MASTER DATA INTEGRATION

Managing product data exchange in an electronic commerce environment

Aleksander Śmierciak

Bachelor's Thesis

May 2013

Degree Programme in Software Engineering

Technology, Communication and Transport

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

Description

Author	Type of publication	Date
ŚMIERCIAK, Aleksander	Bachelor's thesis	2013-05-17
	Pages 48	Language English
	Confidential ( ) until	Permission for Web publication (X)

Title
MASTER DATA INTEGRATION
Managing product data exchange in an electronic commerce environment

Degree Programme
Software Engineering

Tutor
SALMIKANGAS, Esa

Assigned by
Descom Oy

Abstract
This thesis aims to analyse how data is exchanged between two specialized business branches: product information management and electronic commerce, putting emphasis on the former's point of view.
As the topic explicitly concerns data storage, data flow and data processing, it touches the very heart of the definition of information technology. Gaining valuable knowledge and experience is expected in the process of writing and concluding research into this topic.
The basis for the thesis stems from an internship at a company specializing in e-commerce, where the product information management team encounters issues described in this thesis.
The company project that constitutes for the basis of such elaboration requires two different software suites to cooperate by means of data exchange integration.
Particular topics of interest include what measures are taken to provide the webstore side with data prepared and transferred from the information management side, what are the possibilities for creating generic and reusable models and export format templates, lastly, how extensive the automation of otherwise human effort is possible in the data onboarding part.
Conclusions reached are contained in a summary of integration architecture for one-way data transfer and an overview of at least partial reuse in similar projects.

Keywords
application, WWW, e-commerce, product information management, enterprise, data exchange, integration

Miscellaneous

CONTENTS

FIGURES

ACRONYMS

API	Application Programming Interface
B2B	Business-to-Business
B2C	Business-to-Consumers
BOD	Business Object Document
CSV	Comma Separated Values [file]
DSL	Domain Specific Language
GUID	Globally Unique Identifier
IDE	Integrated Development Environment
IT	Information Technology
HPM	Heiler Product Manager
OAGi	Open Applications Group, Incorporated
OAGIS	Open Applications Group Integration Specification
PIM	Product Information Management
RCP	[Eclipse] Rich Client Platform
UI	User Interface
URI	Uniform Resource Identifier
W3C	World Wide Web Consortium
WSC	[IBM] WebSphere Commerce
XML	Extensible Markup Language
XSD	[W3C] XML Schema Definition

1 REUSABLE AND ROBUST INTEGRATION

1.1 Foreword

Data exchange is a hot topic for every enterprise, regardless of its size or structure. In fact, the more sophisticated a project or software suites used are, the more data exchange issues will surface.

No matter the surface of consideration – be it enterprise-level, team-level or project-level – data is exchanged only with existing protocols and specifications that strictly describe its format, or schema. Furthermore, because of impracticality of a unified data model concept stemming from the expensive complexity of resulting software, data integration is considered optional and out of the enterprise software development scope. The act of "gluing" acquired software suites in an attempt to automatize and ease refining development is known as enterprise integration.
(Enterprise Integration Patterns webpage, 2012)

While different branches of business require different approaches to address the specifics of their environment, the enterprise integration is likely to be the one characteristic unifying them all. It has garnered a significant momentum in the business world ever since the 90s (IFIP/IFAC Task Force, 1996). It tackles topics such as electronic data exchange and interconnection between systems. This technical field is strictly connected with software engineering since its early days.

The basis for this thesis is a webstore and information repository project for the company's client. Since both the former and the latter belong to different industry IT types, the software suites in which they are developed are not compatible to the point of deeming enterprise integration obsolete; on the contrary, an integration layer needs to be tailored specifically.

1.2 Project goal

The issue at hand is a need to have the answer to a question: how to provide ready-to-use complete data in one software suite, coming from another, when both use different data formats and communicate differently.

The solution is to prepare a format in which data can be transferred from one suite to another. This solution can then be later used for other company projects, as reusability and elasticity are given high priority from the beginning.

The work requires involvement from both parties in varying degrees of effort and time, but essentially, no integration can be done without strict cooperation. As the author of this work is part of the team responsible for information management, much attention is given to processes and actions necessary to undertake from this development team side.

Topics of importance include:

- preparing the output of information repository and analysing both the present and future needs imposed on export data,
- creating templates for data output in accordance to both team findings,
- performing data validation before the transfer,
- performing automation of data transfer task.

The goal is to have an integration component fitting in every similar enough project in the company, especially where two software suites used: Heiler Enterprise Product Information Manager and IBM WebSphere Commerce play major role. This would cut down the costs of producing software solutions, both in terms of human resources and material resources.

2 MEANS OF ENTERPRISE DATA EXCHANGE

2.1 Introduction

To begin discussion on how data is exchanged and lay the foundations for enterprise integration overview, at least some theoretical knowledge is needed on topics such as how data is stored. In enterprise level software, the popularity and reliability of extensible markup language is widely recognized and will be the starting point here.

2.2 W3C's Extensible Markup Language

Extensible Markup Language, **XML** in short, is a universal markup language designed to represent data in a format easily readable both by human and machine. Its definition as a standard is contained in the XML 1.0 specification (among others) produced by the World Wide Web Consortium (W3C). (W3C XML recommendation, 2008)

Concurrently to XML standard 1.0, another one – 1.1 – was being developed; the newest instalment of the former being the sixth release from 2008, while of the latter being the second release from 2006. (W3C XML recommendation, 2008)

As stated by W3C in XML 1.0 standard specification, the language's design goals were usability across the Internet, generality, flexibility and absolute simplicity. These goals are to be reached by employing open standards for character sets and identification tags, such as Unicode and Internet Assigned Numbers Authority's Language Subtag Registry. (W3C XML recommendation, 2008)

To date, XML is widely used for data exchange in various software development branches, particularly in Web services (XML Coverpages, n.d.).

The code snippet below presents a sample XML message (W3Schools Examples and tutorials, n.d.).

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

<shiporder orderid="889923"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:noNamespaceSchemaLocation="shiporder.xsd">
  <orderperson>John Smith</orderperson>
  <shipto>
    <name>Ola Nordmann</name>
    <address>Langgt 23</address>
    <city>4000 Stavanger</city>
    <country>Norway</country>
  </shipto>
  <item>
    <title>Empire Burlesque</title>
    <note>Special Edition</note>
    <quantity>1</quantity>
    <price>10.90</price>
  </item>
  <item>
    <title>Hide your heart</title>
    <quantity>1</quantity>
    <price>9.90</price>
  </item>
</shiporder>
```

2.3 XML schema

An **XML schema** is a description of syntactical constraints extending the very basic ones imposed by XML itself. Schemas are used for additional validation of documents, ensuring integrity and structural safety; however most commonly they don't guarantee that the data is sensible. (O'Reilly XML.com, 2001)

XML schemas consist of a set of **schema components**; those, in turn, are the effect of processing sets of **schema documents**, containing the actual written definitions. (O'Reilly XML.com, 2001)

To express said constraints, XML schemas contain extended markup and grammar as well as Boolean predicates, ordering rules, data types rules and uniqueness/integrity rules tag properties. (O'Reilly XML.com, 2001)

Native schema language for XML is called the Document Type Definition (DTD) language, while other more widely used languages are, most notably, XML Schema (or W3C XML Schema) and RELAX NG. (O'Reilly XML.com, 2001)

The code snippet below presents a schema sample written in XML Schema language, more thoroughly described in the next sub-chapter (W3Schools Examples and tutorials, n.d.).

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="shiporder">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="orderperson" type="xs:string"/>
      <xs:element name="shipto">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="name" type="xs:string"/>
            <xs:element name="address" type="xs:string"/>
            <xs:element name="city" type="xs:string"/>
            <xs:element name="country" type="xs:string"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="item" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="title" type="xs:string"/>
            <xs:element name="note" type="xs:string" minOccurs="0"/>
            <xs:element name="quantity" type="xs:positiveInteger"/>
            <xs:element name="price" type="xs:decimal"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="orderid" type="xs:string" use="required"/>
  </xs:complexType>
</xs:element>

</xs:schema>
```

2.4 W3C XML Schema language

XML Schema is an XML schema language published by World Wide Web Consortium in May 2001. Due to its name being easily confused with the more general term “XML schema”, another name has been adopted by W3C in version 1.1 of this schema specification, that is **XML Schema Definition** (XSD). (W3C XML Schema recommendation, n.d.)

