der auch die notwendigen Routing- und Transformationskomponenten enthält. Die internen Komponenten eines ESB werden Integrationsdienste genannt. Diese können ähnlich wie in der Anwendungslandschaft verteilt sein. Die wichtigsten Integrationsdienste eines ESB sind (i) die Transformationsdienste, welche die Unterschiede in den Datenformaten und Datenmodellen überbrücken; (ii) der Routingdienst, der eine Nachricht entgegen nimmt und sie nach vordefinierten Routingregeln an die entsprechenden Empfänger weiterleitet; und (iii) der Orchestrierungsdienst, der nach vordefinierten Prozessmodellen, den Fluss von Nachrichten zwischen Dienstkonsumenten und Dienstanbietern steuert. Ein Orchestrierungsdienst ist in der Regel ein 'Workflow Management System (WfMS)', der Prozesse ausführen kann. Ein Produktionsumfeld kann man in fünf Abstraktionsebenen aufteilen. Der MSB befindet sich in der mittleren Ebene und stellt einen Routingdienst und verschiedene Integrationsdienste zu Verfügung, um Ereignisse von der Produktion zu den entsprechenden digitalen Werkzeugen weiterzuleiten. Zusätzlich bietet der MSB einen Orchestrierungsdienst an, um vordefinierte Abläufe auszuführen. Diese Abläufe können zum Beispiel Reaktionsprozesse sein, die auf spezifische Ereignisse in der Produktion reagieren. Ziel dieser Architektur ist die Anpassung der Integrationsprozesse in einer Produktionsumgebung. Die zu verbindenden Systeme benötigen dafür eine Dienstschnittstelle, um an den MSB angeschlossen werden zu können.

Bei der Implementierung einer Erweiterung einer ESB-Infrastruktur für Produktionsumfelder müssen die Eigenschaften der Kommunikation in einer Produktionsumgebung berücksichtigt werden. Produktionsprozesse werden in der Regel nach Eintreten eines konkreten Ereignisses, Alarms oder nach einer Benachrichtigung gestartet. Eine Störung löst z.B. einen Reparaturprozess aus oder ein Kundenauftrag startet den Prozess der Auftragsbearbeitung. In Produktionsumgebungen ist die Kommunikation also üblicherweise asynchron. Die Verwaltung und Automatisierung von ereignisgesteuerten Prozessen spielt eine entscheidende Rolle bei der Wandlungsfähigkeit eines produzierenden Unternehmens. Die Verbindung zwischen den verschiedenen Informationssystemen im Produktionsumfeld darf nicht nur auf eine reine Datenintegration begrenzt werden, sondern muss auch die Integration auf der Anwendungs- und Prozessebene ermöglichen [MRZ11]. Dies ist die Basis für die Wiederverwendbarkeit der Prozesse.

Um die, in Punkt-zu-Punkt Verbindungen entstandene Komplexität zu reduzieren, kann man einen sogenannten 'Broker' einsetzen, der die direkte Kommunikation zwischen Dienstkonsument und Dienstanbieter übernimmt. Die Vorteile eines solchen Ansatzes sind die Abschaffung von Abhängigkeiten zwischen Datenkonsumenten und Datenquellen sowie die Reduzierung der Anzahl von Verbindungen und damit auch der Reduzierung von Wartungskosten. Der MSB bietet durch den Routingdienst und die Unterstützung von mehreren Kommunikationsprotokollen eine Brokerrolle an. Hierfür ist ein ganzheitliches Datenformat notwendig. Ziel des MSB ist, durch die Bearbeitung und das Management von Ereignissen die Reaktionsfähigkeit zu steigern. Deshalb basiert das MSB-Datenformat auf einem ganzheitlichen Ereignisdatenmodel, der die Darstellung und Bearbeitung von Ereignissen ermöglicht. Dieses Ereignisdatenmodell ist XML-basiert und ermöglicht (i) die Modellierung von Ereignis-Metadaten, wie zum Beispiel, Ereignis-Typ oder Ereignis-Herstellungszeitpunkt; (ii) die Spezifizierung von Routingdaten, nämlich Ereignis-Zielpunkt und Ereignis-Quelle; und (iii) die Modellierung von anwendungsspezifischen Ereignisdaten, wie zum Beispiel Störungsmeldungen oder Kundenaufträge [MRM+10]. Dieses Datenmodell soll maschinen-interpretierbare Ereignisse darstellen, da sie an den Routingdienst und durch die Ereignis-Metadaten automatisch interpretiert und weitergeleitet werden. Aus diesem Grund, wurde XML als Modellierungssprache gewählt. Die Interpretation und Bearbeitung von Ereignissen wird im Routingdienst über Ausdrücke einer standardisierten XML-Abfragesprache durchgeführt.

Der Routingdienst bildet den Eingangspunkt im MSB, der alle Ereignisse entgegennimmt und weiterleitet [MRR+10]. Zuerst werden die Ereignis-Metadaten durch Ausdrücke der XML-Abfragesprache XPath analysiert. Im zweiten Schritt speichert der Router-Prozess das Ereignis in einer Datenbank ab, sofern die Registrierung vom Ereignis noch nicht stattgefunden hat. Im dritten Schritt, werden die Abhängigkeiten ausgewertet, um zu erkennen, an welche Systeme die Nachricht weitergeleitet werden soll. Diese Abhängigkeiten sind nichts anderes als Abonnements von Systemen, die für sie wichtige Nachrichten empfangen möchten. Dieses Konzept ist in der Literatur auch als 'Publish-Subscribe' oder Datenpropagation bekannt. Im vierten Schritt, werden die Zielsysteme bestimmt und abschließend wird eine Nachricht an den entsprechenden Dienst weitergeleitet.

Der erste Prototyp des MSB wurde in der Testumgebung der Lernfabrik advanced Industrial Engineering realisiert und getestet [MRM+10]. Die Testumgebung bietet mit einer digitalen Planungsumgebung und einem realen, wandlungsfähigen Montagesystem alle Fähigkeiten, um auf interne und externe Turbulenzen zu reagieren. In verschiedenen Studien wurden interne Turbulenzen wie ein Ressourcenausfall oder externe Turbulenzen wie Änderungen bei den Kundenaufträgen als relevante Turbulenzen für Unternehmen identifiziert.

Umsetzung wandlungsfähiger informationstechnischer Prozesse

Informationstechnische Systeme und Prozesse sind auch Fabrikressourcen, die als sich verändernde Strukturen betrachtet werden können. Ein produzierendes Unternehmen muss im IT-Bereich sowohl Services als auch Integrationsprozesse unter kontinuierlicher Überwachung und Anpassung ausführen [MSM+11]. Die Überwachung und Steuerung von Services und ganzheitlichen Prozessen bildet ein Lebenszykluskonzept, welches unter dem Dach der IT-Governance im Unternehmen betrachtet wird. Im turbulenten Umfeld ist für produzierende Unternehmen extrem wichtig, alle ihre IT-Ressourcen unter kontinuierliche Überwachung zu stellen und entsprechende Anpassungen schnellstmöglich durchzuführen. Dies stellt die zentrale Herausforderung bei der Umsetzung wandlungsfähiger informationstechnischer Prozesse dar. Die Bereitstellung aller notwendigen Kontextinformationen, über die in der

Fabrik installierten Services, erfolgt über ein Service-Repository, das die Analyse- und Anpassungsmethoden für Integrationsprozesse zur Verfügung stellt. Dieses Repository umfasst eine semantische Datenbank für Service-Metadaten (engl. Provenance-aware Service Repository), die als Service-Verzeichnis mit Suche- und Versionierungsfunktionalitäten eingesetzt wird [MNM11]. Die Besonderheit beim Service-Repository ist die Speicherung von Kontextdaten, die das Wissen über Abhängigkeiten zwischen Fabrikkontext und Services darstellt. Die Verbindung zwischen realer Fabrik und Service-Repository wurde über eine webbasierte Schnittstelle realisiert. Die Integration von Produktionsdaten im Service-Repository besteht aus einer Sequenz von Operatoren, die auf einem datenstrombasierten System ausgeführt werden. Durch diese Operatoren werden mithilfe eines Wissensentdeckungs-Algorithmus Datenströme analysiert und in Domainwissen konvertiert. Somit können interpretierbare Daten aus der Fabrik in das Repository importiert werden und in Echtzeit analysiert werden. Die Verarbeitung solcher Daten stellt einen vorher nicht vorhanden, bedeutungsvollen Kontext bereit, der durch logische Schlussfolgerungstechniken einen besseren Ausblick auf die reale Produktion anbietet. Die Nutzung neuer Kontextinformationen für die Optimierung der IT-Prozesse in der laufenden Produktion stellt ein großes Potenzial dar.

Fazit und Ausblick

Zum Validierungszweck wurden verschiedene Anwendungsfälle herangezogen. Mit der Erprobung von vier Szenarien in einem echten Produktionsumfeld wurde die Verbesserung der Integrationsfähigkeit, Flexibilität und Anpassbarkeit von Integrationsprozessen gezeigt. Dies steht im Gegensatz zur bisherigen starren Integration von einzelnen Systemen. Dabei konnten die Potenziale des serviceorientierten Ansatzes in der Verknüpfung von heterogenen Systemen gezeigt werden, indem sowohl die Integration neuer Systeme als auch die Anpassung vorhandener Prozesse deutlich vereinfacht wurden. Des Weiteren hat der vorgestellte servicebasierte Integrationsansatz zur Verwaltung der IT-Prozesse mit dem MSB als zentrale Integrationsebene in einer ereignisgesteuerten Produktionsumgebung den Nachweis der Funktionalität, anhand der Bewältigung von mehreren Turbulenzen, erbracht. Mit der Reduktion der aufwendigen und kostenintensiven Verknüpfung von Systemen, die ein wesentliches Hindernis für den Einsatz der Digitalen Fabrik darstellt, können die Potenziale der Verbindung zur realen Fabrik leichter erschlossen werden. Gerade die Aktualität und Konsistenz der Daten in den verschiedenen Systemen sowie die Mehrfachverwendung sind dabei zu nennen. Eine erfolgreiche Umsetzung des vorgestellten Ansatzes über den ganzen systemübergreifenden Produktlebenszyklus hinweg bietet vereinfachte Möglichkeiten für die Integration von Systemen, die in den Unternehmen bisher als Insellösungen vorhanden sind [MSM11, SMM11]. Deshalb sind weitere Überlegungen notwendig, diesen Integrationsansatz im gesamten Produktlebenszyklus zu nutzen, um auf die aktuellen Daten der Produktion zuzugreifen [SMS+10]. Es wurde ein Prototyp entwickelt, der aufzeigt, wie es in der realen Fabrik gelingt, eine stetige Adaption umzusetzen [MRM+10, MRR+10]. Der Prototyp stellt eine "technische Intelligenz"dar, die aktuelle Daten in Echtzeit an die Produktionsplanung weiterleiten kann.

Abstract

Constantly changing business environments force manufacturing companies to continuously adapt their products, processes and services in order to remain competitive. This adaptation requires also changes in Enterprise Application Integration (EAI) processes, which seamlessly integrate applications across the factory. Therefore, EAI middleware solutions for the manufacturing industry need to not only integrate a large number of heterogeneous applications and legacy systems but also establish a lifecycle management strategy that supports the planning, execution, monitoring and analysis of EAI processes. Nowadays, the support of digital planning tools for the required adaptation has become indispensable. In manufacturing, the leverage of digital tools for visualization, simulation and virtual reality applications is known as the Digital Factory. In order for the Digital Factory to be effective, a dynamic picture of the factory absolutely indispensable. The reflection of the current state of the factory into the future can only be useful to engineers if a permanent feedback loop from the real factory, which enables the monitoring and analysis of the past experience from earlier states of the factory, is provisioned. This concept is known as Real-time Factory. However, the realization of the Real-time Factory presents a number of research issues regarding the in-

tegration of applications and information systems across the factory, such as the heterogeneity of systems, the lack of integration at the application level, the lack of automation tools for the monitoring and analysis of the operational environment and missing mechanisms for the agile adaptation of EAI processes.

In this thesis, a service-oriented integration architecture for manufacturing environments is introduced. This architecture is based on a service bus that allows a loose coupling of distributed services in event-driven manufacturing environments. This platform provides flexible communication between Digital Factory and shop floor components by introducing an application-independent canonical data model for manufacturing events, a content-based routing service, data transformation services as well as event processing workflows. Furthermore, an EAI Process Model is proposed. This model is used by an EAI Process Editor to plan and design the integration processes that enable the exchange of data in the Real-time Factory. In addition to the EAI Process Model, an Adaptation Model for the Real-time Factory is proposed. This model constitutes an adaptability framework for the Real-time Factory that implements the monitor, analyze, plan and execute (MAPE) functions of a self-managing environment and serves as a guideline for the feedback loop established between the execution and the planning environment. The architecture presents self-managing and adaptive mechanisms thanks to the automation of the monitoring and analysis tasks. The monitoring phase is implemented by a mining process that provides real-time domain data evaluation and transforms it into high-level context descriptions that can be processed in the analysis phase. For the analysis of domain context, which comprises the analysis phase of the MAPE-based feedback loop, a Provenance-aware Service Repository is proposed. The Service Repository processes domain recommendations and enables the communication with the EAI Process Editor and other lifecycle applications in order to react responsively to turbulent scenarios in the domain. The Service

Repository manages information about services, processes and their dependencies in a service knowledge base and in a process knowledge base. A semantic data engine in the Repository provides an inference mechanism, based on an algorithm that generates the appropriate corrective actions to attend the recommendations made by the mining process. Both the mining graph and the Service Repository close the MAPE cycle and automate the domain knowledge extraction process, which eases an agile adaptation of EAI processes based on a real-time domain data evaluation. This architecture is capable of managing the life cycle of services and EAI processes in order to achieve the desired agility in an interconnected environment.

A prototype implementation for the integration architecture, the EAI Process Editor, the mining process and the Service Repository are introduced and described in this thesis and serve as a proof of concept. The applicability and validation of the approach are supported by different use cases. The evaluation of the approach has been realized by the examination of different criteria that have been classified in six categories: interoperability, flexibility, mediation, adaptability, agility and integration. These categories are used to evaluate the approach and to compare it with past and current approaches in the EAI domain. Two recent research approaches to integration and context management in the manufacturing domain are also evaluated and compared to the presented approach.

Chapter 1

Introduction

Current manufacturing companies face constantly changing market conditions and turbulent scenarios that require a continuous adaptation of factories in order to customize products according to the customer needs and thus meet market demands. The need for support of Information and Communication Technologies (ICT) for this adaptation has become a fundamental requirement. Digital planning tools, web-based applications and information systems are used on a daily basis to support engineers and are an essential element in factories to run manufacturing processes. However, over the last few years, the growing complexity, the heterogeneity of the involved systems and the pressure of constant change and adaptation have increased the costs of system integration in the manufacturing industry [RM09] as well as in other industries. These problems can be found in almost all domains of manufacturing companies, where ICT is involved, such as in the Digital Factory, Factory Data Management or Product Lifecycle Management (PLM).

The Digital Factory is an example of the need for adaptation in constantly changing environment, such as the factory itself. This adaptation requires the evaluation of the current, past and future state of the factory. Whereas the picture offered by the Digital Factory is a static picture, the real factory changes in a constant manner. In order for the Digital Factory to be effective, a dynamic picture of the factory needs to be provided with help of simulation tools, visualization tools, virtual reality and specific applications. This reflection of the current state of the factory into the future can only be useful to engineers if a permanent feedback loop from the real factory, which enables the monitoring and analysis of the past experience from earlier states of the factory, is provisioned. This concept is known as Real-time Factory and the integration of systems and applications in this domain is the focus of this thesis.

Different heterogeneous applications, legacy information systems and the management of information flows that require the interconnection of several information systems usually define complex integration scenarios. From past experiences [Gun97, Sny91] , this complexity leads to rigid integration, high maintenance costs and a lack of flexibility. In order to deal with this problem, an integration infrastructure acting as middleware between applications is adopted to increase the connectivity and interoperability. At an enterprise level, the set of software architectural principles that govern such an integration infrastructure is known as Enterprise Application Integration (EAI). The purpose of EAI is to provide the appropriate framework for applications and data sources in an enterprise to seamlessly share their data and processes. The problem domain of this thesis is defined by the Real-time Factory operational characteristics and the motivation of this thesis is to find the best suitable EAI-middleware for the Real-time Factory. The problem domain and the motivation are described ahead. Section 1.2 describes the vision and objectives of the proposed architecture as well as the challenges that current manufacturing presents. Finally, Section 1.3 details the structure of this thesis.

1.1 Problem Domain and Motivation

A constantly changing environment pressures manufacturing companies to achieve a high degree of agility in factories. A constantly changing environment and the required adaptability have a tremendous impact on the ICT resources of the factory. The heterogeneity of information systems, growing complexity of integration are two great agility barriers that current manufacturing companies have to deal with. These agility barriers show the need for comprehensive and flexible integration ICT infrastructures.

Nowadays, the support of digital planning tools for the needed adaptation has become indispensable. The so-called Digital Factory offers a complete digital representation of the factory, in order to reduce the "costs of experience" [Wes06] The goals of the Digital Factory are mainly to minimize production costs, efficiently organize the use of resources and thus to increase the productivity of production plants [VDI06]. However, in a constantly changing environment, there is a fundamental requirement for the Digital Factory in order to achieve its goals: the evaluation of the current, past and future state of the factory. The picture offered by the Digital Factory is a static picture, whereas the real factory changes in a constant manner. A dynamic picture of the factory can be achieved with help of simulation tools, visualization tools, virtual reality and specific applications. This reflection of the current state of the factory in the future is known as Virtual Factory. However, the Virtual Factory can only be effective if the past experience from earlier states of the factory is analyzed and a permanent feedback loop from the real factory is provisioned. This concept is known as Real-time Factory, which offers an intelligent, real-time operational management of factory processes and resources. The goal of the Real-time Factory is to integrate the real factory with the virtual factory by continuously communicating, connecting and evaluating the factory's operational data [JWW09].

The motivation of this thesis is to find ways to support the flexible information provisioning and thus increase the agility of a factory regarding the management of its ICT resources in the Real-time Factory. Therefore the motivation is divided into two parts: flexibility and agility. The Real-time Factory is one of the research areas in manufacturing where both flexibility and agility are more needed. This is due to the diversity of information systems to be integrated, namely production planning tools and monitoring and control manufacturing systems that operate under real-time requirements and typically follow an event-driven communication paradigm. Therefore, the heterogeneity of systems and complexity of integration may become extremely unmanageable in the Real-time Factory without a comprehensive integration infrastructure that provides the needed flexibility. The required flexibility for such complex integration scenario represents the first part of the motivation and is described in Section 1.1.1. The second part of the motivation of this thesis is determined by the required agility in the Real-time Factory. Mergers, acquisitions, new planning tools, new products, new software versions, new process flows are all factors of a constantly changing ICT landscape that require manufacturing companies to manage the life cycle of such ICT resources in order to achieve the desired agility in an interconnected environment. The aspect of agility and ICT resource life cycle management is described in Section 1.1.2.

1.1.1 The Real-Time Factory

In order to achieve a seamless integration between applications in the Real-time Factory a number of problems regarding the interconnection of information systems need to be dealt with. These are the communication between heterogeneous applications and the growing complexity of integration. The first problem is the extremely heterogeneous landscape of applications that is encountered for in factories. Different databases, legacy systems, application servers are distributed across different domains. Information flows in a factory typically require the interconnection of the existing information systems with different applications. Heterogeneity has been temporarily solved in many occasions by making direct system connections as needed, thus leading to a so-called accidental architecture. The accidental architecture is a de facto integration approach that develops over time, as a result of not having a coherent corporate wide strategy for integration. This represents an ongoing legacy of point-to-point integration solutions, each with its own flavor of connectivity and integration [Cha04]. This evolution usually leads to unmanageable interdependencies between applications. As a result, applications are extremely tightly coupled. It is precisely the intended solution to heterogeneity what has created the second problem over time: the complexity of integration. As the era of digital manufacturing started, the number of manufacturing systems and applications and the cost of integration have grown to unprecedented limits [Mes08]. Therefore, a comprehensive approach that allows managing the integration of information systems is needed. In the Real-time Factory, the immediate precondition to adapt to a constant changing environment is a flexible ICT integration infrastructure. Such an infrastructure has to be able to seamlessly manage information flows across multiple heterogeneous manufacturing systems and provide the necessary means to support the execution of manufacturing processes.

1.1.2 Life Cycle Management of ICT Factory Resources

The modern view on manufacturing engineering resides in incorporating the life cycle paradigm into the factory as a whole, its corresponding products, manufacturing processes and technologies [JWW09]. This view includes the ICT factory resources, which involve integration processes, services , databases, digital planning tools and domain-specific systems, such as Enterprise Resource Planning (ERP) systems, Manufacturing Execution System (MES) or Customer Relationship Management (CRM). All these resources undergo the same changing environment as the whole factory. This means that these resources, as part of a changing factory, have their own life cycle as well. The adaptation of these ICT resources requires a deep analysis of up-to-date production data and history of previous states of the factory. The up-to-date information about the production environment can be referred to as factory context. The history of data is a concept known as provenance [BKT07], which also contains detailed descriptions about data and process dependencies. Provenance can be very valuable information that can be used to interpret the history of earlier states of the factory and adapt accordingly the affected ICT resources when changes occur. The correct evaluation of the factory context, provenance and resource information to efficiently adapt such ICT resources represents the second motivational aspect that is addressed in this thesis.

1.2 Vision, Objectives and Research Issues

Recent efforts have focused on service orientation to deal with the integration problems that arise when dealing with a very heterogeneous landscape of applications and information systems [Mes08, SML08, HGB05]. Service orientation, or Service Oriented Architecture (SOA), is a software paradigm to design business applications by using — or reusing — services which are self-contained, independent and discoverable. Such services can then easily communicate with other services or even be instantiated by other processes in a coordinated manner. When these processes are in charge of executing transactions that integrate different systems or applications, these will be referred to as integration processes or EAI processes from now onward. Usually such processes are executed with help of a coordination unit, such as a Workflow Management System (WfMS), or an integration middleware, such as a message broker or a service bus. SOA has been widely adopted in the industry due to its service reusability and loose coupling principles. However, when applying these SOA principles to integration, enterprises managing services and EAI processes need to establish a service lifecycle management strategy that supports the modeling, deployment, evaluation and redesign of such integration processes. The management of services and EAI processes is crucial for the application integration in the Real-time Factory. For this reason, only the life cycles of these two ICT resources, services and integration processes, are considered for the purpose of this thesis, which is to achieve flexible information provisioning in the Real-time Factory.

1.2.1 Vision

In this thesis, an integration architecture connecting the real factory with the Virtual Factory is proposed. This architecture, which is shown in Figure 1.1, combines a service-based integration middleware with different services in order to enable the execution of integration processes. The conception of the middleware, inspired by SOA principles, and its implementation, which is based on a service bus, has been named Manufacturing Service Bus (MSB). Such processes enable the bidirectional communication between information systems in the real factory (e.g., production control system and the shop floor data collection system) and a number of manufacturing applications that act as link to the Virtual Factory (e.g. MES, ERP, CRM). Integration processes are modeled in a specific process editor and can seamlessly connect Virtual Factory applications with information systems in the real factory through the internal services of the integration middleware. The connection of applications to the integration middleware is realized by means of service interfaces, as in a SOA. This enables the management of applications and information systems as encapsulated functional units, namely, as services. The life cycle of services and integration processes are managed in a so-called service provenance-aware repository, which keeps the necessary information about EAI processes, services and factory context. EAI process descriptions are sent to the repository by means of the process editor. The purpose of this repository is to provide the necessary provenance information and to adapt the services and processes that are used to connect different applications in the Real-time Factory. The repository receives up-to-date information from the factory through a data stream processing platform, which is connected to the data collection system of the real factory.

1.2.2 Objectives

In order for this vision to be successfully realized, the following objectives need to be accomplished within the presented architecture:

- Objective 1: Flexible information provisioning. The integration middleware must enable the flexible provisioning of information to the involved systems in integration processes. This can be achieved if the following two goals are achieved. The first goal is the precondition for flexible integration: loosely-coupled components. A tightly-coupled approach facilitates the proliferation of rigid integration interfaces which are very difficult to adapt. The second goal consists in minimizing the impact of changes in applications. The adoption of new applications in existing integration processes must be possible without affecting the existing interfaces.
- Objective 2: Improve interoperability. The integration middleware must process production data from multiple applications in a neutral manner that avoids excluding any information system for having an incompatible data format. The proposed middleware must

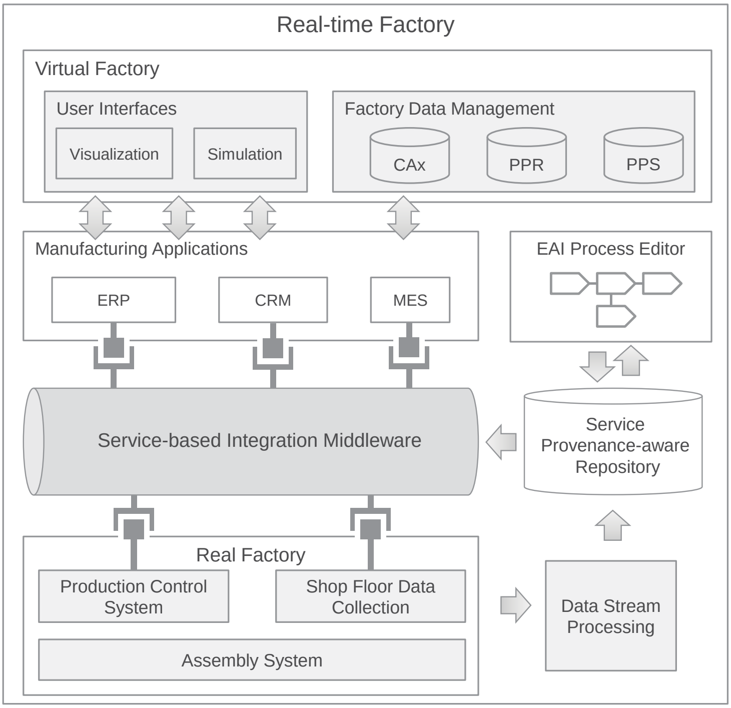


Figure 1.1: Vision for Integration in the Real-time Factory

enable interoperability between applications at the data level and also at the application level. An integration of the application interfaces is necessary while keeping applications decoupled. Such an approach facilitates the reuse of the functionality that applications expose in their interfaces. Further details about the different levels of integration are given in Chapter 2.

- **O**bjective 3: Adaptability. The adaptability of a factory depends enormously on its ICT resources [JWW09, WZ09] as well as on the capability of adaptation of the integration processes and its IT infrastructure. Four goals are considered within the concept of adaptability in this objective: (i) lifecycle management of ICT-resources, (ii) reconfiguration, (iii) agility and (iv) knowledge-driven adaptation. The first goal consists in equipping infrastructures with lifecycle mechanisms to Monitor, Analyze, Plan and Execute (MAPE) integration processes. Such a cycle, known as the MAPE cycle [IBM05a] represents a building block of adaptive, self-managing systems, and thus, it must be considered in the adaptability objective. The second goal is reconfiguration. This goal is complementary to a MAPE cycle, since the reconfiguration of processes is needed for adaptation purposes in every cycle iteration. The third goal is agility. Agility is a key aspect to increase the responsiveness of a factory. An agile adaptation of EAI processes is a fundamental requirement in case it is required by changes in the production environment. For this reason, the composition, configuration and deployment of EAI processes must be realized as fast as possible, assisting process editors and providing high-level context information that can be used to update processes. Finally, the fourth goal is knowledge-driven adaptation. This goal also complements the MAPE-cycle concept by enhancing the monitoring and analysis phases. The adaptation of integration processes has to be done based on knowledge extracted from the domain, which in this thesis represents the Real-time Factory.

1.2.3 Research Issues

In order to accomplish the objectives mentioned above, there are a number of research issues, challenges and barriers that need to be met. These are described ahead:

- Research issue 1: How to deal with multiple vendor installations? In many industries, software product vendors try to provoke vendor lock-in situations. In such cases, the lack of openness in the interfaces of the software modules makes it very complex and expensive to integrate these systems with systems from different vendors.

- Research issue 1: How to avoid tightly-coupled integration? Heterogeneity has been temporarily solved in many occasions by making direct system connections as needed, thus leading to accidental architectures. An accidental architecture develops over time as a result of not having a coherent corporate wide strategy for integration. This represents an ongoing legacy of point-to-point integration solutions, which does not scale: as the number of applications grows, the number of point-to-point interfaces grows exponentially.

- Research issue 3: How to deal with heterogeneity? The architecture must deal with the vast heterogeneity of systems that are present in the Real-time Factory, including legacy systems, applications, event-driven systems, services, integration processes. In addition to this, the presented architecture must ease the efforts of integrating new applications that may be added in the future to the ICT landscape of the factory.

- Research issue 4: How to enable the functional connectivity of applications? The aspect of heterogeneity also reflects in the integration at the application level. An important research issue that needs to be met for the functional integration is the heterogeneity of the application interfaces.

- Research issue 5: How to provide adaptation mechanisms? In order to provide the integration infrastructure with the required adaptability, self-managing components need to be adopted. One of the most important challenges in the adaptability objective is to introduce automation mechanisms in the adaptation lifecycles of EAI processes.

- Research issue 6: How to ease reconfiguration? The ease of reconfiguration tasks aim to hide the technical aspects of reconfiguration from the user. The idea is that the user does not have to deal with interface names or binding information. The abstraction of complexity for the modelers of EAI processes defines this research issue.

- Research issue 7: How to increase agility? An agile adaptation of EAI processes can be provided if the knowledge extraction and analysis tasks are automated or semi-automated in a way that reduces user intervention. The challenge in the agility objective is to reduce the monitoring and analysis tasks that are realized manually.

- Research issue 8: How to manage provenance and context information? In order to provide a knowledge-driven adaptation of EAI processes, the factory context, service provenance information and service dependencies need to be analyzed, processed, filtered and delivered to process modelers. One of the challenges for the realization of these tasks is to manage knowledge information in a way that is both human and machine interpretable. Moreover, knowledge about the reuse of services in a changing production environment has to be managed as well in order to adapt, if needed, the appropriate EAI processes that integrate services with updated interfaces.

In order to achieve the aforementioned vision and objectives in the problem domain that has been described, a number of research issues has to be overcome (see Table 1.1).

Objectives	Research Issues
O1-1.Flexibility of EAI Processes	RI-1. Deal with multiple vendor installations while preventing rigid integration approaches
O1-2. Minimize the impact of changes in EAI Processes and application services	RI-2. Avoid tightly-coupled systems
O2-1. Interoperability at the data level	RI-3. Heterogeneity of applications and information systems in the Real-time Factory
O2-2. Interoperability at the aplicaiton level	RI-4. Enable the functional connectivity of applications
O3-1. Adaptability of ICT integration infrastructure	RI-5. Enable self-management features in the integration middleware. Automation of adaptation cycles
O3-2. Ease of Reconfiguration	RI-6. Simplify configuration tasks and abstract human tasks from the specific requirements of IT systems
O3-3. Agile Adaptation	RI-7. Automation of knowledge extraction and analysis tasks
O3-4. Knowledge-based Adaptation	RI-8. Manage dependencies between data, services and EAI processes. Manage reusable services

Table 1.1: Research Objectives and Research Issues

How these research issues are met by the presented approach is detailed in Chapter 5.

1.3 Structure of the Thesis

The rest of this thesis is structured as follows. In Chapter 2, the foundations and related work to the domain of the presented approach are given. The foundations of this thesis include a detailed description of the paradigms and enabling technologies on which the presented architecture is based, i.e. Enterprise Application Integration [Lin99], Service-oriented Architecture [PH07, Erl05], semantic interoperability [CS06], service lifecycle management [Pap08], Web Services [WCL+05] and Enterprise Service Bus [SHL+05, Cha04]. Related work is given at the end of Chapter 2 and it describes the limitations of current integration approaches.

In Chapter 3, the principles of integration that guide the proposed solution for the Real-time Factory are given. First, a discussion about the deficits and lessons learned in past and current integration efforts is given. These lessons are valuable knowledge that serves as guideline to achieve the objectives of this thesis. These lessons are applied to the scenario of the Real-time Factory and the integration requirements of the Real-time Factory are described. Based on these requirements, the principles of integration for a middleware infrastructure and for the integration of Real-time Factory applications are described.

In Chapter 4, the feasibility of SOA-based integration approaches is investigated based on different research studies in five domains of manufacturing. The investigated domains are: digital factory, ETO enterprises, PLM, PSS and reconfigurable production systems. The most influential studies for this thesis are the integration in the digital factory and the study on the responsiveness of manufacturing companies that was realized on the case of an exemplary ETO enterprise. The first study illustrates the benefits of the integration of the data propagation system Champagne in an SOA-based architecture. In the second study, different event-handling mechanisms in service-oriented processes are described.

In Chapter 5, the architecture of the Manufacturing Service Bus [MLJ+10] is described. The presented approach applies the integration principles of loose coupling and flexibility. The MSB comprises an integration layer that enables a seamless integration of information systems and applications across the factory in a flexible manner and that is capable of increasing the agility of the factory with regard to ICT resources. The proposed architecture is enhanced with two service management components: a service repository and an EAI Process Editor, which are used to manage the services and the integration processes that control the information flows across information systems in the factory. In Chapter 5, an EAI Process Model is proposed. This model is used by the EAI Process Editor to plan and design the integration processes that enable the exchange of data in the Real-time Factory and that are executed in the MSB. Furthermore, monitoring and analysis tasks are automated by a mining process and the inference process of the service repository, respectively. This closes the SOA lifecycle gap with the EAI Process Editor, which permits to increase the capabilities for agile adaptation of the approach. The SOA lifecycle is implemented following a MAPE-based architecture that implements a feedback loop from the manufacturing domain (MSB) to the analysis phase (Service Repository) and from the planning phase (EAI Process Editor) back to the execution environment in the manufacturing domain. In Chapter 5, a Real-time Factory Adaptation Model is proposed, which is followed by the MAPE-based architecture. This architecture implements a feedback loop and enables the execution of self-managing mechanisms, thus improving the agile adaptation of integration processes.

In Chapter 6, four applicability use cases and the validation and evaluation of the approach are given. The first two use cases demonstrate the usability of the MSB in a real production environment. Both use cases show the flexibility, interoperability of the approach thanks to the routing service and other mediation services of the integration architecture, which enables components to remain loosely-coupled. The last two use cases demonstrate the adaptability and agility features of the implemented MAPE-based feedback loop. The validation of the approach aims to demonstrate the technical feasibility and applicability and to evaluate the practical value for the groups of interest. The demonstration of the technical feasibility is described in Chapter 5. The applicability of the approach is illustrated in the use cases. Research results are validated in the assessment of the presented contributions, which is given by an extensive coverage of the objectives that are described in Chapter 1. The evaluation of the approach has been realized by the examination of thirty criteria that have been classified in six categories: interoperability, flexibility, mediation, adaptability, agility and integration. These categories are used to evaluate the MSB approach and to compare it with past and current approaches in the EAI and SOA domain. Two recent research approaches to integration and context management in the manufacturing domain are also evaluated and compared to the presented approach.

In Chapter 7, the conclusions of the thesis and an outlook are given. The conclusions underline the research results in general and remark the benefits of the approach. The outlook is based on the limitations of the approach in comparison to other approaches and illustrates future research challenges.

Chapter 2

Foundations and Related Work

This Chapter is organized in two parts: foundations and related work. The foundations of this thesis, including a detailed description of the enabling technologies on which the presented architecture is based, are described in Sections 2.1 to 2.4. Related work is detailed in Section 2.5. The limitations of current research approaches in the EAI domain for manufacturing environments are described in Section 2.6. A summary is given in Section 2.7.

2.1 Enterprise Application Integration

Enterprises dealing with a large landscape of applications need to face certain integration challenges. The purpose of Enterprise Application Integration (EAI) is to provide the appropriate framework for applications and data sources in an enterprise to seamlessly integrate and share their data and processes [Lin99]. A description of different types of integration as well as different concepts in EAI middleware are given ahead.

2.1.1 Types of Integration

There are four different types of integration [Lin99]. The first type of integration takes place at the data level and connects data stores directly with each other. Its major advantage is that it doesn't require changing, testing or redeploying applications. The second type of integration is done at the application interface level and is based on leveraging application interfaces to connect the business logic of different applications. Within this type of integration, interfaces are used not only to access data, but also to process information. The third type is called method-level EAI and intends to share information within different business processes of an enterprise. The fourth type of integration takes place at the User Interface (UI) level, also known as screen scraping. It pursues to capture information that appears directly on the UI of an application. The advantages of each type of integration depend on the concrete problem domain.

2.1.2 EAI Middleware

Middleware is the term used to refer to any type of software that allows one entity, usually an application, to communicate with another entity. Examples for EAI middleware are application servers or message brokers. An EAI middleware can manage different models of communication. The relevant concepts for this thesis are explained ahead.

2.1.2.1 Synchronous versus Asynchronous

A middleware system may support two types of communication: asynchronous or synchronous. An asynchronous middleware is capable of decoupling applications and is non-blocking. As a consequence, applications can process information independently of what the middleware is doing. Such a model is typical of systems that deal with the consumption and reaction to events. A synchronous middleware makes applications dependent on the middleware because it keeps applications waiting while processing data.

2.1.2.2 Publish-Subscribe

Publish-Subscribe is a model of communication that has been widely used in many areas due to its loose coupling properties. In this model, a broker is in charge of administrating a number of published topics, to which applications can subscribe. Applications send their messages to the broker, under a specific category or topic. The broker then forwards the messages to all interested applications, i.e. to all applications that have previously subscribed to the same category or topic. The advantage of this model is that the sender does not need to know anything about the target applications.

2.1.2.3 Message Brokers

A message broker accounts for all discrepancies in the syntax and semantics of messages that applications share. Message brokers represent a step forward in decoupling applications. Message brokers can perform message transformations, routing tasks and even alter the content of messages. Message brokers are usually integrated in a Message Oriented Middleware (MOM).

2.1.2.4 Repository Services

Many message brokers embrace the concept of a repository to archive all the information that is needed to keep applications decoupled. Such information includes history of data elements, message schema information, metadata, interrelations between applications, process information, rules and logic for message processing.

2.2 Service Oriented Architecture

Service Oriented Architecture (SOA) [Erl05] is a software paradigm to design business applications by using -or reusing- services which are self-contained, independent and discoverable. SOA can empower a business with a flexible infrastructure and processing environment by provisioning independent and reusable automated business processes as services [PTD+07]. The challenge of integration, the cost of managing IT and the inflexibility to respond to changing requirements are the decisive reasons why most organizations adopt SOA. As a matter of fact, for companies that have focused on internal SOA deployments, the leading investment has been application, process and data integration [AC05].

2.2.1 The SOA Principles

SOA presents a number of design principles based on the reuse of existing building-blocks, called services, without requiring any details about the implementation of these services. Services contain a well-defined interface that describes the functions offered by the service and how to invoke these functions. This enables the interoperability of services that are implemented in different programming languages and platforms. Nowadays, SOA, as an architectural style, has still an enormous potential for integration and governance that can be exploited if a number of design principles is followed. The SOA principles most relevant for this thesis are described ahead.

2.2.1.1 Reusability

One of the most distinguishing principles of SOA is the reusability of services. A sustainable and financially prudent approach for IT managers that have to deal with very short application lifecycles demands reusing existing databases and application services, rather than recreating the same business processes and data repositories over and over [Lin99]. Reusability is not a new concept proposed in SOA, since it has already been identified and addressed by previous software paradigms. An example is Component-based development, where a component works as an encapsulated set of functions that other components use. However, the reuse of services in SOA is much more effective since self-defined interfaces guarantee the independence between clients and service providers. This opens the possibility to aggregate services in higher-level compositions. This way, new applications can assemble existing services. Applications that are composed of more than one service are referred to as composite services. Service aggregation makes it possible for an enterprise to orchestrate the execution of more complex composite services that make use of the functionality offered by more fine-grained services, independently of programming language or operating system.

2.2.1.2 Loose Coupling

Unlike in an object-oriented approach, services do not need to be instantiated before they are used. This creates a level of abstraction for the client when using, or reusing, a service from a service provider, allowing client and service providers to remain loosely coupled. Loose coupling is one of the most important preconditions for an enterprise that needs to cope with the pressure of a changing business environment. In an architecture inspired by this SOA principle, services remain loosely coupled from one another and the substitution of a specific service in a composite service can be easily achieved without making any changes to the rest of the services within the composition.

2.2.1.3 Discoverability

Services are discoverable building-blocks allowing their descriptions to be discovered and understood by service requestors that may have interest in using, or reusing, their processing logic. At a SOA level, the principle of discoverability refers to the ability of the architecture to provide a discovery mechanism, such as a service registry or directory [Erl05].

2.2.2 Find, Bind, Invoke

A key communication model that brings the desired level of independence between service providers and service requestors is the find/bind/invoke paradigm [W3C04a, Ora05, Cha04]. This paradigm is also known as service brokering in the literature [PH07]. Here, a service provider publishes a service description in a service registry (1). The process continues when a service requestor asks the registry to find a service that matches certain criteria (find). Once the registry has found a service matching the criteria of the requestor (2), it answers with a contract and the endpoint address of the service. Finally, the service requestor accepts the contract (bind) and invokes the service (invoke) via the endpoint address (3). This process can be seen in Figure 2.1. This paradigm exploits the benefit of discovery, but entails a drawback as well. It requires writing the routing and flow logic into the application, or service requestor that needs to reach the service provided by the service provider. This obstacle can be dealt with if a routing mechanism is included. A method to adopt such a mechanism is explained in Section 2.3.4.

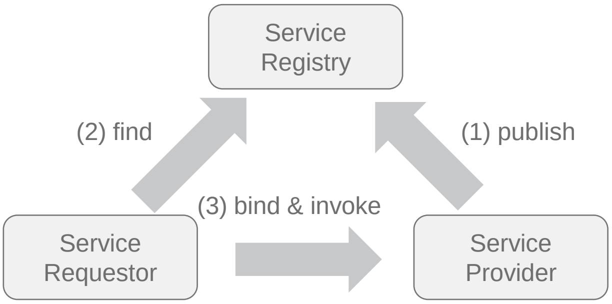


Figure 2.1: The Find, Bind, Invoke Paradigm

2.2.3 Web Services

After the establishment of the XML technology set during the e-Business movement in the late 90s, web services were born as an idea to create a pure, web-based, distributed technology - one that could leverage the concept of standardized communication frameworks to bridge the enormous disparity that existed between and within organizations [Erl05]. Therefore, the success of web services is for the most part due to the standardization efforts that led to the specification of a number of languages, inspired on XML, to support the definition of interfaces, messaging, and the publication of services. These efforts are resumed in the WS-I Basic Profile [WSI04], which comprises three standards. First: the Web Service Description Language (WSDL) [CCM+01], used to describe the public interface of a service. Second: Simple Object Access Protocol (SOAP)

[W3C07a], thought to support the exchange of messages between web services. Third: Universal Description Discovery and Integration (UDDI) [Org04], which would be embraced later to support the publishing of services and discovery. This way, web services were at their early stage a complete technology framework, also known as the WS-I basic profile, robust enough to implement the find/bind/invoke paradigm and collaboration in the definition of SOA as a new architectural style. The definition of a web service provided by the World Wide Web Consortium makes an explicit reference to the standards WSDL and SOAP to support the exchange of messages conveyed over internet protocols [W3C04a]. At the same time, SOA has evolved inspired by the foundational principles of reusability, autonomy, composability and discoverability, amongst others. This has led to a second generation of specifications, known as WS-*. For this thesis, not the whole WS-* stack is relevant, but some standards have been used for the implementation of certain components, such as the Web Service Business Process Execution Language (WS-BPEL). Other standards have been investigated and compared with further alternatives for implementation purposes, e.g., WS-Notification.

