The MT940 Project Requirements Document

M. Janek

R. Khadka

S. Jansen

Technical Report UU-CS-2011-001
January 2011

Department of Information and Computing Sciences
Utrecht University, Utrecht, The Netherlands
www.cs.uu.nl

ISSN: 0924-3275

Department of Information and Computing Sciences
Utrecht University
P.O. Box 80.089
3508 TB Utrecht
The Netherlands

Abstract

This document describes a preliminary support infrastructure for MT940 services in a public Service Oriented Architecture and serves as an initial reference for developing MT940 parsing functionality. Its aim is to present functional, technical and process (organizational) requirements gathered by interviewing companies participating in the project. The document outlines a solution for technical implementation and provides a discussion about the organizational issues faced by a soon-to-be-created MT940 consortium.

1 Introduction

The ServiciFi [3] project aims to identify and extract reusable services from large monoliths of financial software. Software monoliths have several well-known disadvantages (e.g., inflexibility, domain unspecificity, and hard to maintain), recognized by both the vendors and customers of these software products. Thus, ServiciFi aims to extract reusable services from monoliths of financial software using the Service Extraction Process. The MT940 project is the first attempt to transform a closed source, company specific code into a shared service.

The MT940 project is part of the ServiciFi project - it is the first attempt of ServiciFi to create a shared, reusable service based on the functionality that is currently implemented separately by each company. The reason for this is that there is currently no open source component natively available for technological platforms of all the participating companies. The aim of MT940 project is to facilitate parsing of MT940 bank statements for the participating companies. The companies involved in the project currently develop and maintain their own components for parsing MT940 bank statements. Thus, there is lack of specific standard or guideline while parsing the MT940 format. The companies reported that it would beneficial to have a joint initiative in developing and maintaining a shared MT940 parsing component which would bring numerous advantages such as decreased maintenance costs, up-to-date format definitions and a shared knowledge base. With this motivation, we aim to investigate preliminary support infrastructure for MT940 parsing. This report presents the functional, technical and process (organizational) requirements gathered by interviewing the participating companies and outlines a solution for technical implementation of the MT940 parser. Listing 1 depicts snippet of MT940 format.

Listing 1: Example of MT940 format

```
0000 01ABCDNL21XXXX00001
0000 01ABCDNL21XXXX00001
940 00
:20:GTZPB
:25:0001234567
:28C:004
:60F:C060326EUR44,89
:61:060327D0,45NGTNONREF
:86: 0660014650 TEXT ACCOUNT WITH THE REQUIRED TEXT WHICH CAN RUN ON
:61: 060327D0,45NBANONREF
:86:453261094 1318224 AAAAA HIJN HUISS>HUISS
PASNR 013T500 05-08-05 12 UUR 25 TRANSACTIENR 7492221
:61:060327D2,24NAC1234567812345678
.....
```

The participating companies are Internet based companies that use MT940 for parsing customer bank statement data into their systems. They agree that they do not compete on the very parsing and import of MT940 and could therefore benefit from a joint effort in developing and maintaining the MT940 import component. The companies were interviewed for requirements and they suggest that one of the following or the combination of *a community source component* or *an online service* be created. The component or the service is to be used by participating companies to replace or augment their own software components for parsing MT940. The value brought to the participating companies lies in the high quality of the component's code, lower maintenance

costs, availability of up-to-date parsing rules and shared format documentation. This document uses several terms that are defined as follows.

- A *component* is a unit of composition with contractually specified interfaces and explicit context dependencies [4]. For the purpose of this document, we extend the definition to specify that the component runs in the same environment as the existing application. In this case 'the same environment' refers to the physical presence of the component's code on the computer(s) running the existing application. For the sake of simplicity, the term component will be used throughout this document to refer to the software outcome of the joint initiative.
- A *web service* is a software system identified by an URL, whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web Services in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols [1].
- A *consortium* stands for the shared organizational and technical efforts of the companies and individuals participating in the MT940 project.

In order to avoid possible ambiguity with other related terms the definitions of the above mentioned terms was necessary.

1.1 Participating Companies

This sub-section briefly describes the companies involved in the MT940 project. To keep the name of companies anonymous we use arbitrary names instead of the real ones.