Apart from allowing constraints expression, XSD is intended by design to be able to determine document's validity by producing an infoset – a collection of information concerning specific data types found in it. (W3C XML Schema recommendation, n.d.)

XML Schema specification is available on W3C webpages and is divided into two parts, one describing the structures, the other describing the datatypes. (W3C XML Schema recommendation, n.d.)

2.5 XML namespace

Extensible Markup Language namespace is a unique name dictionary for elements and attributes in an XML document.

Namespaces are named after uniform resource identifiers (URIs) for the resources owned by the subject defining the namespace vocabulary, but it is not necessary for the URIs to point directly to specification; on the contrary, URIs are not to be parsed by XML parsers differently than standard strings. Usage of URIs (most commonly long WWW addresses) offering human readers concise information, however, lowers the risk of naming conflict. (W3C XML Namespace recommendation, 2009)

In the previous two drawings, namespaces are specified with `xmlns` attributes. Their elements are prefixed with adequate shortcuts when used later on. (W3C XML Namespace recommendation, 2009)

2.6 Open Applications Group Integration Specification

The **Open Applications Group Integration Specification** (OAGIS) is a complete standard specification for business information integration through a uniform way of communication with XML as the language for defining business messages. (Open Applications Group incorporated website, n.d.)

OAGIS is being developed by the Open Applications Group, Incorporated (OAGi). This organization started as a grouping of ERP vendors agreeing on the incompatibility problem and focusing on providing common content format. (Open Applications Group incorporated website, n.d.)

Acknowledging the unique vocabulary set connected with each and every vertical industry (a single industry type with all its products and services), OAGi intends to create a horizontal (across industries) unification by cooperating and discussing with vertical industry groups. (Open Applications Group incorporated website, n.d.)

Examples of such groups with whom OAGi is already engaged in cooperation include various automotive standards bodies, human resources, chemical and aerospace. Other than those, OAGi is recognized by other standard bodies and, most notably, by the Memorandum of Understanding Management Group of the four recognized standards bodies in the world: (Open Applications Group incorporated website, n.d.)

- International Electrotechnical Commission (IEC),
- International Organization for Standardization (ISO),
- International Telecommunication Union (ITU),
- United Nations Economic Commission for Europe (UN/ECE).

Support for OAGIS was and is constantly added in new implementations by vendors, consulting companies and users. (Open Applications Group incorporated website, n.d.)

The Vendor Challenge Event back in November 2001 is possibly the culmination of OAGi members' efforts to popularize and widen OAGIS usage; while three different scenarios for integration were presented, 21 OAGi members demonstrated their products complying with OAGIS documentation by integrating with other vendors' applications in at least one of the scenario cases. (Open Applications Group incorporated website, n.d.)

This event provided case study empirical data and encouraged new vendors to consider integrating OAGIS by allowing every one of them to realize a large return-on-investment coming from the fact of using a canonical business language. (Open Applications Group incorporated website, n.d.)

The OAGIS comes in three versions known as editions. With version 9.6 of this specification, editions can be summarized as follows (Open Applications Group incorporated website, n.d.):

1. **Community Edition**

   This Edition contains 586 standalone BODs as well as sample instances of them; in other words, content, but not architecture. It is also the only free edition, the others being paid for.

2. **Platform Edition**

   This edition contains 1 BOD (Confirm BOD), all Components definitions, architecture for BOD and messaging as well as Meta Model; in other words, architecture, but not content.

3. **Enterprise Edition**

   This edition contains both the architecture and the content, as described in the other two editions.

OAGIS standard essentially provides a Canonical Model approach to data exchange. OAGIS Business Object Documents are validated by XSD schemas. (Open Applications Group incorporated website, n.d.)

2.7 Business Object Document

The **Business Object Document** (**BOD**) is an open standard developed as part of OAGIS to unify horizontal message architecture among its implementers; a BOD is a multi-area message with nested elements that serves as means of exchanging business data.

OAGIS provides, along with BOD specification, a collection of example integration scenarios that can be used as templates.

A graphical presentation of what BOD message is structured like is found in Figure 1 (below). (Open Applications Group incorporated OAGIS specification, n.d.)

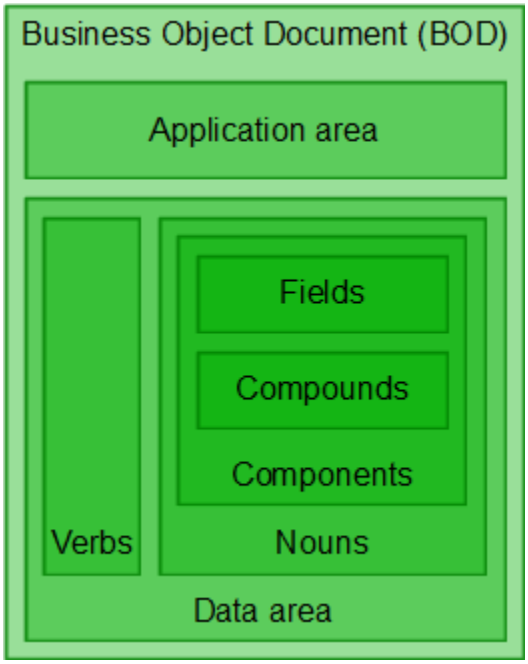

FIGURE 1: Business Object Document structure

BOD wrapper contents

Each BOD contains four attributes in addition to its ApplicationArea and DataArea children (Open Applications Group incorporated OAGIS specification, n.d.). These are data vital to quick recognition of a BOD message, namely:

1. **Release ID**, which identifies the OAGIS release as a number and is a required attribute,
2. **Version ID**, which identifies the BOD version (its level, as documented in OAGIS) and is an optional attribute,
3. **System Environment Code**, which identifies a BOD as belonging to either production level or test level integration and is an optional attribute,
4. **Language Code**, which identifies the data language of a BOD and may be overridden for particular fields with multi-lingual purposes.

Application area contents

The **Application area** (written in the specification as **ApplicationArea**) takes part in application-to-application integration by containing information vital to identifying a BOD and authenticating the sender (Open Applications Group incorporated OAGIS specification, n.d.).

Application area elements are shown on the Figure 2 (below). (Open Applications Group incorporated OAGIS specification, n.d.)

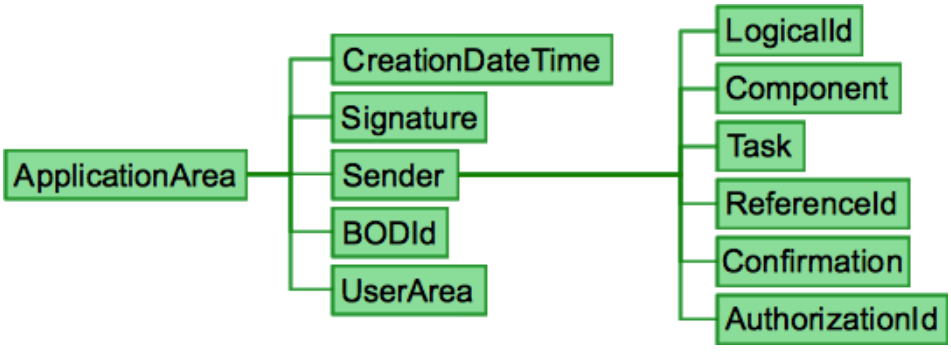

FIGURE 2: BOD structure, ApplicationArea contents

The elements are: (Open Applications Group incorporated OAGIS specification, n.d.)

1. **Sender** defines the source of BOD message and is described deeper in the next paragraph.
    a) **LogicalId**, which specifies location of the server and application of origin of a BOD and is optional.
    b) **ComponentId**, which specifies the exact application of origin of a BOD (note: LogicalId specifies the location only) and is optional.
    c) **TaskId**, which identifies an event behind the need for a BOD creation and is optional.
    d) **ReferenceId**, which identifies an event or task being the source of a BOD creation for the receiving application.
    e) **ConfirmationCode**, which is a request coming from the sender application to the receiving application to send a confirmation BOD back. This confirmation BOD will indicate the outcome of processing, either success or failure, in which case it will contain error conditions.
    f) **AuthorizationId**, which specifies the terminal device user used to cause the creation of a BOD.
2. **CreationDateTime** is a date and time value of a BOD creation.
3. **Signature** is included in case a BOD is to be signed. Any digital signature type supported by OAGIS is allowed.
4. **BODID** is a Globally Unique Identifier (GUID), assuring each BOD is uniquely identifiable.
5. **UserArea** is a special (optional) field that tells a BOD contains user's unique data.