2.2.4 WS-BPEL

WS-BPEL [Org07] is the de-facto standard to model composite services and business processes in service-oriented computing. WS-BPEL is an XML-based language that emerged in the area of Business Process Management (BPM) and enables an enterprise to model and execute its business processes and to aggregate existing services in a flexible way. Moreover, WS-BPEL allows for a recursive aggregation of services since a WS-BPEL process can be used, reused, and aggregated as a web service as well. Worth mentioning for an integration architecture that deals with events are the various resources of WS-BPEL to handle events. Supported event types are (1) events that occur when a certain message arrives at a

WS-BPEL process instance, and (2) events that occur when a timer is expired. The first type of events is known as message-based events and the second type as alarms. The WS-BPEL standard offers different activities and control mechanisms to process events, once they arrive at a process instance.

2.2.5 SOA Governance

Many successful enterprises have created value by selecting the right investments and successfully managing them from conceptualization through implementation to realizing the expected value. Such decisions are generally made on the basis of IT governance. The main purpose of IT governance is to achieve an IT alignment with the goals of an organization. The decision-making about IT investments, policies and practices as well as processes to monitor and control IT decisions are prioritized and are all part of the IT governance of a company. SOA governance is an extension of IT governance that focuses on the lifecycle of services and composite applications in an organization's SOA [HPG06] and that is still considered as a research gap [PH07]. The main purpose of SOA governance is to assure that all deployed services in a company truly contribute to enhance value. A well-implemented SOA governance strategy enables companies to increase process flexibility, improve responsiveness, and reduce IT maintenance costs. The adoption of SOA not only involves the operational aspects of a service life cycle management, but also the management of service design policies, reusability guidelines and service change policies. An enterprise deploying an EAI middleware based on SOA also needs to establish a governance framework. Such a framework helps the organization to maximize the business benefits of SOA and mitigates the business risks of adopting SOA by guiding the definition of appropriate services and by measuring effectiveness [HPG06]. To ensure adequate governance, it is necessary to manage services as first-class assets throughout the lifecycle [KKR09]. In coherence with the objectives stated in Chapter 1, the focus is on the reusability of services and on life cycle aspects of service management within the presupposed SOA governance strategy of a company.

2.2.6 Service Lifecycle Management

In the literature there is no uniform view of the SOA lifecycle. Several models have been proposed in BPM [ML08], semantic BPM [WMF+07], software services [KKR09] and Web services [Pap08]. In BPM, lifecycle models are usually oriented to support process modelers in the development of business processes. This support includes discovery and reusability techniques. An example of such a lifecycle model is given in [ML08], which is based on the reuse of process fragments in business processes.

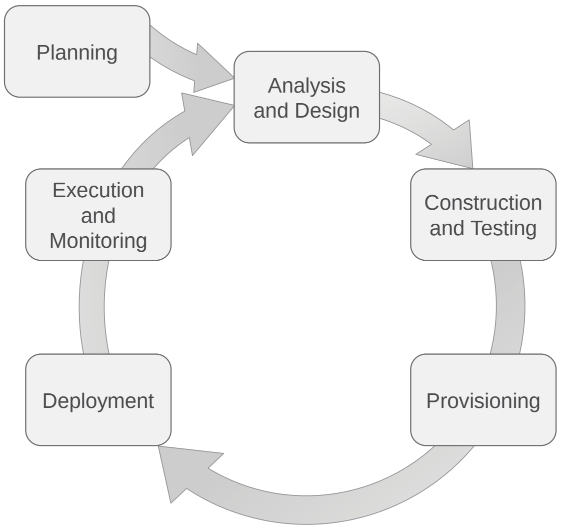


Figure 2.2: Service Development Lifecycle from [Pap08]

This lifecycle model guides the business user in understanding and adopting the concepts of using process fragments in business process modeling and guides the development of a business process modeling tool and a business process repository that support reusing process fragments in business process modeling. Research in Semantic Business Process Management (SBPM) defines the life cycle of processes in four steps: modeling, implementation, execution and analysis [WMF+07]. This model leverages semantic annotations in order to exploit automatic semantic-based discovery in the modeling phase. In the semantic business process analysis two features have to be remarked: the first one is process monitoring which aims at providing relevant information about running process instances in the process execution phase, the second one is process mining that analyzes already executed process instances, in order to detect points of improvement for the process model [WMF+07]. Regarding SOA, IBM defines the SOA life cycle in four phases: model, assemble, deploy and manage [IBM05b]. A more generic business and software service lifecycle is given by Kohlborn et al [KKR09]. Here, the following phases are distinguished: service analysis, service design, service implementation, service publishing, service operation, service retirement. This model aligns with the common management layers in organizations. The Service Development Life Cycle (SDLC) is defined for web services as a highly iterative and continuous approach to developing, implementing, deploying, and maintaining web services in which feedback is continuously cycled to and from phases in iterative steps of refinement [Pap08]. This cycle is shown in Figure 2.2 and described as follows:

The planning phase is a pre-design phase, which is used to observe and evaluate the business environment and decide what services need to be planned. In the planning phase, an analysis of the requirements takes place. In the analysis and design phase, the business requirements are mapped to the requirements of a SOA-based implementation. Additionally, reusable services are identified. The appropriate service granularity is determined, which is a decisive factor for the reusability of the service. In this phase, performance and Quality of Service (QoS) aspects are determined as well. A service repository with possible reusable, process fragments as well as process editing tools may be used to support the design process. After the design phase, the service goes through the construction and test phase. Here, the service is implemented, the service interface is defined. After these steps, the service is tested. The service provisioning phase involves decisions on service governance, service auditing, service metering and service billing. In this phase, mechanisms for the management of QoS need to be put in place in order to enforce the fulfillment of quality agreements and to eventually be able to prove it to the service consumers. In the deployment phase, the service is published in a repository for discoverability purposes and deployed in the running environment. Finally, the service begins to be actually used in the execution phase, where the necessary monitoring mechanisms are also put in place. Monitoring is necessary in order to analyze the business environment and check the effectiveness of the service functions and thus to initialize a new iteration of the loop if needed, based on this feedback.

2.3 Enterprise Service Bus

An Enterprise Service Bus (ESB) is a standards-based integration platform that combines messaging, web services, data transformations and intelligent message routing to connect distributed applications across enterprises while assuring reliability and transactional integrity [Cha04]. The find/bind/invoke operations are inherent to the bus and independent of the business logic, thus enabling loosely coupled integration components to be accessed as shared services and thus providing a highly distributed SOA. An ESB typically supports different messaging, like SOAP [Org07] or Java Message Service (JMS) [MHC00], as well as multiple transport protocols, such as Hypertext Transport Protocol (HTTP) and File Transport Protocol (FTP), allowing diverse applications to be connected to the bus, and thus enable the execution of process flows that require the interoperability between heterogeneous applications.

2.3.1 Evolution from MOM to ESB

The major difference between MOM and an ESB is the total separation between the routing logic of the middleware and the application logic. Services are configured instead of coded [Cha04]. Moreover, the ESB support for multiple transport protocols makes it a preferable option when spanning physical network boundaries. Another characteristic of an ESB, which is not found in regular MOM, is an integrated support for orchestration. An ESB provides a range of process flow capabilities in order to define business processes for one or across multiple business units. Orchestration within an ESB is usually provided by a WS-BPEL engine.

2.3.2 Mediation Services

A very remarkable and distinguishing advantage of an ESB is that mediation services, which are the means by which the ESB can ensure that a service requestor can connect successfully to a service provider (e.g., transformation and routing services), can be provided by third parties [SHL+05]. This feature allows for adding and manipulating mediation services, without affecting service providers or consumers, which contributes to achieve a higher level of flexibility.

2.3.3 Real-time Throughput of Data

The characteristics of an ESB enable applications and services to act as service endpoints that can readily respond to asynchronous events [Cha04]. This concept of integration eliminates the latency problems that originate in the commonly used nightly batch processing. The real-time throughput of operational data up to the business-level applications provides a great opportunity for enterprises to increase their agility in constantly changing environments that are under great adaptation pressure, such as the Real-time Factory.

2.3.4 Content-based Router

Content-based routing is an integration pattern in the category of message routers [HW03] which can be found in messaging systems. A Content-Based Router (CBR) routes a message to the correct recipients based on the message content. This pattern is typical of an ESB. A CBR represents the best match alternative for the implementation of loosely coupled environments. With a CBR put in place, the message sender does not need any knowledge of the interactions or destinations for the messages it produces, since the CBR looks into the content of each message and forwards it to the correct destinations based on this information. A CBR is a type of message routing pattern that is widely adopted in SOA [CN08].

2.4 Semantic Web Technologies

The semantic web initiative [SHB06, BL98] intends to facilitate the understanding of meaning of information [She99]. Semantic interoperability issues often arise when solving EAI scenarios via the SOA paradigm, such as slow discovery processes or required human intervention during the definition of mappings. A number of semantic web standards have been proposed by the W3C to deal with problems regarding semantic interoperability. The standards that are relevant to this thesis are described ahead.

2.4.1 Resource Description Framework

Resource Description Framework (RDF) [W3C04b] is a standard model for data interchange on the Web. RDF offers features that facilitate data merging even if the underlying schemas differ, and it specifically supports the evolution of schemas over time without requiring all the data consumers to be changed [W3C04d]. RDF is used to model the relationships between two entities. This is done in form of subject-predicate-object statements, also known as triples or RDF-triples. All three parts of a triple are identified by a Universal Resource Identifier (URI) [IET05].

2.4.2 The Web Ontology Language

The term ontology originally refers to the metaphysical study of the nature of being and existence. In the field of knowledge engineering, an ontology is a data model that represents a conceptualization of the world within a domain. The term is tightly related with semantics. An ontology usually contains: classes, individuals, attributes, relations, restrictions, rules and axioms. Ontologies are used to model knowledge and thus to share the same definitions of terms. A very important usage of ontologies is the establishment of knowledge bases to be able to perform reasoning processes and thus to derive higher-level context information. A few ontology languages have been developed so far, but one of them has prevailed: The Web Ontology Language (OWL) [W3C04c] proposed by the W3C. OWL possesses certain features inherited from description logics, such as the ability to realize inference operations. It is inspired on the W3C standard RDF Schema (RDFS) [W3C04d], which was designed to share RDF vocabularies. OWL offers a higher expressiveness than RDFS. The OWL language provides two specific subsets: OWL DL and OWL Lite. The complete OWL language is known as OWL Full to distinguish it from the subsets. OWL Lite is preferred if a friendly syntax or decidable inference are on the top priority. OWL Lite has an even simpler syntax but it is less expressive. It is a syntactic subset of OWL DL and it has a more tractable inference. OWL DL puts constraints on the mixing with RDF and requires classes, properties, individuals and data values to be disjoint. Finally OWL Full is the most expressive version and OWL Full allows the free combination of OWL with RDF Schema. Moreover, OWL Full does not enforce a strict separation of classes, properties, individuals and data values. For further information on OWL and other ontology languages, it is recommended to consult the literature [W3C04c, W3C04d, HPH03].

2.4.3 Semantic Web Services

The purpose of semantic web services is to increase quality of discovery of appropriate services by appropriately describing what services do [CS06]. WSDL does not contain semantic descriptions, it only specifies the structure of message components using XML Schema constructs. In order to solve problems regarding semantic interoperability, which arise in the exchange of data, some standardization efforts have been made to support the interoperability between web services. A standard of interest for the annotation of web service descriptions is the Semantic Annotations for WSDL and XML Schema (SAWSDL) [W3C07b]. SAWSDL is an evolution of WSDL-S and defines how semantic annotations can be included into WSDL documents with references to conceptual models. SAWSDL offers direct support for functional and data semantics. Service providers can additionally incorporate non-functional and execution semantics with the WS-Policy framework [W3C06, Spe10]. This is an important feature to improve and to accelerate service discovery [SSV+04]. The most important benefit of SAWSDL is that it allows a free choice of language for the definition of terms in the conceptual model that is referenced from an SAWSDL document. This feature offers great flexibility.

2.5 Current State of Integration

In this section, the current state of integration is described by presenting related work in different research areas that are relevant to the area of application of this thesis. First, an overview of the planning and production environments in manufacturing is given. Then, context-aware applications and currently applied methods of integration are described. In Section 2.5.3, a number of approaches that realize an event-driven SOA are presented followed by a very active area of research, namely adaptability in BPM. An approach following the Model-Driven Architecture (MDA) paradigm to solve EAI scenarios is also described. Finally, the limitations of current research approaches in the EAI domain for manufacturing environments are described.

2.5.1 Overview of Production Environments

In this Section, an overview of production environments is given. Three levels are distinguished: the production planning level, the production control level and the production process control level. The planning level is usually governed by an ERP system and other tools used in the production plannning, such as layout planning tools or Computer-Aided Design (CAD) tools. The production control level is ruled by the MES. The MES

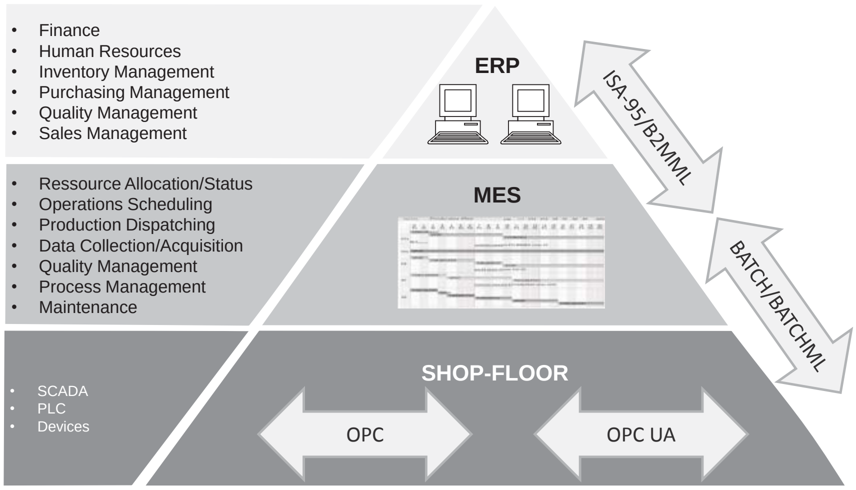


Figure 2.3: Production Planning, Procuction Control and Process Control

performs tasks related to the scheduling of operations and to the acquisition of data from the shop-floor, including resource status data, quality management data and maintenance data. The process control level comprises several production systems, including devices, Programmable Logic Controllers (PLC) and Supervisory Control and Data Acquisition (SCADA) systems. At each level, there are different protocols and stardardized data models that manage production data at each level and transfer it to the next level. These standards include data models, such as the norm ISA-95 [Int00] for production control systems, and communication protocols, such as Object Linking and Embedding for Process Control - Data Access (OPC-DA) [OPC02], which specifies the real-time communication standard to access data in devices from different vendors at the shop-floor level. Recently, these specifications and standards in the manufacturing industry have been updated in order to comply with the recent trends in integration, namely XML and SOA. Thus, ISA-95 data can also be represented by means of the Business to Manufacturing Markup Language (B2MML) [WBF08b], which is a XML-based implementation of the ISA-95 model. In the same manner, BatchML [WBF08a] is becoming a well-accepted alternative to the ISA-88 in order to exchange master data or equipment data, and OPC Unified Architecture (OPC-UA) [OPC09], which is the most recent specification from the OPC Foundation, updates the former OPC specification in order to enable the cross-platform communication between devices in service-oriented architectures. The whole picture of these standards in production environments is shown in Figure 2.3.

2.5.2 Context-aware Applications and Integration

Context-awareness is a fundamental requirement for applications that operate in constantly changing environments, such as a factory. For modeling the context of a factory, the Smart Factory approach represents a context-aware manufacturing environment to handle turbulences in real-time production using a decentralized information and communication platform [LCW08]. This platform, known as Nexus [SFB01], supports a large variety of context-aware applications. A global context model is provided by a federation layer that integrates different local models to an integrated view [NGS+01]. This model of integration has proven to be a valid approach for providing information to location-based and spatially aware applications. On the research of adaptability of information systems in manufacturing environments, the data change propagation system Champagne [CHL+05] has to be remarked. This solution supports the enterprise by a generic approach capable of managing data dependencies between information systems and of transforming data from a source information system to dependent information systems [CHM02]. The prototype comprises a propagation engine that interprets routing scripts based on the eXchangeable Routing Language (XRL) [VVK01], a data dependency manager to edit the routing scripts as well as their embedded transformation scripts, which are written in XSLT files, and a repository to support both [RCH+02]. XSLT [W3C99b] is a language for transforming XML documents into other XML documents. Champagne has been proven to provide a flexible integration solution in a manufacturing environment by integrating a Digital Factory solution, a factory planning table and an assembly configuration tool [CHL+05]. Moreover, Champagne is able to access data of data services by using a self-defined Data Service Description Language (DSDL) to create the appropriate transformation scripts and to request access to the data [HCM05]. Champagne has also been in focus of research to integrate and provide the system as a service in service-oriented integration platforms [MJH+09].

2.5.3 Event-driven SOA

The challenges of integrating explicit and implicit service interactions that can be found in SOA and Event-Driven Architecture (EDA), respectively, have called the attention of several research groups [M06, QYS+08, LC08, WMK+08]. A combination of both architecture styles — SOA and EDA — is introduced in [WMK+09]. The result is a model based on Event-driven Process Chain (EPC) flowcharts representing a standardized, event-centric business process notation for defining an initial process model. This initial model is then transformed into WS-BPEL as a service-centric execution model for actual process enactment. Regarding the mapping of EPC to WS-BPEL, several transformation techniques are combined to preserve the structures of the EPC graph in the WS-BPEL process model and to map these to the correct WS-BPEL activities. A mapping of the EPC graph to simple events and the specification of Event-Condition-Action (ECA) rules in a Complex Event Processing (CEP) system makes the deduction of complex events possible. These complex events trigger the corresponding WS-BPEL process instances.

The main advantage of this approach is the separation of tasks for domain experts, which design the EPC graphs, and for IT experts, which manually add WS-BPEL details at the end of the mapping process. In order to enable the integration of web services into EDA, the specifications WS-BaseNotification [Org06a], WSBrokeredNotification [Org06b] and WS-Topics [Org06c] were defined as a publish/subscribe notification standard. However, these specifications cannot differentiate between simple and complex events. They do not define any mechanism for event correlation and are not as expressive as other CEP solutions. That is one reason why several approaches work on extending the expressiveness of these specifications, as in [DS08].

2.5.4 Adaptive Business Management

An approach to achieve quick adaption of processes is Adept2 [RRK+05], which provides adaptive process management and visualization of the effects of ad-hoc instance modifications. Another option to support domain experts is to mine action patterns from existing process model collections and suggest additional activities to the modeler during the modeling act, as in [SWM+09]. Furthermore, this kind of support for process modelers can be extended to the optimization of processes. To achieve this, a business optimization platform is needed to manage formalized process optimization patterns for detecting and implementing process improvements [NRM11]. SBPM represents another option to optimize the management of business processes. SBPM aims to achieve a higher degree of automation in BPM by using semantic technologies. The functional requirements for each phase of the BPM lifecycle and the benefits of adopting semantic technologies are explored in [WMF+07]. The major benefits are the possibilities for automated service discovery and for dynamic binding of services to process tasks during process execution. A reference architecture [KVL+08] and implementation of a SBPM system has been carried out within the SUPER project [SUP09]. The integration layer is based on a semantic service bus. This contribution is a conceptual architecture that focuses on service orchestrations.

2.5.5 A Model-driven Approach

A different point of view to solve EAI scenarios is to generate executable code based on data integration models. The GENIUS tool [SL09] is a representative example of such solutions. GENIUS is a model-driven approach that uses a parameterization of integration patterns methodology to generate executable EAI artifacts for different execution environments. Comparable approaches can be found in the literature [BSM+06, FYL+08].

2.5.6 Service Provenance

Service provenance is usually adopted in approaches that focus on monitoring quality of service in service runtime environments. In [MRL+09], the authors propose an approach, called VRESCo, where service provenance graphs are built to visualize service events and metadata. The VRESCo runtime environment also processes provenance queries, which can be used to establish provenance subscriptions. Moreover, it comprises mechanisms for advanced event and notification processing in service runtime environments [MRL+08]. The concept of semantic provenance was introduced in [SSH08] for e-science domains. This approach advocates the creation of high-quality information using specialized services.

2.5.7 Limitations of Current Research

The Champagne [RCH+02] approach and the Smart Factory [LCW08] are two research projects designed to integrate manufacturing applications and to manage context in factories, respectively. The missing flexibility and agility is especially remarkable in these approaches. The lack of an integration model in Champagne prevents the framework itself to react to turbulent scenarios or other domain-specific situations that require a quick adaptive reaction of the integration processes. The lack of an integration model does not contribute to reduce the complexity of integration, since users are not assisted in the adaptation of its transformation and routing scripts. In the Smart Factory [LCW08], the management of context information in the factory is realized in a very effective manner [LCW09] by leveraging the benefits of the Nexus augmented reality model [NGS+01]. However, the approach lacks the mediation services and self-adaptation mechanisms that would provide the framework with more adaptability and agility. Current research approaches, such as the Semantic Service Bus [KWV+07] in the SUPER project [SUP09], GENIUS [SL09] or Adept2 [RRK+05], adopt the service-orientation paradigm in the EAI and BPM domains in order to fill the adaptability and agility gap in IT infrastructure systems. The adoption of the SOA paradigm in the EAI domain of manufacturing environments is investigated in this thesis. The adoption of the SOA paradigm in the manufacturing domain entails several integration principles, which are summarized in Chapter 3. The proposed approach of this thesis relies on these integration principles.

2.6 Summary

In this chapter, the foundations and related work to the research issues described in Chapter 1 are given. First, the fundamental concepts of

EAI have been explained, including the distinction between the different types of integration, the concept of middleware as well as the most influential communication and messaging patterns in current approaches. In this Chapter, the paradigm of service orientation has been described, as well as web services, which represent the most accepted implementation of the SOA paradigm. The principles of integration for the Real-time Factory in Chapter 3 and the integration approach presented in Chapter 5 are based on the SOA principles, which are explained in Section 2.2.1, and on the ESB, which is a standards-based integration platform based on SOA concepts (see Section 2.3). A quick overview of semantic technologies is given in Section 2.4. Finally, the current state of integration is given describing different research projects in the domains that are relevant to the area of application of this thesis, such as adaptation [SWM+09, RRK+05, NRM11, WMF+07], factory context management [LCW08, LCW09] and integration [RCH+02, CHL+05], as well as event-driven SOA [M06, WMK+09, Org06a]. Current research approaches in the EAI domain for manufacturing environments have some limitations, such as the lack of flexibility and mechanisms for the agile adaptation of integration processes. Based on these limitations, the adoption of the SOA paradigm for EAI scenarios in the Real-time Factory stands out as the most appropriate approach to follow, as illustrated by other SOA research projects in the EAI and BPM domains [KWV+07, SL09, RRK+05].

Chapter 3

Principles of Integration for the Real-time Factory

In this Chapter, the principles of integration that guide the proposed solution for the Real-time Factory are given. Based on the foundations and current software paradigms, as well as the current state of integration offered in Chapter 2, this Chapter offers an insight into the principles of integration for the Real-time Factory.

First, a discussion about the deficits, potential and needed optimization in past and current integration efforts is given in Section 3.1, which details some of the lessons from past systems integration strategies. These lessons are valuable knowledge that serves as guideline to achieve the objectives of this thesis. In Section 3.2, it is described how these lessons apply to the scenario of the Real-time Factory and which integration requirements the Real-time Factory presents. Based on these requirements, the principles of integration for a middleware infrastructure are given in Section 3.3. In Section 3.4, the principles of integration for Real-time Factory applications are described. How a SOA-based approach matches these principles is explained in Section 3.5. Finally, a summary and some conclusions are given in Section 3.6.

3.1 Lessons Learned

A number of studies [Sny91, Gun97, Eva01, Sin97] show that past and current integration initiatives in manufacturing, e.g. Computer-integrated Manufacturing (CIM) [Wal92], struggle to achieve the expected results. In the case of CIM, a number of integration and adaptability issues have been proven crucial for the successful implementation of such a cross-departmental integration strategy. These issues can be identified as: organization, strategic, behavioral, operational and technological [Gun97]. Regarding the technological issues there are four important findings that have been reported in the past.

- Problem 1: Multiple vendor installations. Most manufacturing firms have many multivendor computer-based systems that have evolved independently. The need to connect computer and departmental system has been addressed by vendors of specialized interface equipment to provide a means for heterogeneous system interconnection [Sny91]. This has led in many cases to an accidental architecture, which is extremely complex and expensive to maintain.

- Problem 2: System incompatibility. The lack of infrastructure and integration methods has been a great obstacle to achieve the desired system compatibility [Gun97]. System incompatibility is perceived to be important and has been discussed in the literature for quite some time. Much progress has been made in addressing the incompatibility problem as it relates to linking automation on the factory floor. It remains problematic, however, for firms trying to achieve a higher degree of functional integration - linking diverse systems across functional boundaries [MS94].

- Problem 3: Lack of flexibility. As pointed out by Babbar and Rai [BR90], while CIM integrates the system components, it does not necessarily introduce flexibility into the system. This is due to a high number of point-to-point interfaces, which lead in many cases to a rigid integration of tightly-coupled applications.

- Problem 4: Lack of Standards. The lack of a unique set of standards that fulfills all the requirements of a CIM system. Although many commendable results have been obtained from various studies, more than one standard has emerged, resulting in confusion for potential users [BDE95, NG94].

In order for an integration strategy to be successful, not only technological aspects are decisive, but also organizational and strategic aspects, such as the commitment of top management or the aversion to risk of investing in new technology, can be crucial. In this thesis, it is assumed that the implementation of the proposed integration architecture would be embedded in the integration strategy of a company that takes into account such non-technological issues as well.

3.2 Integration Scenario: the Real-time Factory

Based on the findings and experience reports regarding the technological aspects of CIM systems, some observations in terms of today's perspective on the described problems are given in this section. Also, how these

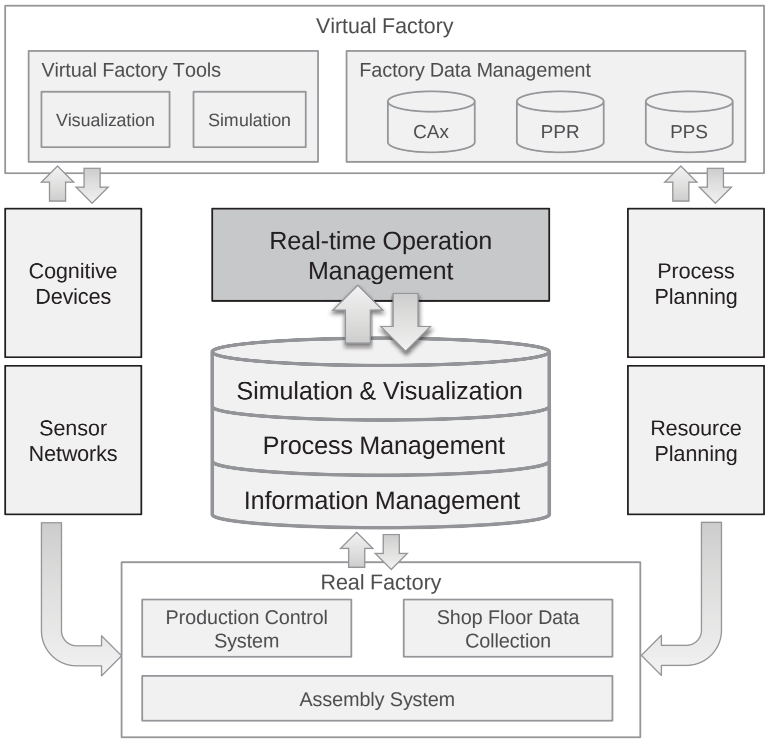


Figure 3.1: The MSB and its Five Layers of Integration

problems relate to the integration scenario of this thesis, the Real-time Factory, is described in this Chapter.

The Real-time Factory tightly integrates the Real Factory with the Virtual Factory by continuously communicating, connecting and evaluating the factory's operational data [JWW09]. As shown in Figure 3.1, the main objective of this integration is to achieve a real-time operational management of the processes and resources of the factory. This can be achieved by supporting the process and resource planning tasks of the Digital Factory with the leverage of sensor networks and cognitive devices that acquire the actual state of the factory, e.g. machine states, flow of material or product quality. By using ubiquitious computing techniques and self-organizing sensor networks, data is collected, aggregated and processed in an intelligent way. These data are integrated and managed in a repository forming the basis of context-aware systems in the Real-time factory [WJE+05]. This basis comprise factory information management as well as process management in order to provide the simulation and visualization tools of the Virtual Factory with the up-to-date information about the operational state of the factory.

In order to aggregate all factory data relative to the resources and processes, it is necessary to provide an integration platform that complies with certain requirements. Regarding the realization of such an integration platform for the Real-time Factory, these requirements derive from the application of the lessons learned of the integration efforts made in the past in the manufacturing domain, e.g. CIM, to the Real-time Factory. These integration requirements are summarized in Sections 3.3 and 3.4. The requirements for the Real-time Factory are also based on some observations in terms of today's EAI technologies and standards. The observations are detailed ahead:

- Multivendor installations. Solution providers rarely cover all areas involved in the Real-time Factory, therefore the problem of multiple vendors remains as part of the motivation of many integration scenarios, including the Real-time Factory, at least for the next few years. The actual problem is the accidental architecture evolved in many companies. In order to solve this issue, companies have widely adopted SOA-based solutions. As the AMR Research survey in 2005 shows, 72% of the internal SOA deployments aimed for application, process and data integration [AC05]. However, this survey also shows a substantial increase in companies that cannot reconfigure business processes as needed (three times higher in companies using SOA compared to companies planning to use SOA). This reveals the need to implement the mechanisms that can ease the reconfiguration of processes (especially in SOA environments). The reconfiguration of processes is a fundamental requirement of the Real-time Factory. Therefore, the challenge of reconfiguration is critical for the success of the presented approach.

- System compatibility. The leverage of XML and Web Services has reduced part of the system incompatibility problem. By separating the interface definition and the application logic, current integration middleware solutions can actually provide the required mechanisms to increase system interoperability. The problem of functional integration that was accounted for in multiple CIM surveys has been addressed by BPM technologies, such as WS-BPEL [Org07]. Moreover, current integration solutions based on an ESB architecture offer diverse connection choices that reduce the effort required to connect applications to the ESB. Some of these choices include SOAP/HTTP, JMS [MHC00], FTP or Simple Mail Transport Protocol (SMTP).

- Flexibility. The lack of flexibility is actually a problem that has persisted over time. The abundance of approaches based on a tight integration and the increasing complexity of current scenarios is responsible for the persistence of this problem. As described in Chapter 1, the lack of flexibility is one of the most important issues in the Real-time Factory. The changing conditions of the production environment require a high degree of flexibility regarding the reorganization of the involved ICT factory resources.

- Standardization. The lack of standards is no longer a problem regarding integration. As it will be shown in the next Chapter, the acceptance and adoption of XML and web services standards (WSDL, SOAP, UDDI) has enabled that different functional domains of the manufacturing industry can speak the same language regarding the exchange of data, such as the B2MML [WBF08b] or STEP-XML [Int07a]. However, standards alone do not guarantee the success of an integration approach. A number of issues need to be taken into account in the Real-time Factory, such as the need for asynchronous communication, the need to manage domain knowledge regarding the life cycles of the installed services and integration processes and the agility that is required to adapt services and processes. An integration scenario with dozens of systems and hundreds of point-to-point interfaces based on the request/reply mechanism can result in a very tightly-coupled and complex integration approach, independently of the fact whether communication is made using SOAP messages over HTTP. This shows that standards are not a sufficient condition to achieve the desired flexibility and agility, provided by more loosely-coupled approaches.

These observations under the current perspective serve as guideline to establish the integration principles that are put into practice in the proposed integration approach for the Real-time Factory.

3.3 Principles of Integration for Middleware Infrastructures

Based on the aforementioned observations on integration issues under the current perspective and on the integration challenges of the Real-time Factory, which were described in Chapter 1, there are a number of principles of integration to follow. The principles of integration that are described in this section are to be applied to the conceptualization and to the design of a middleware architecture that intends to meet the challenges of integration in the Real-time Factory.

3.3.1 Ease of Reconfiguration

One of the most important requirements for the Real-time Factory is the adaptation of its information flows to the constantly changing conditions. Such information flows extract, transform, route and load data from diverse applications and systems thus enabling a seamless integration. Sometimes, changes need to be made in these information flows. For example, a data source may update the XML schema of its output or a data transformation service may require new input data. Sometimes, the evaluation of the factory context may point to wrong operations, such as false failure detection, produced by wrong configuration of an information flow. The configuration of information flows is crucial for the correct execution of all integration services that are needed when two or more systems exchange data. A correct execution of such integration services includes for instance the correctness of the order of execution, the correctness of the exchanged data and appropriate timing. The Real-time Factory is a changing production environment that requires information flows to be changed and adapted. This can be done by means of reconfiguration. Keeping in mind the objective of increasing responsiveness, which was stated in Chapter 1, reconfiguration must be done with ease. This involves abstracting humans from complex parsing expressions or transformation scripts in the phases of modeling and adaptation of information flows.

3.3.2 Loose Coupling

A fundamental requirement for the desired agility in the Real-time Factory is a loosely-coupled integration. Loose coupling is best defined by defining the opposite. Two systems are tightly-coupled when one of them needs explicit knowledge about the other, e.g. by means of a handle that contains a reference to an endpoint, in order to define their dependency.

Therefore, loosely-coupled systems do not need any knowledge about each other in order to exchange data. An integration middleware that keeps information systems and applications loosely-coupled provides a great degree of flexibility in terms of adaptation, enabling the reorganization of integration processes without having to modify the business logic of the involved systems and applications.

3.3.3 Asynchronous Thinking

Information systems and manufacturing applications in production environments are usually interconnected following mostly a pattern of asynchronous communication. Most interchanged messages at the shop floor level are based on some kind of event, alarm or notification. Such events are associated with an event emitter and one or more event consumers that react to the event according to a prefixed procedure. The software architecture paradigm for this type of event-centralized communication and integration of systems is known as EDA. EDAs are needed in the shop floor of factories to propagate business-relevant events to the appropriate destination within the enterprise. This architectural model requires asynchronous messaging mechanisms, such as Message Queuing (MQ) [BHL95]. These mechanisms have proven to provide reliability [HW03, Hoh04], but at the same time asynchronous communication introduces a number of issues. These issues, which were defined as "Architect's Dream or Developer's Nightmare" [Hoh07], are due to the accustomed thinking of synchronous messaging. However, asynchronous messaging mechanisms are needed in the Real-time Factory due to the need of propagating multiple events that originate at the shop floor and that are relevant to other manufacturing applications.

3.3.4 Standards-based Integration

The leverage of standards enables companies to avoid vendor lock-in situations, as well as specialist consultants, which reduces the costs of integration. Additionally, the usage of standards enables a company to integrate internal applications with business partner applications, such as the integration of a procurement system and a customer order application of a supplier. The emergence of standards and industry specifications, produced either by vendor alliances or by standards consortia, is the result of the lessons learned from the past, including the interoperability issues, the lack of flexibility and the increasing costs of integration. Therefore, an integration middleware must support the most common standards and protocols in their domain regarding communication, transport and messaging. Among the most relevant and meaningful standards for integration in the last decade is XML, which is a de facto standard in the industry for sharing structured data among applications. In the manufacturing industry, a number of well-accepted standards and specifications need to be taken into account by a middleware infrastructure. These include data models, such as the norm ISA-95 [Int00] for production control systems or the Standard for the Exchange of Product Model Data (STEP) [Int07b] for the exchange of product-related data, and communication protocols, such as the OPC-DA Specification [OPC02], which specifies the real-time communication standard to access data in devices from different vendors at the shop floor level. Recently, these specifications and standards in the manufacturing industry have been updated in order to comply with the recent trends in integration, namely XML and SOA. Thus, ISA-95 data can also be represented by means of B2MML [WBF08b], which is a XML-based implementation of the ISA-95 model. In the same manner, STEP-XML [Int07a] is becoming a well-accepted alternative to the STEP file in order to represent product data, and OPC-UA [OPC09], which is the most recent specification from the OPC Foundation, updates the former

OPC specification in order to enable the cross-platform communication between devices in service-oriented architectures. The evolution of standards and specifications in order to adapt to current trends and the emergence of new standards need to be taken into account in the conceptualization, design and implementation of a middleware architecture for the Real-time Factory.

3.4 Principles of Integration for Real-time Factory Applications

The following principles of integration serve as guideline for applications that are connected to a middleware in the Real-time Factory. The implementation of these principles is a fundamental requirement for applications in order to achieve the desired degree of flexibility, agility and adaptability in the Real-time Factory.

3.4.1 Well-defined Interfaces

The information flows that seamlessly integrate applications across the enterprise need to be composed before they can be executed. The composition of such information flows requires certain knowledge about the applications' functionality that they are putting together. Moreover, an application needs to describe its interface in order for other applications, or a middleware, to know how to interact with it. Such a description must include the specific protocol bindings that the application supports, how to invoke operations and where to send messages. This information must be well defined in order to be human-readable and machine-readable. The latter is a requirement for environments that need to support a certain degree of automation.

3.4.2 Separation of Implementation and Interface

The interface of an application defines the functionality that is visible to others and provides information about how to access this functionality, e.g. the operations available, protocol bindings and data types. These are publicly available descriptions that other applications can use to know how to invoke the functionality of a specific application. The implementation of the different functionality is usually hidden from external entities. This concept is inherited from modular design principles of software architectures and it's important that this principle is kept in the Real-time Factory. The implementation of business functions can be realized by a specific software package, a suite of components, a legacy application or a commercial application. The implementation details should be kept isolated from the interface definition in order to cause the minimum amount of disruption to other components when changing parts of the implementation. In order to face the challenge of heterogeneity in the Real-time Factory, applications that connect to a middleware need to provide platform-independent interface descriptions. Applications can easily comply with this requirement if the interface is kept separated from the implementation details. It is irrelevant if the implementation of an application's functionality is realized in Java, .NET, or other programming language, as long as these details are independent of the interface definition. This way, a middleware can integrate multiple applications independently of the implementation platform.

3.4.3 Standards-based Interfaces

Provided the separation of the interface from the implementation details, applications must focus on interface definitions that lead to a better interoperability. In order for applications to be effectively integrated and to minimize the costs of integration, the leverage of standards for the description of interfaces becomes a fundamental requirement. For the interoperability of an application, it is essential that other applications understand the capabilities and the functionality, including its syntax and semantics that it provides. In the case of a middleware that decouples applications, the reuse of the business functions that applications provide can be facilitated by leveraging standards that ease the definition and interpretation of the capabilities, which are described in the interface. This way, by easing the reuse of business functions, the composition of complex business services can be accelerated. Furthermore, well-defined standards-based interfaces favor a higher degree of automation regarding the composition of information flows that integrate business functions that are distributed across multiple applications.

3.5 Service Orientation as an Integration Approach

The principles of integration for middleware architectures and applications in the Real-time Factory that have been described in this Chapter can be implemented by a service-oriented approach. The match between the principles of the SOA paradigm and the aforementioned principles has guided the conceptualization, design and implementation of the approach presented in this thesis. Loose coupling, reusability and flexibility are SOA promises that are needed to solve the challenges described in Chapter 1. The maturity and adoption of relevant standards for integration have helped to foster the emergence of the ESB as a technology trend [Cha04] for the implementation of an event-driven enterprise SOA. An ESB provides the integration capabilities of a middleware, such as asynchronous messaging and leverage of XML-based standards, to comply with the principles described in this Chapter. Regarding the principles of integration for applications, web services provide an accepted stack of standards in the industry that ease the publication of well-defined interfaces, via WSDL [CCM+01], for other services to discover, as well as a messaging protocol, i.e. SOAP [W3C07a], which is currently the most widely accepted standard in the message communication domain [Pap08]. Moreover, the leverage of the extended stack of WS-* standards provides additional features in integration architectures, such as reliability, compliance or security. Web services and other technologies and standards that support the SOA principles are mature enough for its adoption in many manufacturing domains, as it will be shown in Chapter 4. In Chapter 5, an extensive description of the proposed approach and the application of these principles of integration are given.

3.6 Summary

Given the lessons learned from past approaches to integration in the manufacturing industry, i.e. CIM [Gun97, Sny91, Sin97], a set of principles of integration is given in this chapter. These principles of integration are requirements for middleware infrastructures and applications in the Real-time Factory. Therefore these principles are also based on the objectives stated in Chapter 1. The principles of integration for middleware are:

- Ease of reconfiguration
- Loose coupling
- Support of asynchronous communication patterns
- Standards-based integration.

The principles of integration for applications in the Real-time Factory that are connected to a middleware infrastructure in order to exchange messages with other applications can be summarized in:

- Separation of implementation and interface
- Well-defined interfaces
- Standards-based definition of interfaces.

At the end of the chapter, a description of the SOA principles that match these requirements is given. Complying with the described principles of integration is the main criterion that explains the choice of SOA and web services as a technological foundation for the integration of information systems in the Real-time Factory.

Chapter 4

SOA in Manufacturing

In order to study the viability of an integration architecture based on service orientation, different research studies on service-based integration in five domains of manufacturing were carried out. The investigated domains are: digital factory, PLM, Product-Service Systems (PSS), Engineering-To-Order (ETO) enterprises and reconfigurable production systems. First, the current state of service orientation in manufacturing is described. Then, a brief description of these five studies is given followed by some conclusions.

4.1 Penetration and Current State of SOA in Manufacturing

The emergence of the web and its associated technologies and standards supported the exchange of manufacturing data over the web, which has been, and is still, very successful in different ERP domains, such as logistics or distribution. ERP vendors developed software solutions, which have defined the so-called web-based manufacturing era. Web services dominate this kind of application integration. The use of XML-based standards has a great responsibility for the success of web-based manufacturing. In a typical web service communication, services are hosted in a server and can be accessed over a network, such as the Internet. The usage of web services and web-based applications quickly expanded to other domains, e.g. customer relations and Supply Chain Management (SCM). The capability to exchange information with external entities, such as suppliers or customers had a positive impact in manufacturing factory operations by accelerating transactions and creating an opportunity to automate others. However, this combination of component-based middleware and Web technologies in order to integrate business processes and applications proved to be insufficient for many reasons. For instance, this type of simple integration approach does not consider issues such as integration of different data models, workflow engines, or business rules, to name just a few [Sta02]. EAI solutions, such as Java Remote Method Invocation (RMI) [Ora], IBM Websphere MQ [IBM], CORBA [Obj91] or Microsoft Message Queuing (MSMQ) [Mic00], solved most of these issues, but most solutions were proprietary, complex to use and could not interoperate with each other [Hen06, GJ05]. Soon, the idea of building a Broker-based middleware using Internet protocols such as HTTP and XML as a data marshaling solution was proposed [Sta02] and started to be put into practice in the manufacturing industry [Mes08]. Not much later, the characteristics of a layer of communication and integration logic between applications combining SOA concepts and Web services to coexist with deployed applications coalesced into the concept of the ESB [Cha04, PH07]. Parallel to these developments, as applications started to proliferate across the enterprise, new standards were needed to model and execute the processes that would coordinate the different activities across multiple systems. Therefore, in the early years of the 2000-2010 decade, several proposals were made to the standardization institutions (OMG, OASIS, W3C) to solve this problem, e.g. BPEL4WS (2003, later renamed to WS-BPEL in its second version) [Org07], Web Services Choreography Description Language (WS-CDL) [W3C05], Business Process Management Notation (BPMN) [Obj09]. At the end, WS-BPEL has prevailed as the most influential execution standard in the market [RLL09]. Its modeling and execution capabilities permit to model business processes, or fragments of them, and execute them as workflows in a workflow execution environment. Part of the explanation for the dominance of WS-BPEL in industry, including the manufacturing industry, is the ability to invoke external entities, which are called services in a SOA environment, and allow this way an orchestration of services. This feature, and the fact that WS-BPEL was a web service-based proposal, has made it a dominant standard in environments where web services were already widely adopted, e.g. the manufacturing industry.