ERPComp1 is an online bookkeeping solution which aims to free entrepreneurs from having to deal with paperwork, thus giving them more time to do business. Users scan their invoices and then upload them to ERPComp1 for OCR processing and archiving. ERPComp1 can match these against MT940 imported bank statements to identify outstanding invoices. Although MT940 is crucial for ERPComp1, it is not its core business. That is why ERPComp1 wants to spend less time on MT940 maintenance and achieve better results on parsing, namely more up-to-date definitions of the format. They report that banks change MT940 format unannounced. Thus a common platform for sharing documentation knowledge is one of ERPComp1's requirements. This is further supported by the reported lack of documentation of MT940. What is also missing is the ability to test files for MT940 compliance. On the technical side, ERPComp1 suggests that the software architecture of the component should allow for future extensibility.

DigitalInvoiceComp1 is a web application for online billing. It allows business users to create professionally looking invoices and automatically send them to their customers on a given date or period. The customers receive invoices in PDF format and can pay the invoice conveniently via iDEAL. DigitalInvoiceComp1 currently does not support MT940 so the users track the status of the invoices based on manually entered information. However, DigitalInvoiceComp1 intends to implement MT940 import functionality. DigitalInvoiceComp1 reported that the most appropriate solution for them would be a service that would supply their custom software with up-to-date format definitions.

DigitalInvoiceComp2 is a web application for online invoicing. Users create invoices through a web interface and send them either instantly or with a predefined periodicity. DigitalInvoiceComp2 supports invoice tracking and payments via iDEAL. Currently, the bank account import functionality supports two banks (X, Y), one of which has to be manually selected by the user. DigitalInvoiceComp2 would welcome the capability to process bank account statements from all major Dutch banks as well as the ability to automatically detect users' banks based on their bank

account statements. DigitalInvoiceComp2 reported that the desired functionality could be implemented in form of a dedicated web service.

2 Requirements

In this section we describe the requirements for MT940 processing. The requirements are based on the interviews and the analysis of the interviews done with the respective companies. We categorized the requirements three groups: functional, process, and technical requirements, which we describe in following subsections.

2.1 Functional Requirements

This section describes what the MT940 project must deliver in order to create value for the participating companies.

Table 1: Functional Requirements

ID	Description
F1	The MT940 component must be able to process MT940 formatted input files in order to allow import into the systems of the participating companies.
F2	The MT940 component must be able to automatically recognize what financial institution is the author of MT940 input files. This functionality is necessary for correct interpretation of MT940 dialects.
F3	Regardless of whether the initiative will result into a physical component or a service, there should exist a common service or a storage place supplying the participating companies with up-to-date definitions of MT940.

2.2 Business Requirements

This section describes the requirements from the organizational point of view (i.e. establishing and maintaining the consortium).

Table 2: Process Requirements

ID	Description
B1	An appropriate organizational model must be created that will describe the governance structure of the consortium. Because the participating companies might differ in size, financial strength or other aspects, it is necessary to create a structure that is flexible enough to appropriately delegate suitable tasks to each company, yet keeping an equilibrium between the resources provided and the value obtained by each participating company. The organizational model has to minimize dependencies among companies in order to avoid possible political or technical issues. It is also essential to consider a case of a company exiting the consortium and the subsequent transfer of responsibilities to other parties.
B2	The MT940 component must be developed using the resources of all participating companies. The resources might be in several forms: human, financial or other as agreed on by the participating companies.
B3	The initiative should create a knowledge community for sharing documentation knowledge regarding MT940.
B4	The consortium needs to agree on the communication of changes pertinent to the software component or the MT940 format definitions. A process ensuring clear communication of changes must be defined.
B5	The consortium has to agree on the financial structure for maintaining the MT940 component considering the possibility of new parties entering the consortium at a later stage. Therefore, it is necessary to make the distinction between financing the project in the stage of building and in the stage of maintenance and upgrades.
B6	The selected organizational structure should sustain product development and support over time, taking into consideration companies entering or exiting the consortium.
B7	The consortium needs to agree on the intellectual property ownership of the jointly developed software component in order to protect the resources invested into the project. Such an agreement should define the rights associated with the component, even for cases when a company decides to exit the consortium.