Data area contents

The **Data area** (written in the specification as **DataArea**) contains at the basic level both a (single) Verb and a Noun or multiple Nouns. In case of providing multiple Nouns, each one is subject to action defined by the Verb (Open Applications Group incorporated OAGIS specification, n.d.).

The Figure 3 (below) shows Data area structure (Open Applications Group incorporated OAGIS specification, n.d.). It is further described below.

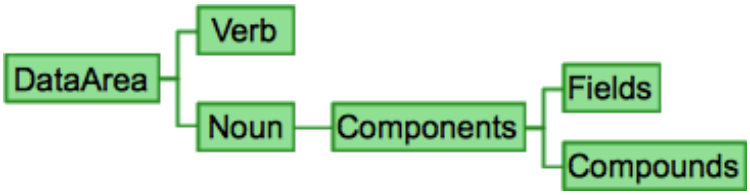

FIGURE 3: BOD structure, DataArea contents

The consecutive elements are:

1. **Verb** specifies an action that is to be applied to the object. Verb contains all kind of information that might be exclusively needed by the action (the rest being contained by the noun).
2. **Noun** specifies an object or document to which an action is to be applied. Noun contains all kind of information that might be needed by any of applicable actions. They are component-based and extensible.
3. **Components** are the basic blocks for Nouns. The elements that a single component may contain are: compound, field or another component.
4. **Compounds** are contextually grouped fields and as such they are generic for all BODs.
5. **Fields**, being the lowest-level building blocks of BOD structure, are used to create Compounds and Components. On the other hand, they themselves can be created either with OAGIS-defined or user-defined templates.

2.8 Modular and plugin-based approach to development

Apache Camel

Apache Camel is a framework focusing on simplifying the process of integration. Camel performs rule-based routing and mediation and is configurable through an API (with Spring through XML configuration files) or Java Domain Specific Language (DSL).

As Camel was intended by its creators to be as flexible and robust as possible, not a single messaging model or data format is favoured; instead, there are many models and formats supported directly. (Apache Software Foundation Camel webpage, n.d.)

To address Endpoints and Routes, **Java Domain Specific Language** usage is recommended. It is available by extending the RouteBuilder class and implementing its configure method. (Apache Software Foundation Camel webpage, n.d.)

Components implement the Factory design pattern and serve to spawn instances of Endpoint. Components can be divided into the standard ones and external ones, the latter being licensed differently, thus not in the specification.

Route is basically a connection between two endpoints. Routes can be created simply – just by specifying two Endpoints – and more sophisticatedly – by using string formatters, filters, predicates list, custom processors or interceptors (Apache Software Foundation Camel Core API, n.d.).

Endpoint is a single end of a Route; it embraces a role of producer, consumer or both of them at the same time. Endpoint is an Apache Camel implementation of the Message Endpoint enterprise integration pattern (Hohpe, Woolf, Enterprise Integration Patterns, 2007).

Uniform Resource Identifiers (URIs) play major role in specifying endpoints within Routes. Like Components, they can also be divided into standard and external.

3 ENTERPRISE SOFTWARE ECOSYSTEM

3.1 Introduction

The following chapter is dedicated to describing more software-oriented terms and definitions, as well as familiarizing the reader with the subject of integration – the electronic commerce and product information management software.

3.2 Product information management

Definition

Product information management is a term used to describe a range of technologies, processes and specifications, all of which focus on centralizing the management of information about products. The main area targeted is distribution and marketing, hence the emphasis put on managing information required by sales and distribution channels. (SapientNitro, n.d.)

PIM systems and software usually involve reliable support for internationalization and localization, such as: multiple geographic locations, multilingual data, multinational currency sets and so on. Those characteristics are necessary especially when discussing the international expansion of products or service providers of many different kinds. (SapientNitro, n.d.)

The Figure 4 (below) presents typical usage of product information management software. (Heiler Software AG resource center, n.d.)

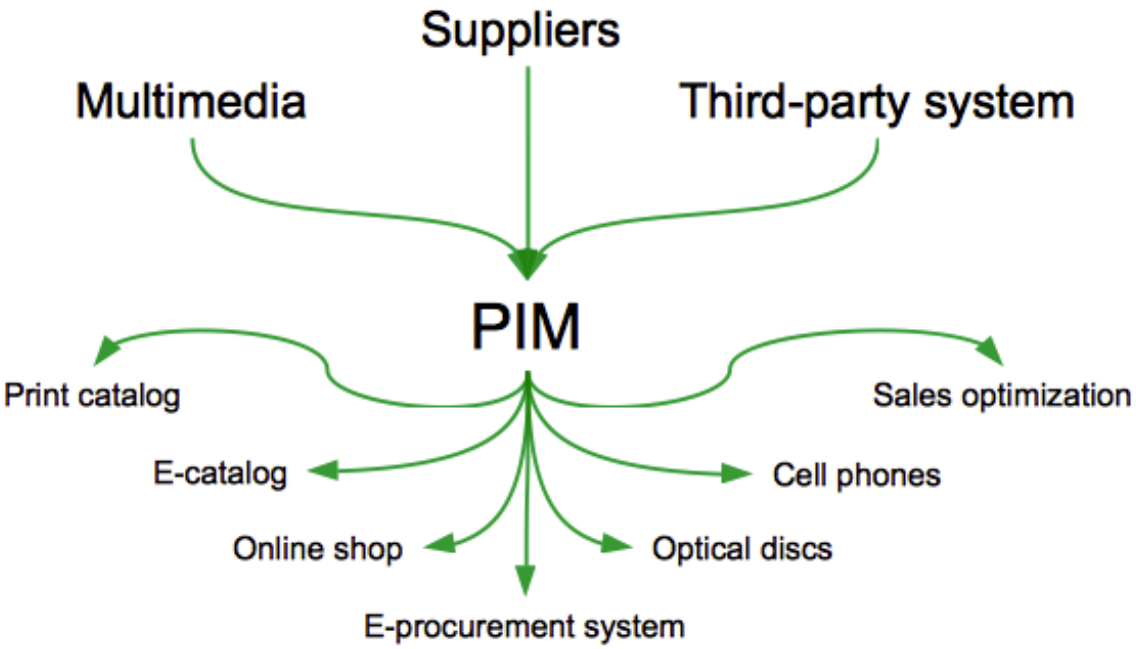

FIGURE 4: Product information management data flow overview

Role of PIM in B2B, e-commerce

PIM systems provide an electronic catalog for information on catalog entries, be it products or items (essentially product variants). As such, a wide array of choices for further use is available, including any type of electronic commerce, as presenting an offer becomes a task of supplying a user interface – data model connection only. This can be any type of website, document, media object or a whole e-commerce solution like a webstore.

Heiler Enterprise Product Information Management

Heiler Enterprise PIM is a software suite for centralized management of highly structured product data, both textual and in form of media assets. It is directed to medium and large companies dealing with retailing and manufacturing. (Heiler Software AG resource center, n.d.)

Product Information Management in Heiler Software is centred around catalog entries (products and items) and their structural assignments (channels, catalogs, structures and their groups). Catalog entries are defined with data fields, either standard or custom ones, as the repository of product information is flexibly extensible. (Heiler Software AG resource center, n.d.)

According to the Ventana Research's Product Information Management Value Index (Ventana Research, 2012), Heiler software belong to a small group of market leaders with mature and high-quality value of what the software offers.

Heiler Enterprise Product Information Management is written in Java programming language and bases its foundation on the Eclipse Rich Client Platform. This is most apparent in the plugin system and the ability to extend base software like with almost any other Eclipse RCP-based program. (Heiler Software AG internal resources, n.d.)