4.2 Leverage of a Data Integration System in SOA

One of the most important characteristics of SOA is the reuse of services. On the data level, a number of middleware solutions have been used in the manufacturing domain over the last few years [LCW08, RCH+02, CW08, SKN+07]. In order to solve the problem of different data models, which was one of the shortcomings of the first EAI middleware solutions, the dependencies between information systems can be managed. A system that follows this approach for a manufacturing environment is Champagne [RCH+02], which has been already introduced in Chapter 2. Champagne is a change propagation system that manages dependencies between applications by means of transformation scripts. A propagation engine routes the changes in relevant pieces of information from one application to another by using routing scripts. Champagne was deployed in a manufacturing environment and tested for several scenarios, including the integration of a PPR-Hub, a factory planning table and an assembly configuration tool [CHL+05]. This change propagation mechanism provides a high degree of interoperability by enabling systems to propagate data in an event-driven message-oriented architecture. However, Champagne imposes some flexibility barriers. Firstly, it does not provide any kind of functionality-based integration. The integration is made exclusively at the data level, impeding business process modelers to integrate the functionality of systems into larger composite services. Secondly, the lack of integration at a functional level hinders having a comprehensive model that provides a view on the information flows that propagate data between applications. Thirdly, assuming the constant change in production systems and that different requirements arise as new technologies establish, it is possible that Champagne needs to extend its functionality in order to provide further propagation methods. If extending the system functionality was needed, most parts of the application logic would need to be modified, which would be tedious and costly. In order to deal with these issues, different alternatives were investigated to integrate Champagne into a SOA [MJH+09] and facilitate its reuse and extensibility by applying the SOA principle of reusing services.

4.2.1 Champagne as a Service

In this first approach it was considered how a data propagation system, like Champagne, can be integrated into a SOA environment. In this case study, a change in a SCM system needs to be propagated to an ERP system. The most basic scenario based on the use of services is to expose all participating applications in a transaction as services. Likewise, each entity has a role assigned to it: the SCM acts as the service consumer, and the ERP as the service provider. Champagne will receive the request from the SCM application and forward it to the ERP application via a change request to modify some of its data. Therefore, Champagne acts as service provider for SCM and as service consumer for the ERP system. In step 1 the registration of services is done by publishing the WSDL descriptions on a service registry. The lookup of the needed services to propagate the data is carried out in step 2. Finally, in step 3 the invocation of services takes place. Here, SOAP over JMS is used in order to preserve the queuing capabilities of the event-driven architecture of Champagne. The communication scheme is depicted in Figure 4.1. This first approach has a major advantage: exposing Champagne as a service allows for business process management tools to find the service and compose automatic change propagation processes that must be executed periodically. This is a major improvement that can be achieved through SOA-based application integration. This approach enables a flexible reuse of the change propagation service.

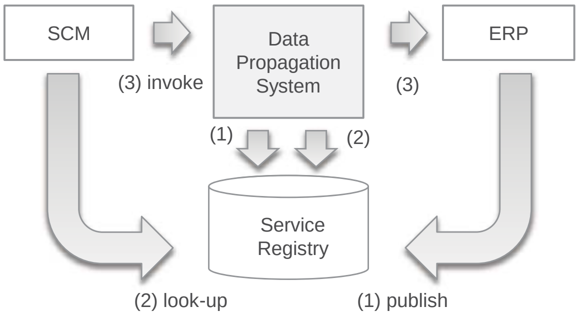


Figure 4.1: Champagne as a Service

4.2.2 Connection of Champagne to an ESB

Exposing Champagne as a service has major advantages as it has been detailed in the first approach. However, some things remain tedious. For example, no communication with external services is possible unless a gateway is built, which is capable of accepting SOAP messages over HTTP and convert them to SOAP over JMS. This gateway should then preserve the queuing capabilities when necessary. For this purpose, a mediation layer is needed to establish proper control of messaging as well as to apply the needs of security, policy, reliability, and accounting. The delegation of these interoperability requirements can be done by means of an ESB. This approach considers the connection of the Champagne Service to an ESB. The ESB is responsible for the proper control, flow, and translations of all messages between services, using any number of possible messaging protocols. An ESB pulls together applications and discrete integration components to create assemblies of services to form composite business processes, which in turn automate business functions in an enterprise [PH07]. An ESB not only promotes loose coupling of the systems taking part in integration, but it can also break up the integration logic into distinct easily manageable pieces. This approach exploits precisely this feature by connecting an already existing integration system to the bus. The ESB support for web communication and transport protocols, including SOAP/HTTP and SOAP/JMS, enables the Champagne-based integration of enterprise applications with external applications, such as customer order applications or supplier applications. This approach, based on the connection of Champagne to an ESB and thus leveraging ESB connectivity and mediation capabilities, is depicted in Figure 4.2.

A third option was considered, in which Champagne is split into two services, instead of just one Champagne service. This approach exposes the dependency manager and the propagation manager, which are the fundamental components of Champagne, as two different services. This option

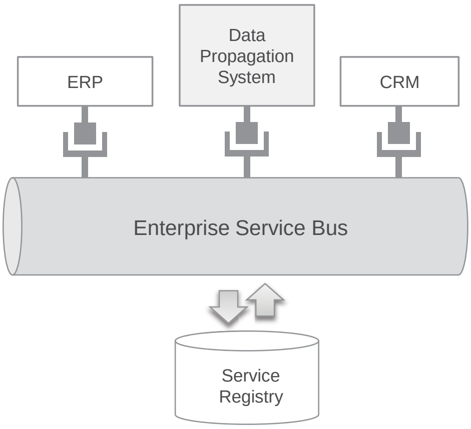


Figure 4.2: Integration of a Data Propagation System in a SOA

was considered due to the importance of choosing the appropriate degree of granularity when making web services available. Splitting a system into multiple services has a major advantage compared to other approaches: the more services, the higher is the degree of flexibility. Finer-grained services enable a more flexible composition of services, which is an important aspect for the aggregation of services in a workflow. However, it also needs to be considered that breaking down a system into a high number of services requires a bigger effort, which involves higher costs.

4.3 SOA Integration Principles for PLM

PLM is another manufacturing domain that has been investigated. PLM has been the focus of integration efforts for over a decade. Here, the problem domain is similar to the scenario of the Real-time Factory. The heterogeneity of information systems, the excessive number of point-to-point interfaces in many companies and the lack of a functional orientation regarding the application integration within departments are factors that avoid manufacturing enterprises from adapting to shorter product life cycles and turbulent scenarios [Bro03]. The adaptability and agility requirements are very present in PLM. Due to a growing number of product variants, shorter product lifecycles and turbulent markets, the importance of an efficient management of the products within their lifecycle is growing continuously [JWW09]. To achieve this, the following aspects of integration need to be taken into account: (1) Information flows need to be more efficient to speed up all processes within the product lifecycle, (2) all phases of the product lifecycle have to be integrated to enable a faster and automated data exchange, improved data quality and data availability over the whole lifecycle and (3) feedback loops have to be enabled in order to improve product quality and production processes [SMM11]. In this case study, an analysis of the requirements for this domain and a matching process of these requirements and SOA integration principles were carried out [SMS+10]. The presented concept is based on a service-based PLM integration infrastructure, called PLM-Bus [SMM11], which provides a mediation platform for several domain-specific ESBs. This integration scheme follows an approach based on incremental adoption, where an ESB is dedicated to a specific project or departmental domain and later integrated into a larger integration network. Incrementally staged deployments of ESB integration projects can provide immediate value while working toward the broader corporate initiatives [Cha04]. The scheme of such an approach is shown in Figure 4.3.

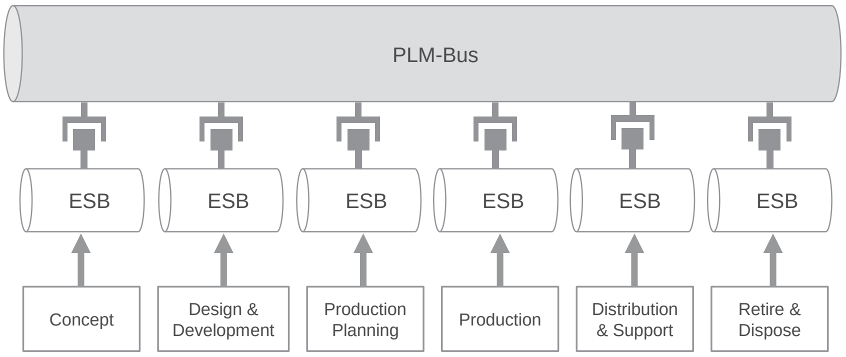


Figure 4.3: Integration Architecture for the Real-time Factory

4.4 SOA Principles applied to Reconfigurable Machines

One of the most important positive effects of flexible production systems is the rapid reconfiguration capabilities that they offer. In this case study an approach to automate the start-up of reconfigurable production machines based on SOA principles was investigated. The introduction of service-based platforms at such a low level in factories has a lot of potential regarding the ease of future integration with higher level SOA-based middleware. The integration of two service-based environments entails a considerable reduction of effort due to the loose coupling, reusability and flexibility principles on which they are designed. The elimination of point-to-point interfaces and the adoption of communication schemes based on a message broker, also contributes to the reduction of complexity. In this study, an architecture was proposed to enable features of an automatic start-up of production machines [ASM+11]. The key concept for designing reconfigurable machines is modularization. Nowadays, modules within machines contain different functionality from different domains, such as mechanical and electrical engineering communication technology. They are also called Mechatronic Modules (MM). MM are characterized by their internal functionality and the interfaces they provide on their boundaries. This concept enables the creation of new machine structures by combinations of MM. However, there is no standard process that automates the configuration of MM. The variety and heterogeneity of the functionality that MM contain makes configuration very tedious and it takes a long time to reconfigure such machines. The proposed architecture in this study aims to ease the communication with MM by integrating these into a SOA. The protocol stack of MM contains the Devices Profile for Web Services (DPWS) [Jam05], which enables the messaging, discovery and description of MM based on web service technologies. A configuration system communicates with the MM by means of an integration middleware. Once the configuration system has retrieved all descriptions of the machine, different configuration services evaluate this information and compute the necessary parameter configuration. An automatically derived start-up sequence is derived. This start-up sequence is defined in WS-BPEL and executed in an orchestration engine. During execution the sequence configures the mechatronic functionality in different MM to be ready for production by means of service invocations.

4.5 Event-driven BPM in ETO Enterprises

One of the most important requirements in the Real-time Factory is the propagation of business-relevant events to the appropriate destination applications. In order to investigate the alternatives to integrate events into business processes within a SOA, a study on Engineering-To-Order (ETO) manufacturers was carried out [MRZ11]. In ETO enterprises, the product is designed especially for one customer. In ETO supply chains, the Original Equipment Manufacturer (OEM) is usually integrated in a complex logistics network, where suppliers are required to comply with strict delivery dates. ETO enterprises have a great interest in the integration of business workflows with manufacturing processes in order to gain the needed responsiveness and flexibility when reacting to manufacturing events. This can be achieved by filling the gap between EDA and SOA paradigms, which requires the integration of complex events into business processes. In this study, the possibilities to handle incoming events in BPEL was reviewed and compared to other approaches. The motivation scenario of this shows the impact of certain events on the entire supply chain of an ETO enterprise and reflects the importance of the integration of such events into specific B2B business processes. The scenario deals with the propagation of certain information related to the machine failure in one of the suppliers production plants. This failure causes the stop of production for an unknown period of time. The consequence of the stop of this supplier part production is the delay in the delivery of this part until the production is restarted. Punctual delivery is considered one of the primary goals in ETO production. This is due to the high degree of product customization that takes place across the supply chain. The OEM needs accurate information about delivery dates for the elaboration of the appropriate production plan. Missing supplier parts at the OEM production line provoke unacceptable delivery delays at the customer side. Even customized products that don’t contain the missing supplier part may suffer delays as well due to the lack of timely information at the OEM to rearrange the production plan. Therefore, the automation of event propagation in the ETO supply chain is very important for the OEM and its customers, as well as for suppliers of higher levels that depend on other suppliers. The main goal in such a scenario is to limit the impact to the product group that needs this affected supplier part by timely rearranging the production plan at the OEM shop-floor. This avoids unnecessary delivery delays in products that don’t contain the supplier part. Figure 4.4 shows two interacting business processes that reflect some parts of this

ETO supply chain motivation scenario. A failure management process of the supplier shop floor system receives a message indicating a machine failure and afterwards invokes a service that logs this failure and stops all manufacturing activities. A maintenance operator analyses the failure and evaluates its impact. Based on this evaluation, the shop floor process propagates the estimated delay of all affected supplier parts to the process of the OEM system. Upon receipt of this delay information, the OEM process calls a service that adapts the production plan and uses this plan to compute an overall production delay and to notify the customer about this delay. In parallel to this, the shop floor process calls a service to maintain and restart the production, informs the OEM process about the restart, as well as the shop floor system about the finished event processing. After the production restart notification, the OEM process rearranges the previously adapted production plan and, based on the new plan, again notifies the customer about the new delivery data.

BPEL offers various possibilities to handle events, as we have shown above. Among the main benefits of BPEL are its modular design, the sophisticated fault and compensation handling capabilities, and its flexibility regarding generic XML data types as well as late binding of services. These and in particular its event handling capabilities make BPEL a valuable asset for the combination of event-driven manufacturing environments and service-based business processes:

1. Events from heterogeneous sources. In order to reduce the number of event handling activities to be processed within a BPEL process instance, it is possible to infer complex events from multiple sources via an event processing service. A complex event may contain data from different sources in a manufacturing environment. BPEL supports the re-partition of such complex events within an event handling activity. For the three options for processing events based on messages (receive, pick and event handlers), the event

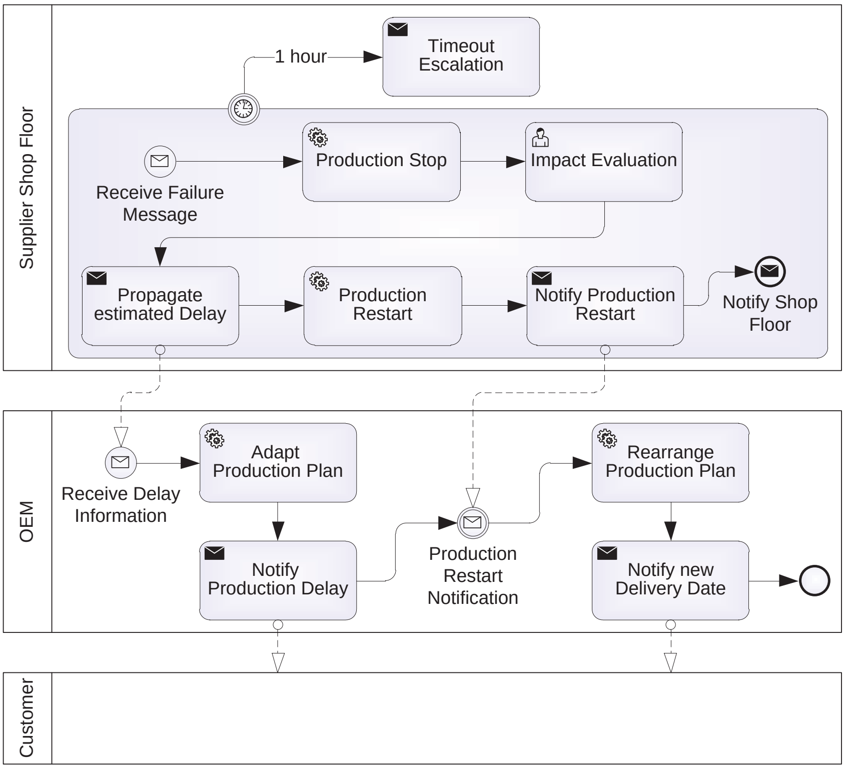


Figure 4.4: Integration Architecture for the Real-time Factory in BPMN Notation

data, i.e., the event message, may be stored in one BPEL variable or in multiple ones, each one storing a certain part of the event data. This avoids unnecessary event message traffic and processing.

2. Complex event processing with real-time requirements. There are cases in which BPEL can be used to optimize the data sent in an event message and therefore help systems to fulfill their real-time requirements. For example, there may be cases in which a piece of

equipment always sends the same event message repeatedly. If the event data is not relevant to the event consumer, BPEL offers the possibility to react to the event based only on the event provider. In this case and when no message parts are contained in the WSDL definition of the event message, the event data does not need to be stored in a BPEL variable. The incoming message is routed to the appropriate event handling activity, depending only on the specified partner information, e.g., partner link, WSDL port type and operation. This can contribute to improve performance and to enable specific manufacturing systems to achieve real-time requirements. Moreover, BPEL offers non-blocking event handling, that is, the possibility to perform other activities while waiting for the notification of a specific event is of special interest in manufacturing environments. In order to fulfill real-time requirements in a high performance manufacturing environment, multiple events need to be processed in parallel. BPEL event handlers onAlarm and onEvent meet exactly this requirement and are the most useful mechanisms to react to an event without blocking the execution of other procedures.

3. Extensibility. Manufacturing companies need to react to relevant events that require action as fast as possible. In order to identify relevant events, the expressiveness of the model used to describe events plays a very important role. The extensibility of a model for event messages and its semantics are key factors of the processing performance. BPEL provides several possibilities to extend the standard. In particular, extension activities may be added to define custom functionality that can be used within a BPEL process, e.g., BPEL4People. Further extensions might be new attributes or new XML elements in existing BPEL constructs as well as extension assign operations, i.e., customized operations for manipulating process data. A customized extension activity can be specified to ad-

dress the special needs of receiving and processing events within a manufacturing environment, in particular within a certain manufacturing scenario. In the case of maintenance operations, human interaction is in particular needed when deciding on the required actions given a certain type of failure. A machine may be repaired manually, the maintenance service has to be contacted or maybe a spare part has to be ordered. These decisions are usually carried out by the shop floor maintenance staff. An extension for human interaction and possibly other extensions for maintenance operations can simplify the BPEL processes considerably. This can contribute to the efficient failure management on the ETO supplier side, which will enable the supplier to react in an agile manner, as required in ETO supply chains.

4. Flexibility through dynamic service binding: Binding of services at runtime is a powerful feature. This feature is of great importance in manufacturing, since there are processes that need to adapt on runtime, depending on the manufacturing context. Several BPEL implementations support dynamic binding via different alternatives, e.g., WS-Policy. However, not all BPEL-based workflow engines offer the possibility to bind services dynamically. In these cases, a BPEL process invokes an external service that contains the routing logic.

4.6 Product-Service Systems

A further area of study for the applicability of service-oriented architectures is PSS, also known as Industrial Product Service Systems (IPS2). PSS are a strategic approach that offers manufacturing companies the possibility of long-term differentiation against competitors by integrating goods and services. The implementation of a PSS entails challenges for the resulting supply chain structure and the IT infrastructure supporting coordinated service offerings, such as conflicting goals and coordination in the integrated business processes. As investigated in this study, the SOA paradigm provides certain design principles of integration that match the needs of PSS networks. The most important challenges of PSS networks include the following:

- Coordination of cross-organizational business processes. Coordination implies transparency of operations and business processes and it is a pre-requisite to assure a unified image of the product-service system to the customer. Splitting services into front-office activities, where the customer is involved, and back-office activities, without customer presence, enables a PSS network to coordinate its B2B processes transparent to the customer.

- Rapid reconfiguration of the network by binding and integration appropriate replacements. The integration of new participants requires a shared strategy for goal definition and matching participant discovery that allows the PSS network to automate the process of reconfiguring itself.

- Monitoring Key Performance Indicators (KPI). A KPI at the business level may be mapped to different nodes of the network at the operational level. Therefore, monitoring these KPIs, analyzing the impact of deviation of the network goals and correcting these deviations represent decisive challenges for efficient PSS networks.

- Goal definition. The heterogeneity of product and services originate conflicting goals. In order to reach an agreement in the service provisioning phase, all participants in a service offering need to define their goals previously. Goal definition must permit all participants to understand and process the goals of others in order to negotiate in a case of conflicting goals.

- Conflict resolution. Opposite interests or conflicting goals may arise when combining the supply chains of a product manufacturer and service providers. For instance, if a service process structure is defined as a service factory with high standardized service operations, which corresponds to a cost minimization strategy, high flexibility becomes a conflicting goal [BNM10]. In order to meet the rapid configuration requirements in a PSS network, participants must be able to reach an agreement as fast as possible, in order to remain agile in case of conflicting goals. Therefore, a high level of automation in conflict resolutions is a fundamental requirement for PSS networks.

The desired agility in PSS networks compares to the agility that is needed in factories. The Real-time Factory is a manufacturing environment that also requires a coordinated and flexible communication between applications. Even a cross-organizational coordination of business processes may be needed if applications on the supplier or customer side are taken into account. This cross-organizational and agile communication is possible thanks to web service technologies. The proposed reference architecture in this study contains three different areas: network configuration, network performance monitoring and conflict resolution [MBN11]. All these three domains aim to provide a reference for companies seeking for an appropriate IT infrastructure that supports communication within a PSS network. The shift from a manufacturing company to a solution provider through PSS offerings was detailed in a strategic roadmap [BNM10], which was previous to this study.

4.7 Summary

In this chapter, the feasibility of SOA-based integration approaches is investigated based on different research studies in five domains of manufacturing. The investigated domains are: digital factory, PLM, PSS, ETO enterprises and reconfigurable production systems. The integration approach in the digital factory (Champagne) presents some limitations, such as the lack of a flexible integration model. The integration of Champagne into a service-oriented environment to meet the requirements of flexibility is thus investigated. The flexibility requirements for the integration of information systems in PLM have also motivated an integration approach based on SOA principles. Further details about this approach can be seen in the literature [MSM11, SMS+10, SMM11]. With regard to the adaptability objectives of this thesis, the use case investigated for ETO enterprises shows the possibilities of the BPEL standard for event-handling. This is an interesting aspect regarding the integration of events into SOA-based environments. Another investigated domain was the reconfiguration of production machines. For this domain, a service-oriented architecture is proposed to enable the communication with the mechatronic modules in production machines. The introduction of service-based platforms at such a low level in factories has a lot of potential regarding the ease of future integration with higher level SOA-based middleware. Finally, the adoption of a SOA-based approach for the integration of different organizations into production networks has also been investigated [MBN11], which has shown a great potential for manufacturing enterprises becoming a PSS provider.

Chapter 5

The Manufacturing Service Bus

In order to solve the integration problems in the Real-time Factory, an integration approach that applies the integration principles of Chapter 3 needs to be adopted. Loose coupling and flexibility are two important requirements that need to be met in order to achieve adaptability objectives described in Chapter 1. The architecture described in this chapter aims to address this need. The presented approach is divided into three parts. The first part comprises mediation services and a mediation infrastructure that together enable a seamless integration of information systems and applications across the factory in a flexible manner and thus increase the agility of the factory with regard to ICT resources. These mediation services are integrated into a service-based integration platform, called the Manufacturing Service Bus [MRR+10, MLJ+10]. The MSB concept and the architecture are detailed in Sections 5.1 and 5.2, respectively. The second part of the presented approach consists of a service repository and an EAI process editor. These, which are detailed in Section 5.3, are used to manage the services that are connected to the MSB as well as the integration processes that control the information flows across information systems in the factory. Additionally, an EAI Process Model is proposed. This model is used by the EAI Process Editor to plan and design the integration processes that enable the exchange of data in the Real-time Factory and that are executed in the MSB. The third part of the presented approach deals with the need of agile adaptation in the Real-time Factory. In Section 5.4, the adoption of the MSB to achieve an agile adaptation of integration processes and the implementation of an autonomic computing feedback loop are described. Here, a Real-time Factory Adaptation Model is proposed. This model constitutes an adaptability framework for the Real-time Factory that implements the monitor, analyze, plan and execute (MAPE) functions of a self-managing environment and serves as a guideline for the feedback loop established between the execution and the planning environment. A MAPE-based architecture is presented in Section 5.4, which closes the SOA lifecycle and implements a feedback loop from the manufacturing domain (MSB) to the analysis phase (Service Repository) and from the planning phase (EAI Process Editor) back to the execution environment in the manufacturing domain. Finally, a summary is given in Section 5.5.

5.1 Concept

As mentioned in the principles of integration in Chapter 3, production environments are usually interconnected following a pattern of asynchronous communication. Asynchrony is needed to propagate events to the appropriate destination applications whenever these events occur. This requirement has caused the development of event-centric integration architectures at the shop floor level, based an the EDA paradigm. However, due to other developments of the last decade, manufacturing companies have adapted their business processes and business applications to new technologies, e.g. Web Services, in order to gain flexibility and interoperability. The principles of reusability and loose coupling of services have made Service Oriented Architecture (SOA) the most used paradigm for software design at the business level. This has led to an integration gap between the shop floor systems and manufacturing applications at the business level, e.g. ERP systems. The Enterprise Service Bus (ESB) is an integration approach that enables an event-driven SOA by monitoring, processing, enriching, and propagating low-level system events to high-level business systems [Cha04]. Current implementations of ESBs are usually adapted to the specific needs of concrete integration scenarios. This adaptation requires usually a great effort for the reconfiguration of the bus mediation services. The MSB and the adaptive architecture that is proposed in Section 5.4 aim to minimize the adaptation effort that has to be invested in the reconfiguration of ESBs. The MSB is based on the ESB concept and enhanced with manufacturing-specific mediation services in order to fill the gap between EDA-based manufacturing environments and SOA-based business processes. In order to fill this gap, the MSB needs to follow the principles of integration for middleware infrastructures in manufacturing as described in Chapter 3. How the MSB applies these principles is described ahead:

- Ease of Reconfiguration. The ESB integration pattern retains centralized control over configuration while allowing bus infrastructure services, such as message routing, mediation or addressing, to be physically distributed [KAB+04]. This pattern is especially relevant when extending ESB capabilities by deploying new services without affecting the existing infrastructure. The MSB is defined for a manufacturing environment based on this service-oriented integration pattern [MLJ+10].

- Loose Coupling. In order to achieve a consistent and flexible integration between digital factory applications and manufacturing systems, the MSB uses a number of mediation services, which comprise transformation, routing and orchestrated service compositions. These services decouple applications and enable the MSB to act as message broker. Transformation services enable the transformation of data between different manufacturing standards, such as ISA-95 [Int00], B2MML [WBF08b], or STEP [Int07b]. The implementation, configuration and deployment of these services are independent of the current state of the MSB. This is possible thanks to loose coupling characteristics of the highly-distributed SOA environment provided by an ESB [Cha04]. The configuration of mediation services is, in this manner, independent of the business logic of the applications that are connected to the bus. This feature enables applications to remain loosely-coupled.

- Asynchronous communication. One of the most important challenges of the MSB is to automate the execution of integration functions in the Real-time Factory. The Real-time Factory is an event-driven environment, which means that data is shared following an asynchronous communication pattern. The MSB acts as a message broker between applications, which receives event-messages, and routes them to their appropriate destinations. This decouples applications from one another and facilitates the asynchronous communication that dominates manufacturing environments.

- Standards-based Integration. The MSB provides multiprotocol support in order for applications to be able to connect and communicate with the MSB using their respective protocols. This way, vendor lock-in situations in factories can be circumvented.

An integration governed by the MSB in the Real-time Factory is depicted in Figure 5.1. Five different levels of abstraction are differentiated: data source (layer 0), data service layer (layer 1), integration layer (layer 2), integration process layer (layer 3) and the business process layer (layer 4). Each layer is described ahead.

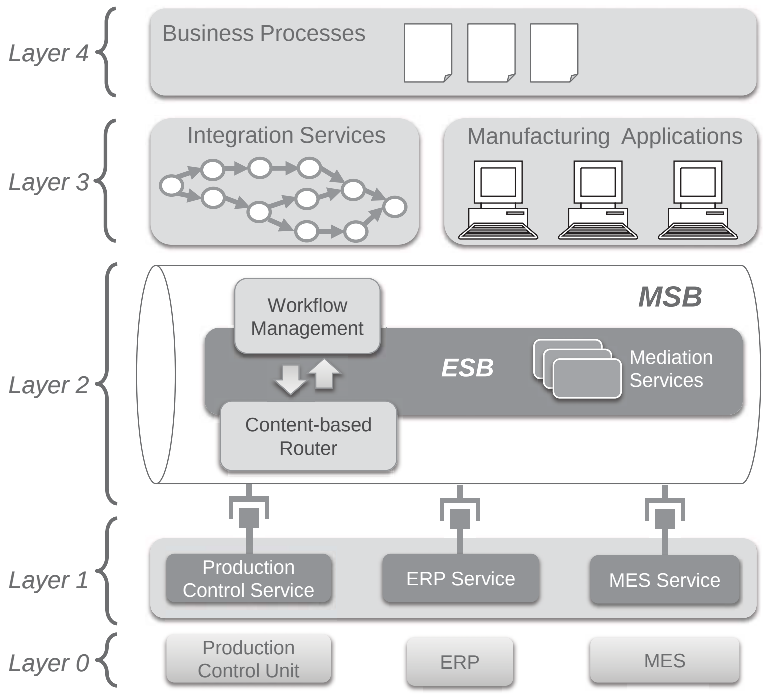


Figure 5.1: The MSB and its Five Layers of Integration

5.1.1 Data Source Layer

All manufacturing systems and digital factory information systems are located in this layer which forms the source of all information flows. These information flows usually perform integration tasks across the Real-time Factory, such as data acquisition, transformations and loading operations. Furthermore, they can notify manufacturing applications of the events that are relevant for the processes at the business level. Such events have their origin also in this layer. This layer comprises a number of heterogeneous data sources, such as unsynchronized databases, diverse data models and different systems that use various communication protocols. Possible systems in this layer are an MES, an ERP system and a Production Control Unit.

5.1.2 Data Service Layer

Due to the heterogeneity of data sources, data provisioning services that resolve these heterogeneities need to be arranged in an upper layer. The data service layer comprises service adapters that enable manufacturing systems and applications to provide data as services which can later be connected to the bus. Enabling a service interface for data providers is the main requirement of service-oriented data integration. In order for applications and information systems to connect to the MSB, they have to follow the principles of integration that are described in Chapter 3, which include:

1. A clear definition of the offered functionality and the corresponding interface.
2. A separation of implementation and interface.
3. The use of standards for the definition of the interface.

Different services can be mapped to a system, depending on the diversity of its functionalities and on the level of granularity that is needed. For example a Manufacturing Execution System may have different services for the different operation areas, e.g. production scheduling, material flow management or quality management.

5.1.3 Integration Layer

The MSB includes a number of mediation services that enhance an ESB as a domain-specific service bus by adapting its infrastructure to manufacturing environments. An example of a mediation service is for instance a transformation service that converts an XML file exported from a MES, which contains scheduling data in the ISA-95 format, into the appropriate file format required by the Production Control Unit, e.g. a file based on Comma-Separated Values (CSV). The MSB facilitates the integration of data provisioning services of layer 1 as well as composite applications and services. Message transformation and mediation services enable the MSB to handle various messaging protocols. A Content-Based Router (CBR) receives, transforms and routes event messages that data services send to the MSB. The CBR communicates with a workflow management system that executes integration services that are needed to process events. Such integration services are BPEL processes that are considered as part of the set of the mediation services that are executed in the MSB.

5.1.4 Integration Service Layer

The information flows that are executed in the Real-time Factory comprise (1) data provisioning services from the data service layer, (2) mediation services from the integration layer, and (3) data destinations that receive the processed data. These three components form a set of atomic services that can be aggregated in composite services with the purpose of integrating different applications. Such composite services are referred to as integration services or integration processes (also EAI processes or mediation workflows). The goal of these processes is to keep manufacturing applications up-to-date regarding the current state of the factory. This layer consists of integration services and manufacturing applications which are usually connected as end-destination of the event notifications that are processed by mediation services. Manufacturing applications are based on human interaction and include digital tools, such as a maintenance console or a customer order portal, to monitor and control the current state of the factory.

5.1.5 Business Process Layer

This layer contains all business processes in a manufacturing environment that are relevant for the execution of production processes, such as a customer order management process or a failure management process. Business processes in this layer comprise different manufacturing domains, such as product quality control or supply chain management. Each business process can be divided into multiple integration services in layer 3. Additionally, these business processes make use of the information that is processed by the manufacturing applications in layer 3. The modeling of these business processes and organization into different integration services is out of the scope of this thesis. Nevertheless, modeling such business processes can be realized by current service-oriented BPM tools. Standards with service orientation features, such as WS-BPEL, facilitate the integration of such tools into this architecture.

5.2 Architecture of the Manufacturing Service Bus

The MSB enhances the functionalities of an ESB by integrating event management services needed in a manufacturing environment. This architectural model integrates the different manufacturing systems that can be found in a production environment. The architecture of the MSB comprises several components that provide a flexible integration of systems. Manufacturing systems and digital factory information systems, such as MES or ERP, are considered as the source of manufacturing information flows. These systems use a service interface that enables manufacturing systems and applications to provide data as services which can be connected to the integration infrastructure and orchestrated in workflows that the MSB can execute. The MSB facilitates the integration of data provisioning services into complex business services. In the MSB, event processing and routing components are used to propagate events to the appropriate event consumers. The MSB uses an XML-based canonical format for event-messages, which facilitates event-processing and routing tasks. Events are stored in the Event Registry. Event-Flows, which determine the relations between separate events, are stored in the Event-Flow Registry. Event attributes contain information about the nature of the event, the current state of the event as well as routing parameters, such as origin and destination. Extended Schemas are used to extend the event model for different manufacturing sub-domains like maintenance, customer relationship or supply chain. The MSB counts on a Content-based Router (CBR), which receives incoming messages and routes them to the correct destinations depending on their content. A Service Registry (SR) and a registry that manages Context Dependencies (CD) support the implementation of this functionality. This approach enables the adoption of CEP techniques in a Service-Oriented Computing (SOC) environment, which is one of the most important requirements in order to fill the gap between the SOA-based business processes and the event-driven manufacturing environments. A Workflow Management System enables the orchestration of different business services. Such business services are executable parts of mediation workflows. Mediation workflows integrate one or more data sources, mediation services, BPEL processes, and end-applications. The architecture of the MSB is depicted in Figure 5.2. The main components of the presented architecture (Event Canonical Model, Event and Event-Flow Registries, Content-based Router, Mediation Services, Workflow Engine) are described ahead.

5.2.1 Event Canonical Model

In shop floors most interchanged messages are based on some kind of event, alarm or notification, which need to be processed under real-time

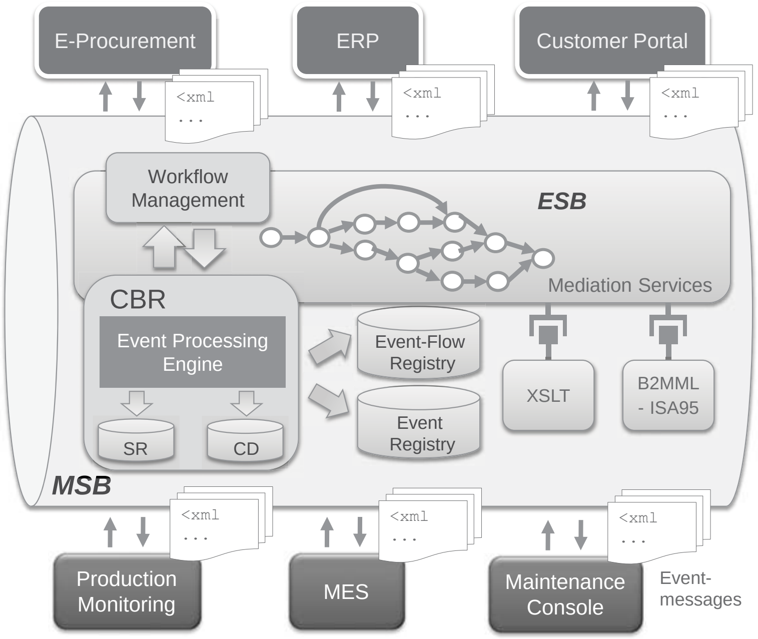


Figure 5.2: The MSB as an Integration Layer

constraints. An event-driven architectural pattern is usually applied to the implementation of systems and applications, that generate, propagate and process events. In a typical EDA, event consumers receive event messages, which are generated by event producers. The concept of the event bus emerges as a solution for brokering event messages between an event producer and multiple event consumers. An event bus acts as a mediation layer, which routes event messages to consumers. This routing process can be implemented by using different methods, such as correlation algorithms, detection of complex patterns and topic subscriptions. A common analysis technique used in event-driven architectures is CEP. The goal of CEP systems is to identify complex events that are inferred from simple events by rule-based event interpreters. However, CEP systems can be a dead-end if no reaction procedures are automatically triggered. Usually, the structure of active rules in an EDA follows the ECA pattern: an event triggers the invocation of the rule, which, if evaluated to true, causes the execution of the action. An expressive representation of events is required in order for a processing engine to evaluate the given conditions and recognize complex situations. In the presented approach, the implementation of the required ECA structure consists of an event model, which is used to describe events, and a routing service, which evaluates certain conditions on incoming events and invokes an external service. From an integration perspective, a manufacturing environment can be seen as a compilation of distributed events generated across multiple heterogeneous applications. These events need to be registered, processed and propagated to the appropriate destinations. This propagation is made possible by introducing an event canonical model (see Figure 5.3), which can reduce complexity over time, as the number of applications increases and as changes are introduced [PH07]. This model is based on XML and has a common schema for events, which defines some basic characteristics, such as registration and routing properties. This common part of

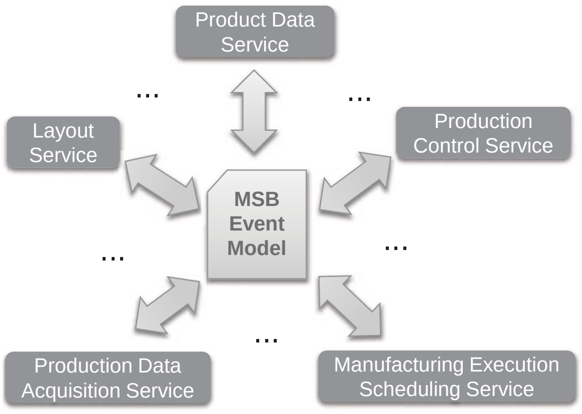

Product Data Service
Layout Service
Production Control Service
MSB Event Model
Production Data Acquisition Service
Manufacturing Execution Scheduling Service
...

Figure 5.3: The MSB Event Canonical Format

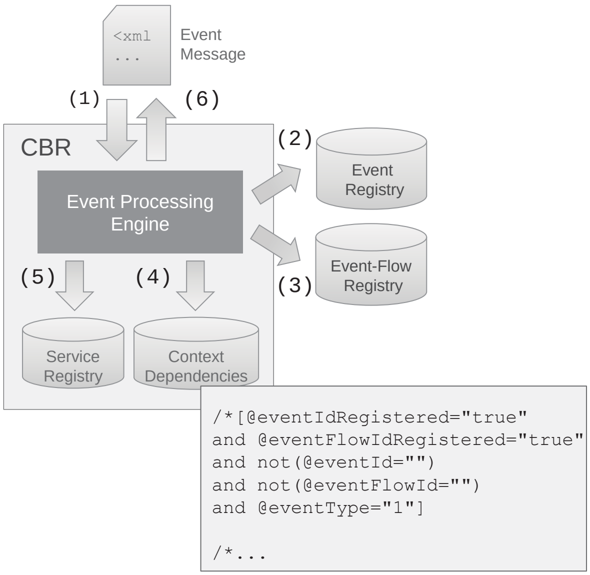
the model also includes an event type attribute, and tags for the event description and timestamp. Depending on the event type, event messages include an additional part for custom data. Custom data vary depending on the event. For instance, the custom part of an event indicating a machine failure contains information about the kind of failure, the location, and the state of the failure as maintenance operations evolve. The XML Schema of this model can be found in Appendix A. More information on this model, including an example of an event message can be found in [MRR+10].

5.2.2 Event and Event-Flow Registries

Event messages are generated by the source applications and come directly into the MSB. Before messages are routed to their correct destinations, these need to be registered. In order to process complex events, it is also important to register the successive actions after a certain event, that is, all events that depend on the first event, which triggers a set of actions. These interdependent events are defined as an event flow. The MSB has two registries for this purpose: Event Registry and Event-Flow Registry. All events are identified by a generated id, which contains the origin system and a timestamp. An event flow is identified by the id of the first event in a flow. The Event-ID is assigned by the Event Registry Service. All Event-IDs, Eventflow-IDs, and their relations are saved in a database. The identification of event flows is a key aspect for the stability and performance of event-driven architectures. The Event-Flow Registry enables the MSB to keep track of event interdependencies, keeping the system stable. Incoming events that result from triggered actions are just stored in the Event Registry and assigned to the corresponding event flow.