2.3 Technical Requirements

This section lists the technical requirements for the MT940 component. A major challenge in the

Table 3: Technical Requirements

ID	Description
T1	The MT940 component must implement the following functionality: reading an MT940-formatted input message, parsing it, interpreting it and outputting it in either a standardized/agreed on XML-based format or transforming it to native constructs of the specific programming language. This entire process must remain semantically correct and ensure data completeness. In case of a web service, it is crucial that the participating companies agree on a semantically unambiguous XML-based (intermediate) format. The intermediate format must not allow misinterpretation of data and must be sufficiently well documented.
T2	Each of the participating companies uses a different development platform, which makes interoperability a central technical requirement. Specifically, ERPComp1 uses ASP.NET, DigitalInvoice1 uses Ruby and DigitalInvoice2 uses PHP. The component must be interoperable with all of the above-mentioned technologies.
T3	The component must be sufficiently well documented at both the source code level and API (interface) level in order to minimize the resources necessary to perform maintenance.
T4	The fact that the MT940 project deals with three different environments means that there might be some duplicate code required in case of a physical software component. The consortium should strive to aggressively minimize duplicate code for two reasons: first, to keep the maintenance costs low and second, to avoid errors in the source code.
T5	The MT940 component must be scalable to a level agreed upon by the consortium. The reason for this is that the consortium involves organizations of different sizes with different numbers of clients. Additionally, even smaller organizations have the potential to acquire more clients. However, the solution should find a good balance between scalability and economic viability in order to avoid unnecessarily high costs.
T6	In case of a web service, the MT940 web service should adhere to a set of agreed-upon performance metrics, such as response time and throughput. Additionally, based on the financial model used, the technical solution should be capable of tracking the number of requests from each party or defining performance aggregation ratios.
T7	In case of a web service, the solution must guarantee an acceptable level of availability. Additionally, fail-over capability should be taken into consideration in case that the MT940 service is deemed crucial for ensuring the continuity of business operations of the participating companies.
T8	The solution has to be designed with several decoupling points in mind in order to allow distributed development of its parts independently by the participating companies and to allow a partial use of its functionality by parties that do not need to use all of its functions. Specifically, the distinction should be made between the MT940 format definitions, the MT940 parser, the XML transformation engine and appropriate management (web) interfaces. This would bring advantages such as easier maintenance, the ability to develop parts of the solution concurrently (and to some extent independently of the programming language used) and future extensibility.
T9	Security of the solution is of utmost importance and should be considered since the very first phase of design. This is given by the sector that the participating companies operate in - dealing with sensitive financial data. Security of the solution should be aligned with the organizational model, specifically, changes that could affect business operations of more participating companies should be only allowed and executed after the approval of the body designated by the consortium.
T10	In case that the result of the initiative is a web service, it is of paramount importance to ensure confidentiality and isolation of the processed customer data.

MT940 project is that not all companies favor the idea of having their business dependent on an external service, which they have no direct control over. The flexibility of the technical solution is thus paramount to the success of the MT940 project.

3 Analysis of Organizational Issues

The consortium will include commercial subjects (the participating companies) as well as individual(s) outside the participating companies, namely the ServiciFi project initiator(s). Therefore

the selected organizational structure needs to take into account the interests of all stakeholder. The consortium has two options when it comes to legal governance:

1. Establish a new joint venture that would acquire a status of an independent legal entity with its own budget, management, etc.
2. Govern the development of the component by a contract without creating a new legal entity.

Additionally, the consortium has three options of developing the MT940 component:

1. *Shared distributed development*: the companies would partition the software component into parts that could be developed in parallel with large degree of independence. The consortium would agree on assigning the development of different parts of the component to different parties based on their capabilities and competence. This option has the advantage of distributed control over the project, however it would require more coordination as well as a more advanced technical solution due to the need of software decoupling points.
2. *Development by a single company*: one company would develop the component and the others would compensate it financially. The advantage of this option is low communication overhead, however a major challenge is the control over the project.
3. *Outsourced development*: the consortium would raise funds and hire another company to develop the component. This option is politically neutral but if the code is developed by another party, maintenance could lead to additional costs since it would have to be performed by another company.