Heiler Enterprise PIM vocabulary

As was said in the foreword, different branches of enterprise have different surroundings and vocabulary; in this, product information management is not an exception. Certain buzzwords exist that might have different meaning elsewhere, but have a strict definition when talking about managing information on products. (Heiler Software AG resource center, n.d.)

The vocabulary commonly used in Heiler PIM follows:

1. **Item**

   An item is a single, tradable and physically existing piece of merchandise. It is the most basic and direct abstract in an information repository. (Heiler Software AG resource center, n.d.)

2. **Product**

   A product is a more general abstract than an item; an item is connected to its single higher-level product definition in a many-to-one relation (there can be many items connected to one product). (Heiler Software AG resource center, n.d.)

3. **Catalog entry**

   A catalog entry is a more general term for what is a product, its variant or a product-dependent item. (Heiler Software AG internal resources, n.d.)

4. **Structure group**

   A structure group is an essential container for catalog entries. Structure groups are usually created to mimic items possible usage. (Heiler Software AG internal resources, n.d.)

5. **Catalog**

   A catalog is a set of structure groups. Catalogs recreate trade purposes (sales, purchasing, export and so on). In Heiler PIM there is always a master (global) catalog. (Heiler Software AG resource center, n.d.)

Heiler PIM export format templates

An export format template is a means of specifying the data format for an export action in Heiler PIM. Templates are, in fact, the only way of retrieving data in mass quantities from the information repository to use with other software. As such, default templates exist and are available for users of the Heiler Software products – those mainly target e-commerce systems and databases of Heiler partners and key market holders. Creating one's own custom template is also possible. (Heiler Software AG resource center, n.d.)

Export format templates are written in a quasi-language allowing use of functions (built-in and custom ones), modular design, data fields and template comments). (Heiler Software AG internal resources, n.d.)

Export format templates are separated into modules and submodules, which allows reuse of single parts of template code. As the software is based on Eclipse RCP, a contextual method of choosing the data fields just like in e.g. Eclipse Integrated Development Environment is present to speed up the process of developing. The collection of data fields that can be used may be vast, as it depends on the number of data fields in the extensible information repository. (Heiler Software AG internal resources, n.d.)

The results of carrying out an export are saved as one file or many files, depending on the export format template configuration. A developer may choose to export single data pieces separately from each other, while exporting other data pieces together. (Heiler Software AG internal resources, n.d.)

The file formats and character encodings are chosen individually for each file. Some supported file formats include: (Heiler Software AG internal resources, n.d.)

- plain text files,
- comma-separated values (CSV) files,
- HTML and CSS files for Web output purposes,
- CIF multimedia files,
- industry-specific, (mostly) XML-based data interchange formats:
  - cXML,
  - xCBL,
  - eCX,
  - BMEcat,
  - Datanorm,
  - Eldanorm.

Exact file(s) name(s) and export paths can also be chosen by the developer. Such precise configuration is needed especially for automating export process as a time-triggered or an action-triggered task. (Heiler Software AG internal resources, n.d.)

Heiler PIM data model

Heiler's approach to data architecture is to store the information in a relational database management system, such as Oracle Database or Microsoft SQL Server. This is similar to IBM's approach, only the data is later interfaced with use of an abstract called repository, allowing to perform extensions as deemed necessary by the developer. This makes an information repository particularly open to changes and end-point customization, but at the same time proves exporting data in unified formats difficult. (Heiler Software AG internal resources, n.d.)

3.3 Electronic commerce

Definition

Electronic commerce (e-commerce) is an industry type which focuses on using computer networks (such as the Internet) in order to buy or sell products or services. Nowadays, e-commerce is most commonly associated with the World Wide Web (WWW), though also with communication technologies such as e-mail, social media and telephones. (WikiBooks, n.d.)

A few elements can be isolated from each other when talking about e-commerce, namely: webstores (e-tailing), demographic data gathering and usage, electronic data interchange, e-mail and fax usage, B2B buying and selling and the transactions' security. (WikiBooks, n.d.)

Role of electronic commerce

Electronic commerce is widely believed to have a positive impact on consumers' information gathering ability regarding products and services they are looking for. Electronic commerce benefits big companies much more than it does with the small or medium ones, as their operational efficiency allows them to cut prices more, as an effect, to be more competitive. (The Economist, 2010)

Ultimately, electronic commerce is viewed as an industry which is to be considered by every company and not overlooked, especially with the constant growth of money flowing through e-markets as well as the rise of online shopping. (eMarketer, 2013) (Mashable, 2013)

IBM WebSphere Commerce

IBM WebSphere Commerce is an e-commerce software platform and also a framework for building web stores and offering electronic commerce. WebSphere Commerce targets all business models, both direct: business-to-consumer (B2C) and business-to-business (B2B) as well as indirect: through channel partners.

WebSphere Commerce is considered a market leader in e-commerce software (Gartner analysis, 2012). With the latest version 7, released on 2009 and receiving patches (called Feature Packs) ever since, IBM introduced mobile clients support as well as social media integration to an already existing PC clients support in its product (Internet Retailer portal, 2009).

IBM WebSphere Commerce data model

WebSphere Commerce is built on an SQL database foundation and the data model directly connects the software usage with database maintenance. (IBM Info Center, n.d.)

The data exchange model for WebSphere Commerce software suite is largely reliant on OAGIS BOD architecture. This is to ensure wide-range compatibility, as BOD messaging is considered complete in terms of data expression. (IBM Info Center, n.d.)

Many Data Load Utilities exist specifically for importing new data into the WebSphere Commerce; those are in fact the only way to perform this task. IBM Data Load Utility is recommended among all, being the official application. (IBM Info Center, n.d.)

IBM Data Load Utility

The **Data Load Utility** is an object-based loading utility made by IBM which purpose is to import information into the WebSphere Commerce database. Being a loading utility recommended by IBM to use with WebSphere Commerce, it supports different types of input data and is customizable. The procedure of configuring the Data Load Utility is depicted in Figure 5 below. (IBM Info Center, n.d.)