5.2.3 Content-based Router

The problem of interconnecting multiple systems by point-to-point interfaces is the required explicit knowledge about each interaction in every data exchange process. This can be avoided by introducing a content-based routing mechanism [HW03]. Incorporating content-based routing into SOAs assists requesters in finding their required services and allows overcoming most limitations in the usual request/reply interaction mechanism by introducing new communication paradigms, such as publish/-subscribe [CN08]. A CBR can be plugged into a service bus architecture and process incoming messages in order to determine the correct destination based on the content of each message. However, a CBR solution can only scale if exchanged messages share the same canonical format. That's the main purpose of the presented MSB event canonical model. Through this event model, the MSB can keep track of events, route messages to the appropriate destinations and perform mediation tasks on messages. The existence of multiple parsing technologies represents an important advantage for the performance of XML-based CBR services. The MSB CBR service is based on a fixed set of XPath [W3C10] expressions, which evaluate certain nodes in incoming event messages as it can be seen in Figure 5.4. Upon the arrival of an event, the CBR routes first all incoming messages to the registration services, namely the Event Registry and the Event-Flow Registry respectively. As messages with assigned eventIds return to the CBR, the routing service looks at its context dependencies and determines where to route the message by evaluating the event data. Context dependencies are represented as XPath expressions and service-event relationships, which are both stored in routing tables. Each event type is mapped to one or more expressions. Once an expression, which matches the event type of the incoming event, is found, the event processing engine looks for the registered services that are subscribed to this specific event type. Services that are subscribed to a specific event are expected to

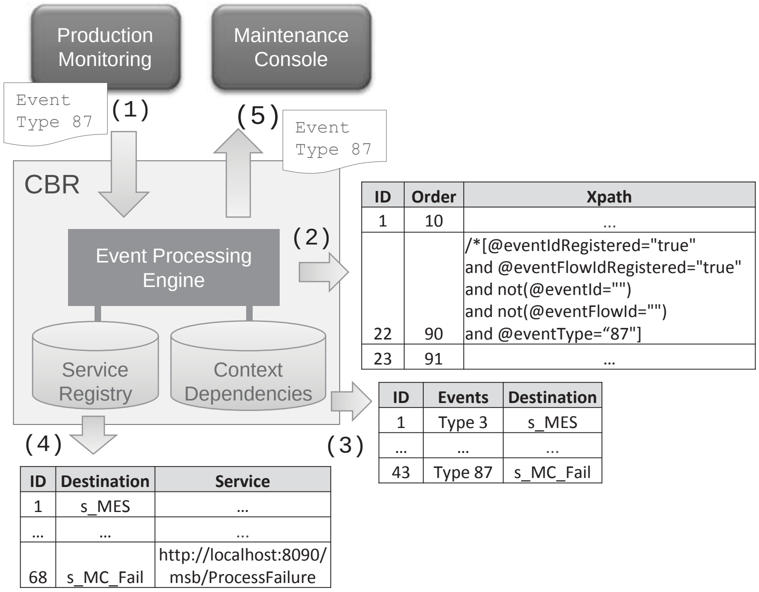


```
/*[@eventIdRegistered="true"
and @eventFlowIdRegistered="true"
and not(@eventId="")
and not(@eventFlowId="")
and @eventType="1"]

/*...
```

Figure 5.4: The MSB Content-based Router

be able to process it. This subscription mechanism lightens the processing requirements of the integration platform and delegates the event processing tasks to the end-destinations. Finally, the Event Processing Engine looks into the Service Library to find out the service endpoints that correspond to the destinations that are subscribed to receive this type of events. As the CBR knows where to route an event message, it adds the appropriate destination endpoints to the routing part of the message and sends the event message to these destinations. The routing logic is exposed as a Web service that is called from the routing process, which is implemented in a BPEL process. The BPEL process takes the event message and sends it to the routing web service, which registers incoming events in the Event Registry and Event-Flow Registry. This routing service looks up the appropriate destination endpoints in a service database and returns the appropriate routing parameters to the routing BPEL process, which then forwards the event to the corresponding destination services. The BPEL process uses for this a parameterized web service call. An example



ID	Order	Xpath
1	10	...
22	90	/*[@eventIdRegistered="true" and @eventFlowIdRegistered="true" and not(@eventId="") and not(@eventFlowId="") and @eventType="87"]
23	91	...

ID	Events	Destination
1	Type 3	s_MES
...	...	...
43	Type 87	s_MC_Fail

ID	Destination	Service
1	s_MES	...
...	...	...
68	s_MC_Fail	http://localhost:8090/ msb/ProcessFailure

Figure 5.5: Failure Event Routing in the MSB

of the routing part of an event message is shown in Figure 5.5. In this example, the source of the message is the Production Monitoring System, which has detected a machine failure. The destination is a maintenance console that will process the event. As the event, marked as event type 87, comes into the Event Processing Engine, XPath expressions are evaluated in the order established in the database. Once an expression is found that matches the attributes of the event, the Engine knows which kind of event type it is, but still doesn't know to which destinations it should be forwarded. In order to find this out, the Event Processing Engine looks into the Manufacturing Context database and discovers that the corresponding destination for this type of event is a system called s-MC-Fail.

In order for the MSB to find out which service endpoint corresponds to this alias, the Event Processing Engine will send a query to the Service Library which returns the corresponding endpoint of the maintenance console (http://localhost:8090/msb/ProcessFailure). Once the service endpoint is retrieved, it is added to the routing destination part of the XML event message and sent to the maintenance console.

5.2.4 Mediation Services

The MSB comprises a number of so-called mediation services. These services are usually transformation services that convert data from one data model to a different one. The heterogeneity of a manufacturing environment and the historical evolution of manufacturing data management have led to many different data models, which include data models from legacy systems as well as currently adopted standardized data models, such as ISA-95 or STEP. Mediation services are needed in order to transform the data that two or more applications exchange. This transformation is needed when the applications use different data models. It can also happen that two applications use the same manufacturing standard, but different data models. This is the case of XML-based data models that were designed to provide the already existing standards with XML support, such as B2MML [WBF08b] for ISA-95 or STEP-XML [Int07a] for STEP. Mediation services may be required to convert data in both directions if needed. Due to the wide adoption of XML for data exchange between applications, transformation services to convert XML data are often needed. Such transformation services are based on the Extensible Stylesheet Language Transformations (XSLT) language [W3C99b]. This W3C standard allows the creation of new documents, based on the content of an original document. XSLT converts data between different XML Schemas [W3C01], but it can also be used to create HTML files or PDF documents. This feature is especially interesting for reporting applications that need information from various systems distributed across the Real-time Factory. The MSB mediation services are not restricted to transformation only. These also include filtering, routing, processing and query services. Additionally, the ESB native support for multiple bindings, such as HTTP or JMS, file transfer and e-mail, allows the connection and usage of mediation services in multiple forms, depending on the requirements of the applications.

5.2.5 Workflow Engine

The MSB workflow engine is part of the ESB infrastructure and allows the execution of BPEL processes. The strengths of BPEL are especially remarkable in three cases: (1) to execute processes with complex logic, to (2) include human tasks, and (3) when state is required [Fas08]. In the first case, BPEL allows to model processes with complex logic and in this case and in terms of integration BPEL is preferred to an ESB due to BPEL's control structures and container activities, such as while loops and scopes, which an ESB does not contain [Fas08]. The second case is when human tasks need to be taken into account. The ESB approach is rather data-centric and it is not possible to bring people into interaction with a mediation flow. In BPEL, thanks to the extension BPEL4People [KKL+05], a number of activities can be used to include tasks performed by humans. This is especially relevant in manufacturing mediation flows, where human interaction is often needed for process control purposes. The third case refers to the execution of stateful processes. Due to the stateless transactional nature of an ESB, a higher performance can be achieved but no state is recorded during the execution of mediation flows. In scenarios with one or more of the aforementioned requirements, BPEL is preferred to an ESB. However, there are cases, in which the strengths from both BPEL and ESB are required. This is the case of a manufacturing environment such as the Real-time Factory. The heterogeneity of applications calls for an ESB adoption due to its support for multiple communication and messaging protocols. Additionally, a number of applications in the Real-time Factory have real-time requirements, which calls for a high-performance ESB that can process messages in the shortest time possible (message in, message out). However, in manufacturing mediation flows, the integration of human tasks is indispensable, such as in change order management or maintenance operations. Moreover, stateful mediation flows may be required as well, e.g. for data analysis purposes. For these reasons, in such an environment as the Real-time Factory, no approach is preferred to the other, but instead, the ESB adopts BPEL processes as well as other mediation services and allows the execution of such processes in a workflow engine. The management of BPEL processes as services can be done thanks to its recursive aggregation of services, which allows for a BPEL process to be used, re-used or aggregated as a service as well. This way, a BPEL process can be seen as a self-contained web service that performs certain tasks. These tasks can be used, as well as other mediation services, in the mediation flows that are managed by the MSB.

5.2.6 Core Implementation of the MSB

The MSB implemented prototype is based on OpenESB, which is a open source ESB based on the Glassfish Server [Ora08]. The implementation of Open ESB uses the Java Business Integration (JBI) standard [Jav05]. JBI is a Java-based standard addressing EAI issues based on the SOA paradigms and principles. JBI defines a plug-in based architecture with three building blocks: the service engines, binding components and the Normalized Message Router (NMR). A service engine provides an environment for the execution of application logic. Examples of service engines are a BPEL engine or an Extensible Stylesheet Language (XSL) transformation. Binding components are connectors to external applications. The most important characteristic of binding components is the support of multiple protocols, such as SOAP, JMS, E-mail, File or FTP. Finally, the NMR finds the appropriate service that is provided by an external component and supports the necessary context for the execution of message exchange sequences. The MSB adapts the functionality of the Open ESB to manufacturing environments by connecting the aforementioned content-based routing and mediation services. The MSB uses SOAP/HTTP bindings for the exchange of event messages with external applications. File bindings are also used in order to import manufacturing data from proprietary applications. The Content-Based Router is implemented as a BPEL routing process, which is executed in the BPEL engine of the Open ESB. The Context Dependencies and the Service Registry tables are managed in a Microsoft SQL Server. The Event Processing Service runs on a Windows Application Server, also known as Internet Information Services (IIS). The Event Processing Service is implemented in the .NET Framework with help of the Windows Communication Foundation (WCF) API. Other WCF Services are the Event Registry and the Event-Flow Registry.

5.3 Service Management

The main objective of transformable factories is to always run on the best economic operating point by the permanent and continuous adaptation of internal and external processes [JWW09]. In order to achieve the adaptability goal in the Real-time Factory, all manufacturing processes, services and resources need to be continuously monitored and analyzed. The integration processes managed by the MSB as well as the data services and applications that are connected to the MSB, are an integral part of the factory-internal processes that need to be permanently adapted. Therefore, the management of these processes and services is a fundamental requirement for the needed adaptation. Process adaptation entails a number of challenges that need to be faced. These challenges include a strategy for the reuse of services, i.e. proper decisions on the granularity of services for each application, and versioning methodologies. In order to address these challenges of adaptation and to ensure the success of the SOA-based approach presented in this thesis, services are managed according to specific service lifecycle guidelines and integration methods, which are described ahead.

5.3.1 Life Cycle Management of Services and EAI Processes

Service adaptability is one of the most important challenges in SOA governance and service lifecycle management [PTD+07]. Services expose different application functionalities that can be reused, thus enabling different composition of services depending on the current business conditions. As these business conditions change, services must be adapted as well. However, when service definitions change, certain precautions need to be taken into account since services may be integrated into and thus affect multiple EAI processes. In order for process modelers to make the right decisions on accepting changes in service interfaces, it needs to acquire sufficient information about the service dependencies of processes and about all processes affected by a service revision. The Service Development Life Cycle (SDLC) is defined for web services as a highly iterative and continuous approach to developing, implementing, deploying, and maintaining web services [Pap08]. In these services, feedback is continuously cycled to and from phases in iterative steps of refinement. The life cycle that is assumed for all services that are connected to the MSB is based on the SDLC, without the restriction to web services. This cycle is depicted in Figure 5.6.

The planning phase is a pre-design phase, which is used to observe and evaluate the business environment and to decide what services need to be planned. In this phase, an analysis of the requirements takes place, followed by the identification of possible reusable services. In the subsequent analysis and design phase, the appropriate service granularity is determined, which is a decisive factor for the reusability of services. Here, performance and QoS aspects may be analyzed as well. A process fragment library managing possibly reusable process fragments [SKL+10] as well as process editing tools may be used to support the design process. After the design phase, services go through the construction and test phase. Here, the service is implemented, its interface is defined and the service is tested. In the deployment phase, it is then published in a repository for discoverability purposes and deployed in the runtime environment. Finally, the service begins to be actually used in the execution phase, where the necessary monitoring mechanisms are also put into

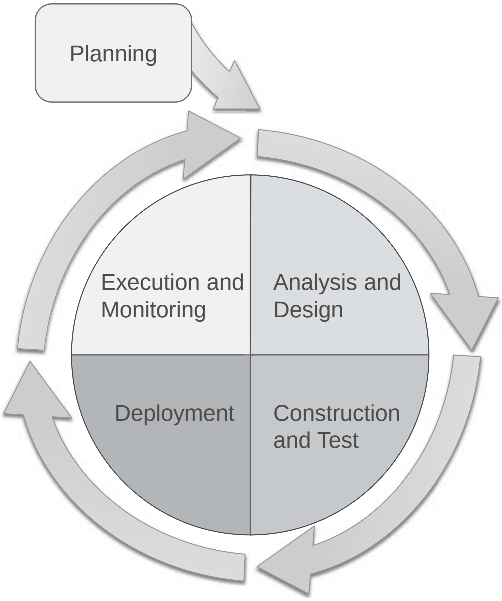


Figure 5.6: Real-time Factory Services Lifecycle, adapted from [Pap08]

place. Monitoring is necessary to analyze the business environment and to check the effectiveness of the service functions and thus to initialize a new iteration of the loop if needed.

In terms of adaptability, two objectives are defined for the Real-time Factory in Chapter 1: responsiveness and knowledge-based adaptation. The components of the proposed approach that address these objectives and the challenges they embrace are presented in this section. These components are the EAI process editor and the Provenance-aware Service Repository.

5.3.2 EAI Process Model for the Real-time Factory

An EAI process editor is used for modeling the EAI processes that are executed in the MSB. These EAI processes are a representation of the mediation flows which integrate two or more manufacturing applications across the Real-time Factory. Such mediation flows are control-oriented flows that process the events that applications produce and consume in the manufacturing environment. These mediation flows leverage the benefits of a service bus, such as the variety of bindings and central configuration, in order to manage a highly distributed environment. A very important benefit of the ESB is performance. An ESB is designed to be able to handle large volumes of messages. Therefore, in a data-centric integration approach, the ESB is currently the best option. However, an ESB does not keep the state of transactions. This may be necessary for the execution of control-oriented processes. Here, there are other standards in the SOA community, such as BPEL [Org07] or WS-RF [Org06d], which present a better alternative. In the case of the Real-time Factory, mediation flows need to leverage the best of both alternatives in order to satisfy the requirements of a large number of applications. The heterogeneity of applications calls for an ESB adoption due to its support for multiple communication and messaging protocols. Some applications in the Real-time Factory have real-time requirements and thus call as well for a high-performance ESB that can process messages in a shorter time than BPEL processes. On the other hand, other applications require to be integrated in control-oriented flows in order to keep the current state of transactions. In such cases, the adoption of BPEL processes is more appropriate.

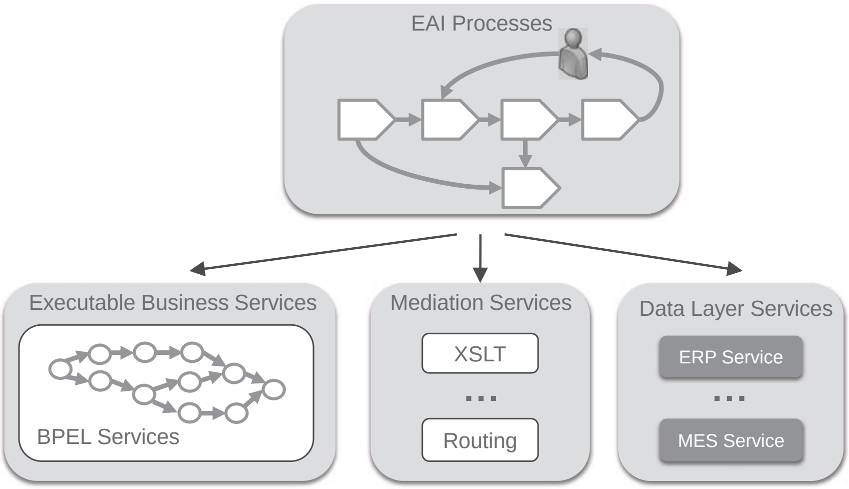


Figure 5.7: Classification of EAI Processes

Therefore, and since there is currently no standard in the manufacturing industry to model mediation flows that can adapt to such different requirements, a process model is needed to integrate both ESB mediation services and control-oriented processes (i.e. BPEL processes). For this purpose, the EAI processes that are executed in the MSB allow the integration of multiple services of different characteristics. From the perspective of an EAI process Such services are referred to as atomic services.

These atomic services are classified in three main groups, which are depicted in Figure 5.7. These services are described ahead:

- Executable Business Services. These are services that implement one or more activities within a business process. These services must be executable in order to be included in an EAI process, i.e. BPEL processes.
- Mediation Services. These services are connected to the MSB and are responsible for all mediation tasks that are executed within an EAI process, such as routing and transformation services.
- Data Layer Services. This group of services include all services from the Data Service Layer of the MSB. Such services act as data consumers or data providers within an EAI process.

In order to design the integration processes that enable the exchange of data in the Real-time Factory and that are executed in the Manufacturing Service Bus, an EAI Process Model is used. This EAI Process Model is based on the service classification described above allowing the integration of control-oriented services, mediation services, data consumer services and data provider services in one process model. This way, the mediations flows of the Real-time Factory can be modeled and make use of diverse services depending on the needs of integration scenarios. This model is defined as the Real-time Factory EAI Process Model and is depicted in Figure 5.8.

An EAI process contains some metadata attributes, which are used for versioning and classification purposes. Furthermore, an EAI process is composed of edges and nodes. A node is the representation of a service, which executes a specific task. Before a node can execute its task, it usually receives one or more events. After having processed the incoming events, it produces one or more output events, each of a specific event type, and sends them to the MSB in order for the event to be routed to

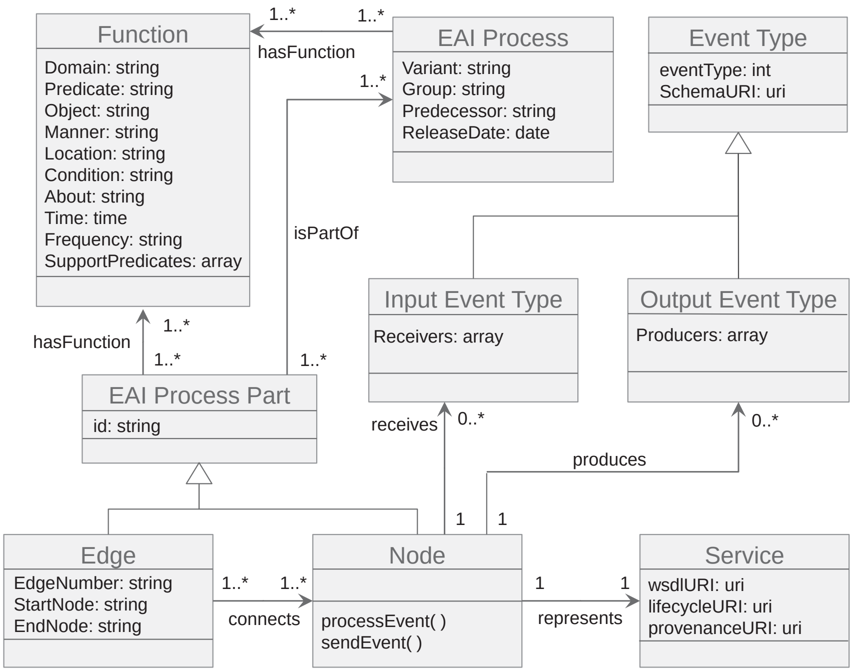


Figure 5.8: Real-time Factory EAI Process Model

the nodes that are subscribed to the specific event type. This mediation operation from a node to the MSB and from the MSB to another node is represented as an edge. An edge has a unique id, as well as nodes, and an EdgeNumber attribute, which differentiates edges that connect the same nodes. Both EAI Processes and EAI Process Parts (edges and nodes) are annotated with one or more Functions. Functions are used to denote which functionality each part of a process and an EAI process in general performs. They comprise a number of attributes, such as domain, predicate and object, which aim to describe a function. These functions are fundamental building blocks of the adaptation of EAI processes, as it is explained in Section 5.4. An example of such Functions is shown in

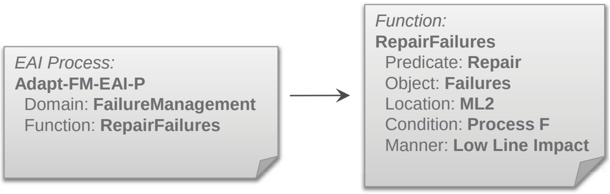


Figure 5.9: Function Description of an EAI Process

Figure 5.9. The Function RepairFailures, which is assigned to an EAI process, is annotated as a function that repairs (predicate) failures (object) at a specific location in the factory. Addtionally, functions contain more attributes, such as condition, manner, time and frequency, which support better adaptation possibilities, as it is described in Section 5.4.

An example of an EAI process is shown in Figure 5.10. The integration process deals with failures in the production system. The communication between the services in the process shown in Figure 5.10 is done by means of the content-based router in the MSB. The MSB is left outside of the process diagram for clarification purposes. More details on the bus communication are given in [WISE10]. The integration process receives a failure message from the shop floor by the SCADA service. The MSB routes the failure message (1) to the Maintenance Console (MC). Here, workers can decide which activity has to be done. Then, a BPEL workflow (BPEL-Repair in Figure 5.10) is triggered with the first evaluation that is made in the Maintenance Console (2). After this failure evaluation, customers are informed about the failure (3) and the estimated delay for their order in a Customer Portal (CP). The BPEL workflow, contains an timeout that allows for an escalation mechanism to be triggered. The escalation consists in notifying the production manager of the long duration of the repair. The workflow calls an e-mail service (POJO Mail) (4), which sends the e-mail to the production manager (5). After the machine

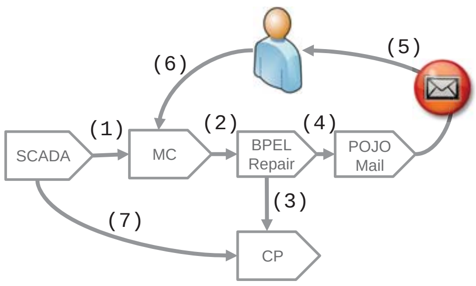


Figure 5.10: EAI Process for Failure Management

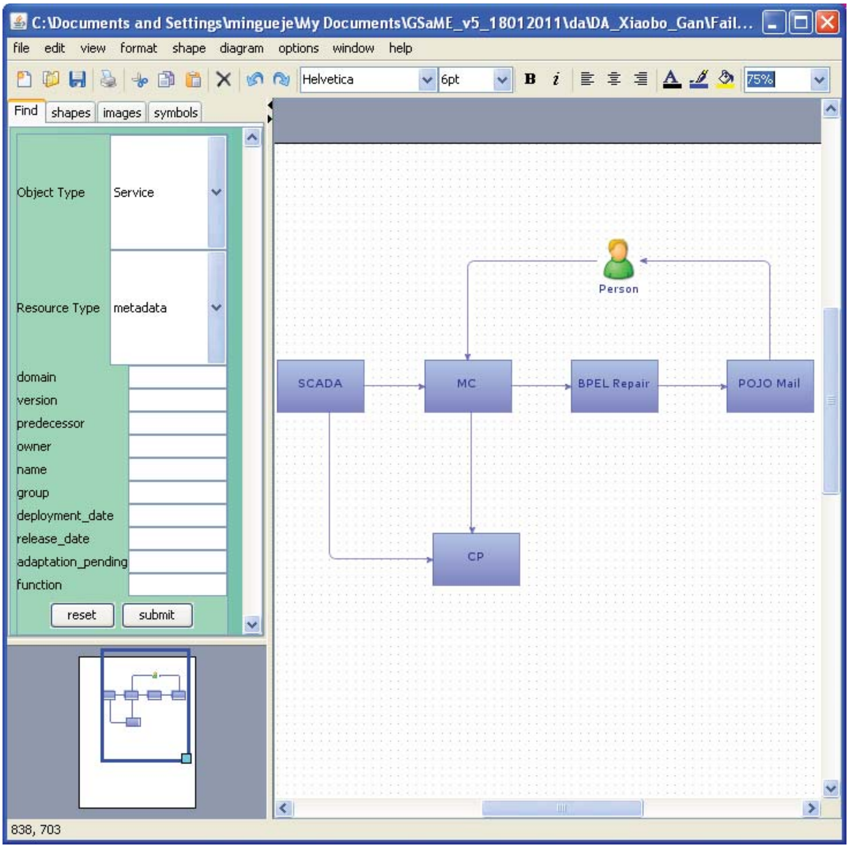
has been repaired, the worker responsible for restarting production has to confirm the repair operation at the maintenance console (6). At that point, production can be restarted and the corresponding event is routed to the customer portal to update the delivery dates of the orders (7). In Section 6.2.2, this process is described in detail, including all messages that are routed by the MSB. The exchange of the event messages with the MSB can be seen in the process described in Figure 6.2.

These EAI processes also have a representation in a written form. The Model shown in Figure 8 serves as a basis to describe EAI processes in the so called MSB Process Description Language (MSB-PDL). This language is based in XML and its purpose is to describe EAI processes in an exchangable format that allows the EAI Process Editor to send and receive descriptions of EAI processes. An example of an MSB-PDL description can be found in Appendix B, where the process shown in Figure 5.10 is described.

5.3.3 EAI Process Editor

To achieve the desired responsiveness, it is crucial to follow the principle of integration that eases reconfiguration of an integration middleware.

Therefore, a mechanism is needed that enables process modelers to easily modify the configuration of the MSB in order to quickly make adapted EAI processes executable. This mechanism is an editor tool for modeling and deploying EAI processes. The EAI Process Editor, which is shown in Figure 5.11, has an UI, which enables process modelers to plan, create, deploy and adapt EAI processes. During the creation of an EAI process, the Find palette on the left side supports the search of services that


C:\Documents and Settings\mingueje\My Documents\GSaME_v5_18012011\da\DA_Xiaobo_Gan\Fail...

Helvetica 6pt 75%

Object Type	Service
Resource Type	metadata
domain	
version	
predecessor	
owner	
name	
group	
deployment_date	
release_date	
adaptation_pending	
function	

838, 703

Figure 5.11: EAI Process Editor

match specific criteria. The fields of the form in this panel correspond to the metadata attributes of the services that are published in the service repository. The communication with the Service Repository is realized by means of an interface that creates concrete queries and sends them to the Service Repository. The corresponding query language is described in Section 5.3.4.2. The query operations supported by the repository are CRUD-operations (Create, Request, Update, Delete) for services as well as for processes. The request operations can refer to a specific resource of a service, such as a WSDL file, a lifecycle description URI or the function assigned to the service. The corresponding resource can be declared by specifying its attributes in the Find palette. After a request operation, the search results are shown on a Service palette. The list of found services, as well as their respective resources, is shown in this palette so that process modelers can drag-and-drop the services onto the main process visualization space on the right side where the editor shows the graph of a process.

As described in Section 5.3.2, a process graph is composed of a collection of nodes and a collection of edges that interconnect these nodes. The EAI Process Editor enables modelers to visualize these graphs and to create MSB-PDL process descriptions. Such descriptions are sent to the MSB in order to be deployed and executed in the platform. The nodes in these graphs are the representations of services that must be connected appropriately in the EAI process. This means that a service acting as node can be connected to another node via an edge if and only if the source node sends the same type of events the destination node is able to process. The editor offers a node compatibility check functionality, which determines the validity of all edge connections in an EAI process. The process visualization space permits to show additional information about the services that are integrated into the EAI process by clicking on them. This information includes metadata information such as version, owner or domain.

Additionally, the EAI Process Editor can receive notifications from the Service Repository. Such notifications include three types of content: (i) process change recommendations, (ii) service in adaptation process, (iii) service finished adaptation. These notifications are important for the adaptation of processes, as it is shown at the end of this Chapter. A process change recommendation is used to recommend corrective actions in EAI processes. The notification "service in adaptation process" notifies the editor of upcoming changes in the interface of a service that is integrated into at least one EAI process. Finally, the third type of notification "service finished adaptation" serves to inform process modelers of new service variants. After a service has been updated, a process modeler has to adapt the EAI processes, which integrate the new service variant.

The EAI Process Editor is implemented via the Java Swing Visualization library (JGraphX) [JGr01]. The Editor UI has been customized to the specific requirements of the EAI Process Editor. This customization includes the Find and Service palettes, a dialog box to enable the annotations of edges and nodes, as well as a Notification palette, which opens when the editor receives notifications from the Service Repository. The EAI Process Editor communicates with the Service Repository by means a specific language: the Service Provenance Query Language (SPQL), which is described in the next Section. The business logic of the EAI Process Editor includes an SPQL Serializer, an SPQL Deserializer to process notifications and an EAI process model serializer, which creates XML-based representations of all EAI processes that are ready to be deployed. The process modeler can start the deployment of an EAI process by pushing the Deployment button in the editor. This button triggers the process model generator, which creates the XML-based representation of the process model and sends it to the MSB.

5.3.4 Provenance-aware Service Repository

A knowledge-based adaptation is one of the objectives of this thesis. The service registry used by the CBR, which is described in Section 5.2.3, does not satisfy the requirements of a knowledge-driven service management system. The purpose of the Provenance-aware Service Repository is to manage the EAI processes that are executed in the MSB. The functions of the repository include the management of service-process dependencies by means of a publish/subscribe mechanism, a notification service, a query processing service and a number of components to manage high-level context information about the EAI processes and the manufacturing domain as well. The management of this context information is based on a semantic service provenance concept, which is described ahead. The components of the repository are described in the subsequent sections.

5.3.4.1 Semantic Service Provenance

The concept of data provenance deals with describing the history and lineage of data, namely the process of tracing and recording the origins of data and its movement and manipulation between databases [BKT07]. Data provenance is especially relevant in scientific applications where data follow a long trail across multiple databases [SSH08]. In the service repository, the concept of provenance is applied to services to describe their versions and variants. The description of service provenance in the repository has to be rich enough to provide high-level context information about services and their effects on the domain they run. For this reason, the repository offers semantic descriptions of the provenance of services in order to provide the necessary knowledge about the deployed services to EAI process modeling tools, as it will be described later in this Chapter. In addition to this, domain data analysis has to be an integral

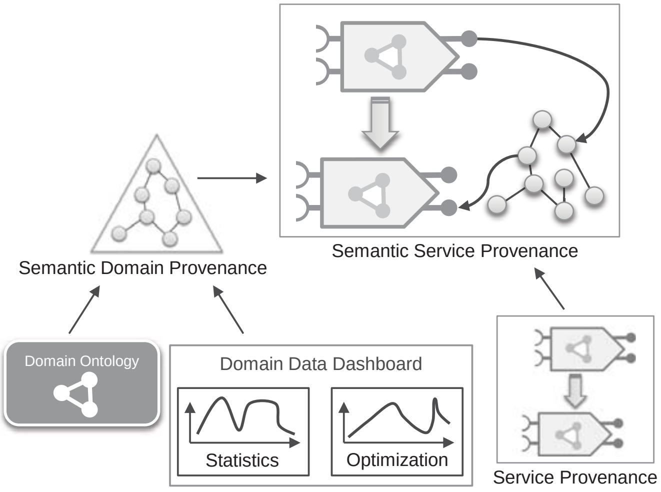


Figure 5.12: Semantic Service Provenance

part of the service lifecycle since services are often redesigned on the basis of optimization parameters derived from analyzing domain data. The concept of semantic provenance is introduced in [SSH08] for e-science domains. This approach advocates the creation of high-quality information using specialized services. This way, provenance information can be automatically interpreted and processed. The main concept is based on two degrees of separation, distinguishing between system provenance and semantic provenance. The combination of domain ontologies with records of data provenance can be used to compute domain-specific comprehensive provenance, which can later be processed to derive higher-level context information. The approach presented in this thesis to describe service provenance is based on this concept of semantic data provenance. Domain-specific ontologies are not only combined with data provenance information. Additional service provenance information is also provided and combined in a further step. This extension of the concept presented by Sheth [SSH08] supports an expressive representation of service interfaces and service provenance, establishing relations to the domain data that services manage. In the presented approach, the combination of service provenance with domain-specific semantic data is defined as semantic service provenance. The resulting provenance information can be used to derive high-level context information about service versions and variants as well as their relations to different domain data. As shown in Figure 5.12, the three basic pillars for building a semantic service provenance framework are (i) the domain data dashboard, (ii) the domain ontology, and (iii) the service provenance data. The domain data dashboard provides standardized metrics to support the analysis of the domain environment. This information is relevant for governance purposes, namely to characterize the effectiveness of running processes and to suggest actions for process optimization, based on domain data analysis. The domain ontology is used to build semantic domain provenance, which can be used to establish relations to service provenance data. Finally, service lifecycle management applications provide service provenance data for the published service variants and versions, such as revision IDs, predecessors, successors, due dates and owners of revisions. A service ontology supports a rich description of service interfaces. This approach to describe service provenance is integrated into a service repository, as it is described in Section 5.3.4.3.

5.3.4.2 Service Provenance Query Language

The Service Provenance Query Language (SPQL) is designed to support the communication of service lifecycle applications with the Provenance-aware Service Repository. SPQL is a markup language and supports eight query types. These query types include four CRUD-operations (Create, Request, Update, Delete) and four types of notifications. Notifications are used to keep track of the lifecycle information of services and processes that are managed in the Service Repository. All SPQL operations are shown in Table 5.1.

A summary of all SPQL operations is given ahead:

- CRUD: These queries are used to Create, Request, Update or Delete services and EAI processes from the Repository. The objects — service or EAI process — in these queries can be referred to by means of different resources, such as metadata, input, output or WSDL references. An example of an SPQL query to create a process is given in Appendix C.

- Notify Object Found: This message is sent back to an application that has previously sent a Request query. The object data that is embedded in the message may contain information about any of the requested resources associated with services and processes.

- Notify Change in Progress: This message is used by applications that enter the re-design phase of a service by planning changes in the service interface. Once changes are planned, they can already be communicated to the Service Repository via this kind of notification. Changes in the interface of a web service affect the following resources: input message, output message and WSDL. The metadata of the service can also be changed.

- Notify Change Finished: Once an application deploys a new version of a service or process, it can send this message to the Service Repository in order to communicate the changes that have been committed. This message may contain any of the resources associated with services and processes.

- Notify Change Recommendation: This message is sent to process management tools, such as the EAI Process Editor. Its purpose is to

Query Type	Object	Resource
Create Request Update Delete	Service, Process	Metadata, Graph, Input, Output, WSDL, Lifecycle, Provenance
Notify Object Found	Service, Process	Metadata, Graph, Input, Output, WSDL, Lifecycle, Provenance
Notify Change in Progress	Service, Process	Metadata, Input, Output, WSDL
Notify Change Finished	Service, Process	Metadata, Graph, Input, Output, WSDL, Lifecycle, Provenance
Notify Change Recommendation	Process	Graph

Table 5.1: Service Provenance Query Language

indicate the changes that need to be carried out in a process graph. This notification is especially interesting from the perspective of adaptive systems. An example of its usage is described later in this Chapter.

The object of a query can either be a service or a process. An object includes different types of resources, which can be referred to in a query. These resource types are described ahead :

- Metadata: This resource refers to the attributes that are used to describe the service itself. These metadata attributes include versioning, classification information such as owner or group, as well as lifecycle timestamps such as release date and deployment date. Furthermore, they include a function attribute, which describes the functionality of the service.
- Graph: A process graph is a collection of nodes and edges, which connect pairs of nodes. An example of a process graph is shown in Section 5.3.2.
- Input: This resource describes the input event messages that a service can process. This description contains a collection of event types that can be empty if a service does not include any operation to process events.
- Output: This resource describes the event types of the output event messages that a service produces. This description can analogously be empty if a service does not include any operation that produces output events.
- WSDL: This resource refers to the WSDL file of a web service.
- Lifecycle: This resource describes the lifecycle information of a service or process. This description includes information about previous versions and other active variants of the service.

- Provenance: This resource describes the provenance information of a service. Provenance information includes the relations of a service or process to its previous versions, to other variants, as well as the service dependency with respect to domain context information.

5.3.4.3 Architecture Overview

The architecture of the Provenance-aware Service Repository is shown in Figure 5.13. The communication between the Service Repository and the EAI Process Editor is based on an interface that can process and generate SPQL queries. This interface can also be used by other applications that update any of the services managed by the repository. These applications send a description of a service revision by means of SPQL notifications: a change-in-progress notification when the changes are planned but not deployed yet and a change-finished notification when the service is redeployed. In addition to the SPQL-interface, the repository can be populated with domain-specific context - domain knowledge - as shown in Figure 5.12. This context information is combined with service provenance information in order to complete the semantic service provenance schema, shown in Section 5.3.4.1. The resulting information is processed by the semantic data engine and stored in a Process Knowledge Base (PKB). The PKB contains the description of service dependencies as well as metadata of EAI processes. The description of atomic services is stored in the Service Knowledge Base (SKB). By atomic services, it is meant all types of services that can be mapped to a node in EAI process graphs, as explained in Section 5.3.2. The PKB and the SKB are based on a process ontology and a service ontology, respectively. The Service Repository architecture consists of the following components: the SPQL-Processor, the Service Discovery component, the Subscription Manager, the Semantic Service Provenance component, the Semantic Data Engine, the PKB and

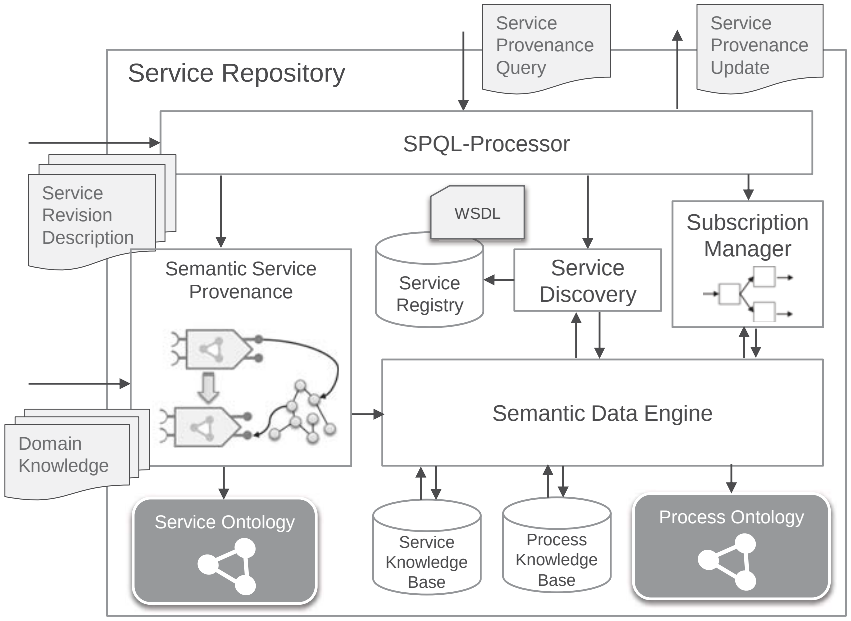


Figure 5.13: Provenance-aware Service Repository Architecture

the SKB. The functionality of these components is explained ahead.

SPQL Processor

The SPQL Processor receives service discovery and process-related CRUD-queries from the EAI Process Editor. Service revision descriptions are sent from external service life cycle applications. When these applications change a service implementation or interface, they send the corresponding update information in provenance update queries to the repository through the SPQL Processor interface. Once a query for create, update and delete operations has been parsed, the SPQL Processor sends the parsed query to the Semantic Service Provenance module, which executes the query operation. When these query operations refer to a process graph, they usually entail a collection of dependencies between the nodes of the graph that needs to be managed. The Subscription Manager is responsible for the management of such dependencies. The information contained in a change-recommendation notification is determined by the Subscription Manager, which informs the SPQL Processor of which processes need to be notified of the recommendation. A change recommendation is made by the Semantic Data Engine. Request queries are sent to the Service Discovery module, which searches for the specific resources of the process or service in the respective knowledge bases. If the relevant resource of the process or service is found, the resource information is sent back to the Service Discovery module. Then, the SPQL Processor receives the resource information and creates an SPQL service-found notification message to send it to the EAI Process Editor. The other types of notification messages are also sent from the SPQL Processor.

Service Discovery

One of the functions that EAI process modeling tools need is service discovery. Integration processes make use of reusable process fragments, which encapsulate repeatable tasks. These can be data transformation services, domain-specific extraction services and application-level services with specific functionality. These reusable services are represented by nodes in a process graph of the EAI Process Editor. All services intended to be reused, need to be discoverable by the EAI Process Editor. In order to allow the discovery of available services, search queries are transformed into ontology model queries that are sent to the service and process knowledge bases. If one or more services are found that match the search criteria, the required resources are sent back to the Service Discovery module. Certain resources are neither stored in the PKB nor in the SKB. Such resources comprise WSDL files and XML Schema files, which describe the interfaces of the managed services. These resources are stored in a Service Registry, which the Service Discovery module can access and look up for the appropriate resource given a specific service ID.

Subscription Manager
One of the most important features of a Provenance-aware Service Repository is to be able to subscribe to service changes. This allows process modeling tools to receive notifications when a service changes if the corresponding subscription exists. The implementation of the subscription manager is done similar to publish/subscribe systems. The subscription manager receives messages from the Semantic Service Provenance component. These messages describe service provenance events, which can affect EAI processes, such as service revisions. Once a service is used in a process, the EAI Process Editor sends a create query to the repository in order to create a process. The SPQL-Processor receives the query and creates a subscription request that is sent to the Subscription Manager. For each edge in a process graph, a subscription entry is created in the subscription store. Every subscription has a Subscriber that issues the subscription. A subscriber can subscribe to different types of messages within the same Subscription object. For instance, a process can subscribe to several provenance events within the same subscription, such as specific service revisions or a change in service operations. This concept associates the subscription object with a process instance, easing the management of subscriptions for the versioning of processes. As new IT systems come into scene, existing integration processes have to be adapted, as well as existing services. When existing services are adapted, there are certain reusability concerns to consider. For instance, services adapted to function properly within new integration processes might also be reused by other processes, which do not require an update. In this case, it is important that the corresponding subscriptions are created when modeling an EAI process. The presented Service Repository allows for this subscription registration functionality to be used by modeling

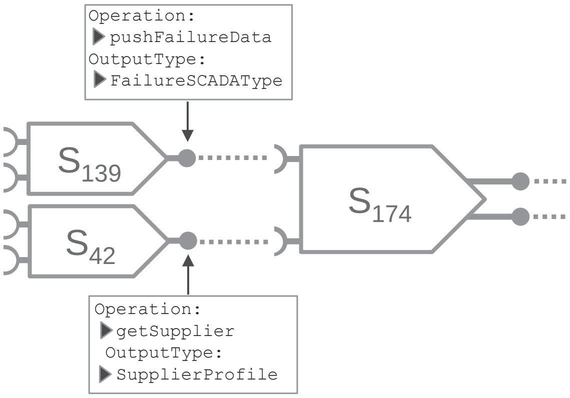


Figure 5.14: Example of Service Dependencies in an EAI Process

tools. For instance, assuming that the EAI process modeler establishes a service integration as shown in Figure 5.14, the corresponding service dependencies have to be translated accordingly into the appropriate subscription. The process subscribes to changes in any of the service operations that deliver input data for other services, like the operations pushFailureData and getSupplier in the example. A process instance may also subscribe to new operations that deliver new data in new service revisions. Finally, all interconnections in the process modeling tool are semantically annotated according to the terms of service and domain ontologies. This gives process modeling tools the possibility to subscribe to any updates or revisions that semantically match the service interactions in the EAI process graph.

Semantic Service Provenance

This module accepts service revision information. Change-in-progress and change-finished queries are sent to the repository from external applications and the SPQL Processor forwards the resulting notifications to the Semantic Service Provenance module. In this module, a semantic description of the revision is combined with domain context information, as given in the scheme of Section 5.3.4.1. Then, the resulting semantic service provenance information is analyzed in the Semantic Data Engine in order to derive new knowledge. Depending on the type of service revision and on the impact of the changes, the derived new knowledge could be the need of adaptation of an EAI process that integrates the updated service. This would imply having to notify the EAI Process Editor of the necessary changes by sending a change-recommendation notification.