One of the options is to base it on code metrics, such as Lines Of Code (LOC), however such a comparison is difficult due to the fact that each party uses a different programming language. The parties would therefore have to either neglect the differences of productivity of the programming languages or agree on some form of ratios. This is however extremely difficult and can be seen as an academic problem. Compensation based on code metrics has the advantage of transparency because every party can easily check the work done by the others. Another option is to use time based metrics such as man-hours, which eliminate the problem with quantifying different programming languages. There are two options with this: the parties would structure the work and agree on the expected workload in advance or report the actual workload during the execution of the project. The latter option, however, introduces trust into the equation. Because the parties would have to report the man-hours dedicated to the project, a considerable level of trust would be required among the parties.

A Project Board of institutional leaders should be created who will decide on the strategic direction of the consortium and oversee its operations. The board will involve one person from each company plus a Coordinator from ServiciFi. The coordinator should also act as an independent mediator in case that the participating companies have different interests and expectations about the direction of the project. The coordinator needs to govern the expectations and objectives of the partners. The consortium has to agree on the distribution of power within the board. The distribution of power can be based on criteria such as financial contributions, contribution of human resources or other. The board should agree whether the MT940 project has a finite time, after which the partnership will be dissolved.

Because the project needs to maintain and update a set of definitions of the MT940 dialects, a special MT940 Format Committee should be established, which would regularly monitor the changes made to the format by financial institutions and map these onto the format Transformation Rules . The committee should include people from several companies for political reasons as well as validation reasons. A ServiciFi representative should also join this committee to provide academic/scientific insights. These people should have technical background and be comfortable with abstract thinking because defining the MT940 transformation rules will require the understanding of mathematical concepts such as regular expressions. Affinity with financial sector would

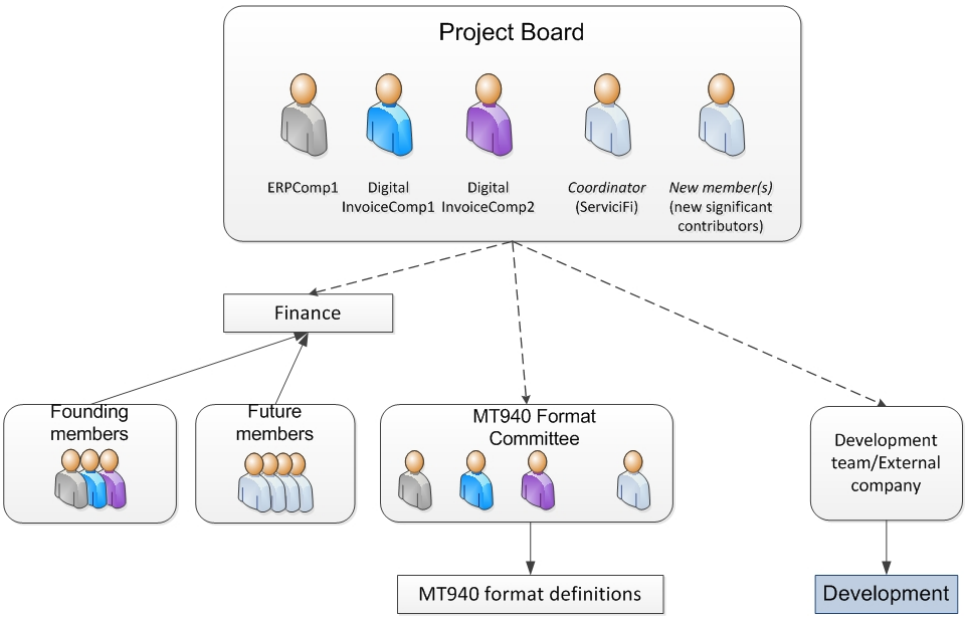

Figure 1: Consortium Organization

also be an advantage. Figure 1 shows a high level view of consortium members and the influence over finance, development and format definitions. A distinction is made between the founding members and future members because the financial model has to consider the resources provided by the founding companies during the development stage.

Upon analyzing the requirements, ServiciFi proposes the following solution: a consortium will come to existence as a joint venture and will develop the software mainly with its own resources, different parts being developed by different companies. A technical solution presented in section 4 supports this arrangement. The question that has to be agreed on is how to quantify the development effort in order to equally compensate all parties.

4 Proposed Technical Solution

This section explains a possible technical solution of the MT940 project.