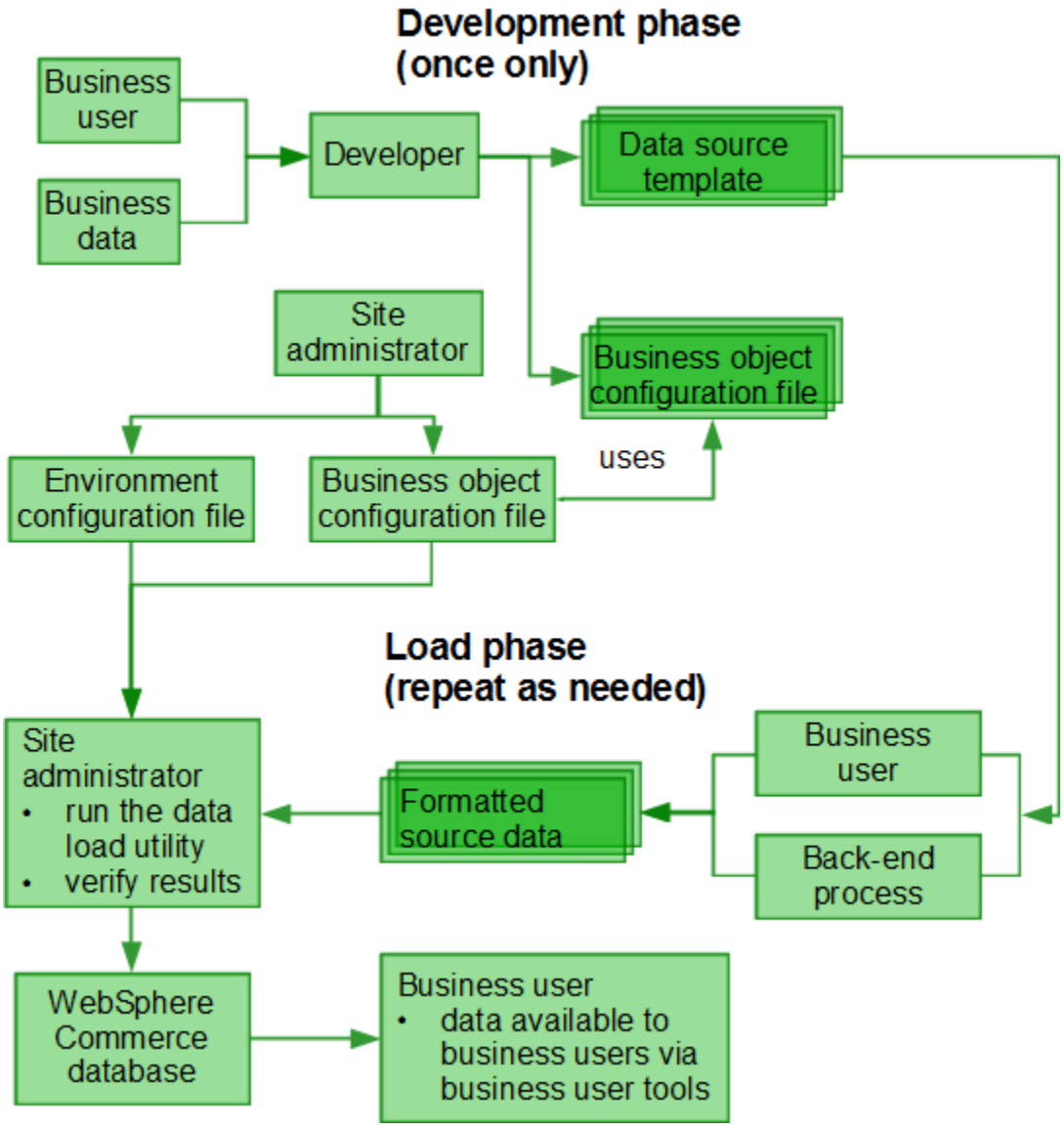

FIGURE 5: Data Load Utility configuration procedure

Development phase

As soon as the developer is provided with business data from business user, a data source template can be created. It defines source data formatting, not unlike XML schemas do for XML documents. A business object configuration file is also created, this in order to define input data mapping and transformations needed which are used to save business object as physical data.

Following those tasks, the site administrator uses the object configuration file to create the load order configuration file, as well as sets the database in the environment configuration file. (IBM Info Center, n.d.)

Load phase

Before the data is loaded to the database, it undergoes formatting according to the data source template rules, after which it is provided to the site administrator. The Data Load Utility is ran, supplied with all of the three configuration files and the formatted source data is inserted into the database. Post-run, a results verification is concluded by the site administrator; any errors are always found in Data Load Utility reports. The Utility can also be used to delete data from database. (IBM Info Center, n.d.)

3.4 Data exchange design patterns

Message Translator design pattern

The **Message Translator** design pattern is an analogy to the Adapter design pattern from the so-called “Gang of Four” group.

Message translation can occur at different levels (or rather layers) of communication: transport (protocol change), data representation (data format change with or without encryption), data types (fields and types changes) or data structures. (Hohpe, Woolf, Enterprise Integration Patterns, 2007)

The Figure 6 (below) presents an overview of the architecture of Message Translator.

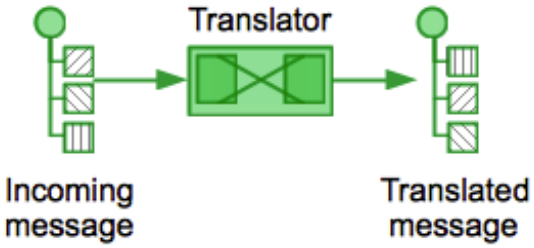

FIGURE 6: Message Translator design pattern architecture

The Figure 7 (below) shows the communication approach which Message Translator uses. This, called many-to-many connection model, trades growing complexity for the sake of simplicity with small number of participants.

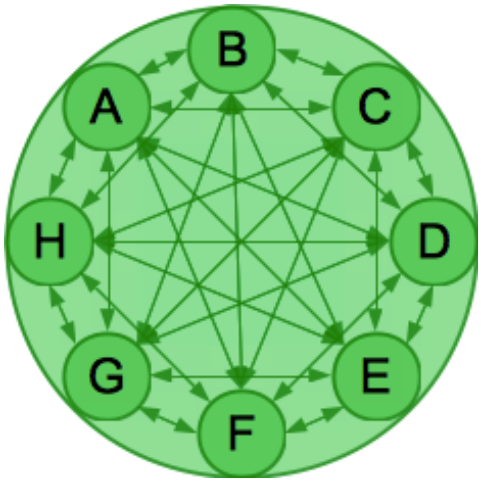

FIGURE 7: Many-to-many communication approach

Canonical Data Model design pattern

To robustly and solidly approach the issue of data incompatibility, an architectural design pattern named **Canonical Data Model** is proposed. Being an additional layer of communication, causing it to become more indirect between the applications in mention, the software implementing this pattern functions as a data format translator both back and forth. (Hohpe, Woolf, Enterprise Integration Patterns, 2007)

The Figure 8 (below) presents an overview of the architecture of Canonical Data Model.

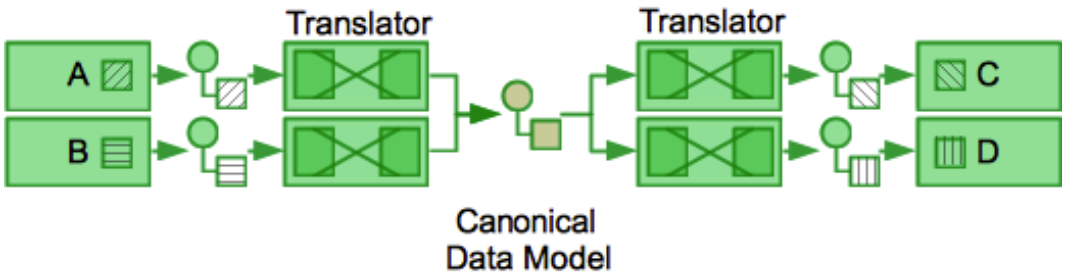

FIGURE 8: Canonical Data Model design pattern architecture

Canonical Data Model is built on many-to-one / one-to-many approach, as presented in the Figure 9 below.

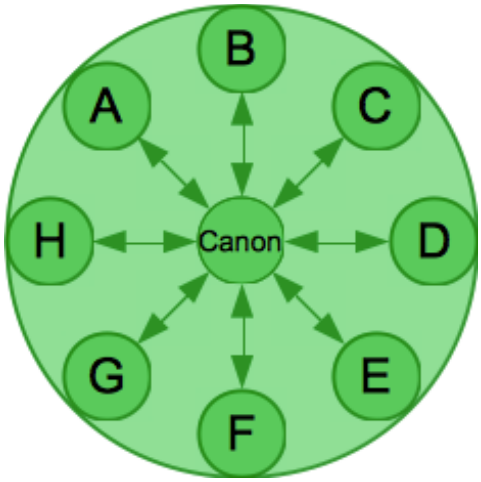

FIGURE 9: Many-to-one, one-to-many communication approach

4 INTEGRATION PROJECT SPECIFICATION

4.1 Integration design pattern choice

Canonical Data Model is a solution to **Message Translator** design pattern's lack of scalability. Whereas the latter is designed for small numbers of senders and receivers because of indirectness of communication, the Canonical Data Model gains significant advantage the bigger the number of nodes is.

The complexity of implementing inter-application exchange with Message Translator design pattern expressed as number of connections required to specify is dependent on the number of senders and receivers and can be written as:

$$\begin{aligned} number\ of\ connections_{MT} &= \left(number\ of\ senders + number\ of\ receivers\right) \times \left(number\ of\ senders + number\ of\ receivers - 1\right) \\ &= \left(number\ of\ senders + number\ of\ receivers\right)^2 - \left(number\ of\ senders + number\ of\ receivers\right) \end{aligned}$$

And, in a special case of all participants being senders and receivers at the same time:

$$number\ of\ connections_{MT} = number\ of\ participants \times \left(number\ of\ participants - 1\right)$$

Whereas the same for the Canonical Data Model design pattern can be written as:

$$number\ of\ connections_{CDM} = number\ of\ senders + number\ of\ receivers$$

And, in a special case of all participants being senders and receivers at the same time:

$$number\ of\ connections_{CDM} = 2 \times number\ of\ participants$$

To compare both solutions in terms of connections complexity, we search for the expressed below inequality's dependency on the number of participants:

$$number\ of\ connections_{CDM} < number\ of\ connections_{MT}$$

$$2 \times number\ of\ participants < number\ of\ participants \times \left(number\ of\ participants - 1 \right), \quad number\ of\ participants > 0$$

$$2 < number\ of\ participants - 1$$

$$3 < number\ of\ participants$$

When the number of participants (senders and receivers total) is higher than three, the Canonical Data Model begins to have a lower effort advantage over the Message Translator solution. For every data exchange participant added, its advantage grows even more significant.