Semantic Data Engine

The Semantic Data Engine triggers an internal reasoning process to derive possible new knowledge. The sources to start a reasoning process are notifications from external applications or CRUD-operations on the SKB and PKB. In some cases, service revisions require to notify the EAI Process Editor of a change recommendation so that it may adapt a specific process according to the changes introduced in one of the integrated services. However, in other cases, service revisions do not affect the existing edges of a process graph. An example for such a service revision is the increase of the number of available service operations. This change does not affect the existing service operations which are used to connect a pair of nodes in a specific process graph. In this case, the Semantic Data Engine would reason about the new knowledge and conclude that it is not necessary to adapt the affected processes. The main benefit of this approach is the use of semantics to extract high-level context information about service dependencies and about the impact of service revisions on EAI processes.

Process and Service Knowledge Bases

Both the PKB and the SKB represent storage places of semantic descriptions that are used to manage process and service information. Furthermore, they store the new knowledge generated by the reasoning process of the Semantic Data Engine. In the same manner, if the reasoning process requires updating or deleting specific service or process information, the Semantic Data Engine updates or deletes the corresponding semantic descriptions.

5.3.4.4 Implementation

An overview of the implementation of all components in the service repository is given ahead.

Ontology Models and Knowledge Bases

All components of the repository are implemented in Java. The process and service ontologies are designed with Protégé [Sta], which is an ontology management tool that supports the visualization of all ontology elements. The ontology models can be imported into the repository by means of an OWL file, which contains all information relevant to the process and service ontologies. The management of ontology models, which is realized by the Semantic Data Engine, is implemented using the Jena Framework [HP06]. The PKB and SKB are two ontology models of the Jena Framework that are managed in memory. The knowledge models of the PKB and of the SKB are represented in RDF triples.

Interfaces

As mentioned above, the repository has two interfaces for external applications. The first interface is the SPQL-Processor interface, which is used to communicate with the EAI Process Editor. It is realized as Web Service interface. A WSDL file defines one operation to process queries. This operation answers with a confirmation reply message after having processed the query. Messaging is based on SOAP messages over HTTP. The body part of the SOAP message contains the XML part that comprises the SPQL query. The SPQL Processor has a web service interface in case its integration with other service-oriented platforms is needed. The second interface is the Semantic Service Provenance interface, which is used to receive Service Revision Descriptions from a specific execution domain. In this case, the domain is the Manufacturing Service Bus. This interface is also a web service interface, which contains an operation to process RDF data. Messaging is again based on SOAP messages over HTTP. The body of the SOAP message contains RDF triples encoded in RDF/XML in a CDATA part. This type of message is generated by domain applications that send specific domain data to the repository. The Semantic Service Provenance that processes the RDF triples expects to receive semantic information about a concrete instance of the process ontology, namely a DomainRecommendation instance. The purpose of this information is to adapt EAI processes accordingly. The adaptation process is described later in this Chapter, in Section 5.4.

Reasoning Process

The reasoning process of the Semantic Data Engine is composed of two reasoners. The first one is an OWL reasoner, which is provided by the Jena Framework. It uses rules to propagate the if-and-only-if-implications of the OWL constructs on instance data. Reasoning about classes is done indirectly, i.e., for each declared class a prototypical instance is created and elaborated [HP06]. If the prototype for a class A can be deduced as being a member of class B, the Jena reasoner concludes that A is a subClassOf B. Some of these OWL-implicit rules can be seen in Table 5.2. The purpose of this reasoner is to check the fulfillment of description-logical constructs and the consistency of the ontology model, such as inverse properties, class subsumption and consistency.

The second reasoner is a generic rule reasoner that realizes a rule-based inference over RDF graphs. The set of rules that is managed by the Semantic Data Engine is loaded from a file. Once the reasoner has been instantiated and the rules have been loaded, it can be used as any other reasoner to answer queries against the resulting inference model. A rule for the rule-based reasoner is defined via a Java Rule object with a list of body terms (premises) and a list of head terms (conclusions). Each term is either an RDF triple pattern or a call to a built-in primitive. Some of the most common built-in primitives are shown in Table 5.3.

Property	Condition	Conclusion
rdfs:subClassOf	(?a subClassOf?b) (?b subClassOf ?c)	(?a subClassOf ? c)
owl:inverseOf	(?a property1 ?b)	(?b property2 ?a)
owl:SymmetricProperty	(?a property3 ?b)	(?b property ?a)

Table 5.2: Ontology implicit rules

Builtin Primitive	Description
lessThan(?a,?b)	Test if a < b
greaterThan(?a,?b)	Test if a > b
equal(?x,?y)	Test if x=y. The equality test is semantic equality so that, for example, the xsd:int 1 and the xsd:decimal 1 would test equal.
regex(?t,?p)	Matches the lexical form of a literal (?t) against a regular expression pattern given by another literal (?p).

Table 5.3: Jena built-in functions

Built-in primitives are commonly found in the premises of rules constructs. An example of a rule is given ahead in order to illustrate these the usage of such constructs:

```
rule1:  (?process pkb:hasFunction ?function)
        (?function pkb:hasLocation ?location)
        (?domainRecommendation pkb:hasFunction ?domainFunction)
        (?domainFunction pkb:hasLocation ?domainLocation)
        equal(?domainLocation, ?location)
        -> (?domainRecommendation pkb:actionOnProcess ?process)
```

The premises of this rule check the condition of the *location* attribute of two instances: a the function of a *domainRecommendation* instance and the function of a *process* instance. If their *location* attributes are equal, then the attribute *actionOnProcess* of the *domainRecommendation* instance is pointed to the *process*.

5.3.5 Real-time Factory Integration Architecture

The integration architecture for EAI process, applications and IT services in the Real-time Factory is defined by the components presented in this Chapter: the MSB, the EAI Process Editor and the Provenance-aware Service Repository. Figure 5.15 shows the connection of these components. As described in Section 5.3, the purpose of the EAI Process Editor and the Service Repository is to manage the lifecycle of services and EAI processes that are executed. The information relative to the lifecycle of these components is managed by the Service Repository. In case an adaptation of EAI processes is needed, the update information provided by the Repository is used in EAI Process Editor in order to adapt EAI processes and redeploy them in the MSB. The communication between the EAI Process Editor and the Service Repository is realized by means of an SPQL interface (see Section 5.3.4.2). The SPQL queries supported by the repository allow the execution of CRUD-operations (Create, Request, Update, Delete) on services and EAI processes as well as sending Notifications relative to the changes in services and EAI processes. The deployment of EAI

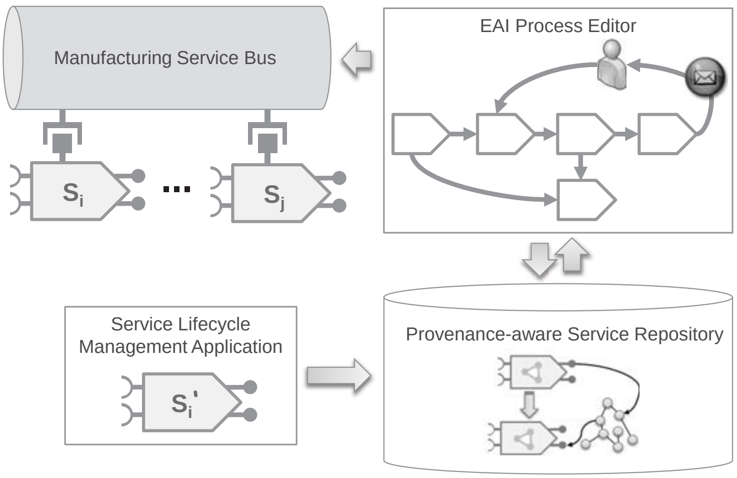

Manufacturing Service Bus
EAI Process Editor
S_i
...
S_j
Service Lifecycle Management Application
S_i'
Provenance-aware Service Repository

Figure 5.15: Real-time Factory Integration Architecture

processes is possible thanks to an MSB interface that is able to process descriptions of EAI processes. Such process descriptions are known written in the XML-based MSB Process Description Language (MSB-PDL). This language is used by the EAI Process Editor to describe the graph of processes. An example of such graphs is depicted in Figure 5.10. The MSB-PDL description of this graph is given in Appendix B. The Real-time Factory Integration Architecture shown in Figure 5.15 also shows a Service Lifecycle Management Application that sends update notifications from the Service Repository, as described in Section 5.3.4.3. The architecture shown in Figure 5.15 provides the desired flexibility and interoperability in the Real-time Factory. However, it lacks a monitoring phase, as shown in the Real-time Factory Service Lifecycle (see Figure 5.6), which would close the service life cycle. The lack of a monitoring system prevents the Real-time Factory to gain the needed agile adaptation for EAI processes. In the next Section, it is described how this loop can be closed.

5.4 Agile Adaptation of EAI Processes

Enterprise agility is defined as the ability of an organization to sense environmental changes and to respond efficiently and effectively to these changes [MP06]. Agile manufacturing has long been recognized as a new expression for an enterprise that states its ability to produce goods and services in the presence of continuous change [ZTZ07]. Both "information architecture adaptability" and "service-based communication" [DGM97] are key aspects to support manufacturing agility. In this thesis, the main line of argument is that a service-based integration platform meets the needs of a manufacturing company that seeks to react flexibly in a changing environment, such as the Real-time Factory. The Real-time Factory is a constantly changing production environment that obliges engineers to continuously adapt applications and services based on the needs of new products and production processes. Similar to the service lifecycle, which is described in Section 5.3.1, EAI processes go through a lifecycle that permits their adaptation to the changing conditions and to new scenarios in the production environment. This adaptation is usually preceded by a monitoring and an analysis phase, in which the execution of EAI processes are evaluated. Production systems register events and data related to the current state of production, whereas the evaluation of these data is carried out manually so far. This means that the evaluation process that monitors and analyzes the current state of the factory usually comprises human tasks, which are executed by factory workers. Such tasks have to manage large amounts of data and therefore take a long time. Moreover, these tasks deal with a high level of complexity and thus require a high level of expertise. This type of monitoring and analysis processes is an obstacle for a quick adaptation of processes. The lack of automated monitoring and analysis mechanisms prevents the Real-time Factory to gain the desired responsiveness in turbulent scenarios. It thus creates a gap in the lifecycle of processes that needs to be bridged. Bridging this gap requires to engineer a self-adaptive system based on a feedback loop. Self-adaptive systems are capable of dealing with a continuously changing environment and newly emerging requirements that may be unknown at design time [BSG+09]. Self-adaptation can furthermore deal with the complexity and uncertainty that govern systems composed of heterogeneous interconnected parts. In such environments, a feedback loop can be established by monitoring and analyzing domain data in order to provide engineers working on the design of software components with the appropriate domain knowledge. The requirement of an agile response when facing turbulent scenarios in manufacturing requires forming this feedback loop by introducing the necessary mechanisms that implement a self-healing, self-configuring and self-optimizing system. In this Section, the mechanisms for an agile adaptation of EAI processes in the Real-time Factory are described. Firstly, the requirements for an agile adaptation of EAI processes in the Real-time Factory are detailed. Secondly, the lifecycle that guides the adaptation of EAI processes in the Real-time Factory is given. Thirdly and finally, the mechanisms that allow the MSB to execute self-adaptive EAI processes are described.

5.4.1 Requirements for an Agile Adaptation in the Real-time Factory

The EAI processes that are deployed and executed in the MSB aim to solve specific integration scenarios in the Real-time Factory. Such EAI processes control the information flows that integrate multiple applications, systems and services across the factory. This integration is needed to achieve an optimal use of resources and to guarantee the communication between systems at all levels of production, including the business level (ERP), plant level (MES) and control level (SCADA) . Production events constantly change the current state of the factory. Turbulent scenarios, such as the introduction of a new product variant or an anomalously high number of machine failures, may move the factory away from its optimal operational point. Under such circumstances, a revision of EAI processes is necessary. EAI processes contain the itinerary information of messages that are exchanged among applications. The identification of specific context situations in these itineraries is crucial in order to be able to act preemptively and to avoid suboptimal production states. An agile adaptation of EAI processes is possible only if the current state of the factory is permanently evaluated, establishing a continuous feedback loop between the execution environment, namely the MSB, and the design tools, namely the EAI process editor. The implementation of a continuous feedback loop enables engineers to react in a responsive manner when facing turbulent scenarios that move the factory away from its optimal operational point. In order for an agile adaptation of EAI processes to take place in the Real-time Factory, some requirements need to be fulfilled. These requirements are described ahead.

5.4.1.1 Flexible Integration

One of the preconditions for agile manufacturing systems is a flexible integration infrastructure. Rigid and tightly coupled integration has been a problem for the realization of agile manufacturing systems since the first implementations of CIM systems . The lack of flexibility brings in difficulties to update when it is required due to evolving enterprise requirements [FSW99]. The lack of interoperability also complicates the incorporation of new systems or technologies. For the context of EAI processes that enable the exchange of data among applications, a flexible integration can be achieved if the following conditions are fulfilled:

- The interaction between two applications is easily reconfigurable.

- Changes in the interface of an application do not affect any other application.
- The incorporation of new applications is possible and does not affect the existing interactions between other applications, and it neither requires changes in the interfaces of these other applications.

These properties, when fulfilled, define a flexible integration. The implementation of these properties leads to an environment of loosely coupled applications that follows the principles of integration described in Chapter 3. Such a flexible integration environment can be achieved if integration is based on open standards, if applications make use of standards-based, well-defined interfaces and if the separation between implementations and interfaces is complete.

5.4.1.2 Capability of Reconfiguration

Reconfiguration is a very important requirement for agile adaptation. A continuously changing environment, such as the Real-time Factory, imposes turbulent scenarios, provoked either by new internal requirements or by external demands, which require the integration processes to be adapted. In order for a manufacturing company to react with agility to turbulent scenarios, this adaptation has to be done in a responsive manner. Such an agile adaptation of integration processes may require the runtime of the integration architecture to be dynamically reconfigured for this purpose. In service-oriented computing, the dynamic reconfiguration of runtime architectures is one of the most important challenges. Dynamic reconfiguration is enabled by automatically leveraging distributed service components and resources to create an optimal architectural configuration according to both user requirements and application characteristics [PTD+07]. In case the reconfiguration of EAI processes and of the integration infrastructure involves human tasks, process modelers should be abstracted from the complexity of parameterizing the integration infrastructure. This way, reconfiguration can be eased for modelers, which accelerates adaptation.

5.4.1.3 Knowledge-driven Adaptation

Knowledge is a fundamental requirement for agility in manufacturing . The concept of knowledge-driven enterprises derives from increasing recognition of knowledge and information as the main differentiators of a successful business [YSG99]. In an enterprise, knowledge can be distributed across multiple resources, including people, reports, but also databases and other repositories [Kid94]. Regarding the agility that is required for the adaptation of EAI processes in the Real-time Factory, domain knowledge needs to be extracted, processed and evaluated. The analysis of the factory context is the most important knowledge source for the adaptation of integration processes. Based on this data analysis, complex events in the factory are identified. The impact of such events on the current state of production must be analyzed as well, in order to react in a timely manner when facing turbulent scenarios that may move the factory away from its optimal operational point.

5.4.1.4 Autonomic Computing Mechanisms

The last, but not less important requirement for an agile adaptation deals with the problem of complexity. In changing environments with a vast ICT-landscape of interconnected applications, the effort to install or interconnect new systems and to maintain integration can be a very tedious task due to the complexity that arises as the number of systems grows. As systems become more interconnected and diverse, architects are less able to anticipate and design interactions among components, leaving such issues to be dealt with at runtime [KC03]. Therefore, a possible solution to this problem is based on self-managing systems, which is also known as autonomic computing. The essence of autonomic computing systems is self-management. The intent of self-mangement is to free system administrators from the details of system operation and maintenance and to provide users with a machine that runs at peak performance 24/7 [KC03]. However, a new construct is still needed nowadays for next-generation communication networks, a pervasive system within the network that builds and maintains high-level models of what the network is supposed to do to provide services and advice to other network elements [DSN+10]. Such a construct involves moving from a data-driven network to a knowledge-driven, self-governing and automatically reconfigurable network.

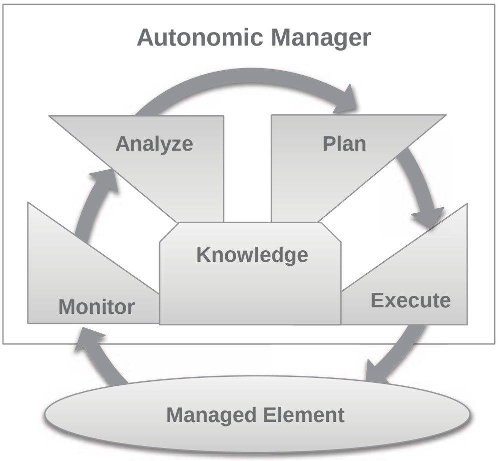


Figure 5.16: MAPE Cycle from [KC03]

In the Real-time Factory, the purpose is to achieve such a level of autonomy to guarantee that self-managing and self-adaptive integration processes can cope with turbulent scenarios by using agile adaptation techniques. The paradigm of Autonomic Computing provides an adaptation mechanism that consists in the realization of the MAPE loop. The MAPE loop (also MAPE cycle) comprises four phases: Monitor, Analyze, Plan and Execute (MAPE) [IBM05a]. An autonomic computing system typically manages such a control loop, which governs the adaptation of managed resources. The autonomic element — introduced by Kephart and Chess [KC03] and popularized with IBM's architectural blueprint for autonomic computing [IBM05a] — is the first architecture for self-adaptive systems that explicitly exposes a feedback control loop [BSG+09]. This feedback control loop is depicted in Figure 5.16. The four phases (Monitor, Analyze, Plan and Execute) represent the control loop that governs the adaptation of the managed resource. The Monitor phase comprises monitoring functions, which collect information about the managed resource. This information includes, amongst others, disruptions in the domain and the status of involved resources. This data is aggregated into complex events and the goal of the monitor phase is to correlate these complex events to symptoms, which can be identified as anomalous situations. The Analyze phase provides the mechanisms to analyze the anomalous situations that have been identified beforehand. The purpose of this phase is to determine if the managed resource has to be changed. For example, the requirement to enact a change may occur when an analyze function determines that some policy is not met. The analyze function is responsible for determining if the autonomic manager can abide by the established policy, now and in the future. If changes are needed, a change request is sent to the plan phase, which creates a procedure to realize the changes. The Plan phase generates an appropriate change plan, which represents a necessary set of changes for the managed resource, and logically passes this change plan to the Execute phase. The latter provides the mechanism to schedule and perform the necessary changes.

As described in Section 3.2, the main objective of the Real-time Factory is to integrate the Real Factory with the Virtual Factory by continuously

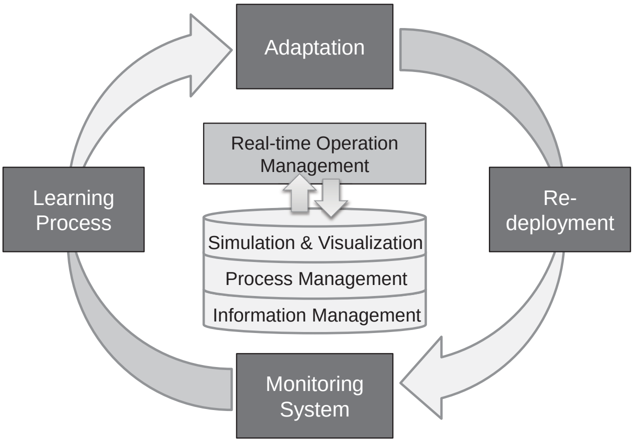


Figure 5.17: Real-time Factory Feedback Loop

communicating, connecting and evaluating the factoryŠs operational data (see Figure 3.1). In order to achieve an agile adaptation in the Real-time Factory, the ultimate goal goes a step further, which is to leverage this continuous evaluation of the current state of the factory for adaptation purposes. Therefore, the integration platform that connects the Real Factory with the Virtual Factory needs to adopt the appropriate learning processes that permit the adaptation of the EAI processes and resources of the factory. Such a learning loop for the Real-time Factory is depicted in Figure 5.17. This loop is based on the MAPE functions and its purpose is to implement a self-managing and self-adaptive system that governs the EAI processes and IT resources — services and applications — of the Real-time Factory. This learning loop comprises a learning process that evaluates the production data that is collected and aggregated by a monitoring system. The results of the evaluation of the learning processes are given to an adaptation system, that is in charge of performing the necessary changes in the managed elements (EAI processes, services and applications) and redeploy them in the production environment. The implementation of such a control loop is a fundamental requirement for the achievement of an agile adaptation approach in the Real-time Factory.

The technique for agile adaptation in the Real-time Factory presented in this Thesis is based precisely on this control loop. The modelling of this loop, based on the MAPE functions, is given by the Real-time Factory Adaptation Model in Section 5.4.2. The implementation of this model is described in Section 5.4.3 and 5.4.4.

5.4.1.5 The MSB as Enabling Platform for Agile Adaptation

The architecture presented in Figure 5.15 fulfills three of the aforementioned requirements for an agile adaptation of EAI processes. Firstly, the architecture provides a flexible integration thanks to the application of SOA principles, which permits to reduce the effort to connect new applications or to change existing interfaces and interactions. Thus, flexibility is achieved by replacing point-to-point interfaces with an integration middleware, which keeps applications loosely coupled and reduces complexity. Secondly, the dynamic reconfiguration of the integration platform is eased for integration experts by a visualization tool, namely the EAI process editor, which abstracts from the complexity of configuring the MSB mediation services. Thirdly, the Provenance-aware Service Repository provides the process editor with the knowledge of the factory and production context that is required to adapt EAI processes.

In terms of adaptation, the goal is to keep a versioning strategy for EAI processes and services that is capable of meeting the aforementioned requirements of adaptation. The management of EAI processes as compositions of different interconnected process fragments which can be adapted to different context situations leads to the existence of multiple process variants. Each variant then represents an adjustment of a particular process that meets specific requirements of a concrete domain context situation. In the presented approach, process variants, as well as service variants, are treated as first-class citizens. Variants of a particular process differ in their process graph description. A service variant differs from another in the interface description. Changes in the implementation of a service variant are not considered, as they do not affect the integration of a service in an EAI process.

The MSB is supported by other components to adapt EAI processes. However, the platform is not self-adaptive as there is no feedback loop so far that permits the automation of adaptation within the process and service lifecycles. The Real-time Factory aims at a high level of autonomy to guarantee self-managing and self-adaptive integration processes that can face turbulent scenarios by using agile adaptation techniques. The MSB is conceived as an integration platform that can provide flexible information provisioning by loosely coupling applications. However, an agile adaptation also requires the introduction of a feedback mechanism that enables the platform to adapt itself. In the case of the MSB, the goal is to provide a feedback loop that allows EAI processes to become self-managing elements. In the next Section, it is described how such a feedback loop can be implemented in the MSB in order to provide the Service Repository with the necessary domain knowledge about the service, process and factory context in an automated manner.

5.4.2 Autonomic Computing in the Real-time Factory

The complexity of current software systems and the uncertainty in their environments have led the software engineering community to look for new ways for the design and management of systems with the capability to adjust their behavior in response to the environment in a self-adaptive way [BSG+09]. This form of self-management has become one of the most promising research directions [KC03, C+08]. An option to implement self-adaptation is Autonomic Computing, which comprises computing systems that can manage themselves given high-level objectives from administrators [KC03]. An autonomic computing environment has the ability to manage itself and to dynamically adapt to changes in accordance with business policies and objectives. Self-managing environments can perform such activities based on situations they observe in the IT environment, rather than requiring IT professionals to initiate necessary tasks [IBM05a]. These environments guarantee self-configuration, resiliency and self-optimization.

The Real-time Factory is a production environment that has to adapt its integration processes in a responsive manner. Responsiveness can be implemented by means of mechanisms for automation and self-configuration and is a fundamental requirement for an agile adaptation. Therefore, the concept of Autonomic Computing is applied to the agile adaptation strategy in the Real-time Factory. Unfortunately, no adaptability models exist that are based on autonomic computing and that can be applied for ICT systems in manufacturing. The known adaptability models for manufacturing systems do not support the adaptation of information flows [LCW08, PM07, WYM11], whereas approaches based on adaptive SOA have not been applied yet to real manufacturing environments in order to support self-managing ICT infrastructures. Therefore, an adaptability model is needed to realize an autonomic computing system for the Real-time Factory that permits self-adaptation, self-configuration and self-management.

5.4.2.1 Definitions

In this Section, some definitions of basic terms are given. These definitions are based on the architectural blueprint for autonomic computing that has been published by IBM in 2005 [IBM05a]. They briefly introduce the terms that are used in the adaptation model for the Real-time Factory, which is described in the rest of this Section.

Autonomic Computing System is a computing system that senses its operating environment, models its behavior in that environment and takes actions to change its environment or its behavior. An autonomic computing system has the properties of self-configuration, self-healing, self-optimization and self-protection. The Real-time Factory behaves as a constantly changing environment and its state is based on sensor data, machine states, material flow, product quality information and other operational data. These data need to be collected, aggregated and processed in an intelligent way [JWW09]. Their integration is managed by factory-context-aware systems that provide the necessary information on the current state of the factory. The factory context information is used as the main knowledge source to execute appropriate adaptation mechanisms that constitute the basis for the Real-time Factory to behave as a self-configuring, self-healing and self-optimizing system.
Self-configuration is a property of autonomic computing systems. Self-configuring components can dynamically adapt to changes in the environment, using pre-defined policies. In the context of the Real-time Factory, such changes include factory-internal planned changes as well as external turbulent scenarios. Self-configuring systems comprise a number of automation mechanisms that reduce the number of manual tasks needed for reconfiguration.
Self-optimization is a property of autonomic computing systems which allows for tuning resources and balancing workloads to maximize the use of information technology resources. In the context of the Real-time Factory, self-optimization focuses on EAI processes in order to adapt them in ways that are aware of factory context situations. The goal of a self-optimizing integration system for the Real-time Factory is to manage such situations and to adapt EAI processes in order to reduce the event

management traffic in the MSB.

Self-healing is a property of an autonomic computing system which permits to discover, diagnose and react to disruptions. Self-healing components can detect system malfunctions and initiate policy-based corrective actions without disrupting the environment. In the Real-time Factory, corrective actions are based on adaptation steps to reconfigure one or more integration processes. The system as a whole becomes more resilient thanks to continuous monitoring and evaluation of the production operational data, which define the policies used to execute corrective actions.

Policy is a set of considerations that are designed to guide the decisions that affect the behavior of a managed resource task. Policies define the goals and objectives to govern the behavior of intelligent control loops. They determine which actions should be taken for the resources being managed.

Managed Resource is a hardware or software component that can be managed. A managed resource could be, for example, a server, storage unit, database, application server, service, application or other entity. Intelligent control loops may be embedded in the runtime environment of a managed resource. These embedded control loops are one way to offer self-managing autonomic capabilities. In the Real-time Factory, EAI processes are entities that are treated as managed resources, as well as all other factory integration resources that govern the plan, execution, monitoring and analysis of these processes, as it is described later in this Section.

Autonomic Manager is a component that manages other software or hardware components using a control loop. The control loop of the autonomic manager includes monitor, analyze, plan and execute (MAPE) functions.

Knowledge Source is an implementation of a registry, dictionary, database or other repository that provides access to knowledge according to the interfaces prescribed by the architecture. In an autonomic system, knowledge consists of particular types of data with architected syntax and semantics, such as symptoms, policies, change requests and change plans. This knowledge can be stored in a knowledge source so that it can be shared among autonomic managers.

5.4.2.2 MAPE Cycle for EAI Processes

As mentioned in Section 5.4.1.4, the adoption of a control loop based on the MAPE functions is a fundamental requirement for an agile adaptation of the EAI processes in the Real-time Factory. As it can be seen in the architecture shown in Figure 5.16, the enhancement needed by the MSB consists in providing a feedback loop that allows EAI processes to become self-managing elements. The conceptualization of a MAPE cycle for this purpose is shown in Figure 5.18. The implementation of this cycle is a control loop that provides the mechanisms to monitor and analyze events of the execution environment and, based on the analysis results, to adapt EAI processes accordingly. This forms one of the major remaining challenges in service-oriented computing [PTD+07, KC03]. The adaptation of EAI processes requires a deep analysis of up-to-date production data, which represents the factory context, and of the history of data, which is known as provenance. Service provenance information as well as production data provenance is analyzed in the presented loop in order to adapt EAI processes. The purpose of each phase is described ahead:

The Planning Phase In the planning phase, the process modeler evaluates the environment to decide what services need to be planned. This comprises an analysis of the requirements, followed by the identification of possibly reusable services. The Service Repository provides the EAI process editor with provenance information about services. This way, modelers can be notified of all changes in the interfaces of registered services. This is possible thanks to the subscription management service in the repository that sends notifications to the editor when services are changed. These notifications describe the process-service dependency as well as the affected processes.

The Execution Phase Once process modelers have created a new process or adapted an existing one, a MSB-PDL description of the process is sent to the MSB. Then, the MSB configuration engine extracts the description of the new process and adapts the internal bus services, e.g., for message routing or transformation, to enable the new control flow of the process.

The Analysis Phase In this phase, context and provenance information is analyzed. Provenance information comprises data about the capabilities of the previous versions of a process and also about the domain. The latter is based on domain knowledge, which must be extracted from production context information.

The Monitoring Phase The monitoring phase comprises monitor functions that collect information from the managed resources and correlate them into symptoms that can be analyzed. The collected data includes information about configuration, status and performance of managed resources. Some of the data is static or changes slowly, whereas other data is dynamic changing continuously through time. A monitor function aggregates, correlates and filters these details until it identifies a symptom that needs to be analyzed [IBM05a]. Once a symptom has been found, it is passed to the analyze function. Production systems register events and data related to the current state of production. Factory monitoring systems usually collect operational data of the factory in order to present it in a visualization tool, such as a factory cockpit [KAW07]. The most part of the evaluation of this data has to be carried out by factory workers in human tasks. Such tasks have to manage large amounts of data and therefore take a long time. Moreover, they deal with a high level of complexity and thus require a high level of expertise. This type of monitoring processes is an obstacle for a quick adaptation of processes and therefore creates a gap in the MAPE cycle, which prevents EAI processes to be adapted in a responsive manner.

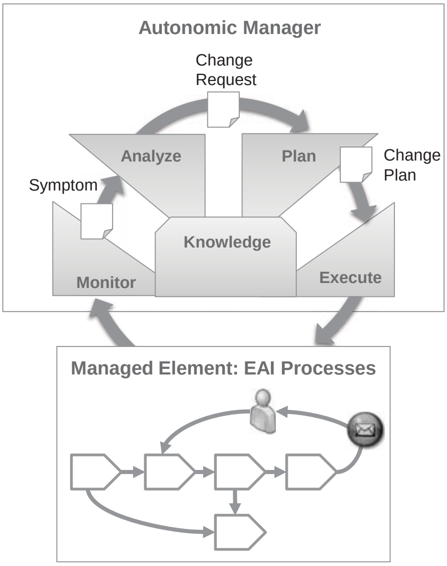


Figure 5.18: Model of a Feedback Loop as a MAPE Cycle

5.4.2.3 The Real-time Factory Adaptation Model

The adaptability of the integration approach for the Real-time Factory is based on an adaptation model, which is inspired by the self-managing concept of autonomic computing systems [KC03]. The adaptation model for the Real-time Factory, which is depicted in Figure 5.19, comprises the components that are needed to implement the MAPE functions that are described in Figure 5.18. These components are related to each other in a way that establishes a learning feedback loop, as described in Section 5.4.1.4 (see Figure 5.17). It serves as basis for the implementation of the adaptive mechanisms for the Manufacturing Service Bus, which are described later in this Chapter. As described above, the existing adaptability models for manufacturing ICT systems are not expressive enough to implement an autonomic computing system for the Real-time Factory.

Some of the concepts in the adaptation model are already defined above, e.g., EAI processes or services. The Production Environment is monitored by a Monitoring System, which identifies Disruptions during the execution of manufacturing processes. Disruptions are aggregated into complex events, which are correlated to symptoms. These symptoms are identified as anomalous situations. Once the monitoring system has evaluated the Symptoms, it elaborates a Diagnosis, which is stored in a knowledge base. Knowledge bases are used as Knowledge Sources by Autonomic Managers. In an autonomic system, knowledge consists of particular types of data with specific syntax and semantics, such as symptoms, policies, change requests and change plans. Policies are among the knowledge data types of a Knowledge Source that can be loaded by an Autonomic Manager in order to consult whether changes need to be made or not. The aim of Policies is to establish the expected behavior of the environment. When changes need to be made, a Corrective Action is applied to specific Managed Resources. In the Real-time Factory, Managed Resources are EAI Processes and Services. Changes of an EAI processes are made by means of the Process Editor, which acts as Planning GUI. Once changes have been planned in the Editor, the Integration Platform (MSB) is reconfigured in order to adapt to the changes made in the corresponding EAI Processes and Services. This adaptation model constitutes an adaptability framework for the Real-time Factory that implements the Monitor, Analysis, Plan and Execute functions of a self-managing environment. These functions form a feedback loop, as in autonomic computing systems, which is explained ahead.

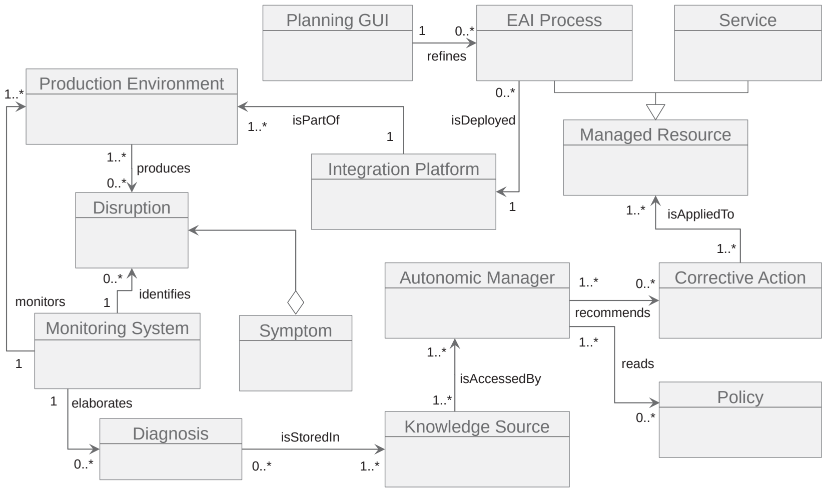


Figure 5.19: The Real-time Factory Adaptation Model

5.4.2.4 SOA Lifecycle Gap in Manufacturing

The gap, which is created by the lack of automatic monitoring mechanisms, needs to be bridged in order to fulfill the autonomic computing requirement, which is a fundamental component for an agile adaptation of EAI processes in the Real-time Factory. Bridging this gap aims at providing the repository with the knowledge that is needed for adaptation. To extract the necessary knowledge from the production environment, production data streams have to be analyzed. However, the MSB cannot process these data streams in real-time as they would have to be aggregated beforehand to integrate them into the control flow of the bus. A possible solution to fill this gap is to add an automatic monitoring phase to the current EAI process execution environment that integrates a real-time monitoring mechanism to timely process production data streams. This way, the analysis phase can be implemented by the inference functions of the Service Repository. Altogether, this establishes a complete

MAPE cycle that enables a manufacturing company to actually improve their agility by means of an adaptive IT infrastructure.

5.4.3 Monitoring and Analysis of Factory Data Streams

The adoption of continuous and automatic monitoring mechanisms has the purpose of extracting knowledge from the production environment. The proposed solution combines the control flow characteristics of the MSB and the data-flow-oriented processing features of a data stream processing platform to integrate high-level context information into the analysis phase, i.e., into the service repository. The latter then uses this input to calculate recommendations for EAI process adaptation and to send these recommendations back to the planning phase. Data stream processing is typically done in two steps: Data elements are collected from data sources and are then processed according to some processing definition consisting of a clearly defined set and ordering of operators. The purpose of the integration of a stream processing platform into the manufacturing environment is to fill the gap between the execution environment of EAI processes, namely the MSB, and the domain knowledge base that provides the high-level context information to the EAI process editor. Closing this feedback loop enables the completion of the MAPE cycle by providing the missing autonomous monitoring functionality.

5.4.3.1 NexusDS

The platform chosen for the pre-processing of the manufacturing data streams during the monitoring phase is NexusDS [CEB+09]. This choice is based on a number of arguments such as its extensible operator base, support for structured and unstructured data, a flexible query graph model and the abstraction of the underlying runtime constraints. These properties are detailed ahead. Depending on the situations or disruptions that need to be diagnosed in the Real-time Factory, specialized operators may be required. NexusDS offers an extensible operator base which allows the seamless integration of custom processing functionality. This NexusDS property enables the creation of custom operators that can be adapted to the needs of specific data streams. It provides a great flexibility in monitoring and adaptatation for changing conditions in the manufacturing environment. Applications in the Real-time Factory manage different types of data, such as data streams from the assembly line, customer orders or sensor data streams, amongst others. This heterogeneity of data enforces NexusDS operators to support many different data types in manufacturing environments. NexusDS operators allow - besides others - to define the accepted input and delivered output data types via embedded operator-specific metadata. This property makes them well-suited for this kind of problem since it improves the flexibility in defining new operators to process different types of factory data streams. A data streaming process is defined by arranging and interconnecting different operators in a processing pipeline. NexusDS distinguishes between three kinds of operators: Source operators, processing operators and sink operators. Source operators represent a connection wrapper for data sources which pushes data into the processing pipeline. Each processing operator constitutes a single step that processes its input data items according to its functional definition and that delivers the resulting output data items to subsequent processing operators or sink operators. Finally, sink operators represent an end of a process definition and provide external applications or processes with the result data. In NexusDS, processing pipelines can be defined in two different ways: Defining a Nexus Plan Graph Model (NPGM) query or defining a Nexus Execution Graph Model (NEGM) query. A NEGM-query is a deployable representation of the respective NPGM-query. This means that a NEGM-query provides complete deployment specifications, such as the concrete physical operator to perform a certain task or the execution node to execute a specific operator. In contrast, NPGM-queries do not provide full deployment specifications. So, these deployment specifications have to be completed by the query processor of NexusDS in order to get a NEGM-query. Some operators may impose certain constraints on their execution environment. For example, some may only be deployed on specific hardware or may require a certain amount of memory at runtime. As another important example of constraints, some operators are only allowed to be executed on a well-defined set of computing nodes to guarantee that sensible factory data does not cross certain administrative domains. In such cases, NexusDS supports the definition of corresponding operator constraints. This can be very useful in manufacturing domains since it permits the definition of processing graphs that are annotated by domain-specific constraints. As a consequence, it provides a natural support for such heterogeneous environments in terms of hardware and software configurations.

5.4.3.2 Integration of NexusDS into the MSB

The integration of NexusDS into the MSB execution environment can be seen in Figure 5.20. The data stream processing platform is connected with the rest of the environment via four interfaces. The main functionality of NexusDS focuses on monitoring production data. The first interface (1) therefore enables the propagation of such production data from the SCADA system to NexusDS processing pipelines. The SCADA system starts this propagation via SQL triggers and forwards the relevant data over a JMS interface in order to ensure asynchronous and reliable communication. The connection to NexusDS is realized as a source operator that copes with this SCADA-proprietary interface. After having monitored and transformed the SCADA data via intermediate processing operators, the processing pipeline send the results to the Service Repository

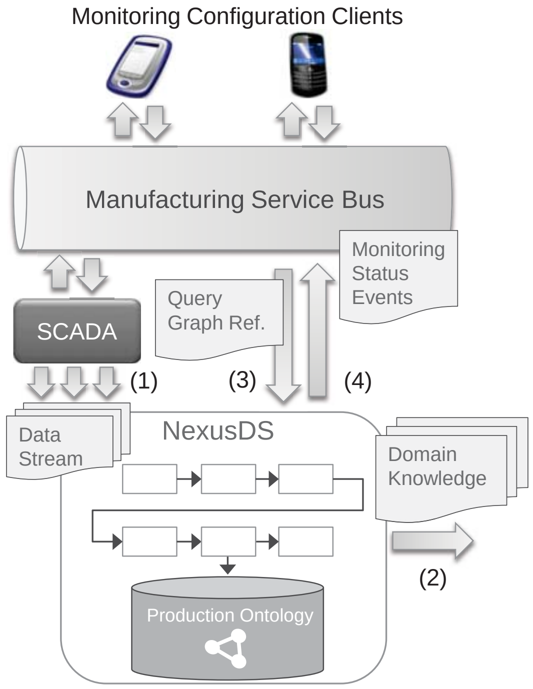
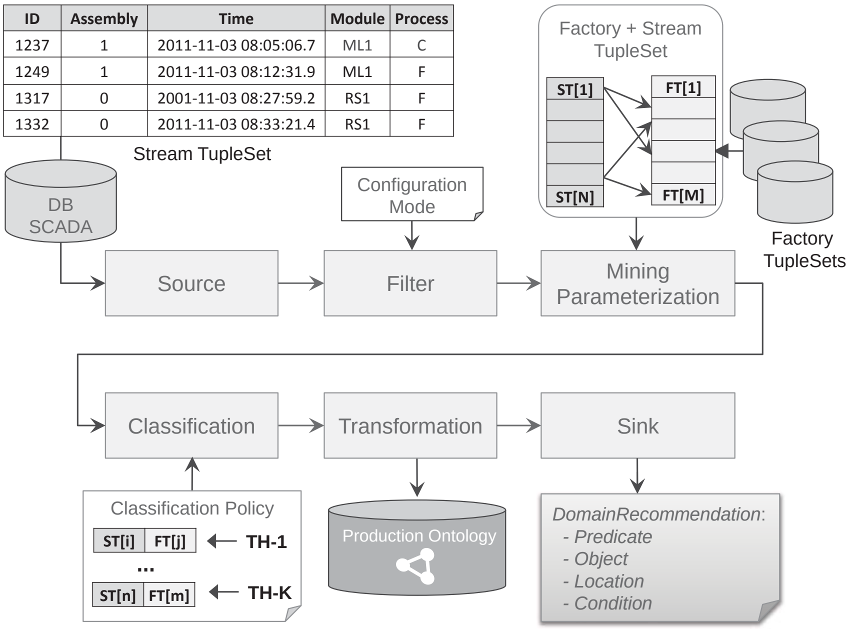


Figure 5.20: Integration of NexusDS with the MSB

via the second interface (2). This interface is implemented as sink operator that encapsulates Web Service operations as needed by the repository. The repository can then use the results of the processing pipeline for analysis purposes. The processing pipeline makes use of a Production Ontology (see Figure 20) in order to convert the results of the analysis into semantic data that can be processed by an inference process in the Service Repository. These semantic data is referred to as Domain Knowledge. The third interface (3) considers the Stream Processing Platform as a Service (SPPaaS) and allows for integrating its functionality into composite applications or processes. These applications or processes issue references to NPGM or NEGM queries that are then installed on NexusDS.

