Standardization of financial transactions

<UNIversal Financial Industry message schemes of the Single Euro Payment Area/>

Master Thesis Software & Systems Engineering
University of Groningen
Faculty of Mathematics and Natural Sciences
7 July, 2007

By
G. D. Craens
Student number: S1450522
gcraens@gmail.com

Supervisors:
Ir. S. Achterop
Faculty of Mathematics and Natural Sciences
University of Groningen

Drs. H. Stevens
Senior Technical Consultant
Pecoma

Ir. E. Wildschut
Consultant
Pecoma

Version: 35
Status: final version

RuG *University* of Groningen

pecoma business technology
denkkracht daadkracht voelkracht

Abstract

Key words: standardization, SEPA, UNIFI, XML, XSD.

Banks and other financial organizations are using a lot of different formats to realize services (e.g. payments by direct debit or credit transfer) for their customers today. Also the difference in law of a country causes different formats. The rules for a direct debit are for example different in Italy compared with the Netherlands.

The Single Euro Payments Area (SEPA) initiative involves the creation of a zone for the euro in which all electronic payments are considered domestic and where a difference between national and international payments does not exist. The project aims to improve the efficiency of international payments and also to develop common financial instruments, standards, procedures, and infrastructure to enable economies of scale. This will replace the complex and costly international infrastructures that are currently in operation. The result is a reduction of the overall cost to the European economy of moving capital around the region. SEPA is based on ISO 20022 and uses the XSD format. The agreement of standardization was made in December 2006. It seems that XSD will be the solution for standardization in the future. But is XSD suitable? What are the constraints of XSD?

This thesis contains answers to these and other related questions. Furthermore a design and implementation of an architecture of a SEPA UNIFI Test tool has been made, which is also described in this thesis. This tool is able to 1) validate XML against XSD, 2) generate XML based on an XSD and 3) be able to generate a form based on XSD for user friendly entry. 4) The tool is also smarter than XSD "out of the box" like for example XX123456789 is a valid bank account according to the regular expression [A-Z]{2}[0-9]{9}, but not likely to be an existing account as XX is not a country code and 123456789 is not likely to be given out by a bank.

Preface

For my final research project I was interested in standardization in the domain of software engineering. Both the semantic aspect and the technical aspect, like for example the implementation of electronic business XML, XBRL, security, etc. could be in the scope of the research.
D. Rinkes came with the thought to do something with the standards of the Single Euro Payment Area (SEPA). I want to thank him, because it matched with my interest.
I want to thank my supervisor Ir. S. Achterop for his supervision and in particular the useful feedback he gave to me. I also want to thank him for teaching me during the courses Embedded Systems, Real Time Systems and Technology for System Realization, which were also part of my Master.
Thanks go also to my supervisors of the company Pecoma: H. Stevens (Groningen) and E. Wildschut (Amsterdam), because of their help and support.
I also want to thank C. Braun for organizing seminars about SEPA. It gave me more insight in SEPA and an opportunity to discover what people of the financial world want to get standardized. I also want to thank H. Voskuilen and G. Aussems of NIBESVV/Ereon and M. R. Colthoff of Capgemini for sharing their knowledge about SEPA. Further I want to thank J. J. Nienhuis of Innopay for answering my questions about the UNIFI standard.

George Craens
Groningen, June 2007

Table of content

1 Introduction

This chapter gives a short introduction of the topic; it describes the purpose, the research question, and gives an overview of the structure of this thesis.

1.1 Topic

Standardization in the financial world with mainly the Single Euro Payment Area (SEPA), is the topic of this thesis. The UNIFI standard is an important element for the realization of SEPA, because it provides the financial industry a common platform for the development of messages in syntax of standardized XML. It uses a modeling methodology which is based on UML to capture in a syntax-independent way financial areas, business transactions and associated messages flows. It also uses a set of XML design rules to convert the messages that are described in UML into XML schemas. These XML schemas or XML Schema Documents (XSDs) play a central role during the final research project.

1.2 Purpose

The goal of the final research project is to extend experience of scientific research in Software and Systems Engineering. Besides the theoretical part, the project also includes a practical part. The output of the theoretical part exists of answers to the research questions that are defined below. The output of the practical part exists of a SEPA UNIFI Test Tool including several designs of the architecture and other related documents.