Consequently, if the number of participants is predicted to be more than three or change dynamically, the Canonical Data Model solution seems a better option.

On the other hand, the complexity of the Canonical Data Model solution is further made larger by bringing another layer – the canonical model layer – into consideration. The Message Translator solution, while complex in implementation and maintenance, is based on direct data exchange with only specific data modifications to meet the requirements of the receiver.

Ultimately, as the data exchange is one-way only and no other participants than the sending Heiler PIM and receiving IBM WebSphere Commerce are expected, the Message Translator solution was chosen. As has been proved, there are no advantages in such situation for Canonical Data Model solution.

4.2 Architecture overview

The simplest visualisation of the project goal is to transfer data from Heiler PIM information repository to IBM WebSphere Commerce database (Figure 10, below).

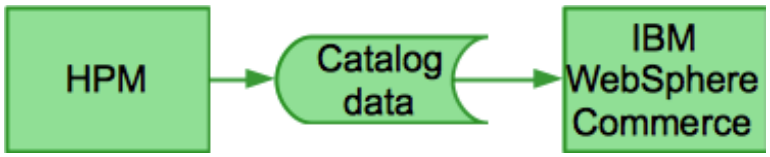

FIGURE 10: Adapter project architectural overview no. 1

Such straight-forward implementation is, of course, impossible due to the incompatibilities between the software suites. The process of data exchange needs to be divided into data export [from Heiler PIM], data transfer [to Data Load Utility] and data import [to IBM WSC]. The corrected overview reflects the actual operations (Figure 11, below).

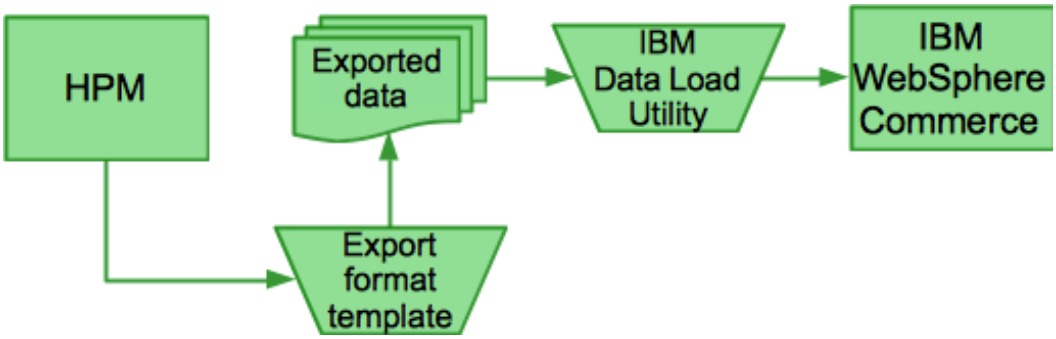

FIGURE 11: Adapter project architectural overview no. 2

This is further expanded by separating tasks and delving into the export mechanism. The process of creating export format templates (by a Heiler PIM developer) and the process of configuring Data Load Utility (by an IBM WSC developer) need also be presented for the full picture.

Since some operations are one-time only (configuring and writing templates, testing) and others are to be conducted automatically on every new data piece that appears, the final overview is presented on Figure 12 (below).

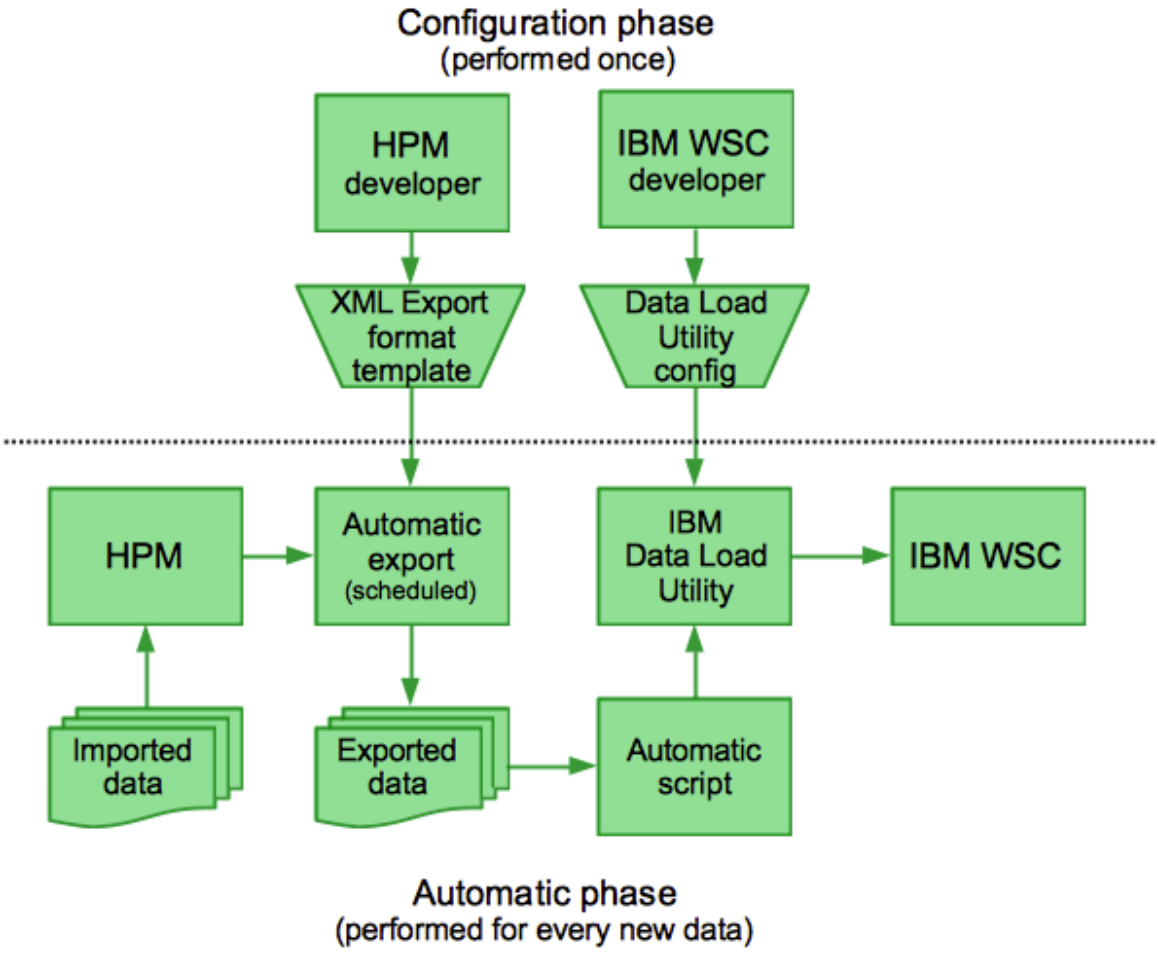

FIGURE 12: Adapter project architectural overview no. 3

The middle part of automatic phase, that is sending the data from Heiler PIM to IBM WSC with means of data export and import, is implementing the Message Translator integration design pattern. Apache Camel is being used as a post-processing tool; it checks for any new data in a given directory, provides middle-point validation and sends the compressed data via SFTP.

4.3 IBM WebSphere Integration templates choice

As Heiler and IBM are both leaders of their own markets, mutual recognition has been noted between them. Heiler products bear IBM recommendation for assured compatibility. Due to commonness of a solution incorporating both Heiler products and IBM WebSphere Commerce information front-end, Heiler provides export format templates bundle for data output tasks, called IBM WebSphere Integration templates. Those feature: (Heiler Software AG internal resources, n.d.)

- support for data retrieval, including:
  - items,
  - products,
  - catalogs,
  - structure systems,
  - structure groups,
  - digital assets,
  - data (prices, logistical data, references),
- product – item relationship compatibility with variant mapping of user's e-commerce solution,
- customizable export format templates themselves,
- integrated QA of channels, including validating to the point of data approval in the e-commerce system.