As soon as all needed input data are available to the underlying processing pipelines, the specified data transformations are carried out by the corresponding operators. Applications that access NexusDS in this way particularly encompass different clients for configuring monitoring processes. For example, such a client may be the maintenance console of a worker that is responsible to control the repair of machines in a failure management process. Other workers might use a Personal Digital Assistant (PDA) when they have to continuously change their operating positions. These different monitoring configuration clients may impose heterogeneous interfaces. To cope with this heterogeneity, the message routing and message transformation capabilities of the MSB are leveraged by connecting NexusDS with the bus via a Web Service interface. The fourth interface (4) enables the communication from NexusDS to the MSB. NexusDS may use it to report on the status of its monitoring processes. For example, this status may be indicated to the monitoring configuration clients or other monitoring dashboards. Furthermore, NexusDS processing pipelines may identify certain noticeable and complex constellations in production processes the SCADA system is not able to identify on its own. Such constellations are then reported to other applications, such as the maintenance console. For this interface, the same integration principle as for the third interface is used, namely a Web Service interface for a connection to the MSB. This again enables an easy connection with the heterogeneous interfaces of the applications that receive processing information from NexusDS. The last two interfaces permit the integration of NexusDS into the control flow environment, represented by the MSB, and its adoption as processing platform for monitoring processes. For this integration, the different spheres of control between the MSB and NexusDS may cause a problem, i.e., the control flow in the MSB wants to control the data flow in NexusDS and vice versa. In order to solve this problem, the event-driven capabilities of the bus and its CBR are used to decouple both spheres of control, as for all other data services connected to the MSB. Altogether, the integration of NexusDS completes the MAPE cycle in a manufacturing environment by adding the missing automatic monitoring functionalities. It thus enables the adoption of adaptability measures in real manufacturing scenarios.

5.4.3.3 Mining Graph for Factory Data Streams



ID	Assembly	Time	Module	Process
1237	1	2011-11-03 08:05:06.7	ML1	C
1249	1	2011-11-03 08:12:31.9	ML1	F
1317	0	2001-11-03 08:27:59.2	RS1	F
1332	0	2011-11-03 08:33:21.4	RS1	F

Figure 5.21: NexusDS Mining Graph

As mentioned above, the purpose of the integration of NexusDS into the MSB environment is to gather domain context information and to use this information for EAI processes adaptation if needed. In order to carry out a pre-processing of factory data streams before the subsequent data analysis, a processing graph has to be established in NexusDS. The implemented processing graph for this purpose is shown in Figure 5.21. The graph follows the Nexus Execution Graph Model and comprises six operators, which are described ahead:

- Source Operator: As described above, the SCADA system represents the source of factory data streams for the integration of NexusDS with the MSB. The database of this system has a trigger that sends every update in the log table to a JMS application client, which sends this data to the source operator. The source operator is implemented as a JMS listener application, which receives the production data asynchronously.

- Filter Operator: This operator filters the incoming production data, depending on the data that the processing graph has to monitor. The whole processing graph can be configured to monitor certain production context situations. Every context situation requires specific data to be monitored. The rest can be left out of the analysis. For example, if an analysis on failure distribution is to be carried out, only production data related to failures has to be monitored, whereas the rest can be filtered out. The configuration of the filter operator is based on a configuration mode parameter.

- Mining Parameterization Operator: In order to find correlations between different data tuples, the classification of tuple sets is configured according to specific analysis requirements. These analysis requirements are defined in the configuration parameters and define which dimensions have to be combined in the mining algorithm. These parameters determine which tuples from the stream-based tuples [ST(1) ... ST(N)] have to be merged with tuples from additional databases in the factory. The additional tuples are defined as Factory Tuples [FT(1) ... FT(M)] and are located in factory databases that manage data, which do not change often, such as worker, assembly and maintenance historical data. For instance, for monitoring failures at the shop floor, the analysis of failure management procedures requires merging incoming failure data with data related to workers or maintenance history. The configuration provided by this operator defines which dimensions have to be combined in the mining algorithm. This is done by merging the incoming tuples with other existing tuples that usually contain static data, such as historical data or process data. This operator defines the tuples that will be used in the classification operator. The mining algorithm is detailed later in this Section.

- Classification Operator: this operator has two functions: it classifies the tuples provided by the former configuration operator and then it evaluates the distribution of the tuple values. The result of the evaluation of the values distribution is passed to the next operator. The evaluation of the distribution is configured by the thresholds [TH-1 ... TH-K] that are given in the classification policy.

- Transformation Operator: This operator transforms the information provided by the classification operator into semantic information. An ontology model of the manufacturing domain provides the necessary knowledge to transform the received information into semantic information. The ontology is based on an OWL model. Incoming data is transformed into RDF triples, which are added to the instance ontology model. A reasoning process in this operator enables the extraction of domain-specific knowledge. This knowledge is modeled as DomainRecommendation instance in the ontology and comprises the resulting RDF triples of the reasoning process, which are passed to the sink operator.

- Sink Operator: The DomainRecommendation RDF triples passed from the former operator are embedded in a SOAP message and sent to service endpoint, which is given in the configuration of this operator.

The processing graph of Figure 5.21 represents a data mining algorithm to monitor a production environment. This algorithm analyzes data streams and is shown in Algorithm 1. It is expressed as a procedure that takes two arguments as input and returns a DomainRecommendation (DR) instance. The two arguments are the parameter list used by the Mining Parameterization Operator and the analysis mode used by the filter operator. The first task is to initialize all parameters. Then, a classification model is created with the parameter list and the analysis mode as input parameters. The classification model defines which tuples must be merged in order for the classification operator to classify the incoming data. In the next step, a factoryTupleSet is composed by feeding the defined tuples with the most recent historical factory data. This data is loaded from different factory databases that manage data about workers, objects location, assembly line configuration and past maintenance operations. The factoryTupleSet and the classification model are used to create a classification policy, which defines the thresholds that determine the results of the evaluation of the data distribution. This is the last configuration task, before the mining actually begins. The mining operations are four: filter, merge, classify and transform. These four operations of the algorithm match one-to-one the operators of the NexusDS mining graph, excluding the source and sink operators. The transformation operator uses an ontology model (OModel) to convert the incoming results of the evaluation into RDF statements. These RDF statements are combined in the ontology and analyzed by a rule-based inference process. The ontology model OModel is contained in the Production Ontology that is shown in Figure 5.21. The algorithm finishes by returning a DomainRecommendation (DR) instance, which contains all generated RDF statements. If an analysis does not provide any domain recommendation, it means that all monitored values are within the permitted value domain.

Algorithm 1: Stream Analysis

```
Data: a list of parameters $paramList$, a configuration mode $CMode$
Result: a domain recommendation $DR$
begin
    foreach $p \in paramList$ do
        $initialize(p)$
    end
    $CModel \leftarrow createClassificationModel(paramList, CMode)$
    $OModel \leftarrow getOntologyModel(paramList)$
    $factoryTupleSet \leftarrow getFactoryTupleSet(paramList)$
    $thresholdList \leftarrow createCPolicy(cModel, factoryTupleSet)$
    foreach $logRecord \in logList$ do
        $fLR \leftarrow filter(logRecord, CMode)$
        $tupleSet \leftarrow mergeTuples(fLR, factoryTupleSet)$
        $distribution \leftarrow$
        $classifyTupleSet(tupleSet, thresholdList)$
        $RDFStatementSet \leftarrow createRDF(OModel, distribution)$
        $DR \leftarrow update(DR, RDFStatementSet)$
    end
end
```

5.4.3.4 Domain Knowledge Analysis

The purpose of the presented NexusDS processing graph is to monitor the production environment and provide domain knowledge that can be analyzed. The analysis of the domain knowledge is a task that is implemented by the Service Repository (see Figure 5.13). As mentioned before, the interface of the Service Repository receives domain knowledge by a web service interface that consumes RDF triples. This interface is connected to the sink operator of the NexusDS Mining graph in order to process the DomainRecommendation instances that result of the monitoring process.

The DomainRecommendation OWL class has a collection of properties that describe the recommendation. Some of these properties are the domain, indicating a field of operation, a predicate and object pair, describing what the goal of the recommendation is, or a location, which describes the physical location that the recommendation refers to. Functions are used to define which tasks EAI processes and services perform. For instance, a failure management process may have a function with the following properties: domain="failure management", predicate="manage, repair" and object="failures". In order to deduce the corrective actions that are needed to adapt an EAI Process or a service upon the arrival of a domain recommendation, the rule-based reasoner starts an inference process that checks three conditions. Once a condition is true, the rest is no longer checked. The algorithm of this sequence is shown in the Algorithm 2. A description of this algorithm and these three conditions is given ahead.

The procedure takes the DomainRecommendation (DR) instance, a list of all services (sList) and a list of all processes (pList) that are stored in the SKB and PKB, respectively. A CorrectiveAction (CA) instance is returned.

The first condition checks if there is a process with a function that has the same domain, predicate and object as the domain recommendation. If this is the case, it is checked if there is a node in the graph of the process found that has a function with the same location of the DomainRecommendation. If this condition is true, the adaptation object of the corrective action points to the process found and all properties of the corrective action are set to the values of the domain recommendation. This is possible because CorrectiveAction and DomainRecommendation share the same properties. The rest of the conditions is not checked.

The second condition checks if there is a service with a function that has the same domain, predicate, object and location as the domain recommendation. If a service is found that fulfills this condition, the adaptation object reference is updated to point to this service and all property values of the domain recommendation are copied as values of all properties of the corrective action.

The last condition checks if there is a service that has a function with a supported predicate that contains the predicate of the domain recommendation. The supported predicate is a property of a function that describes a list of predicates that can be implemented by using the predicate of the function. An information terminal which gives instructions to workers is a service that has a function with the predicate "inform". Supported predicates of this function could be "repair" or "assemble" because these are actions that can be supported by an information terminal if the appropriate instructions are given. If there are more than one service that support the predicate of the domain recommendation, a similarity index is calculated for the service functions. The similarity index is based on the number of matching values for the rest of the function properties. The service with the highest similarity index is selected as the adaptation object of the corrective action.

5.4.4 Architecture for EAI Process Adaptation

The adaptation process is realized in four steps following the MAPE-cycle scheme. Firstly, the data streams of the manufacturing domain are preprocessed by the corresponding processing graphs in NexusDS. Secondly, the domain knowledge extracted by the processing graph is sent to the Provenance-aware Service Repository. The repository starts a reasoning process and analyzes the impact of the extracted knowledge on the running EAI processes and services in the factory. If necessary, a change

Algorithm 2: Creation of a Corrective Action

```
Data: a domain recommendation $DR$, a process list $pList$, a service list $sList$
Result: a corrective action $CA$
begin
    if $\exists\, p \in pList : getFunctionDomain(p) = getDomain(DR)$ and
    $getFunctionPredicate(p) = getPredicate(DR)$ and
    $getFunctionObject(p) = getObject(DR)$ then
        $setObject(CA, p)$
        if $\exists\, n \in getNodes(p) : getLocation(n) = getLocation(DR)$
        then
            $setMetadata(CA, DR)$
        end
    else if $\exists\, s \in sList : getFunctionDomain(s) = getDomain(s)$
    and $getFunctionPredicate(s) = getPredicate(DR)$ and
    $getFunctionObject(s) = getObject(DR)$ and
    $getLocation(s) = getLocation(DR)$ then
        $setObject(CA, s)$
        $setMetadata(CA, DR)$
    else
        $i = 0$
        foreach $s \in sList$ do
            if $getPredicate(DR) \subset$
            $getSuppPredicateSet(getFunction(s))$ then
                $j = calcSimilarity(getFunction(s), DR)$
                if $j > i$ then
                    $i = j$
                    $setObject(CA, s)$
                    $setMetadata(CA, DR)$
                end
            end
        end
    end
    return $CA$
end
```

recommendation notification is sent to the EAI Process Editor. Thirdly, the EAI Process Editor adapts the affected processes. Finally, the last step corresponds to the deployment of the adapted processes in the MSB execution environment. The cycle starts over again with the monitoring phase in NexusDS, where execution of the adapted processes is observed. In Figure 5.22, the integration of this adaptation loop into the overall system architecture is depicted. Here, the four phases of the MAPE cycle (Monitor, Analyze, Plan, and Execute) are distinguished. This adaptation scheme follows the adaptation model of the Real-time Factory described in Section 5.4.2.2. In the adaptation model, an autonomic manager accesses a knowledge source in order to read the diagnosis elaborated by a monitoring system. Based on this diagnosis information, the autonomic manager creates corrective actions, if needed, in order to adapt managed resources. In the implemented MAPE cycle for the Real-time Factory, the

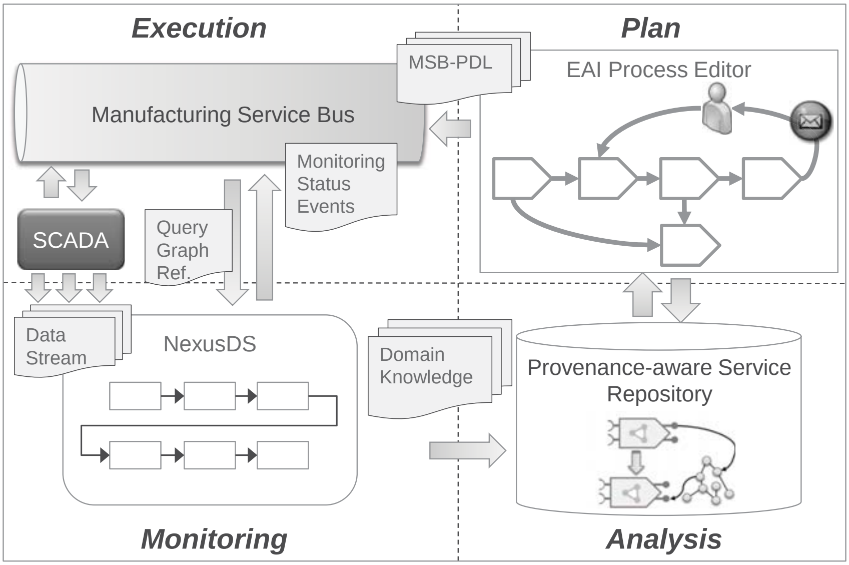


Figure 5.22: MAPE Cycle in the MSB

autonomic manager is the Semantic Data Engine of the Service Repository, which creates the necessary corrective actions based on the information extracted from the DomainRecommendation instances that NexusDS produces as monitoring system.

5.5 Summary

In this Chapter, the Manufacturing Service Bus and the MAPE-based architecture are presented. The MSB concept, which is described in Section 5.1, introduces SOA principles into manufacturing environments, which provides more flexibility and agility in the adaptation of integration processes. The MSB support of different communication and messaging standards enable the interoperability of the platform. Interoperability is provided at the data level and at the application level thanks to the MSB Event Model, which provides a common data model for applications to exchange messages and to connect service interfaces by means of event processing and producing methods. The content-based router and other mediation services decouple service providers and requestors which provides a high level of flexibility, as described in Section 5.2. An EAI Process Model has been proposed in Section 5.3.2 (see Figure 5.8). This model is used by an EAI Process Editor to plan and design the integration processes that enable the exchange of data in the Real-time Factory and that are executed in the Manufacturing Service Bus. Furthermore, monitoring and analysis tasks are automated by the NexusDS mining graph (see Section 5.4.3.3) and the inference process of the Provenance-aware Service Repository (see Section 5.4.3.4), respectively. This closes the SOA lifecycle gap with the EAI Process Editor, which permits to increase the capabilities for agile adaptation of the approach. The SOA lifecycle is implemented following a MAPE-based architecture that implements a feedback loop from the manufacturing domain (MSB) to the analysis phase (Provenance-aware Service Repository) and from the planning phase (EAI Process Editor) back to the execution environment in the manufacturing domain. This architecture is based on the Real-time Factory Adaptation Model (see Section 5.4.2.3), which has also been proposed in this Chapter and can be seen in Figure 5.19. The MAPE-enabled architecture based on the presented Adaptation Model for the Real-time Factory follows the scheme of a feedback loop and enables the execution of self-managing mechanisms. This architecture is depicted in Figure 5.22.

Chapter 6

Applicability and Evaluation

In this chapter, the applicability of the presented approach is demonstrated for different scenarios in a real manufacturing environment. The MSB is adopted to provide a flexible integration of production systems and has been tested under different production situations. The prototype of the MSB has been deployed in the Learning Factory for advanced industrial engineering [RKK+07] at the Institute of Industrial Manufacturing and Management in Stuttgart, Germany. The description of this environment is given in Section 6.1. In Section 6.2, four use cases illustrate the applicability of the presented approach. The first two use cases demonstrate the usability of the MSB in a real production environment. Both use cases show the flexibility, interoperability of the approach thanks to the routing service and other mediation services of the integration architecture, which enables the components to remain loosely-coupled. The last two use cases demonstrate the adaptability and agility features of the implemented MAPE-based feedback loop. In Section 6.3, the impact of the approach in the product life cycle management is given and a model of integration regarding the production planning phase is detailed. The validation and evaluation of the approach are given in Section 6.4. These aim to demonstrate the technical feasibility and applicability and to evaluate the practical value for the groups of interest. The demonstration of the technical feasibility is described in Chapter 5, whereas the applicability of the approach is illustrated in the use cases of Section 6.2. The research results are validated in the assessment of the presented contributions, which is given by an extensive coverage of the objectives that are described in Chapter 1. The evaluation of the approach has been realized by the examination of thirty criteria that have been classified in six categories: interoperability, flexibility, mediation, adaptability, agility and integration. These categories are used to evaluate the MSB approach and to compare it with past and current approaches in the EAI and SOA domain. Two recent research approaches to integration and context management in the manufacturing domain are also evaluated. In Section 6.5, some conclusions are given.

6.1 The Learning Factory: a Field for Evaluation

The Learning Factory comprises a digital planning environment and a real transformable assembly system, also known as iTRAME [RKK+07]. The assembly system is composed of a transport belt conveyor, as well as of different manual stations, where assemblers perform one or more assembly tasks, and of robot stations. This test environment has all capabilities to be able to react to external events like a customer order or internal turbulences like the breakdown of resources. Such events have been identified as relevant turbulences by several studies in industry [WZ09, RKK+07]. The usage of digital tools and the iTRAME enable the execution of production processes from the production planning down to the assembly line of the factory. A test environment description and the specific workflows that react to these turbulences are shown ahead. The Learning Factory for advanced Industrial Engineering has been developed to qualify people in methods and tools in the field of process planning and production optimization. In order to reach the target of a short reaction time in turbulent scenarios, a high planning quality and the validation of planning results before their realization, industrial engineers are supported by a digital planning environment. In each planning phase, digital tools can be used to support the execution of information flows between shop floor and digital factory. Other important aspects to react to turbulences are the transformability of the production system to realize the planning results in short time and the close contact between the digital planning and the real production system. This close contact is important because the planning needs actual information from the production to make decisions on a wide base of heterogeneous information. The transformable assembly system of the learning factory collects and delivers this information for the production planning environment through the Production Control Unit in the shop floor. In order to evaluate the applicability of the presented MSB as central integration layer in an event-driven digital planning environment, four scenarios have been chosen: (1) management of a customer order, (2) the internal turbulence of the breakdown of resource, (3) service revision and (4) the adaptation of the failure management process due to the introduction of a new product variant. These scenarios, which are detailed in the next Section, are used to demonstrate the functionality of the MSB. The following information systems in the Learning Factory are participants of the information flows shown in the scenarios of the next Section:

- Supervisory Control and Data Acquisition (SCADA): this system is located in the shop floor and is responsible for the production monitoring and control. The assembly structure of the iTRAME is managed from the SCADA terminal. This system receives information related to production orders from the Manufacturing Execution System.

- Manufacturing Execution System (MES): The MES has a link function between the SCADA system and the ERP system. The MES receives information from production in real-time. At the same time, the MES receives customer order information through the ERP system.

- Enterprise Resource Planning (ERP) System: the ERP system sends information related to the production resources, such as assembly stations, tools, product variants and processes, to the MES.

- Maintenance Console: This application shows the failures detected in the shop floor that have pending review or repair operations by maintenance workers.

- Customer Portal: This application informs customers of the current state of their orders. The state of customer orders is updated when the order enters production, when the order is finished or when production must be stopped due to an internal failure in the assembly line.

In all scenarios, some of the triggered actions are coordinated through a BPEL workflow. In the scenario of the breakdown of a machine, the workflow integrates the production monitoring and controlling system, a customer portal and a maintenance portal. In the scenario dealing with changes in the production planning phase, the integration of a production planning application, a simulation tool, an ERP system and the MSB is realized through an ESB approach, corresponding to the production planning phase of the product lifecycle management.

6.2 Case Studies

In this Section, the description of the four case studies that illustrate the functionality of the implemented approach is given. The first use cases describe the scenario of a customer order workflow and a failure management workflow. Both scenarios comprise the data exchange operations between applications across the MSB. The failure management workflow of the second scenario represents the execution flow of the failure management process that is depicted in Chapter 5 as an exemplary EAI process. The third and fourth scenario illustrate the service revision and adaptation of the failure management process, respectively. Such adaptation is possible by integrating the inference results of the data stream analysis procedure and the Provenance-aware Service Repository into the EAI Process Editor, as detailed in Figure 5.22.

6.2.1 Customer Order

The customer order process starts by saving the order in the ERP System. The whole process is shown in Figure 6.1. An incoming customer order generates an event with all relevant information to manufacture the product. This information includes customer data, order specific data like order date and information about the ordered products. The event is sent to the MSB for further processing (1). At the MSB each message has to run through the registration and routing (2). The event processing service receives the event message and checks the condition for customer order messages: all order messages from the ERP-System have to be routed to the MES. The customer order event is sent to a BPEL-service (3), which sends an e-mail with well formatted information to the production manager (4) which creates the production order by using the MES terminal (5). Once the production order has been created, the MES creates a text

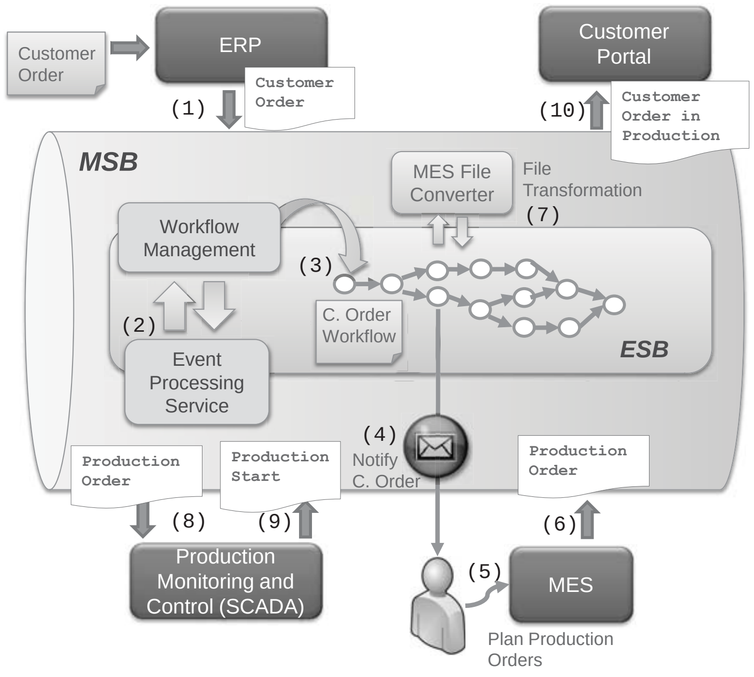


Figure 6.1: Customer Order Workflow

file, which contains all necessary information to start the production in the shop floor area. The text file, which does not contain information about the customer, is sent to the MSB (6). The text file is converted to a specific format (7) and then sent back to the event processing service. The message runs through the same process of registration and routing. At this point, the message is routed to the shop floor (8). At the shop floor, the production control unit creates a new production order with all machine instruction and the execution of the order is started. During the production process the factory is continuously saving state entries in a special logging database table. These logging entries are observed by a trigger in the database which sends notifications to the MSB about the production process (9). The customer portal changes the state of the or-

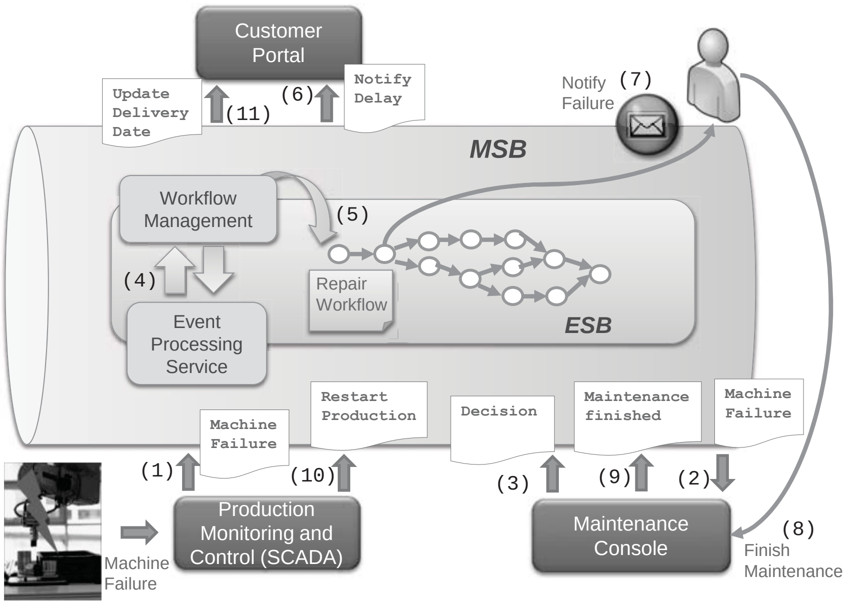


Figure 6.2: Failure Management Workflow

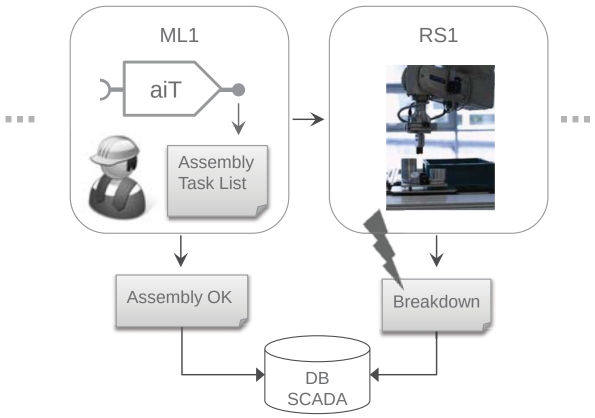
der to "in process" when the production order starts to be processed in the shop floor (10). All relevant updates in the production process are routed to the customer portal. In this scenario, the customer portal is the main event consumer, following a customer service policy that keeps customers permanently informed during production.

6.2.2 Failure Management

This scenario describes the processing of a machine failure event (see Figure 6.2). This failure event is generated from the shop floor using the same database functionality like normal production update messages, which are sent to the MSB in order to be forwarded to the appropriate destinations (1). A new failure message is routed to the maintenance portal (2). At the maintenance portal workers can decide which activity has to be done. There are two possible activities to react to machine failures: repair or replace. The decision about the activity is automatically sent back to the MSB as an event message that contains the decision about the failure (3). In this case, we consider the decision is a repair procedure and that no part of the machine needs to be replaced. The dependency with the initial event is set through the Eventflow-ID attribute. After performing the registration, the Event Processing Engine sends the event to the Workflow Management System (4), which triggers the Repair Workflow (5), which is executed in the runtime environment of a Workflow Management System. With the information related to maintenance decision, the delay in the scheduled orders can be calculated and the customers may be informed about the delivery delays their orders will suffer due to the machine failure. This is done by the BPEL process, which sends a notification of delay to a customer portal application (6). After some time, according to an escalation timeout, the workflow sends an Email notification to the production manager (7). The production manager supervises all maintenance operations related to the machine failure that originated the stop of production and once all maintenance operations are concluded and production is ready to be restarted he logs into the maintenance console the appropriate data that represents the end of all maintenance operations in the machine that failed (8). The maintenance console sends then the "Maintenance finished" event to the MSB (9). From the production control unit, production is restarted (10) and customers receive the updated information about the delay of their orders (11).

6.2.3 Service Revision

The following scenario describes the adaptation capabilities of the integration platform and the challenge of optimization when the production of a new product variant starts. In this scenario, the production system manufactures more than fifty variants of the same product, which is a simple table-set. The different variants are assembled on the transformable assembly system: iTRAME. The initial situation for this scenario is given by the introduction of a new product variant (variant F). The configuration of the assembly line for this product variant comprises several manual stations and a robot station. The assembly of the new variant requires that an assembler mounts the set light on the table plate base at one of the manual stations (ML1). In this station, assemblers are assisted with assembly instructions by means of an advanced Information Terminal (aiT). Once a worker has mounted the product part, it confirms the assembly operation. The confirmation action is logged as a successful assembly task in the database of the SCADA system. In the next station (RS1), a robot arm is configured to mount a pen holder. In the SCADA database, the log table contains the result of assembly operations. Successful assembly operations are logged with 1, whereas unsuccessful assembly operations, due to some kind of failure for instance, are logged with 0. The module field defines the assembly station and the process field defines the product variant. This initial situation is depicted in Figure 6.3. The log table shows a correct assembly operation on variant C at the RS1 station and three incorrect assembly operations on variant F also at the RS1 station. The problem is given by an elevated number of failures in the robot station, which are logged after the start of production of the new variant. When the robot arm cannot mount the pen holder correctly, it triggers an alarm, which stops the material flow in the assembly line and prevents the production to be resumed until the failure has been repaired or until the robot arm is restarted manually in the case of a breakdown. Such repair operations on the assembly line are very expensive due to the production stop that the robot arm breakdown provokes. Therefore the production supervisory team decides to configure and run a monitoring process in NexusDS in order to analyze the failures that are produced in the assembly line. The monitoring process is configured based on the suspicion of



ID	Assembly	Time	Module	Process
1237	1	2011-11-03 08:05:06.7	RS1	C
1249	0	2011-11-03 08:12:31.9	RS1	F
1317	0	2001-11-03 08:27:59.2	RS1	F
1332	0	2011-11-03 09:05:06.5	RS1	F

Figure 6.3: Misleading Failure Detection Scenario

a correlation between the increased number of failures in the assembly line and the new variant. The configuration of the monitoring graph is depicted in Figure 6.5. The entries at the log table of the SCADA database are used as input data stream in the source operator. The Filter operator is configured to use the Failure Analysis mode, which lets all entries that are not related to failure logs out of the analysis. The Mining Parameterization operator is configured to merge three additional tuples from external databases. These tuples are:

- (Worker,Module): this tuple is imported from an assembly workers database. A worker is registered on a specific module during a period of time.

- (FailureType,AssemblyID): this tuple is imported from a maintenance database, which keeps records about the registered failures on the assembly line and maintenance operations. All failures from the assembly line have a failure type assigned to it. The Failure-Type value is a 5-digit code that contains five properties of a failure. This code is shown in Figure 6.4. The Material property indicates if the failure lies on the material or on the product part that is assembled. The Repair property defines the difficulty to repair a failure. The highest value in the scale (5) is used to define a high difficulty in the repair operations, which require experienced maintenance personal. The lowest value in the scale (1) defines a very low difficulty to repair the failure, which can be repaired by assembly workers or other non-maintenance personnel. The Priority property describes the priority, with which the failure has to be repaired. The highest priority is 5 and the lowest is 1. The priority is directly proportional with the impact of the failure on the assembly line. The Tool property defines if the failure is on an assembly tool. The Machine property defines if the failure is caused by a machine. The possible values of the Failure Type parameter can be seen in Figure 6.4.

Material	Repair	Priority	Tool	Machine
1/0	1-5	1-5	1/0	1/0

Figure 6.4: Failure Type Value Scope

- (TaskListQuality-Module): this tuple is imported from the database of a quality assurance system. Every assembly station (module) has an assigned value for the quality of its task list description. Values are assigned by assembly workers.

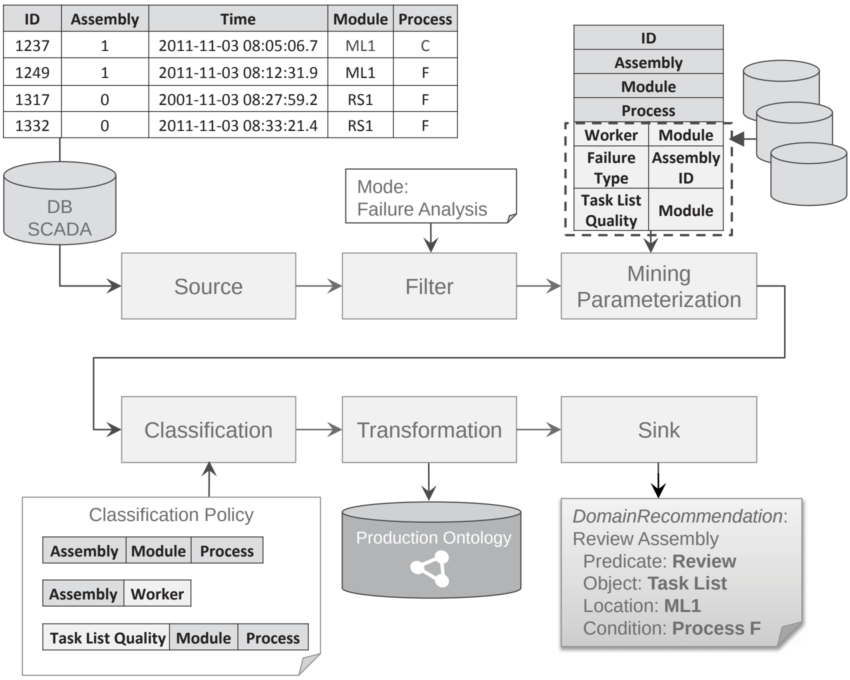
These tuples are merged with the tuples of the input data stream from the SCADA database. This is done in the Classification operator in order to classify production data according to different criteria. These criteria are given in a policy document. The purpose of merging different tuple combinations is to find correlations. In this example, three classifications are made, which are described ahead:

- The first classification is done by the tuple (Assembly, Module, Process). This classification looks for the correlation between failures, variants and assembly stations. In this case, no external tuple is merged. The results of the first classification reveal a correlation between the product variant F and the RS1 station for registered failures.

- The second classification merges the (Worker, Module) tuple with the (Assembly, Module) tuple. The goal of the mergence is to find correlations between assembly workers and failures. The result of this analysis does not show any relevant finding.

- The third classification merges the (TaskListQuality, Module) tuple with the (Module, Process) tuple of the SCADA Log table. A classification of values is made in the resulting (TaskListQuality, Module, Process) tuple in order to find correlations between the quality of the task list descriptions and the product variants for each module. This analysis reveals that more than 65% of the workers that have worked at ML1 station value the task list description for variant F as 'bad' or 'very bad'.

The results of the classification analysis describe a possible explanation for the elevated number of failures of variant F at the robot arm station. The conclusion of the monitoring process is that the failures that are registered at the robot arm station are actually originated at the previous manual station due to an incorrect assembly operation. The monitoring results show that the failures detected at the robot arm station are due to an overlapping previously-assembled part when an order of F-variants is



ID	Assembly	Time	Module	Process
1237	1	2011-11-03 08:05:06.7	ML1	C
1249	1	2011-11-03 08:12:31.9	ML1	F
1317	0	2001-11-03 08:27:59.2	RS1	F
1332	0	2011-11-03 08:33:21.4	RS1	F

Figure 6.5: The NexusDS Monitoring Graph for Failure Analysis at RS1

in production. This indicates that the previous assembly task has probably not been carried out correctly, although it is registered as correct by the assembly workers. This situation defines a misleading failure detection scenario, which can be detected by the established mining graph. The final steps of the mining graph include a Transformation operator and the Sink operator. The Transformation operator converts the results of the classification analysis into RDF statements based on the given Production Ontology. Once all RDF statements are created, a reasoning process is started in order to infer the corrective actions that are necessary to correct the failure situation. This reasoning process is a rule-based inference process that creates a DomainRecommendation OWL instance. This instance is sent over SOAP/HTTP to the repository, as described in Section 5.4.3.4. The domain recommendation requests a review the assembly task list, which is given by the output of the aiT service, since it seems that these instructions are the cause of the wrong assembly of the product variant. The review of the description of the assembly tasks at ML1 gives evidence of missing instructions for the assembly. This discovery and the consequent improvement of the assembly task description, brings the failure rate of the variant F down to average values.

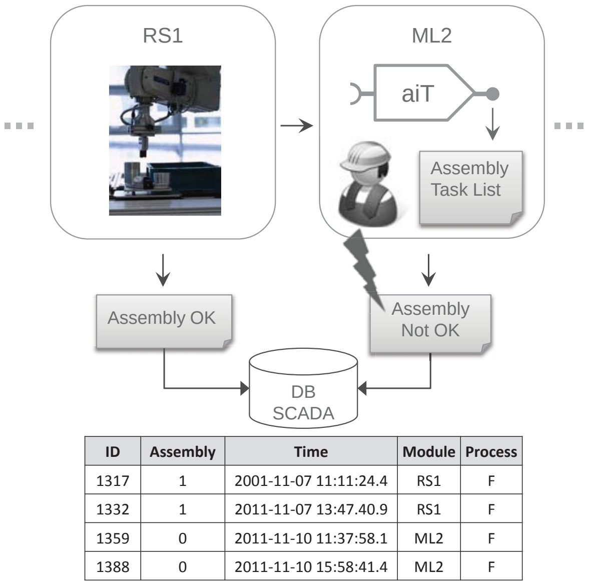
6.2.4 Adaptation of the Failure Management Process

This scenario is based on the same premises as the last scenario. The same product variant is in production. However, the failures are this time registered at a different assembly station. The scenario is depicted in Figure 6.6. This time workers are having difficulties to assemble the product part at the ML2 assembly station, which follows after the RS1 station. This provokes an elevated number of failures at the ML2 station when the product variant F is being assembled. The procedure to analyze the situation and to detect the origin of the problem is the same as in the last scenario. The NexusDS mining graph is configured to investigate possible correlations in the production data. The configuration of the mining graph is depicted in Figure 6.7. As in the scenario described in Section 6.2.3., the entries at the log table of the SCADA database are used as input data stream in the source operator. The Filter operator is configured to use the Failure Analysis mode. The Mining Parameterization operator is configured to merge three additional tuples from external databases. These tuples are:

- Worker-Module: (see scenario in Section 6.2.3).
- FailureType-AssemblyID-RepairTime: this tuple is imported from the maintenance database. This tuple shows the repair time that is invested in fixing each failure of the registered failures. This

tuple contains the specific description of failures in the failure type column. Failure types are described in Section 6.2.3.

- LineImpact-FailureType: this tuple, which is also imported from the maintenance database, shows the assigned impact on the assembly line of each failure type. The impact on the assembly line is a measure to define the importance of a failure. The impact of a failure is a value that goes from 1 to 10, being 10 the highest impact. For example, a failure that provokes a stop of production has a high impact (8), whereas a failure that is repaired as soon as it is detected and does not require stopping production has a relatively low impact (2).



ID	Assembly	Time	Module	Process
1317	1	2001-11-07 11:11:24.4	RS1	F
1332	1	2011-11-07 13:47.40.9	RS1	F
1359	0	2011-11-10 11:37:58.1	ML2	F
1388	0	2011-11-10 15:58:41.4	ML2	F

Figure 6.6: Misleading Failure Detection Scenario

These tuples are merged with the tuples of the input data stream from the SCADA database in the Classification operator in order to classify production data according to different criteria. The criteria to realize the classification of data are given in a policy document. In this example, five classifications are made, which are described ahead:

- The first classification is done based on the tuple (Assembly, Module, Process) (see Section 6.2.3). As suspected by the production management team, the results of the first classification reveals a correlation between the variant F and the ML2 station for registered failures.

- The second classification merges the tuple (Assembly, Module) from the SCADA log table and the tuple (Worker, Module) from the maintenance database. The resulting tuple (Assembly, Worker) is analyzed. No correlation is found.

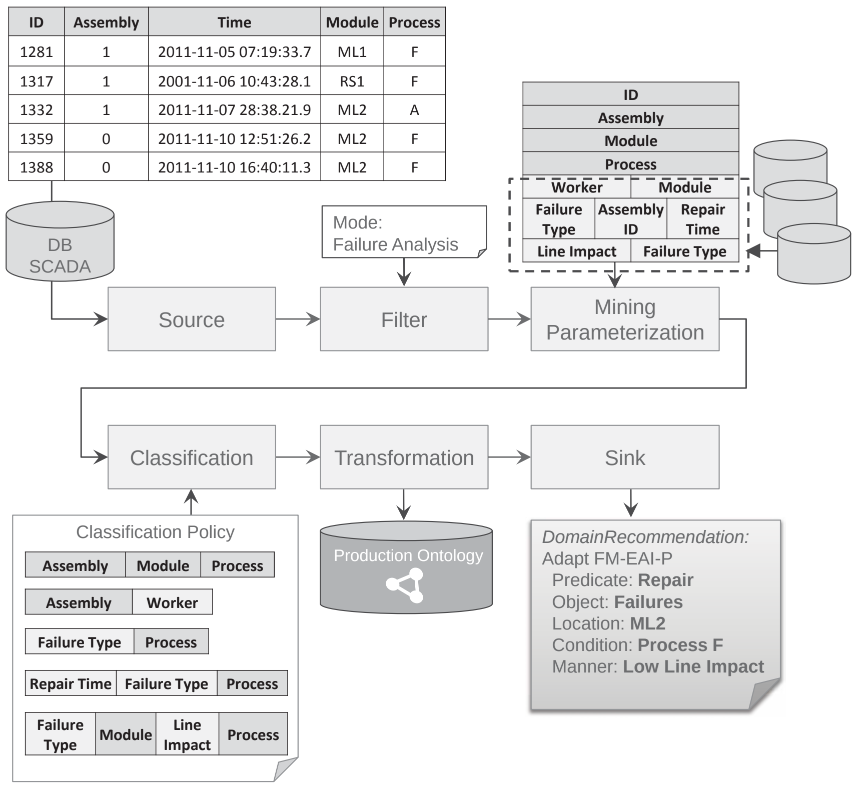
- The third classification merges the tuple (FailureType, AssemblyID, RepairTime) from the maintenance console and the tuple (ID, Process). The resulting tuple is (FailureType, Process). The results of the classification realized for this tuple show that over 90% of the failures registered at ML2 have the failure type ’11400’ (the codification follows the scheme described in Section 6.2.3). This means that most failures at ML2 are a failure related to the material or the product part, that is easily repairable and that has a high priority (4). The high priority means that the impact of this type of failures is high, meaning a probable stop of production.

- The fourth classification merges the tuple (FailureType, AssemblyID, RepairTime) from the maintenance database with the tuple (ID, Process) from the SCADA Log table. The resulting (FailureType, RepairTime, Process) tuple is analyzed in order to find out the repair times of the failures provoked by variant F. The results show low repair times, whereas the most failure types are coded as

’11400’. This means that the risk of stop of production is elevated although the repair times are low.

- The fifth classification merges two tuples of the maintenance console, which are the (LineImpact, FailureType) tuple and the (FailureType, AssemblyID, RepairTime), with the tuple (ID, Process, Module) of the SCADA log table. The resulting tuple (FailureType, Module, LineImpact, Process) is analyzed in order to look for the impact index of easily repairable failures (code pattern ’X1XX0’) provoked by variant F at the ML2 station. The results of the classification show that a high number of failures related to variant F at the ML2 station are easily repairable but with a high impact on the assembly line.