1.3 Research questions

The main research question is: "is XSD a suitable solution for standardization of financial messages?" To answer this question the following sub questions are defined.

1) What is standardization and which aspects are related to standardization?

2) What are the current tools to develop and view (UNIFI) XSDs?
    a) What are the extra features of the concerned tool compared with the features of XSD?
    b) What are the constraints and problems of the tool? With which functionality could the concerned tool be extended?
    c) How does a SEPA UNIFI Test Tool look like?

3) What are the possibilities and constraints of (UNIFI) XSD?
    a) What kinds of UNIFI messages do exist?
    b) How are (UNIFI) XSD messages implemented?
    c) Are organizations who implement UNIFI still able to add new functionality/wishes, without being incompatible with UNIFI? Is it for example possible to add extra attributes to the standard as defined in the ISO 20022?
    d) Does the ISO 20022 define schemas for e-invoicing?
        i. If not, how could an e-invoice schema look like?
        ii. What are the advantages of e-invoicing?
        iii. What are the disadvantages of e-invoicing?
    e) What are the possibilities around inheritance in XML/XSD?
    f) How much of the inheritance related features of XSD are applied on the XSDs of UNIFI?
    g) Are the schemas of UNIFI generic enough to apply (more) inheritance?
    h) How is inheritance applied in Object Oriented languages like Java? Give a description of the generics and inheritance of this language.
    i) How will a UNIFI message with Java syntax look like?

1.4 Methodology and methods

This paragraph describes how the research has been accomplished. Figure 1 illustrates the global research approach. The final research project started with an orientation to get more knowledge of the topic. Next, a project plan has been made with i.e. a time schedule and a table for risks management. After that two almost independent threads have been started. The first one has been divided in: defining research questions, desk research, interviewing and writing the thesis. The second one has been divided in: requirement analyses, design, implementation and validation plus verification. The former has a theoretical character and the latter a practical character. The final research project ends with a presentation, including a demonstration.
[r50] This research can be classified as qualitative instead of quantitative, because there are open questions defined and most of the knowledge about the topic has been retrieved 'through' the eyes of others' during the desk research. Another principle of qualitative research that has been applied is the cyclical interaction instead of linear interaction between the stages. The arrows in the figure below represent the flow of actions between the stages.
Information for this research is mainly of the first order i.e. objective data. This has been replenished with data of the second order, i.e. information obtained from involved people via e.g. interview, mail and seminars.

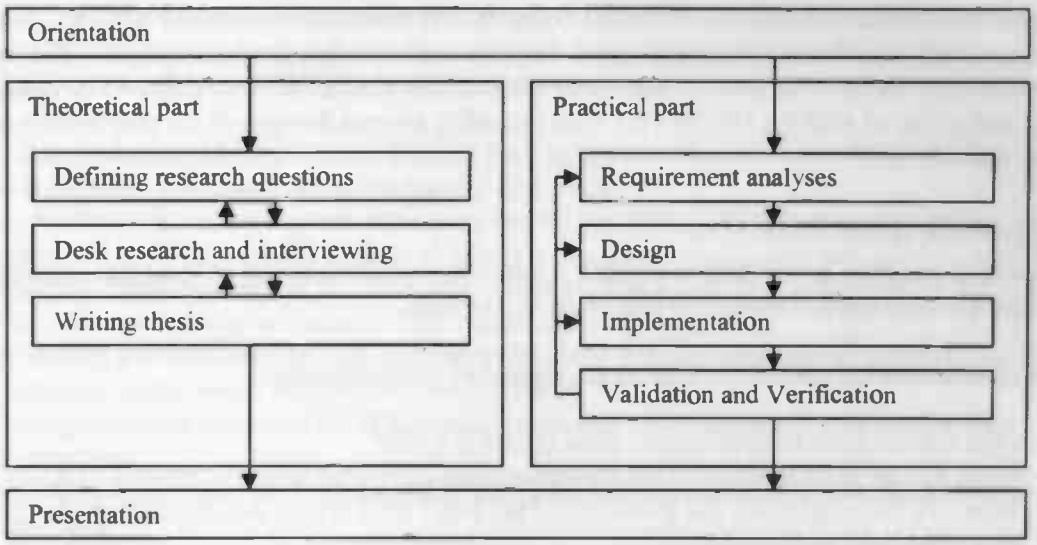


Figure 1: Global research approach

1.5 Structure of this thesis

The next chapter provides information about the financial world to get an understanding of the current and new situation. The third chapter describes what UNIFI is, why it has been developed, which kind of messages do exist and how these messages are implemented. Also XML, XSD and e-invoicing are treated in this chapter. Chapter four describes theory and technology of standardization, languages and notations of these languages. Furthermore it contains a description of inheritance and a description of a UNIFI message in a programming language. Available tools are mentioned in chapter five.
Requirements, design, implementation, validation and verification of the SEPA UNIFI Test Tool are mentioned in chapter 6. This paper ends with a summary of answers and a conclusion.