The export format templates made by Heiler are not used in the project that this thesis concerns. The reasons include, but are not limited to what follows:

1. More fine-grained corrections and custom modifications are possible when creating export format templates from the scratch – unused features would lower visual clarity for developers;
2. At the point of making the decision, the WebSphere Integration templates only supported CSV file format. XML file format was preferred instead, as it allowed better integration into the software tailored for this solution;
3. Different Data Load utility is in use, different methods of post-processing are used.

4.4 Expected data and its format

IBM WebSphere Commerce side expects to receive extensive information on master and sales catalogs and their entries, that is products and items. These information need to be initially validated, as there are constraints imposed by IBM WebSphere Commerce not apparent in Heiler PIM software. Those include:

1. **Catalog entry structure group assignment**

   Each product and each item must be assigned to an existing structure group.

2. **Item product relationship**

   Each product must have at least one item assigned. Each item must be assigned to a higher-level product.

3. **Structure group parent relationship**

   Each structure group must have its parent specified. The only exception is a top-level structure group; in its case the parent (which would be the structure itself) must not be specified.

4. **Structure group features specification**

   Structure group features need to be specified in a separate data source called attribute dictionary. This file, belonging to a catalog as a whole, must list all structure group features and the groups they belong to.

5. **Catalog entry attributes specification**

   Attributes of each product and each item need to be specified along them.

6. **Multimedia attachments placement**

   Multimedia files (such as images) must be transported separately to where IBM WebSphere Commerce expects them to be located. This is not strictly an export format template job, so it will not be described in this thesis.

4.5 In-depth look at export format template

Description by example

The whole export format template used in the project concerns not only products, but also their items and structure groups, all of which are intertwined together. To help in understanding templates, their syntax and their abilities, an example template part for Products is presented in fragments below.

For the list of buzzwords and commonly used technical words, see subchapter Heiler Enterprise PIM vocabulary on page 20.

```
<?xml version="1.0" encoding="UTF-8"?>
<des:Push catalog="CATALOG_NAME" xmlns:des="XMLNS_DES_URI">
    <_cat:CatalogEntry catalogEntryTypeCode="PRODUCT"
        xmlns:_cat="http://www.ibm.com/xmlns/prod/commerce/9/catalog"
        xmlns:_wcf="http://www.ibm.com/xmlns/prod/commerce/9/foundation"
        xmlns:oa="http://www.openapplications.org/oagis/9">
...
```

As XML format was chosen, the template opens with an XML version and encoding specification. UTF-8 encoding is chosen here, as it is the recommended encoding for multi-language, multi-alphabet texts.

The following tag (`des:Push`) opens the actual data area, with its first attributes specifying the namespaces that this document will draw from. Those mostly contain OAGIS BOD elements and WebSphere Commerce-specific elements.

It is also specified here what kind of data this document contains. WebSphere Commerce element `_cat:CatalogEntry` tells the parser that it is a catalog entry, while the attribute that follows specifies it further – it is a product (not an item).

```
...
<_cat:CatalogEntryIdentifier>
    <_wcf:ExternalIdentifier>
        <_wcf:PartNumber>PRODUCT_ID</_wcf:PartNumber>
    </_wcf:ExternalIdentifier>
</_cat:CatalogEntryIdentifier>
...
```

All products are identified by a unique number or an alphanumeric character string both in Heiler PIM information repository and in IBM WebSphere Commerce database. Thus the first data fragment in a product template specifies this identifier by which references are created later on.

```
...
<_cat:Description language="LANGUAGE_ID">
    <_cat:Name>PRODUCT_NAME(LANGUAGE)</_cat:Name>
    <_cat:Thumbnail>PRODUCT_THUMBNAIL_PATH(LANGUAGE)</_cat:Thumbnail>
    <_cat:FullImage>PRODUCT_FULLIMAGE_PATH(LANGUAGE)</_cat:FullImage>
    <_cat:ShortDescription>PRODUCT_DESCRIPTION_SHORT(LANGUAGE)</_cat:ShortDescription>
    <_cat:LongDescription>PRODUCT_DESCRIPTION_LONG(LANGUAGE)</_cat:LongDescription>
    <_cat:Keyword>PRODUCT_KEYWORDS(LANGUAGE)</_cat:Keyword>
    ...
</_cat:Description>
...
```

All text data and also some number data is bound to a specific language. Differences can also be seen on pictures or other media formats, in which case we need to supply a complete language-specific data set for a product. The following part is repeated for every language data exists in. As a product is specified by a myriad of different data fields and attributes, only a handful is presented below. Some other language-specific data might include prices in different currencies, availability in regions, product segment, logistics and so on.

```
...
<_cat:CatalogEntryAttributes>
    <_cat:Attributes usage="ATTRIBUTE_USAGE" displaySequence="ATTRIBUTE_SEQUENCE"
        language="ATTRIBUTE_LANGUAGE">
        <_cat:AttributeIdentifier>
            <_wcf:ExternalIdentifier>
                <_wcf:Identifier>ATTRIBUTE_ID</_wcf:Identifier>
            </_wcf:ExternalIdentifier>
        </_cat:AttributeIdentifier>
        <_cat:Name>ATTRIBUTE_NAME(LANGUAGE)</_cat:Name>
        <_cat:Description>ATTRIBUTE_DESCRIPTION(LANGUAGE)</_cat:Description>
        <_cat:AttributeDataType>ATTRIBUTE_DATATYPE</_cat:AttributeDataType>
        <_cat:Value>ATTRIBUTE_VALUE(LANGUAGE)</_cat:Value>
    </_cat:Attributes>
    ...
</_cat:CatalogEntryAttributes>
...
```

The next part lists all attributes connected with a product. Those are data restrained to a specific product type and not necessarily applicable to the products pool as a whole; a good example might include a catalog of electrical equipment, some of which are active devices while others passive ones. The attributes for the former might include operating voltage or battery requirements, which may make no sense for the latter. On the other hand, the latter might be described with such attributes as parts-related technical sizes, wiring veins quantity and these will be ambiguous when applied to more complex devices.

The attribute names are repeated for every language found within the product. A data type necessary for database insertion is also specified herein.

```
...
<_cat:ParentCatalogGroupIdentifier>
    <_wcf:ExternalIdentifier>
        <_wcf:GroupIdentifier>PRODUCT_PARENTGROUP_ID</_wcf:GroupIdentifier>
    </_wcf:ExternalIdentifier>
</_cat:ParentCatalogGroupIdentifier>
...
```

Product is contained in one or more groups of a structure. The fragment below references a parent group (or groups) for every product exported with this template.

As product is obligatory placed in a structure group, so does item. The item part of this export format template also contains the parent group identifier (or identifiers for many relationships).

```
...
<_wcf:UserData>
    <_wcf:UserDataField name="HasItems">PRODUCT_HAS_ITEMS</_wcf:UserDataField>
</_wcf:UserData>
...
```

The last part of a template specifies whether a product has at least one item in the repository. It is only sensible to insert products with their item counterparts, as no product entry is complete without actual items. This is one of the constraints imposed by the IBM WebSphere Commerce system.

```
...
    </_cat:CatalogEntry>
</des:Push>
```

The template's product part ends with closing tags, in accordance to XML.

The aforementioned item template part is similar to the product one. The only one type left to describe is the structure group template part, which allows for structure hierarchy to be recreated on the IBM WebSphere Commerce part. This is in most cases analogical, even with the parent group identifier, as groups are hierarchical and form a data structure known as tree.

Automation of export format template usage

As the previous sub-chapter tackled the topic of export format template contents and usage, it also mentioned the process of automation. Indeed, it is a desirable way of data transfer to do so without constant human supervision.

Export automation requires setting two features of export format template settings; those are limiting the export to new and/or changed entities and settings a time interval for otherwise one-off export default. This will ensure a steady stream of data being output to where the developer has pointed in the configuration, while also preventing duplicates from showing up.

An export format template in Heiler PIM can be scheduled in virtually any possible way. Supported configurations include:

- periodic execution with seconds precision,
- odd days or even days execution,
- specific days of week execution,
- specific weeks of month execution,
- specific days of month execution.