The results of the classification and analysis of the distribution of failures indicate that failures at ML2 are correlated again with the new product variant F and that these failures are independent of the assembly workers. The results also show that most failures are: (i) related to the material flow or the assembly product parts, (ii) easily repairable and (iii) have a high impact on the line. The reason for this distribution of failures is that the assembly at RS1 is not working correctly and is assembling a product part at an incorrect position, which should be free at ML2. This leads to a false positive assembly at RS1 and a false negative assembly at ML2. The robot arm must be inspected by an external maintenance service and possibly replaced. Since the maintenance operations of the RS1 take several weeks, an alternative solution is proposed. Failures of product variant F registered at ML2 related of a type pattern ’11X00’ are directly repaired by the assembly workers following maintenance instructions given by the aiT service. This type of failures is easily repairable and related to the material flow or product part, which means it is probably due to a wrong assembly of the robot arm at RS1. This solution is proposed at the Transformation operator after an inference process. The policy that leads to



ID	Assembly	Time	Module	Process
1281	1	2011-11-05 07:19:33.7	ML1	F
1317	1	2001-11-06 10:43:28.1	RS1	F
1332	1	2011-11-07 28:38.21.9	ML2	A
1359	0	2011-11-10 12:51:26.2	ML2	F
1388	0	2011-11-10 16:40:11.3	ML2	F

DomainRecommendation:
Adapt FM-EAI-P
Predicate: **Repair**
Object: **Failures**
Location: **ML2**
Condition: **Process F**
Manner: **Low Line Impact**

Figure 6.7: The NexusDS Mining Graph Configuration Failure Analysis at ML2

the conclusion mentioned above is coded in rules that are used to execute the inference process and generate a DomainRecommendation instance. These rules also contain the necessary information to require an adaptation of the EAI process for failure management. The adaptation process that follows after the DomainRecommendation instance is received by the service repository is depicted in Figure 6.8. The phases that follow the monitor phase (analyze, plan, execute) to complete the MAPE cycle in the given scenario are described ahead.

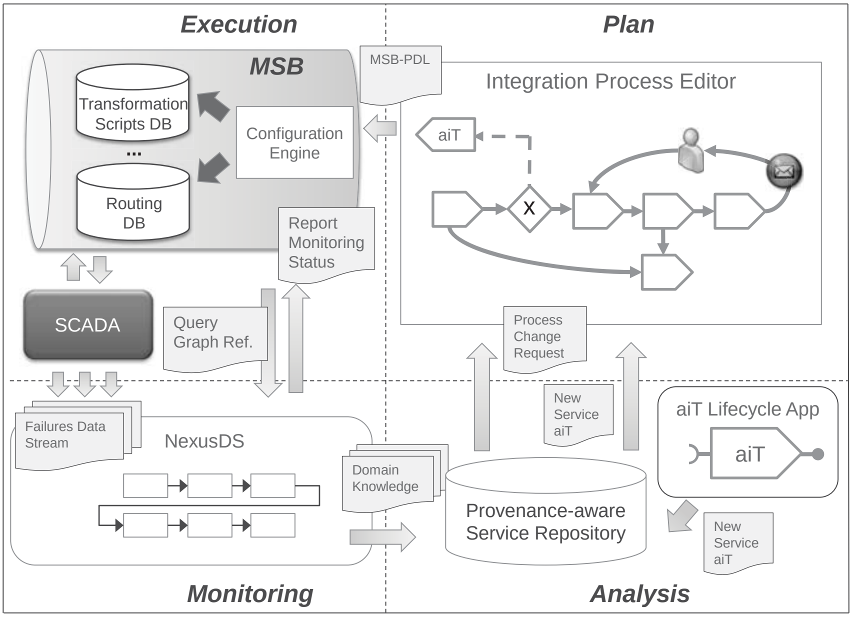


Figure 6.8: Domain Recommendation in the MAPE Cycle

Analysis. Once the domain knowledge is stored in the repository, it starts an inference process that derives high-level context information in order to send the corresponding corrective actions to the Process Editor. The information managed in the repository comprises service provenance data, process-service dependencies and domain knowledge. The service repository can combine these data and use its internal inference functions to make suggestions on process optimizations to modelers. This way, process modelers are able to adapt EAI processes in the planning phase when needed. In this scenario, the mining graph from the monitoring phase requires to repair failures of variant F at the assembly station ML2 with a low impact on the assembly line. In terms of failure management operations, a low impact is equivalent to a repair ad-hoc without having to stop production. In the service repository, the algorithm that looks for corrective actions is executed. This algorithm finds the description of the failure management process, which is given in Section 5.3.2, since the process matches the predicate, object and domain of the domain recommendation instance. A match for the location attribute is not found in any node of the process and therefore, services are looked up with the same location attribute (ML2). The aiT service is found. The algorithm creates a CorrectiveAction instance, which is translated into a change process request, which is sent to the Process Editor, and a change service request, which is sent to the lifecycle application of the aiT service. The adaptation of the aiT service requires the service to also show information the repair of failures of Variant F, if necessary. The lifecycle application of this service realizes the required changes that enable the visualization of repair instructions and registers the update in the service repository. The notification to the EAI Process Editor contains a request to integrate the updated version of the aiT service into the process into an edge that causes a low impact on the assembly line. The evaluation of these conditions is made by means of the internal rules of the service repository and transformed into specific requests that are sent to the EAI Process Editor as recommendation for the adaptation of the failure management process. The purpose of the integration of the aiT service into the process is to assist assemblers for ad-hoc repair operations for the product variant F.

Plan and Execute. The process change request can be interpreted by the editor-internal logic and used to make adaptation suggestions to process modelers. Once, process modelers have a new version of the process to be deployed, a description of this new process version is sent to the MSB in a new MSB-PDL description. The MSB configuration engine extracts this description and adapts the internal bus services, such as routing or transformation services, to enable the execution of the new control flow of the process.

6.3 Integration of the MSB into PLM

As described in Chapter 4, the integration of systems in the PLM domain comprises similar problems to the ones that are dealt with in the Real-time Factory, such as the heterogeneity of systems, the excessive number of point-to-point interfaces and the lack of a functional orientation with regard to the integration of applications. These problems cause a lack of flexibility and agility in PLM, which are indispensable nowadays due to the growing number of product variants, shorter product lifecycles and turbulent markets that require an efficient management of the products within their lifecycle. Furthermore, PLM needs a flexible and efficient support for the execution of the business processes within the product life cycle. The similarity of the ICT requirements for PLM to the requirements of the Real-time Factory in terms of integration ensures the reuse of the service orientation concept to increase interoperability and flexibility in the PLM domain. The extension of the MSB concept to the PLM domain and how the interconnection of PLM phases can be done is detailed ahead.

6.3.1 Extending the MSB Concept to other phases in PLM

First, the leverage of a unique service bus, as the MSB, to connect all applications of the product lifecycle was considered. But there are several problems, which have to be considered, when integrating hundreds of applications with a single ESB. The first problems are the different requirements within each phase of the product lifecycle like scalability, throughput and real-time demands. Further problems are of an organizational nature like the organizational responsibility for the development and maintenance. The resulting fear of loss of predominance between the different responsible persons in the individual departments and business units can be a great obstacle, before implementing the integration solution [MSM11]. Another problem is the geographical distribution of the departments, which leads to an increased coordination effort when using a holistic ESB implementation. To reduce the effect of these problems, an approach with phase-specific ESB's for each phase of the product lifecycle can be used. Each ESB is adapted to the requirements of its phase like the MSB to the manufacturing phase. Subsequently, the specific ESB's have to be connected with an additional ESB, the PLM-Bus [SMM11], to get a continuous integration of the whole product life cycle. The PLM-Bus supports the execution of cross-departmental business processes and provides the necessary mediation services to act as gateway between the different phase-specific buses. An example of the integration approach to connect two phases of the product life cycle is given ahead.

6.3.2 Approach to Connect the Planning and the Production Phases

The integration approach that is followed to integrate the MSB with other phases of the product life cycle is depicted in Figure 6.9. From the perspective of the MSB, as well as from the perspective of any phase-specific service bus, the PLM-Bus acts as gateway for event-messages that must be routed to applications that are not directly connected. This is done by connecting the routing service of the PLM-Bus to the MSB. In order to receive messages from other phase-specific buses, the routing service of the MSB is connected to the PLM-Bus. The PLM-Bus manages the business processes, mediation flows and mediation services that are needed to execute transactions across different phase-specific buses. Mediation services comprise a number of transformation services capable of converting data between data models specific to the applications of each phase. These are called inter-phase mediation services. Additionally, the inter-phase mediation flows control the message flows that go between phases. The approach is shown in Figure 6.9, where the MSB is connected to the service bus specific to the production planning phase, also called Production Planning Service Bus (PPSB).

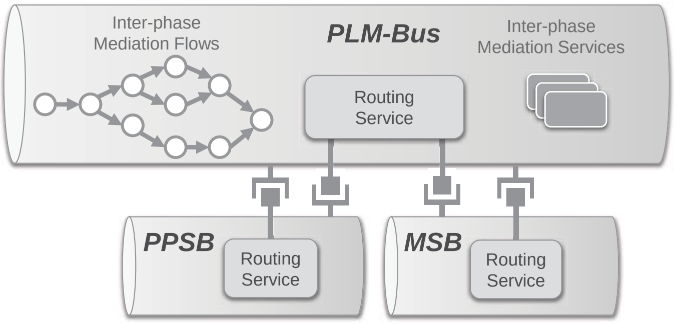
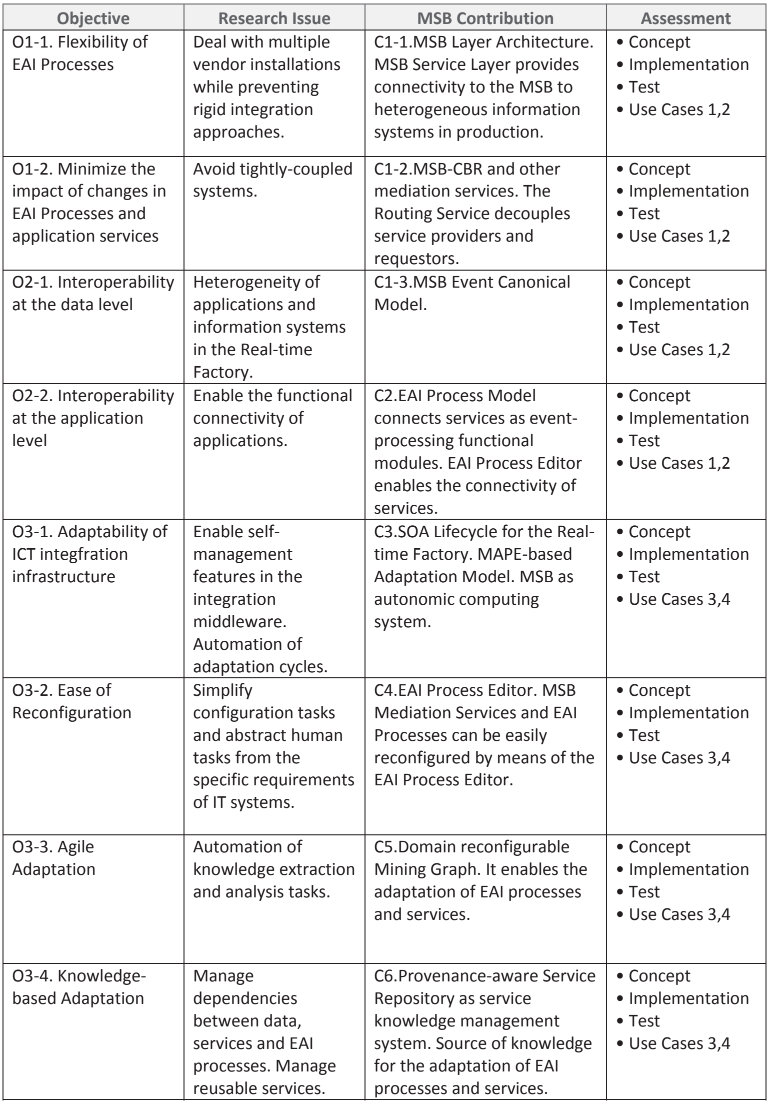


Figure 6.9: Integration of the MSB into a PLM Service-based Architecture

6.4 Validation of the Approach

The validation of the approach aims to demonstrate the technical feasibility and applicability and to evaluate the practical value for the groups of interest. The demonstration of the technical feasibility is described in Chapter 5. The applicability of the approach is illustrated in the use cases of this Chapter. The validation of research results is the assessment of the presented contributions, which is given by an extensive coverage of the objectives that are described in Chapter 1. This assessment is described in Section 6.5.1. The evaluation of the approach is given at the end of this Chapter as well as a comparison with other current approaches in the manufacturing, EAI and SOA domains.

6.4.1 Objectives Coverage



Objective	Research Issue	MSB Contribution	Assessment
O1-1. Flexibility of EAI Processes	Deal with multiple vendor installations while preventing rigid integration approaches.	C1-1.MSB Layer Architecture. MSB Service Layer provides connectivity to the MSB to heterogeneous information systems in production.	• Concept • Implementation • Test • Use Cases 1,2
O1-2. Minimize the impact of changes in EAI Processes and application services	Avoid tightly-coupled systems.	C1-2.MSB-CBR and other mediation services. The Routing Service decouples service providers and requestors.	• Concept • Implementation • Test • Use Cases 1,2
O2-1. Interoperability at the data level	Heterogeneity of applications and information systems in the Real-time Factory.	C1-3.MSB Event Canonical Model.	• Concept • Implementation • Test • Use Cases 1,2
O2-2. Interoperability at the application level	Enable the functional connectivity of applications.	C2.EAI Process Model connects services as event-processing functional modules. EAI Process Editor enables the connectivity of services.	• Concept • Implementation • Test • Use Cases 1,2
O3-1. Adaptability of ICT integfration infrastructure	Enable self-management features in the integration middleware. Automation of adaptation cycles.	C3.SOA Lifecycle for the Real-time Factory. MAPE-based Adaptation Model. MSB as autonomic computing system.	• Concept • Implementation • Test • Use Cases 3,4
O3-2. Ease of Reconfiguration	Simplify configuration tasks and abstract human tasks from the specific requirements of IT systems.	C4.EAI Process Editor. MSB Mediation Services and EAI Processes can be easily reconfigured by means of the EAI Process Editor.	• Concept • Implementation • Test • Use Cases 3,4
O3-3. Agile Adaptation	Automation of knowledge extraction and analysis tasks.	C5.Domain reconfigurable Mining Graph. It enables the adaptation of EAI processes and services.	• Concept • Implementation • Test • Use Cases 3,4
O3-4. Knowledge-based Adaptation	Manage dependencies between data, services and EAI processes. Manage reusable services.	C6.Provenance-aware Service Repository as service knowledge management system. Source of knowledge for the adaptation of EAI processes and services.	• Concept • Implementation • Test • Use Cases 3,4

Figure 6.10: Objectives Coverage

The assessment of the presented contributions is structured by the objectives that are described in Chapter 1. The major objectives of the Thesis given in Chapter 1 are three: flexibility, interoperability and adaptation of EAI processes. These three objectives can be divided into sub-objectives that have guided the realization of the approach presented in this thesis. The objectives are described in Chapter 1 and are also listed in Table 1.1 as well as in Figure 6.10. The realization of the work packages entails challenges that have been met by the presented contributions. The assessment of the contributions is based on the conception and design of the solution, the implementation and the necessary tests that have been carried out to prove the technical feasibility of the approach. Additionally, the use cases described in this Chapter are also a piece of the validation to prove the applicability of the approach. The coverage of the objectives is shown in Figure 6.10 by the contributions of the approach and the respective assessments.

6.4.2 Evaluation and Comparison with other Approaches

The use cases in this Chapter have been tested in the Learning Factory. Thus, the use cases are applicability tests in a manufacturing environment that is representative of a Small or Medium Enterprise (SME) of the manufacturing industry. Therefore, the evaluation of the MSB and the presented MAPE cycle are valid for the Learning Factory as well as for manufacturing SMEs. The evaluation of the approach is made by the compliance assessment of thirty criteria that have been selected as the most important factors for EAI middleware in the manufacturing industry. The selection of these criteria is based on the objectives of this Thesis, which are described in Chapter 1, and on the principles of integration that are applied in current EAI solutions. The thirty criteria points are divided into six categories: interoperability, flexibility, mediation, adaptability, agility and integration. Each of these categories contains five criteria. The criteria in each category are arranged in a scale from 0 to 4. The fulfillment of a criterion presumes the fulfillment of the lower criteria, except for the lowest criterion. The evaluation criteria for each category are defined ahead.

6.4.2.1 Evaluation Criteria

The first category of the evaluation criteria is interoperability. This category corresponds to the first objective stated in Chapter 1. The lowest criterion is communication of components with no integration. The second criterion is the usage of standards. The third and fourth criteria are the interoperability at the data level, application level. The fifth criterion is the support of multiple bindings, i.e. SOAP/HTTP, FTP or file transfer. These criteria can be seen in Figure 6.11. The second category is flexibil-

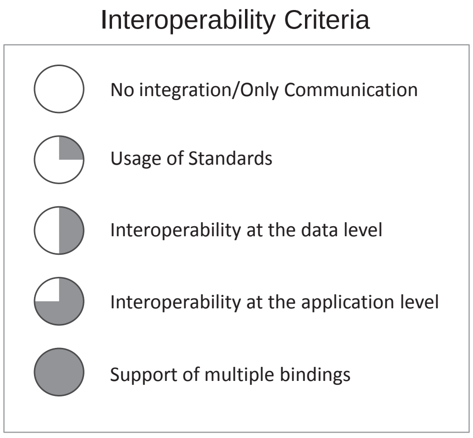

Interoperability Criteria
No integration/Only Communication
Usage of Standards
Interoperability at the data level
Interoperability at the application level
Support of multiple bindings

Figure 6.11: Interoperability Criteria

ity, which corresponds to the second objective of the Thesis. The lowest criterion is tightly-coupled components, which is equivalent to no flexibility. The first precondition for flexibility is to enable the integration of loosely-coupled components. Thus, this is the second criterion. The third criterion is service reuse, which presumes an integration approach based on loosely-coupled components. Finally, the highest criteria are defined by the reduction of complexity and the possibility to realize dynamic bindings (also called late bindings). The reduction of complexity can be achieved in different ways, but it usually aims to simplify the design, configuration and deployment of services and EAI processes. Flexibility criteria are shown in Figure 6.12. The third category is mediation. Mediation

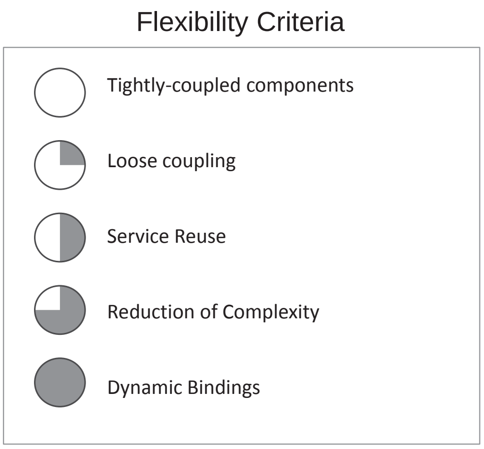


Figure 6.12: Flexibility Criteria

comprises several components or services in an integration architecture that support to decouple clients and servers, or service providers and requestors in a SOA. The lowest criterion represents the absence of mediation services. The most simple and common mediation services are transformation services, which define the second lowest criterion. The third criterion is represented by routing services, such as the CBR in the MSB. Orchestration or workflow management represents the fourth criterion. Finally, the fifth criterion of mediation is the support of stateful services. Mediation criteria are shown in Figure 6.13. The adaptability

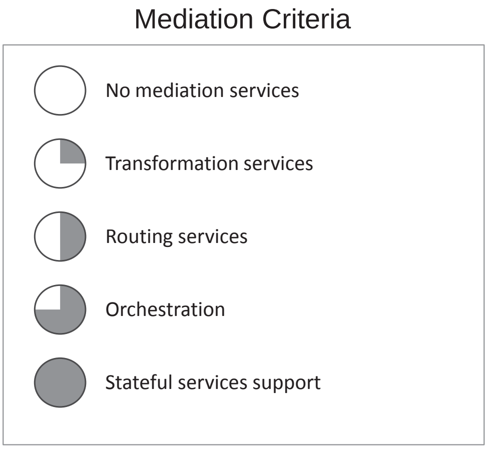


Figure 6.13: Mediation Criteria

category coincides with the third major objective of the Thesis. This category gathers the criteria to evaluate the capability of adaptation of an integration approach. The first criterion is given by manual or complex reconfiguration procedures. This criterion is equivalent to the absence of adaptability. The second criterion is versioning support. This feature is an important building block for adaptive integration infrastructures that manage different versions and variants of services. The management of data provenance and dependencies between the different entities in an integration scenario defines the third criterion. The fourth criterion is given by the features of self-managing components. Self-managing and self-governance mechanisms are usually present in autonomic computing systems that are capable of collecting domain data, analyzing it and adapting to new situations if needed. The monitoring of quality of service data defines the fifth criterion. The adaptability category and the corresponding criteria can be seen in Figure 6.14. The fifth category is

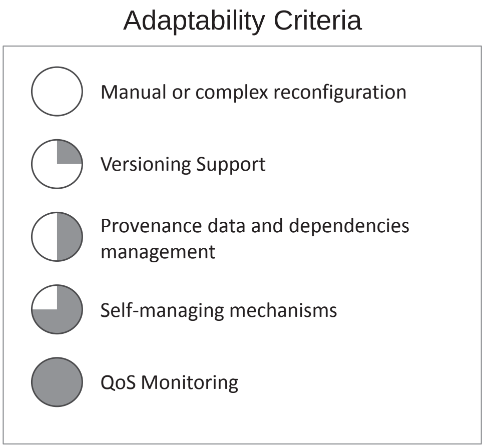


Figure 6.14: Adaptability Criteria

defined by one of the most important aspects in the adaptability of IT infrastructures in changing environments: agility. The ability to react responsively is a fundamental requirement in adaptation. An agile adaptation presumes the automation of certain tasks, such as monitoring and analysis. The first criterion is given by the absence of automated tasks in the domain analysis. Triggering corrective actions asynchronously represents the first criterion. The second criterion is given by the automation of monitoring and domain analysis tasks. The possibility to adapt configuration or services at runtime is the fourth criterion. The fifth criterion is represented by the implementation of a MAPE feedback loop (see Section 5.4). The agility criteria can be seen in Figure 6.15. The integration category defines the influence of an integration model that governs the integration processes. The absence of an integration model defines the first criterion, whereas the existence of an integration model complies with the first criterion. The expressiveness of the model is represented by the third criterion. The management of business or integration rules to analyze data or to adapt processes is given by the fourth criterion. The

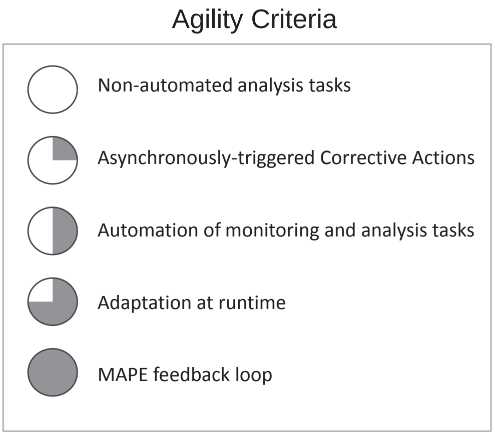

Agility Criteria
Non-automated analysis tasks
Asynchronously-triggered Corrective Actions
Automation of monitoring and analysis tasks
Adaptation at runtime
MAPE feedback loop

Figure 6.15: Agility Criteria

last criterion defines if the model is executable or if it can be converted to executable code. The criteria of this category can be seen in Figure 6.16.

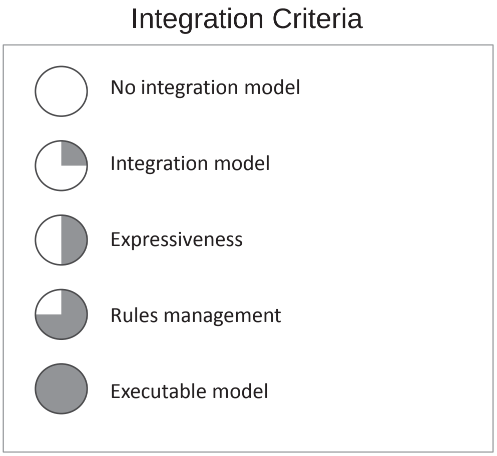


Figure 6.16: Integration Criteria

6.4.2.2 Evaluation and Comparison

The presented categories are used to evaluate the MSB approach and to compare it with other approaches. The result of the evaluation can be seen in Figure 6.17. The categories are placed on the left-side column and the MSB, along with other integration approaches are placed on the top. The evaluation of the MSB shows compliance with the categories that are derived from the objectives of the Thesis, such as interoperability, flexibility and agile adaptation. The MSB gets the highest score at interoperability thanks to the support of multiple bindings in the JBI implementation of the service bus and also thanks to the integration at the data level and at the application level, which are enabled by the MSB Event Model. This model provides a common data model for applications to exchange messages and to connect service interfaces by means of event processing and producing methods. The implemented CBR and mediation services decouple service providers and requestors, which in addition to the EAI Process Editor facilitate the highest score in flexibility. The implemented MAPE feedback loop (see Figure 5.22), which is described in Section 5.4.2 and Section 5.4.3 gives the MSB the highest score in agility. The executable EAI process model (MSB-PDL), which is described in Section 5.3.2 (see also Appendix B), provides the MSB with the highest score in integration. Only the lack of support for QoS monitoring and missing support of communication for stateful services prevent the MSB to score the maximum possible points in all categories. However, the comparison with other approaches reveals that none of the approaches is capable of supporting stateful services. Moreover, the monitoring of QoS runtime data is only supported by the VRESCo environment [VRESCo]. The adoption of QoS monitoring techniques is increasing in current research projects due to the growing importance of compliance and governance in service-oriented computing. However, more approaches have not been included since not all have the focus on integration. The Champagne

[RCH+02] approach and the Smart Factory [LCW08] are two research projects designed to integrate manufacturing applications and to manage context in factories respectively. The missing flexibility and agility is especially remarkable in these approaches. The lack of an integration model in Champagne also prevents it to point in the integration category. On the contrary, there are four approaches - Adept2 [RRK+05], GENIUS [SL09], Padres [FJM05], SSB [KWV+07] - that show comparable scores with the MSB. One of these approaches is the research project Adept2 [RRK+05], which is a system to assist process modelers to adapt business processes. Adept2 obtains lacks a MAPE feedback loop to achieve the agility score of the MSB. A MAPE loop is also missing in the Semantic Service Bus (SSB) [KWV+07] and in the Padres architecture [FJM05]. The SSB shows, except for the agility category, identical scores to the MSB. The reason for this match is the similar implementation characteristics of both approaches, which are based on an ESB with extended functionality.

	MSB	Champagne	Smart Factory	Adept2	VRESCo	GENIUS	Padres	SSB
Interoperability	●	◑	◑	●	◔	◕	●	●
Flexibility	●	○	◔	●	◕	●	●	●
Mediation	◕	◑	◔	◕	○	◕	◕	◕
Adaptability	◕	◑	◑	◕	●	◕	◕	◕
Agility	●	◔	◑	◕	◑	○	◕	◕
Integration	●	○	◑	●	◕	●	◕	●

Figure 6.17: Evaluation and Comparison with other Approaches

6.5 Summary

In this chapter, four applicability use cases and the validation and evaluation of the approach are given. The first two use cases demonstrate the usability of the MSB in a real production environment. Both use cases show the flexibility and interoperability of the approach thanks to the routing service and other mediation services of the integration architecture, which enables components to remain loosely-coupled. The last two use cases demonstrate the adaptability and agility features of the implemented MAPE-based feedback loop. The architecture presents self-managing and adaptive mechanisms thanks to the automation of the monitoring and analysis tasks. The validation of the approach is given by a complete coverage of the objectives with the contributions of this thesis. The coverage of the objectives in Chapter 1 can be seen in Figure 6.10. The evaluation of the approach has been realized by the examination of thirty criteria that have been classified in six categories: interoperability, flexibility, mediation, adaptability, agility and integration. The MSB support for standards, the integration at the data level and at the application level, along with the support of multiple bindings provide the highest score at interoperability. The flexibility provided by the MSB and the implemented CBR and mediation services by decoupling applications has been demonstrated and evaluated with the highest score in flexibility category. The executable EAI process model (MSB-PDL), which is described in Section 5.3.2, and the implemented MAPE feedback loop (see Figure 5.22) provide the MSB with the highest score in agility and integration as well. The lack of QoS monitoring and the lack of support for stateful service prevent the MSB to score the highest score at the adaptability and mediation categories. The support of stateful services is still a research issue in many other SOA research projects based on web services. The evaluation of the MSB in comparison with other approaches is very positive due to the lack of agility and adaptation capabilities in some of the projects. Other projects, like the Semantic Service Bus [KWV+07], Adept2 [RRK+05] or Padres [FJM05] obtain similar results and would also be a valid approach if applied to the Real-time Factory. The results of the complete evaluation can be seen in Figure 6.17.

Chapter 7

Conclusions and Outlook

In this Chapter the conclusions and an outlook are given.

7.1 Conclusions

In this thesis, an integration solution to meet the challenges of flexible information provisioning and agile adaptation in the Real-time Factory has been proposed. The approach comprises the concept, implementation and applicability tests of an integration middleware that connects digital production systems in a manufacturing environment. The challenges of integration in the Real-time Factory include dealing with the heterogeneity of IT systems, the avoidance of tightly-coupled components that hinder agile adaptation of processes and the automation of monitoring, analysis and reconfiguration tasks. The proposed solution in this thesis addresses these challenges and presents six contributions:

1. The first contribution is given by the Manufacturing Service Bus and its Layer Architecture (see Section 5.1), which classifies the manufacturing environment of the Real-time Factory into five layers. The layered view of the environment enables software architects to connect applications to the service bus using different levels of integration: communication and connectivity (layer 0 and 1), data level integration (layer 2), application level integration (layer 3) and process level integration (layer 4). The MSB concept introduces SOA principles into manufacturing environments [MLJ+10], which provides more flexibility and agility in the adaptation of integration processes, as it is described in Chapter 6. The integration layer of the MSB, which is described in Section 5.2, comprises a routing service, several transformation services and a BPEL workflow engine that enables the execution of BPEL-processes. These mediation services (routing, transformation and orchestration) provide the means to decouple service providers and requestors in a service-oriented environment. Loose coupling is one of the fundamental integration principles and preconditions for more flexibility and for an agile adaptation of processes, as described in Chapter 3. An important part of this contribution is the MSB Event Canonical Model, which provides a common messaging model for all applications that are connected to the MSB. The event canonical model reduces complexity over time, as the number of applications increases and as changes are introduced [Cha04]. This model is based on XML and has a common schema for events, which defines some basic characteristics, such as registration and routing properties. This XML Schema, which is shown in Appendix A, can be extended with custom parts, which can be included in event messages. This is useful to include additional information on the event type and ensures the extensibility of the model [MRR+10, MRM+10].

2. The second contribution is the EAI Process Model, which is described in Section 5.3.2. The model integrates services, which are represented by nodes. Nodes send events of a specific event type to the MSB. The MSB processes the event and sends it forward to the Nodes that are subscribed to this specific event type. This mediation operation from a Node to the MSB and from the MSB to another Node is represented as an Edge. This model is used to plan and design the integration processes that enable the exchange of data in the Real-time Factory and that are executed in the Manufacturing Service Bus. Thus, this model is the basis for the mediation flows that enable the integration of heterogeneous applications in the Real-time Factory. The model of a sample EAI process is shown in Appendix B.

3. The third contribution is an EAI Process Editor to plan, design, adapt and deploy EAI processes, as described in Section 5.3.3. The EAI Process Editor uses the EAI Process Model in order to provide process modelers with a graphical representation of EAI processes. The representation of EAI process models is possible thanks to a graphical UI that comprises process modeling features to connect nodes with edges and to annotate processes, nodes and edges with metadata information. Additionally, the EAI Process Editor is able to receive update notifications about the services and EAI processes that it manages. The notification mechanism assists process modelers with recommendations based on the domain knowledge and service provenance information.

4. The fourth contribution is the MAPE-based Adaptation Model (see Figure 5.19), which is described in Section 5.4.2.3. This model serves as guideline for the feedback loop established between the MSB and the planning environment. The adaptation of EAI processes follows a MAPE-based feedback loop approach, in which

the domain is monitored and analyzed in order to re-design and re-deploy EAI processes if needed. The given MAPE feedback loop provides a SOA lifecycle of the EAI processes that seamlessly integrate the information systems of the Real-time Factory. The MAPE-based Adaptation Model provides the basis for the implementation of an autonomic computing system with self-managing and self-adaptation capabilities.

5. The fifth contribution is a reconfigurable domain Mining Graph. This Graph is described in Section 5.4.3.3 and is implemented as a sequence of operators in the NexusDS platform [CEB+09]. The graph comprises four processing operators, a source and a sink operator. The source operator receives factory data streams from the shop floor. The following processing operators filter the data stream, classify and correlate the configured sets of data, and transform the results of the correlation analysis into RDF statements. An ontology of the production environment and a set of rules serve as basis for these RDF statements and for the inference process that derives higher-level context information. This information is modeled as an OWL instance that contains a recommendation based on the analyzed data. This recommendation is referred to as domain knowledge and is sent to a Service Repository, which represents the sixth contribution.

6. The sixth contribution is a Provenance-aware Service Repository (see Section 5.3.4), which processes domain recommendations and enables the communication with the EAI Process Editor and other lifecycle applications in order to react responsively to turbulent scenarios in the domain. The Service Repository manages information about services, processes and their dependencies in a service knowledge base and in a process knowledge base. The semantic data engine provides an inference mechanism, based on an

algorithm that generates the appropriate corrective actions to attend the recommendations made by the Mining Graph. The Service Repository has an interface to communicate with the EAI Process Editor which is based on a specific XML language. The Service Provenance Query Language (SPQL) is used for this interface in order to query data and service provenance information. This language also provides constructs to send change process notifications and service change requests. An example of an SPQL-query is given in Appendix C. The Service Repository comprises the analysis phase of the MAPE cycle that has been proposed for the Real-time Factory in Chapter 5.

One of the most important benefits of the MSB in comparison to other approaches based on point-to-point interfaces is the extensibility and scalability provided by the MSB event model, which requires data translation to and from an application-independent canonical format. This is a best-practice and recommended strategy in most integration scenarios [Cha04]. This bypasses approaches based on point-to-point interfaces, whose number of transformation instances increases exponentially with the number of applications. When extending the number of applications to be integrated, such implementations are very error-prone, hard to maintain, thus implying high costs. In the MSB, as applications change, extensions to the model are possible by adding extra XML attributes or by using the custom data part of an event-message. Thus, the impact is limited to message transformation to and from the canonical format. This presents a great advantage in the extensibility and scalability of the integration strategy. In addition to this, the MSB components follow a SOA-based approach. The routing service, the event registries as well as the interface to the shop floor and applications are implemented as services. This provides a level of abstraction, typical in Service-Component Architectures (SCA), which makes the computing environment technologically agnostic. Thus, the cost to replace components reduces drastically.

Consequently, if the implementation of the CBR be replaced or enhanced, the impact on the rest of the applications involved in message exchange workflows would be minimal. The routing service and other mediation services provide an integration backbone that facilitates the realization of a loosely-coupled integration.

Moreover, an agile adaptation of EAI processes has been demonstrated by automating the domain knowledge extraction process. This automation is possible thanks to the integration of a data stream processing platform (NexusDS) with the Manufacturing Service Bus, which governs the execution of EAI processes. This integration enables data streams from the production environment to be processed in real-time and transformed into domain context information, which can be further evaluated by the internal inference mechanisms of the Provenance-aware Service Repository. The adoption of NexusDS entails closing the MAPE cycle and thus being able to perform an agile adaptation of EAI processes in manufacturing environments. The automation of the domain knowledge extraction process eases an agile adaptation of EAI processes based on a real-time domain data evaluation. Additionally, the flexibility of NexusDS and its connection to the service bus enables different client applications to parameterize diverse monitoring processes.

The validation of the MSB approach shows a complete coverage of the objectives that are described in Chapter 1. The assessment of the contributions comprises the concept, implementation of the architecture and the applicability tests that have been described in the use cases in Chapter 6. The MSB and the MAPE feedback loop have demonstrated to increase the flexibility and interoperability in a real manufacturing environment by providing a service-oriented integration middleware that supports the exchange of event-messages between applications in a loosely-coupled manner. The evaluation of the approach shows the practical value for companies in manufacturing industries that face similar challenges as the presented use cases for the Real-time Factory. In addition to this, a remarkable advantage in comparison with other approaches can be seen in the adaptability and agility features of the approach.

7.2 Outlook

As described in this Thesis, the service-oriented approach taken by the MSB provides a flexible integration framework and an agile adaptation of EAI processes in the Real-time Factory. However, some limitations exist and these leave room for future research efforts. One of the limitations of the MSB is the lack of support for stateful services. This is in general one of the most discussed aspects of web services. Web services do not keep any state between requests of a client, which means that a web service does not know if a subsequent request will be received or if previous requests from the same client have been sent. This type of communication follows the one-way message exchange pattern [Jos07], also known as fire-and-forget, and it is a common pattern in ESB implementations. However, in manufacturing there are long-running processes and transactional flows that are required to be stateful. This is the case of simulation processes in the production planning phase. The integration of such simulation processes in an integration infrastructure based on an ESB or web services needs to be further investigated. Another important challenge for further research is monitoring QoS parameters. This issue is gaining importance in service-oriented computing environments due to the impact of QoS in governance and inter-organizational transactions. The integration at the process level of multiple organizations requires establishing contracts (service level agreements) and monitoring the quality of service that is offered by service providers. Although some research projects are working on this issue [MRL+09], the automation of QoS monitoring processes remains a challenge [PTD+07] and has a great potential in the integration of different organizations into production networks [MBN11].