2 The financial world

This chapter provides information about the current and new situation of the financial world. It gives financial background information to get a picture of the kinds of financial transactions, organizations and systems that do exist. Furthermore it describes the payment processes of the old and new situation. This chapter continues with an overview of SEPA. The players, components, time line, advantages and disadvantages are treated.

2.1 Financial background information

2.1.1 Organizations

This paragraph mentions organizations that are related to SEPA and financial standardization in general. See table 1 for an overview of abbreviations and names of these organizations.

Abbreviation	Name
EACT	European Association of Corporate Treasurers
EBA	European Banking Association
ECB	European Central Bank
ECBS	The European Committee for Banking Standard
EPC	European Payments Council
ISO	International Organization for Standardization
	Omgeo
SWIFT	Society for Worldwide Interbank Financial Telecommunication
TWIST	Transaction Workflow Innovation Standards Team/ Treasury Workstation Integration Standards Team)
UN/CEFACT	United Nations / Centre for trade facilitation and e-business

Table 1: Organizations

EACT (European Association of Corporate Treasurers)

[r10] The Purpose of EACT is to group the National Associations of Corporate Treasurers and Financiers of the countries belonging to the European Union. EACT's aims are:

- to develop and strengthen relations with European authorities and institutions;
- to share experiences, express common points of view, undertake joint actions on financial and treasury matters as well as relationships with financial partners;
- to carry out and publish joint surveys and working papers.

EACT represents 8,000 people of 4,5000 corporates of 16 European countries. It supports ISO 20022 for financial information. CAST (Corporate Action on Standards) is an EACT initiative. The aims of this initiative are to define end-user requirements, compare these with any existing standards, detect gaps and identify the best solution in the following areas:

- extended remittance information and e-reconciliation;
- e-invoicing and electronic bill presentment and payment (EBPP);
- trade financing based on standard invoice header;
- unique entity identifier;
- digital identity and e-signature;

CAST is also an example of e-SEPA. The latter regards to the extension of SEPA with the standardization of end-to-end processes for corporations and the development of 'forward-looking' features.

EBA (European Banking Association)
[r5] EBA has been founded in 1985 by 18 banks and the European Investment Bank with support of the European Commission (EC) and the Bank for International Settlements (BIS). Now[1] there are 200 members from 28 countries and 33 associate members. Examples of members are ABN AMRO Bank, DNB, ING Bank and the Rabobank. Examples of associate members are Accenture, ATOSEurnext, Capgemini, LogicaCMG and VOCA.
EBA is an initiator and developer of European payment infrastructures in the financial industry. Their core activities have led to the private large-value clearing system EURO1, the low-value payment system STEP1 and the successor STEP2. EBA Clearing, which consists of 70 shareholder banks, owns these systems. EBA has furthermore the role to collect the European payment related viewpoints of its members and communicate these with regulatory and industry bodies. They also make recommendations and formulations about industry-wide business practices for payments.

ECB (European Central Bank)
[r12, r51, r52] The ECB is the successor of the European Monetary Institute (EMI) and exist since June of 1998.
It is the responsibility of the ECB to carry out the monetary policy, but the main task is to maintain the purchasing power of the euro. This means that there has to be price stability in the euro area. This area counts 13 of the 27 EU countries[2] that have introduced the euro since 1999.
The ECB and the national central banks (NCBs) of all EU Member States, irrespective they have adopted the euro or not, form together the European System of Central Banks (ESCB).
The ECB and the NCBs of EU Members States that have adopted the euro is known as the Eurosystem.
Both, Eurosystem and ESCB will co-exist as long as there are EU Member States that have not adopted the euro. The Eurosystem has mainly four tasks.
One of the tasks of the Eurosystem is to carry out the monetary policy that has been adopted by the Governing Council of the ECB. A second task is to conduct foreign exchange operations. Thirdly, they manage the official reserves of the euro area countries. The fourth task is to promote the smooth operation of payment and settlement. This will be done by:

1) providing payment and security settlement facilities like the euro system TARGET or the Correspondent Central Banking Model (CCBM);
2) setting standards to ensure the efficiency of (clearing and settlement) systems for euro transactions;
3) assessing the continuous compliance of euro payment and settlement systems with these standards;
4) providing a framework for securities settlement systems (e.g. in the framework of the cooperation between the ESCB and the Committee of European Securities Regulators (ESCB-CESR);
5) stimulating changes like for example the efficiency in payment systems and the adaptation of the infrastructure to the needs of SEPA. It also promotes an efficient securities market by encouraging the removal of barriers towards integration;
6) giving advice to legislators and compiling monetary and financial statistics.

ECBS (The European Committee for Banking Standard)
[r13] The ECBS was formed in December 1992 by three European Credit Sector Associations (ECSAs). These are the Banking Federation of the European Union (EBF), the European Savings Banks Group (ESBG), and the European Association of Co-operative Banks (EACB). The ECSAs represent the interests of the European banks and associations from the countries of the European Union (EU), the European Economic Area (EEA), and the European Free Trade Association (EFTA).

[1] Counted on 07-06-07
[2] Euro area countries since 1999: 1) Austria, 2) Belgium, 3) Finland, 4) France, 5) Germany, 6) Ireland, 7) Italy, 8) Luxembourg, 9) Netherlands, 10) Portugal, 11) Spain; since 2001: 12) Greece; since 2007: 13) Slovenia. Other EU contries are: 14) Denmark, 15) United Kingdom, 16) Sweden, 17) Cyprus, 18) Czech Republic, 19) Estonia, 20)Hungary, 21) Latvia, 22) Lithuania, 23) Malta, 24), Poland, 25) Slovakia, 26) Slovenia), 27) Romania.

The goal of ECBS is to improve the European technical banking infrastructures by developing standards after clear business and commercial interests have been identified. ECBS monitors, at European and international levels, the related activities around standards that could have an impact on the European banking community. Furthermore they provide a forum for the European banking sector's opinion about standards and industry bodies.

EPC (European Payments Council)
[r18] This is the decision making and coordination body of the European banking industry in relation to payments since June 2002. Its purpose is to support and promote the creation of SEPA. EPC defines common positions for core payment services within a competitive market place, provides strategic guidance for standardization, formulates best practices and supports and monitors implementation of decisions taken.

EPCA (European Payments Consulting Association)
[r14] EPCA is a pan European association of national consultancies with knowledge of payment systems and payment products and exists since 1998. They consist of six consultancy firms that are located in Belgium, Denmark, France, Germany, Great Britain and The Netherlands[3]. These firms focus on:

- strategy consulting;
- payments infrastructure and organization;
- payments product development;
- financial modeling and business-case calculations;
- research and analysis.

ECOFIN (Economic and Finance) Council
[r8, r66] ECOFIN is one of the 9 configurations of the Council of the European Union. The Council is the main decision-making body of the EU. If the Council acts as a legislator, it makes decisions of proposals that are made by EC.
ECOFIN exist of the economic and finance ministers of the member states. If budgetary issues are discussed it contains as well budget ministers. The ECOFIN Council is active with EU policy in a number of areas. These areas include: economic policy coordination, economic surveillance, monitoring of member states' budgetary policy and public finances, the euro (legal, practical and international aspects), financial market, capital movements and economic relations with third countries. ECOFIN prepares and adopts every year, together with the European Parliament. The budget of the EU is approximately 100 billion euros. Decisions are made mainly by qualified majority and with consultation or co decision of the European Parliament. If dossiers related to the euro and EMU get examined, than only the representatives of the member states whose currency is the euro takes part in the vote of the Council.