In addition to the frequency setting, a start and an end hour (or a date for smaller frequencies) must be specified.

Such advanced scheduling covers any possible usage pattern. In particular, the integration project which is the basis for this thesis requires periodic data transfer and no trigger-based methods.

5 TOPIC SUMMARY

5.1 Results

Architectural choices

As with any engineering, proper tool needs to be taken for a proper task. This is also the case with integration projects, in which choosing an adequate solution of design patterns approaches is crucial. A Message translator integration design pattern was chosen for the task in the wake of discussions among team members.

The design pattern choice is not everything, of course. It is a tool for a problem specified strictly and profoundly among the project developers.

Developers' work is about making the right design and creation choices. This was mirrored closely in having to choose between implementation-ready and stable, albeit large templates supplied by Heiler Software AG and custom ones, tailored specifically for exact team purposes.

Final template

Finally, the data format needs to be agreed upon and in this case it was the data format imposed by the IBM WebSphere Commerce intake possibility.

The final project export format template allows for transferring:

1. Structure groups
   a) from the master catalog,
   b) from the sales catalog,
2. Attribute dictionary for the selected structure groups' features,
3. Catalog entries
   a) products,
   b) items.

Having all of these parts in one single export format template guarantees they are coupled and exported together, which is the most important with attribute dictionaries and structure hierarchy. It is also convenient for it ensures data overall integrity and simplifies task scheduling.

Automation

As was later described, an interval for concluding exports and a reference date for limiting export scope to only new and/or changed entities are set. These tasks essentially limit the developer's effort to periodic review and conducting any changes deemed necessary on the run.

Reusability

In terms of elasticity and reusability of this export format templates, both its parts and the whole template after changes prove useful in other company projects. Developers' effort was put into ensuring that the solution is generic, in a way that allows using it in other e-commerce-related projects. This was not limited to only abiding by the Business Object Document message format, however; good practice rules in such projects were taken into consideration, while some new ones were also developed.

5.2 Discussion

Further development

The software solution and architecture prepared as the main subject of this thesis is not the ultimate integration. In fact, many improvements can be applied.

First of all, as the IBM WebSphere Commerce integration templates provided by Heiler Software AG were rejected in the early process of development, a review of the resulting implementation and comparison to the former might be the starting point for ensuring appropriateness of such integration in any upcoming project in the company.

Secondly, the validation steps are not implemented. A follow-up discussion is required after test extensive enough are conducted. The validation could be applicable to both the transferred data format (thus be a kind of template testing altogether) and the data itself. The latter option could be later on extended with error reporting in case of missing data, other than the mechanism present in Heiler software suite.

Architecture extension

When discussing further development possibilities, the likelihood of new data exchange participants needs to also be raised as a valid issue. Those could include logging servers and return receipts mechanisms – those could benefit validation tasks, but might as well be e.g. server holding media assets of all kind, linked to the product information.

Possible benefits in similar projects

Export format templates remain neutral to the type of data held in information repository; the only customization obligatory in all kind of projects is the proper choice of data fields for transfer.

The especially sought after characteristic is that of unifying the in-house approach to webstore integration; the template, with minor changes, can be distributed to other projects and managed in a centralized way.

As normally we would have to consider developing a custom solution with similar goals within every company project were it not for the template applicability, the possibilities this characteristic poses involve replacing development with integration of already existing software; in this particular case, the export format templates.

Possible new issues

Every solution has its drawbacks, no matter the nature of issue. The same can be applied here, as the integration solution found and described in this thesis is not free from problems.

As with any type of automation, data consistency needs to be ensured by means of integration tests. Tasks concluded successfully by skilled developers will be initially more prone to errors when done by automated scripts, until all loopholes are eliminated.

The enterprise integration itself might pose problems magnified by the solution, namely those connected with encoding when dealing with multi-lingual data or those connected with the data volume.

REFERENCES

Apache Camel, Core 2.10.0 Application Programming Interface. n.d. Apache Foundation. Accessed on 2013-04-22. http://camel.apache.org/maven/current/camel-core/apidocs/overview-summary.html

Camel architecture documentation. n.d. Apache Software Foundation. Accessed on 2013-04-21. https://camel.apache.org/architecture.html

E-Commerce and E-Business: Concepts and Definitions. n.d. WikiBooks. Accessed on 2013-05-12. http://en.wikibooks.org/wiki/E-Commerce_and_E-Business/Concepts_and_Definitions

eMarketer, "Double-Digit Ecommerce Growth Story Continues in Western Europe". 2013. eMarketer. Accessed on 2013-05-11. http://www.emarketer.com/Article/Double-Digit-Ecommerce-Growth-Story-Continues-Western-Europe/1009777

Gregor Hohpe, Bobby Woolf. 2007. Enterprise Integration Patterns. Addison-Wesley

Enterprise Integration Patterns webpage by Gregory Hohpe. 2012. Enterprise Integration Patterns webpage. Accessed on 2013-05-11. http://www.eaipatterns.com/

Heiler Software AG. 2012. Heiler Product Manager, User Manual Export, Version 6.0 (English). Heiler Software AG

Heiler Software AG Resource Center. n.d. Heiler Software AG. Accessed on 2013-04-24. http://www.heiler.com/en/resources/index.php

IBM WebSphere Commerce 7 review. 2009-11-12. Internet Retailer portal. Accessed on 2013-04-21. http://www.internetretailer.com/2009/11/12/ibm-s-websphere-commerce-7-supports-mobile-and-social-commerce

IFIP/IFAC Task Force onArchitectures for Enterprise Integration. 1996. IFIP/IFAC Task Force. Accessed on 2013-05-11. http://www.ict.griffith.edu.au/~bernus/ei.references/task.force.info.html

Magic Quadrant for Horizontal Portals analysis. 2012-09-20. Gartner. Accessed on 2013-04-21. http://www.gartner.com/technology/reprints.do?id=1-1C5H0I7&ct=120925&st=sb

Mashable, "Forrester: U.S. Online Retail Sales to Hit $370 Billion by 2017". 2013. Mashable. Accessed on 2013-05-11. http://mashable.com/2013/03/12/forrester-u-s-ecommerce-forecast-2017/

OAGIS version 9.0 Business Object Document message architecture. n.d. Open Applications Group incorporated. Accessed on 2013-04-21. http://www.oagi.org/oagis/9.0/Documentation/Architecture.html

Open Applications Group Integration Specification webpage, v. 9.6.1.. n.d. Open Applications Group. Accessed on 2013-04-22. http://oagi.org/dnn2/DownloadsandResources/OAGIS961PublicDownloadPage.aspx

Overview of the Data Load Utility. n.d.. IBM. Accessed on 2013-05-01. http://pic.dhe.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.data.doc/concepts/cmlbatchoverview.htm

Product Information Management: Definition, Purpose and Offering. n.d. SapientNitro. Accessed on 2013-05-12. http://www.slideshare.net/sapient/product-information-management-definition-purpose-and-offering

The Economist, “E-commerce and the Market Structure of Retail Industries”. 2010. The Economist. Accessed on 2013-05-11. http://www.economist.com/node/16478931

Ventana Research Product Information Management Value Index for 2012. 2012. Ventana Research. Accessed on 2013-05-11. http://www.ventanaresearch.com/pimvalueindex/

W3Schools Examples and tutorials. n.d. w3schools.com. Accessed on 2013-05-10. http://www.w3schools.com/schema/schema_example.asp

World Wide Web Consortium XML Namespace recommendation. 2009. World Wide Web Consortium. Accessed on 2013-05-11. http://www.w3.org/TR/REC-xml-names/

World Wide Web Consortium XML recommendation. 2008. World Wide Web Consortium. Accessed on 2013-05-11. http://www.w3.org/TR/REC-xml

World Wide Web Consortium, XML Schema recommendation. n.d. World Wide Web Consortium. Accessed on 2013-05-11. http://www.w3.org/XML/Schema

XML Coverpages, XML Applications and Initiatives. n.d. OASIS. Accessed on 2013-05-11. http://xml.coverpages.org/xmlApplications.html

XML.com "Comparing XML Schema Languages". 2001. O'Reilly. Accessed on 2013-05-12. http://www.xml.com/pub/a/2001/12/12/schemacompare.html