4.1 System architecture

Figure 2 shows a high-level architecture of the solution. The solution is partitioned into a Business Rules Provider service and a Transformation Engine service. This partitioning is made in order to meet the decoupling requirement (T8) described in section 2.3 (Technical Requirements). The two parts can thus be implemented independently as stand-alone web services in different programming languages by different parties (requirements B2 and T8).

Business Rules Provider (BRP) is a service responsible for storage, maintenance and publishing of Transformation Rules for the Transformation Engine. Transformation Rules consist of XSD schemata defining each format (MT940 dialects, intermediate XML format) and a set of semantic mappings between the source format and the output format. BRP can also be queried for XSD schemata only for a purpose of testing input files for compliance to a specific format. This architecture allows further extensibility of the import mechanism. It is possible to easily extend the system with other commonly used formats such as bank-specific CSVs. The BRP service is designed to be a non-critical component. Transformation rules are fetched and cached by the Transformation Engine regularly but not with every import request, which makes the solution resistant to BRP

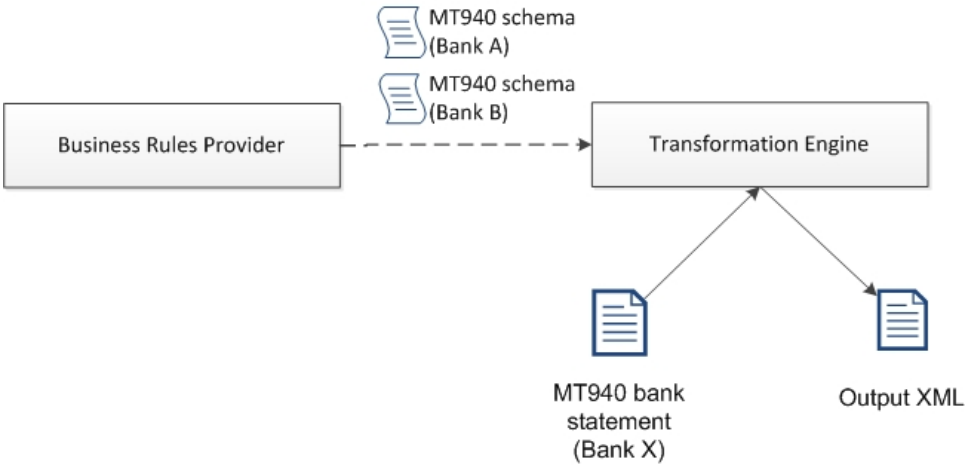

Figure 2: High-level Architecture

downtime. Ideally, a notification service should exist that would notify all registered instances of the transformation engine upon a change in the Transformation Rules.

The Transformation Engine is the core of the system. Upon initialization, it will load the business rules. An MT940 file submitted to the engine will firstly be checked for consistency and compliance. After this, it will be matched against an appropriate XSD schema to identify the bank that generated the file. Upon successful completion of the previous two steps, the transformation engine knows the semantics of the input file and can perform one of the following actions:

- Execute a transformation of the input file into the output XML format using the host platform's XSLT processor (such as the XSLT processor available in Microsoft .NET Framework)
- Invoke a method that will instantiate a new native structure (class, struct) that can be used directly from the program code. In order to achieve a 'clean' implementation based entirely on the business rules acquired from the BRP, the native structure will be instantiated with a dynamic mechanism such as reflection or XML deserialization in case of MS .NET platform.

4.2 Deployment scenario

This sub-section describes the way of deploying the MT940 component based on business and technical requirements of all parties. The most important requirements of the specific parties are:

1. ERPComp1's preference of a physical component over a shared web service
2. DigitalInvoiceComp1's preference of getting only up-to-date MT940 format definitions without using the Transformation Engine
3. DigitalInvoiceComp2's interest in getting the entire MT940 project functionality as a web service

These requirements can all be met. There can be an instance of the Transformation Engine running as a hosted web service for the present and future members of the consortium willing to use an external service. In addition, companies which are too concerned about quality parameters of the shared web service (availability, performance, etc.) can run their own instance of the Transformation Engine. This would be the case for ERPComp1.

Figure 3 shows the deployment scenario. The non-critical component of the solution - the Business Rules Provider - can be used as a web service by all participating companies, since

a temporary unavailability of the BRP does not affect the Transformation Engine. DigitalInvoiceComp1 can only fetch the MT940 format definitions without having to run or interface the Transformation Engine at all. Figure 3 shows a deployment scenario satisfying all requirements. DigitalInvoiceComp2 can use the instance of the Transformation Engine hosted as a web-service.