Appendix A

MSB Event Canonical Model

Listing A.1: MSB Event Model XML Schema

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema elementFormDefault="qualified"
           targetNamespace="http://tempuri.org/MSB
              "
           xmlns:xs="http://www.w3.org/2001/
              XMLSchema"
           xmlns:tns="http://tempuri.org/MSB">
 <xs:element name="Event" type="tns:EventType"/>
 <xs:complexType name="EventType">
  <xs:sequence>
   <xs:element minOccurs="0" maxOccurs="1" name="
      Description"
              type="xs:string"/>
   <xs:element minOccurs="0" maxOccurs="1" name="
      Routing"
              type="tns:RoutingType"/>
   <xs:element minOccurs="0" maxOccurs="1" name="
      TimeStamps"
              type="tns:TimeStampsType"/>
```

```
    <xs:element minOccurs="0" maxOccurs="1" name="
        Procedure"
                type="tns:ProcedureType"/>
    <xs:element minOccurs="0" maxOccurs="1" name="
        Responsible"
                type="tns:ResponsibleType"/>
    <xs:element minOccurs="0" maxOccurs="1" name="
        CustomData"
                type="tns:CustomDataType"/>
   </xs:sequence>
   <xs:attribute name="scheduled" type="xs:boolean"
                 use="required"/>
   <xs:attribute name="inCourse" type="xs:boolean"
      use="required"/>
   <xs:attribute name="eventType" type="xs:int" use
      ="required"/>
   <xs:attribute name="routed" type="xs:boolean"
      use="required"/>
   <xs:attribute name="eventIdRegistered" type="xs:
      boolean"
                 use="required"/>
   <xs:attribute name="eventId" type="xs:string"/>
   <xs:attribute name="eventFlowIdRegistered" type
      ="xs:boolean"
                 use="required"/>
   <xs:attribute name="eventFlowId" type="xs:string
      "/>
  </xs:complexType>
  <xs:complexType name="RoutingType">
   <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="1"
                name="OriginSystem" type="xs:string
                   "/>
    <xs:element minOccurs="0" maxOccurs="unbounded"
                name="DestinationSystem"
                type="tns:RoutingTypeDestination"/>
   </xs:sequence>
  </xs:complexType>
```

```
<xs:complexType name="RoutingTypeDestination">
 <xs:attribute name="SystemId" type="xs:string"/>
 <xs:attribute name="SystemUrl" type="xs:string
    "/>
</xs:complexType>
<xs:complexType name="TimeStampsType">
 <xs:sequence>
  <xs:element minOccurs="0" maxOccurs="1" name="
     started"
             type="xs:dateTime"/>
  <xs:element minOccurs="0" maxOccurs="1" name="
     ended"
             type="xs:dateTime"/>
 </xs:sequence>
</xs:complexType>
<xs:complexType name="ProcedureType">
 <xs:sequence>
  <xs:element minOccurs="0" maxOccurs="1" name="
     workflow">
   <xs:complexType>
    <xs:sequence>
     <xs:element minOccurs="0" maxOccurs="1" name
        ="description"
                type="xs:string"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:string"/>
   </xs:complexType>
  </xs:element>
 </xs:sequence>
 <xs:attribute name="required" type="xs:boolean"
    use="required"/>
</xs:complexType>
<xs:complexType name="ResponsibleType">
 <xs:sequence>
  <xs:element minOccurs="0" maxOccurs="1" name="
     Intern">
   <xs:complexType>
    <xs:sequence>
```

```
         <xs:element minOccurs="0" maxOccurs="1" name
            ="FirstName"
                     type="xs:string"/>
         <xs:element minOccurs="0" maxOccurs="1" name
            ="LastName"
                     type="xs:string"/>
         <xs:element minOccurs="0" maxOccurs="1" name
            ="Mail"
                     type="xs:string"/>
         <xs:element minOccurs="0" maxOccurs="1" name
            ="Phone"
                     type="xs:string"/>
        </xs:sequence>
       </xs:complexType>
      </xs:element>
      <xs:element minOccurs="0" maxOccurs="1" name="
         Extern">
       <xs:complexType>
        <xs:sequence>
         <xs:element minOccurs="0" maxOccurs="1" name
            ="FirstName"
                     type="xs:string"/>
         <xs:element minOccurs="0" maxOccurs="1" name
            ="LastName"
                     type="xs:string"/>
         <xs:element minOccurs="0" maxOccurs="1" name
            ="Mail"
                     type="xs:string"/>
         <xs:element minOccurs="0" maxOccurs="1" name
            ="Phone"
                     type="xs:string"/>
        </xs:sequence>
       </xs:complexType>
      </xs:element>
     </xs:sequence>
    </xs:complexType>
    <xs:complexType name="CustomDataType" abstract="
       true"/>
```

```
<xs:complexType name="OrderType">
 <xs:complexContent mixed="false">
  <xs:extension base="tns:CustomDataType">
   <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="1" name
       ="CustNo"
               type="xs:string"/>
    <xs:element minOccurs="0" maxOccurs="1" name
       ="Company"
               type="xs:string"/>
    <xs:element minOccurs="0" maxOccurs="
       unbounded" name="User">
     <xs:complexType>
      <xs:sequence>
       <xs:element minOccurs="0" maxOccurs="1"
          name="Login"
                  type="xs:string"/>
       <xs:element minOccurs="0" maxOccurs="1"
          name="Password"
                  type="xs:string"/>
      </xs:sequence>
     </xs:complexType>
    </xs:element>
    <xs:element minOccurs="0" maxOccurs="
       unbounded" name="Task">
     <xs:complexType>
      <xs:sequence>
       <xs:element minOccurs="0" maxOccurs="1"
          name="TaskNo"
                  type="xs:string"/>
       <xs:element minOccurs="0" maxOccurs="1"
          name="Process"
                  type="xs:string"/>
       <xs:element minOccurs="0" maxOccurs="1"
          name="ProductName"
                  type="xs:string"/>
       <xs:element minOccurs="0" maxOccurs="1"
```

```
             name="ProductDescription" type="xs:
                string"/>
        <xs:element minOccurs="1" maxOccurs="1"
           name="Quantity"
                  type="xs:int"/>
       </xs:sequence>
      </xs:complexType>
     </xs:element>
    </xs:sequence>
   </xs:extension>
  </xs:complexContent>
 </xs:complexType>
 <xs:complexType name="FailureType">
  <xs:complexContent mixed="false">
   <xs:extension base="tns:CustomDataType">
    <xs:sequence>
     <xs:element minOccurs="0" maxOccurs="1" name
        ="Action">
      <xs:complexType>
       <xs:attribute name="Preventive" type="xs:
          boolean"
                  use="required"/>
       <xs:attribute name="Required" type="xs:
          boolean"
                  use="required"/>
      </xs:complexType>
     </xs:element>
     <xs:element minOccurs="0" maxOccurs="1" name
        ="CurrentTask"
                type="xs:string"/>
     <xs:element minOccurs="0" maxOccurs="1" name
        ="Decision"
                type="xs:string"/>
     <xs:element minOccurs="0" maxOccurs="1" name
        ="ErrorCode"
                type="xs:string"/>
     <xs:element minOccurs="0" maxOccurs="1"
```

```
                name="ErrorDescription" type="xs:
                   string"/>
      <xs:element minOccurs="1" maxOccurs="1"
                name="EstimatedSolvingDate"
                   nillable="true"
                type="xs:dateTime"/>
      <xs:element minOccurs="0" maxOccurs="1" name
         ="Kind"
                type="xs:string"/>
      <xs:element minOccurs="0" maxOccurs="1" name
         ="MachineId"
                type="xs:string"/>
      <xs:element minOccurs="1" maxOccurs="1" name
         ="ReportingDate"
                nillable="true" type="xs:dateTime
                   "/>
      <xs:element minOccurs="0" maxOccurs="1" name
         ="State"
                type="xs:string"/>
     </xs:sequence>
    </xs:extension>
   </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="StatusType">
   <xs:complexContent mixed="false">
    <xs:extension base="tns:CustomDataType">
     <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="1" name
         ="TaskNo"
                type="xs:string"/>
      <xs:element minOccurs="1" maxOccurs="1" name
         ="Started"
                nillable="true" type="xs:dateTime
                   "/>
      <xs:element minOccurs="1" maxOccurs="1" name
         ="Ended"
                nillable="true" type="xs:dateTime
                   "/>
      <xs:element minOccurs="1" maxOccurs="1" name
         ="Bulk"
                 type="xs:unsignedByte"/>
      <xs:element minOccurs="1" maxOccurs="1"
                 name="CurrentInProduction"
                 type="xs:unsignedByte"/>
      <xs:element minOccurs="1" maxOccurs="1" name
         ="OrderQuantity"
                 type="xs:unsignedByte"/>
      <xs:element minOccurs="0" maxOccurs="1" name
         ="ProcessName"
                 type="xs:string"/>
     </xs:sequence>
    </xs:extension>
   </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="StartProductionType">
   <xs:complexContent mixed="false">
    <xs:extension base="tns:CustomDataType">
     <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="
         unbounded" name="Task">
       <xs:complexType>
        <xs:sequence>
         <xs:element minOccurs="0" maxOccurs="
            unbounded"
                    name="ProcessStep">
          <xs:complexType>
           <xs:attribute name="OperationCode"
                        type="xs:unsignedByte" use
                           ="required"/>
           <xs:attribute name="OperationName" type
              ="xs:string"/>
           <xs:attribute name="Modul" type="xs:
              string"/>
           <xs:attribute name="Par1" type="xs:
              string"/>
            <xs:attribute name="Par2" type="xs:
               string"/>
            <xs:attribute name="Par3" type="xs:
               string"/>
           </xs:complexType>
          </xs:element>
         </xs:sequence>
         <xs:attribute name="Exclusive" type="xs:
            boolean"
                      use="required"/>
         <xs:attribute name="IsMesCommand" type="xs:
            boolean"
                      use="required"/>
         <xs:attribute name="ProcessName" type="xs:
            string"/>
         <xs:attribute name="TaskNo" type="xs:string
            "/>
         <xs:attribute name="OrderQuantity" type="xs
            :unsignedByte"
                      use="required"/>
         <xs:attribute name="StartScheduled" type="
            xs:dateTime"
                      use="required"/>
        </xs:complexType>
       </xs:element>
      </xs:sequence>
     </xs:extension>
    </xs:complexContent>
   </xs:complexType>
  </xs:schema>
```

Appendix B

MSB Process Description Language (MSB-PDL)

Listing B.1: MSB-PDL for Failure Management Process

```
<pdl:pdl targetNamespace="http://www.msb-eai.uni-
    stuttgart.de/pdl/example"
  xsi:schemaLocation="http://www.msb-eai.uni-
     stuttgart.de/ns/pdl pdl.xsd"
  xmlns:pdl="http://www.msb-eai.uni-stuttgart.de/
     ns/pdl"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-
     instance"
  xmlns="http://www.msb-eai.uni-stuttgart.de/pdl/
     example">
<pdl:nodelist>
 <pdl:service id="SCADA">
  <pdl:edge id="SCADA-MC-1" edgeNumber="1.0" start
     ="SCADA"
```

```
           destination="MC" type="push" event="
              true"
           eventType="" routing="true" trigger="
              true"
           endOfProcess="false" />
 <!-- Failure detected -->
<pdl:edge id="SCADA-CP-1" edgeNumber="7.0" start
   ="SCADA"
          destination="CP" type="push" event="
             true"
          eventType="" routing="true" trigger="
             false"
          endOfProcess="true" />
 <!-- Production restarted - Notify Customers -->
</pdl:service>
<pdl:service id="MC">
 <!-- Maintenance Operation decided -->
 <pdl:edge id="MC-BPELRepair-1" edgeNumber="2.0"
    start="MC"
           destination="BPELRepair" type="push"
              event="true"
           eventType="" routing="true" trigger="
              false"
           endOfProcess="false" />
 <!-- Maintenance finished -->
</pdl:service>
<pdl:service id="BPELRepair">
<pdl:edge id="BPELRepair-CP-1" edgeNumber="3.0"
   start="BPELRepair"
           destination="CP" type="push" event="
              true"
           eventType="" routing="true" trigger="
              false"
           endOfProcess="false" />
  <!-- Notify Customers -->
 <pdl:edge id="BPELRepair-Mail-1" edgeNumber
    ="4.0"
```

```
              start="BPELRepair" destination="Mail"
                 type="push"
              event="false" eventType="" routing="
                 false"
              trigger="false" endOfProcess="false"
                 />
     <!-- Maintenance Instructions -->
    </pdl:service>
    <pdl:service id="Mail">
     <pdl:edge id="Mail-Person-1" edgeNumber="5.0"
        start="Mail"
              destination="Person" type="push" event
                 ="false"
              eventType="" routing="false" trigger="
                 false"
              endOfProcess="false"/>
     <!-- Maintenance Escalation -->
    </pdl:service>
    <pdl:service id="Person">
     <pdl:edge id="Person-MC-1" edgeNumber="6.0"
        start="Person"
              destination="MC" type="push" event="
                 false"
              eventType="" routing="false" trigger="
                 false"
              endOfProcess="false"/>
     <!-- Maintenance finished -->
    </pdl:service>
    <pdl:service id="CP">
    </pdl:service>
   </pdl:nodelist>
```

Appendix C

SPQL Sample Request

Listing C.1: SPQL Query to Create Process

```
<?xml version="1.0" encoding="UTF-8"?>

<spql:spql targetNamespace="http://www.msb-eai.uni
    -stuttgart.de/spql/DiscoveryExample/"
  xsi:schemaLocation="http://www.msb-eai.uni-
      stuttgart.de/ns/spql spql.xsd"
  xmlns:spql="http://www.msb-eai.uni-stuttgart.de/
      ns/spql"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-
      instance"
  xmlns="http://www.msb-eai.uni-stuttgart.de/spql/
      CreateProcess/">

<spql:query type="create">
 <spql:object value="process" id="aIT-142" >
  <spql:metadata  domain="assembly"
    version="1.42"
```

```
      predecessor="1.41"
      owner="John Smith - Production Manager"
      name="assembly Information Terminal"
      group="assembly"
      deployment_date="2011-10-21 00:00:00"
      release_date="2011-10-22 00:00:00"
      adaptation_pending="false"
      function="(process, failures, type:assembly)">
     <spql:triggers>
      <spql:simpleEvent trigger="true" review_edge="
         yes"
                        domain="failure" pattern="
                           Trigger-1"
                        action_request="yes"/>
      <spql:complexEvent trigger="true" type="chain"
          review_edge=""
                         domain="failures" pattern="
                            Trigger-2"
                         action_request="
                            service_adoption">
       <spql:simpleEvent review_edge="LR-MC-1"
          domain="failure"
                         pattern="NewProduct_VS1
                            [0-9]"
                         action_request=""/>
       <spql:simpleEvent review_edge="" domain="
          failure"
                         pattern="Failure_ML2[0-9]"
                         action_request=""/>
       <spql:complexEvent type="repetition"
          review_edge=""
                          domain="assembly" pattern
                             ="RE_min-3-1-72"
                          action_request="
                             service_adoption" >
        <spql:complexEvent type="repetition"
           review_edge=""
```

```
                          domain="assembly" pattern
                             ="MAX-1970"
                          action_request="
                             service_adoption" >
        <spql:complexEvent type="repetition"
           review_edge=""
                           domain="assembly"
                              pattern="min-1991"
                           action_request="
                              service_adoption" >
         <spql:simpleEvent review_edge="" domain="
            failure"
                           pattern="Stop[VS1|ML2
                              ][0-120][ID]"
                           action_request=""/>
        </spql:complexEvent>
       </spql:complexEvent>
      </spql:complexEvent>
     </spql:complexEvent>
    </spql:triggers>
   </spql:metadata>
   <spql:resource type="SAWSDL" uri=""/>
  </spql:object>
 </spql:query>
 </spql:spql>
```

List of Figures

List of Tables

List of Algorithms

Bibliography

[AC05] Austvold, E., Carter, K., Service-Oriented Architectures: Survey Findings on Deployment and Plans for the Future, AMR Research, 2005.

[ASM+11] Abel, M.; Klemm, P.; Silcher, S.; Minguez, J.: Start-Up of Reconfigurable Production Machines with a Service-Oriented Architecture. In Proceedings of the 21st International Conference on Production Research, 2011.

[BDE95] Boubekri, N.; Dedeoglu, M.; Eldeeb, H.: Application of standards in the design of computer-integrated manufacturing systems. In Integrated Manufacturing Systems, Vol. 6 No. 1, 1995, pp. 27-34, MCB University, Press Limited, 0957-6061, 1995.

[BHL95] Blakeley, B.; Harris, H.; Lewis, R.: Messaging and queueing using the MQI. New York, NY, USA, McGraw-Hill, Inc. ISBN 0-07-005730, 1995.

[BKT07] P. Buneman, S. Khanna, W.C. Tan, "Data Provenance: Some Basic Issues". Foundations Of Software Technology and Theoretical Computer Science, Vol. 1974/2000, 87-93, Springer, 2000.

[BL98] Berners-Lee, T.: Semantic web road map, Internal note, World Wide Web Consortium, 1998. - http://www.w3.org/DesignIssues/Semantic.html

[BNM10] Baureis, D.; Neumann, D.; Minguez, J.: From a Product to a Product-Service System Supply Chain: A Strategic Road-map. Proceedings of 12th International MITIP Conference. Aalborg, Dänemark, 2010.

[BR90] Babbar, S.; Rai, A.: Computer-integrated Flexible Manufacturing: an Implementation Framework. In International Journal of Operations & Production Management, 10, 42-50, 1990.

[Bro03] Brown, J., The Many Faces of PLM, Tech-Clarity, Inc., 2003. - http://tech-clarity.com/documents/-Many_Faces_of_PLM.pdf

[BSG+09] Brun, Y.; Serugendo, G.M.; Gacek, C.; Giese, H.; Kienle, H.; Litoiu, M; Müller, H.; Pezzè, M.; Shaw, M.: Engineering Self-Adaptive Systems through Feedback Loops. In Software Engineering for Self-Adaptive Systems. In: Betty H. Cheng, Rogério Lemos, Holger Giese, Paola Inverardi, and Jeff Magee (Eds.), Lecture Notes In Computer Science, Vol. 5525. Springer-Verlag, Berlin, Heidelberg 48-70, 2009.

[BSM+06] Balasubramanian, K.; Schmidt, D. C.; Molnár, Z.; Lédeczi, Á.: System Integration Using Model-Driven Engineering, 2006. - http://www.dre.vanderbilt.edu/ kitty/pubs/bookchapter-final.pdf

[C+08] Cheng, B. T.C. et al: Software Engineering for Self-Adaptive Systems: A Research Road Map. In Dagstuhl Seminar on Software Engineering for Self-Adaptive Systems in January 2008.

[CCM+01] Christensen, E. ; Curbera, F. ; Meredith, G. ; Weerawarana, S.: Web Services Description Language (WSDL) 1.1. März 2001. - www.w3.org/TR/wsdl

[CEB+09] Cipriani, Nazario; Eissele, Mike; Brodt, Andreas; Großmann, Matthias; Mitschang, Bernhard: NexusDS: A Flexible and Extensible Middleware for Distributed Stream Processing, ACM (Hrsg): IDEAS ’09: Proceedings of the 2008 International Symposium on Database Engineering and Applications, 2009.

[Cha04] D. Chappell. Enterprise Service Bus. O’Reilly Media, Inc., 1st edition, June 2004.

[CHL+05] Constantinescu, C., Heinkel, U., Le Blond, J., Schreiber, S., Mitschang, B., Westkämper, E.: Flexible Integration of Layout Planning and Adaptive Assembly Systems in Digital Enterprises. In: Proceedings of the 38th CIRP International Seiminar on Manufacturing Systems, 2005.

[CHM02] Constantinescu, C, Heinkel, U., Meinecke, H.: A Data Change Propagation System for Enterprise Application Integration, in Proceedings of the 2nd International Conference on Information Systems and Engineering (ISE 2002), pp. 129-134, San Diego, USA, 2002.

[CLS05+] Curbera, F. ; Leymann, F. ; Storey, T. ; Ferguson, D. ;Weerawarana, S.: Web Services Platform Architecture: Soap, WSDL, WS-Policy, WSAddressing, WS-BPEL, WS-Reliable Messaging and More. Prentice Hall PTR, 2005.

[CN08] Cugola, G., Di Nitto, E.: On adopting Content-based Routing in service-oriented architectures, Information and Software Technology, Elsevier Science, ISSN: 0950-5849,

[CS06] Cardoso J., Sheth, A. P., Semantic Web Services, Processes and Applications, Springer, 2006.

[CW08] Constantinescu, C., Westkäamper, E.: Grid engineering for networked and multi-scale manufacturing. In: The 41st CIRP Conference on Manufacturing Systems May 26-28, Tokyo, Japan, 2008.

[DGM97] DeVor, R., Graves, R., & Mills, J. J.: Agile manufacturing research: Accomplishments and opportunities. IIE Transactions, 29(8), 813-823, 1997.

[DS08] De Labey, S.; Steegmans, E.: Extending WS-Notification with an Expressive Event Notification Broker. In: Proc. of the 2008 IEEE International Conference on Web Services, Beijing, China, 2008.

[DSN+10] Dobson, S.; Sterritt, R.; Nixon, P.; Hinchey, M.: Fulfilling the Vision of Autonomic Computing. In IEEE Computer Society, 2010.

[Erl05] Erl, T., Service-Oriented Architecture: Concepts, Technology, and Design. Prentice Hall International, 2005.

[Eva01] Evans, M.: The PLM Debate. Cambashi Inc. Article, 2001.

[Fas08] Fasbinder, M.: BPEL or ESB: Which should you use? IBM Developer Works, 2008.

[FJM05] Fidler, E.; Jacobsen, H.-A.; Li, G.; Mankovski, S.: The PADRES distributed publish/subscribe system. In Proceedings of Feature Interactions in Telecommunications and Software Systems VIII (ICFI'05), 2005.

[FSW99] Fan, Y.; Shi, W.; Wu, C.: Enterprise wide application integration platform for CIMS implementation. In Journal of Intelligent Manufacturing (1999) 10, 587-601, 1999.

[FYL+08] Fu, S. S.; Yang, J.; Laredo, J.; Huang, Y.; Chang, H.; Kumaran, S.; Chung, J.-Y.; Kosov, Y.: Solution Templates Tool for Enterprise Business Applications Integration. In: Sensor Networks, Ubiquitous, and Trustworthy Computing, International Conference on, pp. 314-319, ISBN 978-0-7695-3158-8, 2008.

[GJ05] Gudivada, V.N.; Jagadeesh, N.:Enterprise Application Integration Using Extensible Web Services. In Proceedings of the IEEE International Conference on Web Services (ICWS'05), 2005.

[Gun97] Gunasekaran, A., Implementation of computer-integrated manufacturing: a survey of integration and adaptability issues. Int. J. Compu. Integr. Manuf., 10(1-4): 266-280, 1997.

[HCM05] Heinkel, U.; Constantinescu, C.; Mitschang, B.: Integrating Data Changes with Data from Data Service Providers. In Proceedings of the ISCA 18th International Conference on Computer Applications in Industry and Engineering (CAINE), 2005.

[Hen06] Henning, M.: The rise and fall of CORBA, Queue, v.4 n.5, June 2006.

[HGB05] Haller A, Gomez JM, Bussler C. Exposing Semantic Web Service principles in SOA to solve EAI scenarios. In: Proceedings of the WWW2005, Chiba, Japan, 10-14 May 2005.

[HLN03] Hau, J.; Lee, W.; Newhouse, S.; "Autonomic service adaptation in ICENI using ontological annotation," Proceedings of the Fourth International Workshop on Grid Computing, pp. 10-17, 2003.

[Hoh04] Hohpe, G.: Enterprise Integration Patterns: Asynchronous Messaging Architectures in Practice, 5th Internation Middleware Conference, Tutorial, 2004. - http://www.eecg.toronto.edu/middleware2004/tp04.htm

[Hoh07] Hohpe, G.: Architect's dream or developer's nightmare? In: DEBS '07: Proceedings of the 2007 inaugural international conference on Distributed event-based systems. New York, NY, USA. ACM Ű ISBN 978-1-59593-665-3, S. 188, 2007.

[HP06] HP Labs Jena - A semantic web framework for Java, 2006. - http://jena.sourceforge.net

[HPG06] K. Holley, J. Palistrant, S. Graham, Effective SOA Governance, IBM, 2006.

[HPH03] Horrocks, I., Patel-Schneider, P., Harmelen, F.: From SHIQ and RDF to OWL: The Making of a Web Ontology Language. In: Journal of Web Semantics, 2003.

[HW03] Hohpe, G. ; Woolf, B.: Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Addison-Wesley Professional, ISBN 0321200683, 2003.

[IBM] IBM, Websphere MQ. - http://www-01.ibm.com/software/integration/wmq

[IBM05a] IBM, An architectural blueprint for autonomic computing, White Paper, June 2005.

[IBM05b] IBM: IBM's SOA Foundation - An Architectural Introduction and Overview. IBM White paper, 2005.

[IET05] Internet Engineering Task Force (IETF): RFC 3986 - Uniform Resource Identifier (URI): Generic Syntax. 2005. - http://www.ietf.org/rfc/rfc3986.txt

[Int00] International Society of Automation: ISA-95 Manufacturing Enterprise Systems Standards. 2000. - http://www.isa.org

[Int07a] International Organization for Standardization (ISO): ISO 10303-28:2007 Product data representation and exchange - Part 28 (STEP-XML), 2007.

[Int07b] International Organization for Standardization (ISO): ISO 10303: Standard for the Exchange of Product Model Data (STEP), 2007.

[Jam05] Jammes F.: Service-oriented device communications using the devices profile for web services. MPAC '05 Proceedings of the 3rd international workshop on Middleware for pervasive and ad-hoc computing. ACM, 2005.

[Jav05] Java Community Process: JSR-208 Java Business Integration (JBI), 2005. - http://jcp.org/aboutJava/communityprocess/final/jsr208/index.html

[JGr01] JGraph Ltd. JGraph (Java Graph Visualization Library), 2001. - http://www.jgraph.com

[Jos07] Josuttis, N.M.: SOA in Practice. O'Reilly, ISBN-10: 0-596-52955-4, 2007.

[JWW09] Jovane, F., Westkämper, E., Williams, D., The ManuFuture Road, Springer Verlag, Berlin, 2009.

[KAB+04] Keen, M.; Acharya, A.; Bishop, S.; Hopkins, A.; Milinski, S.; Nott, C. Robinson, R.; Adams, J.; Verschueren, P.: Patterns: Implementing an SOA Using an Enterprise Service Bus. Redbooks, IBM Intl Tech Support Organization, July 2004.

[KAW07] Kapp, R.; Aldinger, L.; Westkämper, E.: Real-time factory cockpit system. In International Conference on Computer-Aided Production Engineering (CAPE), 2007.

[KC03] Kephart, J.O.; Chess, D.M.: The Vision of Autonomic Computing. Computer, v.36 n.1, pp. 41-50, 2003.

[Kid94] Kidd, P.T.: Agile Manufacturing: Forging New Frontiers, Addison-Wesley, Reading, MA, 1994.

[KKL+05] Kloppmann, M.; Koenig, D.; Leymann, F.; Pfau, G.; Rickayzen, A.; von Riegen, C.; Schmidt, P.; Trickovic, I.: WS-BPEL Extension for People - BPEL4People, July 2005.

[KKR09] T. Kohlborn, A. Korthaus, M. Rosemann: Business and Software Service Lifecycle Management, 13th IEEE International EDOC Conference, 1-4 September, Auckland, New Zealand, 2009.

[KVL+08] Karastoyanova, D.; van Lessen, T.; Leymann, F.; Ma, Z.; Nitzsche, J.; Wetzstein, B.; Bhiri, S.; Hauswirth, M.; Zaremba, M.: A Reference Architecture for Semantic Business Process Management Systems. In: Bichler, M. (Hrsg); Hess, T. (Hrsg); Krcmar, H. (Hrsg); Lechner, U. (Hrsg); Matthes, F. (Hrsg); Picot, A. (Hrsg); Speitkamp, B. (Hrsg); Wolf, P. (Hrsg): Multikonferenz Wirtschaftsinformatik 2008.

[KWV+07] Karastoyanova, D.; Wetzstein, B.; Van Lessen, T.; Wutke, D.; Nitzsche, J.; Leymann, F.: Semantic Service Bus: Architecture and Implementation of a Next Generation Middleware. In Proceedings of the 23rd International Conference on Data Engineering Workshops, ICDE 2007, 15-20 April 2007, Istanbul, Turkey, 2007.

[LC08] Laliwala, Z., Chaudhary, S., Event-driven Service-Oriented Architecture. In: International Conference on Service Systems and Service Management, July 2008, pp.1-6, 2008.

[LCW08] Lucke D., Constantinescu C. Westkämper E.: Smart factory - a step towards the next generation of manufacturing. In 41st CIRP Conference on Manufacturing Systems, Proceedings: 115-118, 2008.

[LCW09] Lucke D., Constantinescu C. Westkämper E.: Context data model, the backbone of a smart factory. In 42nd CIRP Conference on Manufacturing Systems, Grenoble, June 3-5, 2009.

[Lin99] Linthicum, D. S.: Enterprise Application Integration. Addison-Wesley Professional, 1999.

[M06] Maréchaux, J.-L.: Combining Service-Oriented Architecture and Event-Driven Architecture using an Enterprise Service Bus, IBM Developer Works, 2006.

[Mes08] MESA Intl., IBM Corporation, Capgemini, SOA in Manufacturing Guidebook, MESA International white paper, 2008.

[MBN11] Minguez, J.; Baureis, D.; Neumann, D.: Providing Coordination and Goal Definition in Product-Service Systems through Service-oriented computing, in: Conference Proceedings of the 44th CIRP Conference on Manufacturing Systems, Madison/USA, June, 2011.

[MHC00] Monson-Haefel, R. ; Chappell, D.: Java Message Service. O'Reilly, ISBN 978-0596522049, 2000.

[Mic00] Microsoft Message Queuing, 2000. - http://msdn.microsoft.com/en-us/library/ms834460.aspx

[MJH+09] Minguez, J.; Jakob, M.; Heinkel, U.; Mitschang, B., A SOA-based approach for the integration of a data propagation system, 2009, IEEE International Conference on Information Reuse and Integration, 2009, vol., no., pp.47-52, 10-12 Aug. 2009.

[ML08] Ma, Z.; Leymann, F.: A Lifecycle Model for Using Process Fragment in Business Process Modeling. In Proceedings of BPDMS 2008 Workshop at CAiSE'08, Montpellier, 2008.

[MLJ+10] Minguez, J., Lucke, D., Jakob, M., Constantinescu, C.; Mitschang, B., Westkämper, E., Introducing SOA into Production Environments: The Manufacturing Service Bus, Proceedings of the 43rd CIRP International Conference on Manufacturing Systems (ICMS), pp. 1117-1124, Vienna, Austria, 2010.

[MNM11] Minguez, J.; Niedermann, F.; Mitschang, B.: A provenance-aware service repository for EAI process modeling tools, 2011 IEEE International Conference on Information Reuse and Integration (IRI), pp.42-47, 2011.

[MP06] McCoy, D.; Plummer, D.: Defining, Cultivating and Measuring Enterprise Agility. Gartner Research, 2006.

[MRL+08] Michlmayr, A.; Rosenberg, F.; Leitner, P.; Dustdar, S.: Advanced event processing and notifications in service runtime environments. In Proceedings of the second international conference on Distributed event-based systems, ISBN: 978-1-60558-090-6, pp. 115–125, 2008.

[MRL+09] Michlmayr, A.; Rosenberg, F.; Leitner, P.; Dustdar, S.: Service Provenance in QoS-Aware Web Service Runtimes. In IEEE International Conference on Web Services, pp.115-122, 2009.

[MRM+10] Minguez, J.; Riffelmacher, P.; Mitschang, B.; Westkämper, E: Servicebasierte Integration von Produktionsanwendungen. In wt-online 3-2011, Seite 128-133, 2011.

[MRR+10] Minguez, J; Ruthardt, F; Riffelmacher, P; Scheibler, T; Mitschang, B: Service-based Integration in Event-driven Manufacturing Environments, Proceedings of the 1st Symposium on Web Intelligent Systems and Services (WISS), 11th International confernce on Web Information System Engineering (WISE), Hong Kong, 2010.

[MRZ11] Minguez, J.; Reimann, P.; Zor, S.: Event-driven business process management in Engineer-to-Order supply chains. In Proceedings of the 2011 15th International Conference on Computer Supported Cooperative Work in Design, CSCWD, 2011.

[MS94] McGaughey, R.E.; Snyder, C.A.: The obstacles to successful CIM. International Journal on Production Economics 37 (1994) 247-258, 1994.

[MSM11] Minguez, J.; Silcher, S.; Mitschang, B.: A Service Bus Architecture for Application Integration in the Planning and Production Phases of a Product Lifecycle, International Journal of Systems and Service-Oriented Engineering (IJSSOE), Vol. 2, 2011.

[MSM+11] Minguez, J.; Silcher, S.; Mitschang, B.; Westkämper, E: Towards Intelligent Manufacturing: Equipping SOA-based Architectures with advanced SLM Services, Proceedings of the 44th CIRP International Conference on Manufacturing Systems (ICMS), Madison, Wisconsin (USA), 2011.

[NG94] Ngwenyama, O.K.; Grant, D.A.: Enterprise Modeling for CIM Information Systems Architectures: an Object-oriented Approach. In: Computers md Engng Vol 26, No 2, pp 279-293, 1994.

[NGS+01] Nicklas, D.; Großmann, M.; Schwarz, T.; Volz, S.; Mitschang, B.: A Model-Based Open Architecture for Mobile, Spatially-Aware Applications. In: Jensen, C. et al. (Eds.), Advances in Spatial and Temporal Databases. Lecture Notes in Computer Science, 2121. Springer, Berlin, pp. 117-135, 2001.

[NRM11] Niedermann, F.; Radeschütz, S.; Mitschang, B.: Business Process Optimization Using Formalized Optimization Patterns. In Proceedings of the 14th International Conference, BIS 2011, Poznan, Poland, June 15-17, 2011.

[Obj09] Object Management Group (OMG): Business Process Model and Notation (BPMN), 2009. - http://www.omg.org/spec/BPMN/1.2/

[Obj91] Object Management Group (OMG): Common Object Request Broker Architecture (CORBA) 1.0, 1991. - http://www.omg.org/spec/CORBA/1.0/

[OPC02] OPC Foundation: OPC Data Access Specification (OPC DA) v2.05a, 2002. - http://www.opcfoundation.org

[OPC09] OPC Foundation: OPC Unified Architecture (OPC UA) v2.05a, 2009. - http://www.opcfoundation.org

[Ora] Oracle: Java Remote Method Invocation - Distributed Computing for Java. White Paper. - http://www.oracle.com/technetwork/java/javase/tech/index-jsp-138781.html

[Ora05] Oracle: Service-Oriented Architecture (SOA) and Web Services: The Road to Enterprise Application Integration (EAI), 2005. http://www.oracle.com/technetwork/articles/javase/soa-142870.html

[Ora08] Oracle: Glassfish Open Source Server, 2008. - http://glassfish.java.net/

[Org04] Organization for the Advancement of Structured Information Standards (OASIS): UDDI Version 3.0.2, 2004. - http://www.uddi.org/pubs/uddi_v3.htm

[Org06a] Organization for the Advancement of Structured Information Standards (OASIS): Web Services Base Notification 1.3, 2006. - http://docs.oasis-open.org/wsn/wsn-ws_base_notification-1.3-spec-os.pdf

[Org06b] Organization for the Advancement of Structured Information Standards (OASIS): Web Services Brokered Notification 1.3, 2006. - http://docs.oasis-open.org/wsn/wsn-ws_brokered_notification-1.3-spec-os.pdf

[Org06c] Organization for the Advancement of Structured Information Standards (OASIS): Web Services Topics 1.3, 2006. - http://docs.oasis-open.org/wsn/wsn-ws_topics-1.3-spec-os.pdf

[Org06d] Organization for the Advancement of Structured Information Standards (OASIS): Web Services Resource Framework (WSRF) Primer v1.2, 2006. - http://docs.oasis-open.org/wsrf/wsrf-primer-1.2-primer-cd-02.pdf

[Org07] Organization for the Advancement of Structured Information Standards (OASIS): Web Services Business Process Execution Language (WS-BPEL) Version 2.0 - OASIS Standard. 2007. - http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.htm

[Pap08] Papazoglou, M., Web Services: Principles and Technology. Pearson, Prentice Hall, Harlow, 2008.

[PH07] Papazoglou, M., v. d. Heuvel, W., Service oriented architectures: approaches, technologies, and research issues. The VLDB Journal, 2007.

[PM07] Papakostas, N.; Mourtzis, D: An Approach for Adaptability Modeling in Manufacturing ŰAnalysis Using Chaotic Dynamics. In CIRP Annals - Manufacturing Technology, Volume 56, Issue 1, pp. 491-494, 2007.

[PTD+07] M. Papazoglou, P. Traverso, S. Dustdar, F. Leyman, Service-Oriented Computing: State of the Art and Research Challenges, IEEE Computer, pp. 38-45, 2007.

[QYS+08] Qian, J., Yin, J., Shi, D., Dong, J., Exploring a Semantic Publish Subscribe Middleware for Event-Based SOA. In: Asia-Pacific Services Computing Conference, APSCC '08. IEEE, pp.1269-1275, 2008.

[RCH+02] Rantzau, R.; Constantinescu, C.; Heinkel, U.; Meinecke, H.: Champagne: Data Change Propagation for Heterogeneous Information Systems. In Proceedings of the 28th VLDB Conference, Hong Kong, China, 2002.

[RKK+07] Riffelmacher, P.; Kluge, S.; Kreuzhage, R.; Hummel, V.; Westkämper, E.: Learning factory for the manufacturing industry. In Proceedings of the 20th International Conference on Computer-Aided Production Engineering, CAPE, 2007.

[RLL09] Ryan, K.L.K.; Lee, S.S.G.; Lee, E.W.: Business process management (BPM) standards: a survey. In Business Process Management Journal Vol. 15 No. 5, pp. 744-791, 2009.

[RM09] Radeschütz, Sylvia; Mitschang, Bernhard: Extended Analysis Techniques For a Comprehensive Business Process Optimization. In: Proc. of the International Conference on Knowledge Management and Information Sharing, Portugal, 2009.

[RRK+05] Reichert, M.; Rinderle, S.; Kreher, U.; Dadam, P.; , "Adaptive Process Management with ADEPT2", Proceedings. 21st International Conference on Data Engineering, pp. 1113- 1114, 2005.

[SFB01] SFB 627: Nexus - Spatial World Models for Mobile Context-Aware Applications, 2001. - http://www.nexus.uni-stuttgart.de

[SHB06] Shadbolt, N.; Hall, W.; Berners-Lee, T.; , "The Semantic Web Revisited," Intelligent Systems, IEEE , vol.21, no.3, pp.96-101, 2006.

[She99] Sheth, A. P.: Changing Focus on Interoperability in Information Systems: From System, Syntax, Structure to Semantics, Interoperating Geographic Information Systems. C. A. Kottman, Kluwer Academic Publisher: 5-29, 1999.

[SHL+05] Schmidt, M.-T., Hutchinson, B. ; Lambros, P. ; Phippen, R.: The Enterprise Service Bus: Making service-oriented architecture real. In: IBM Systems Journal, 44(4), 2005.

[Sin97] Singh, V.: The Cim Debacle: Methodologies to Facilitate Software Interoperability. Springer August 1997, ISBN 981-3083-21-2, 1997.

[SKL+10] Schumm, D.; Karastoyanova, D.; Leymann, F.; Strauch, S.: Fragmento: Advanced Process Fragment Library. In Proc. of the 19th International Conference on Information Systems Development (ISD'10), 2010.

[SKN+07] Schuh, G.; Kampker, A.; Narr, C.; Potente, T.; Attig, P.: myOpenFactory. In: International Journal of Computer Integrated Manufacturing, 2nd ed. vol. 21, no. 2, p. 215, 2007.

[SL09] Scheibler, T.; Leymann, F.: From Modelling to Execution of Enterprise Integration Scenarios: the GENIUS Tool. in Proc. KiVS, pp. 241-252, 2009.

[SML08] Scheibler, T.; Mietzner, R.; Leymann, F.: EAI as a Service - Combining the Power of Executable EAI Patterns and SaaS. In 12th IEEE International EDOC Conference, 15-19 September, Munich, Germany, 2008.

[SMM11] Silcher, S.; Minguez, J.; Mitschang, B.: Adopting the Manufacturing Service Bus in a Service-based Product Lifecycle Management Architecture, in: Proceedings of the 44th International CIRP Conference on Manufacturing Systems, Madison, Wisconsin, USA, 2011.

[SMS+10] Silcher, S.; Minguez, J.; Scheibler, T., Mitschang, B.: A Service-Based Approach for Next-Generation Product Lifecycle Management, 11th IEEE International Conference on Information Reuse and Integration, 219-224, 2010.

[SMT06] Schaffner, J., Meyer, H., Tosun, C.: A Semi-automated Orchestration Tool for Service-based Business Processes. In: Proceedings of the 2nd International Workshop on Engineering Service-Oriented Applications: Design and Composition, Chicago, USA, 2006.

[Sny91] Snyder, C. A., CIM networking: promise and problems. International Journal of Production Economics, 23, 205-212., 1991.

[Spe10] Speiser, S., Semantic annotations for WS-Policy, In: 17th. International Conference on Web Services, 2010.

[SSH08] Sahoo, S.; Sheth, A.; Henson, C: Semantic Provenance for eScience, IEEE Internet Computing, pp. 46-54, 2008.

[SSV+04] Sriharee, N., Senivongse, T., Verma, K., Sheth, A.: On Using WS-Policy, Ontology and Rule Reasoning to Discover Web Services. Springer, 2004.

[Sta] Stanford Center for Biomedical Informatics Research: Protégé Ontology Editor and Knowledge Acquisition System. - http://protege.stanford.edu/

[Sta02] Stal, M.: Web Services: Beyond Component-based Computng - Seeking a Better Solution to the Application Integration Problem. In Communications of the ACM, 2002.

[SUP09] SUPER - Semantics Utilized for Process Management within and between Enterprises, 2006-2009. - http://www.ip-super.org

[SWM+09] Smirnov, S., Weidlich, M., Mendling, J., Weske, M.: Action Patterns in Business Process Models. In: 7th International Conference on Service Oriented Computing ICSOC/ServiceWave, 2009.

[VDI06] VDI4499, Digitale Fabrik - Grundlagen. (Digital factory - basic concepts), 2006.

[VVK01] Van der Aalst, W.M.P.; Verbeek, H.M.W.; Kumar, A.: Verification of XRL: An XML-Based Workflow Language, in: Proceedings of the Sixth International Conference on CSCWin Design,NRC Research Press, Ottawa, Canada, pp. 427-432, 2001.

[W3C01] W3C: XML Schema, 2001. - http://www.w3.org/XML/Schema

[W3C04a] W3C: Web Services Architecture, 2004. - http://www.w3.org/TR/ws-arch/

[W3C04b] W3C: RDF Primer, 2004. - http://www.w3.org/TR/rdf-primer/

[W3C04c] W3C: OWL Web Ontology Language Reference, 2004. - http://www.w3.org/TR/owl-ref

[W3C04d] W3C: RDF Vocabulary Description Language 1.0: RDF Schema (RDFS), 2004. - http://www.w3.org/TR/rdf-schema/

[W3C05] W3C: Web Services Choreography Description Language Version 1.0, 2005. - http://www.w3.org/TR/ws-cdl-10/

[W3C06] W3C: Web Services Policy 1.2 - Framework (WS-Policy), 2006. - http://www.w3.org/Submission/WS-Policy/

[W3C07a] W3C: SOAP Version 1.2, 2007. - http://www.w3.org/TR/soap/

[W3C07b] W3C: Semantic Annotations for WSDL and XML Schema Ů Usage Guide, 2007. - http://www.w3.org/TR/sawsdl-guide/

[W3C10] W3C: XQuery 1.0 and XPath 2.0 Functions and Operators, 2010. - http://www.w3c.org/TR/xpath-functions

[W3C99a] W3C Recommendation: XML Path Language (XPath) Version 1.0. 1999. - http://www.w3.org/TR/1999/REC-xpath-19991116

[W3C99b] W3C: XSL Transformations (XSLT), 1999. - http://www.w3.org/TR/xslt

[WAL+07] M. Wei, I. Ari, J. Li, and M. Dekhil. ReCEPtor: Sensing Complex Events in Data Streams for Service-Oriented Architectures. Technical report, HP, 2007.

[Wal92] Waldner, J.B.: Principles of Computer-Integrated Manufacturing. John Wiley & Sons 1 edition (September, 1992), ISBN 0-471-93450-X, 1992.

[WBF08a] Worlf Batch Forum (WBF): Batch Markup Language (BatchML) v0401, 2008. - http://www.wbf.org

[WBF08b] Worlf Batch Forum (WBF): Business To Manufacturing Markup Language (B2MML) v0401, 2008. - http://www.wbf.org

[WCL+05] Weerawarana, S. ; Curbera, F. ; Leymann, F. ; Storey, T. ; Ferguson, D. F.: Web Services Platform Architecture. Prentice Hall, 2005.

[Wes06] Westkämper, E., 2006, Digital Manufacturing in the global Era, 3rd International CIRP Conference on Digital Enterprise Technology, Setúbal, Portugal, 2006.

[WJE+05] Westkämper, E; Jendoubi, L.; Eissele, M.; Ertl, T.: Smart Factory - Bridging the gap between digital planning and reality. In: Weingärtner, Lindolfo (Chairman); Proceedings of the CIRP 38th International Seminar on Manufacturing Systems, Florianopolis, Brazil, 2005.

[WMF+07] Wetzstein, B.; Ma, Z.; Filipowska, A.; Kaczmarek, M.; Bhiri, S.; Losada, S.; Lopez-Cobo, J.; Cicurel, L.: Semantic Business Process Management: A Lifecycle Based Requirements Analysis. In: SBPM07, Innsbruck, Austria, June 7, pp. 1-11, 2007.

[WMK+08] Wieland, M.; Martin, D; Kopp, O; Leymann, F.: Events Make Workflows Really Useful. Technical report, University of Stuttgart, IAAS, Germany, 2008. - http://www.informatik.uni-stuttgart.de/cgi-bin/NCSTRL/NCSTRL_view.pl?id=TR-2008-09&engl=1

[WMK+09] Wieland, M., Martin, D., Kopp, O., Leymann, F.: SOEDA: A Methodology for Specification and Implementation of Applications on a Service-Oriented Event-Driven Architecture. In: Proceedings of the 12th International Conference on Business Information Systems, 2009.

[WRR08] Weber, B.; Reichert, M.; Rinderle-Ma, S: Change patterns and change support features - Enhancing flexibility in process-aware information systems. Data Knowl. Eng. 66, 3, 2008.

[WSI04] Web Services Interoperability (WS-I) Basic Profile, 2004. - http://www.ws-i.org/Profiles/BasicProfile-1.1-2004-08-24.html

[WYM11] Wang, Y.; Yan, H.; Meng, X.: Matching Decision Model for Self- adaptability of Knowledge Manufacturing System. In International Conference on Information Science and Technology, Nanjing, Jiangsu, China, March 26-28, 2011.

[WZ09] Westkämper, E., Zahn, E., (Hrsg.), Wandlungsfähige Produktionsunternehmen, Das Stuttgarter Unternehmensmodell (The Stuttgart Enterprise Model), Springer, Berlin, 2009.

[YSG99] Yusuf, Y.Y.; Sarhadi, M.; Gunasekaran, A.: Agile manufacturing: The drivers, concepts and attributes. In Int. J. Production Economics 62 (1999) 33-43, 1999.

[ZTZ07] Zhao, J.L.; Tanniru, M.; Zhang L.: Services computing as the foundation of enterprise agility: Overview of recent advances and introduction . In Inf Syst Front (2007) 9:1-8, 2007. to the special issue

Curriculum Vitae

Jorge Mínguez

Date and place of birth:	March 25th, 1983; Madrid, Spain
Nationality:	Spanish

09/1989 – 06/1997	Primary School at Colegio Público Joaquín Costa in Madrid, Spain
09/1997 – 06/2001	Secondary School at IES Cervantes Madrid, Spain
09/2001 – 09/2007	Dipl.-Ing. Telecommunication Engineering Universidad Politécnica de Madrid, Spain
10/2004 – 9/2005	Software Development Internship Solar Decathlon 2005, Madrid, Spain
10/2005 – 10/2007	M. Sc. INFOTECH (Information Technology) Universität Stuttgart, Germany
09/2006 – 03/2007	Software Development Internship Hewlett-Packard GmbH, Böblingen, Germany
04/2008 – 03/2012	Doctoral Candidate at the Graduate School of Excellence advanced Manufacturing Engineering, Universität Stuttgart, Germany