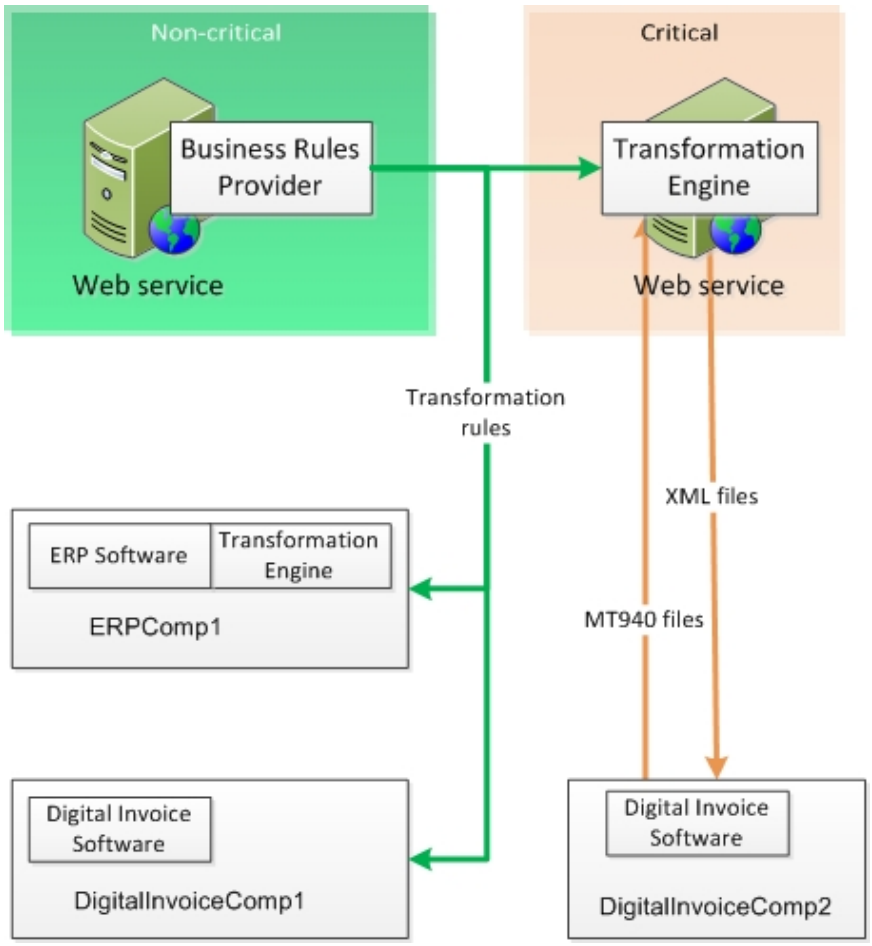

Figure 3: A Deployment Scenario

4.3 Roles in Consortium

A major decision that needs to be made is the choice of a native platform (and a programming language) for the transformation engine. In case that the consortium decides to develop the solution with its own resources (people), without outsourcing the development, the choice of a platform and a programming language is crucial.

Based on the technical expertise of the participating companies, ServiciFi suggests the following solution: since ERPComp1 wants to run the component in their own environment, ERPComp1 is the preferred developer of the Transformation Engine. Thus, the Transformation Engine will be based on Microsoft .NET Framework and C#. This will allow ERPComp1 to develop the engine with its own resources as well as natively embedding it into their software without the need to invoke a web service.

DigitalInvoiceComp1 will implement the Business Rules Provider in Ruby as this is their development environment. Due to the fact that the BRP will run as a web service, there is no conflict between one part of the solution being written in C# and another in Ruby. DigitalInvoiceComp2 will be responsible for creating and maintaining the MT940 knowledge base and supportive parts

of the project such as the consortium's web-site, software SVN directory, etc. All companies will jointly analyze the MT940 dialects and create dialect-specific schemata and XML transformation rules. The participating companies might find the proposed solution imbalanced in terms of workload. In such a case, they should agree on an appropriate financial or other compensation.

5 Related Initiatives

This section describes the related initiatives of standardizing the message types, processes and associated metadata that has been shared across the financial industry.

5.1 Single Euro Payments Area (SEPA)

SEPA is an initiative of the European Payments Council (EPC), which aims to establish a single market for retail euro payments by overcoming the technical, legal and market barriers. This will allow customers to make euro payments throughout Europe as easily, securely and efficiently as they do today within their own countries. Once SEPA has been completed (in 2011), there will no longer be any differentiation between national and cross-border euro payments: they will all be domestic. The implications on European financial industry stemming from the introduction of SEPA are faster payments, equal fraud-risk levels, higher number of competitors, reduction of transaction costs, reduction of cash money and increased measures against money laundering. The European Parliament has already called on the Commission to set a "clear, appropriate and binding end-date, which date should not be later than 31 December 2012, for migrating to SEPA products" [2] (European Parliament, 2009).

5.2 ISO 20022 (UNIFI)

ISO 20022, also known as UNIFI (UNIversal Financial Industry message scheme), is a set of standards for message types, processes and associated metadata that has been shared across the financial industry. UNIFI offers UML business models of its messages and a set of XML design rules to convert the UML models into XML schemata. Most importantly, the use of UNIFI is mandatory for access to a SEPA-compatible processing system, therefore it will become widespread in the upcoming years as banks that wish to participate in SEPA must implement UNIFI. ServiciFi's proposal to use UNIFI is anchored in the expected ubiquity of SEPA and UNIFI in the pan-European financial market (beginning in 2011). Choosing UNIFI as the component's output format will force the participating companies to make their bank statement import mechanisms UNIFI-compatible. After the financial institutions start offering bank statements in UNIFI format, the participating companies will be able to import UNIFI-compatible bank statements directly, without the need of the MT940 component. In addition, the participating companies will still be able to import legacy MT940 bank statements. The UNIFI camt (Cash Management) family of messages contains three messages which are specifically designed for the information that banks provide to their customers regarding payment transactions processed on their account. Those messages are:

- camt.052 (Bank to Customer Account Report)
- camt.053 (Bank to Customer Statement)
- camt.054 (Bank to Customer Debit/Credit Notification)

For the purpose of this project, it is assumed that camt.053 will be used. Camt.053 is used for end-of-cycle bank statements and such statements should be therefore accessible to retail customers. Society for Worldwide Interbank Financial Telecommunication (SWIFT) offers documents that describe MT940 to camt.053 translation rules, which could be useful for implementing the Business Rules Provider and the Transformation Engine.

6 Conclusion

We have analyzed a problem situation consisting of three companies with similar objectives but using different technology platforms and with different organizational characteristics. We have interviewed the companies and gathered requirements for implementing a common component for importing MT940 bank statement. Based on the elicited requirements we have proposed a technical solution satisfying the needs of all concerned parties. Our initial observation in this project suggests that to comply with TR1, the output of the MT940 component must be an intermediate XML-based format that is strictly defined, well documented, unambiguous and all companies can use it seamlessly. Two possible solutions exist: (1) Create a custom XML-based format defined by the consortium and (2) Use an existing XML-based format which fulfills all the necessary requirements. Hence, ServiciFi aims to adopt ISO 20022 as a basis for choosing the intermediate XML-based format.

The proposed solution is compatible with both legacy bank statement format (MT940) as well as the upcoming European format (UNIFI). It should therefore not only provide the required MT940 import functionality but also ensure a seamless transition to the UNIFI format once banks adopt it.

Bibliography

[1] Austin, D., Barbir, A., Ferri, C., Sharad, G.: Web Services Architecture Requirements (Febuary 2004), http://www.w3.org/TR/wsa-reqs/

[2] SEPA: European Parliament resolution on the implementation of the Single Euro Payments Area (SEPA) (March 2009), http://www.europarl.europa.eu/sides/getDoc.do?pubRef=-//EP//TEXT+MOTION+B6-2009-0111+0+DOC+XML+V0//EN

[3] ServiciFi: Decomposing Monolithic Software Systems in the Finincial Domain (November 2010), http://www.servicifi.org/

[4] Szyperski, C., Pfister, C.: Component-Oriented Programming: WCOP'96 Workshop Report. In: Special Issues in Object-Oriented Programming: Workshop Reader of the 10th European Conference on Object-Oriented Programming ECOOP96, Linz. pp. 127–130 (1996)