ISO (International Organization for Standardization)
[r21] ISO is a non-government standardization organization that exists of national institutes of 157 countries. They identify which international standards are required by business, government and society. They develop the standards in partnership with the sectors that will put them into use, adopt them by transparent procedures based on national input and publish them for worldwide implemention.
The standardization work is spread over various 'Technical Committees' (TCs) to which all the ISO member countries can decide to participate. Technical Committee TC68 is responsible for all ISO standards related to financial services (such as BIC, IBAN, ISIN, CFI, MIC, ISO 8583, ISO 15022 and many others including UNIFI-ISO 20022). The committee can be divided into sub-committees (SCs), which are aimed to a specific area. ISO members that participate actively are called 'P'-member countries. Furthermore there are standard organizations that can be authorized for a 'liaison' with a TC or SC they want to contribute. Active 'liaison' organizations are called categorie 'A' liaison

[3] EPCA member of the Netherlands is Innopay, which is a e-payments consulting and product development company.

organizations. Examples of these organizations that are active in TC68 are: UN/CEFACT TBG5, ISDA/FpML, FIX Protocol Ltd (FPL), MDDL, ISITC, TWIST, AMEX, VISA, Euroclear, Clearstream and SWIFT. The organization for the development of financial messages can be divided into three parts (see figure 2).

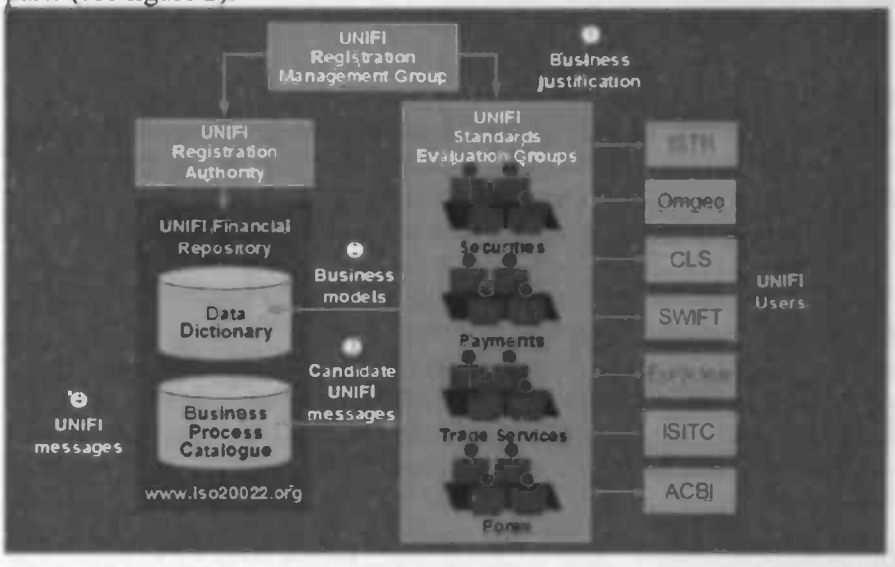


Figure 2: Organization; Source: [r21]

These are the Registration Management Group (RMG), Registration Authority (RA) and the Standard Evaluation Groups (SEGs). SWIFT has been chosen to fulfill the role of the RMG, which exists of senior financial industry experts that monitor the overall registration process. They also support the development of submitting organizations to guarantee the compliance of the developed models with the technical specifications of UNIFI. Furthermore they perform the transformation of message models into XML schemas. The RA is responsible for the UNIFI repository. The SEGs contain industry experts that represent the future users of the financial UNIFI messages. There task is to evaluate if the candidate messages address the needs of the future users. Another task of the SEGs is to inform the relevant industry groups of proposed development to ensure that all business requirements will be addressed. Created SEGs are: 1) the Payment SEG, 2) the Security SEG, the FX SEG and 4) the Trade Service SEG. The following text describes more about these SEGs.

Payment SEG

[r21] The Payment SEG is active on payment instruments like credit transfers, cheques, direct debit, debit & credit cards. Involved business actors are the initiators or beneficiaries of payments. Examples of actors are: 1) private and corporate customers, 2) financial institutions, 3) clearing houses and real time gross settlement systems (RTGSs), 4) payment 'factories' and 5) central banks.
Examples of business areas are:

- payment initiation: retail / commercial payments, communications between the ordering customer and its bank, etcetera;
- clearing and settlement: interbank transfers via correspondent banking or automatic clearing houses (ACHs), high value payments, low value bulk payments, RTGS, etcetera;
- cash management between various actors: account opening, standing orders, transaction and account information, advices & statements from the account servicing institutions to account owners, including reporting from the financial institution to the ordering & beneficiary customers, reconciliation, exceptions & investigations handling.

Security SEG

[r21] This group has validated messages for the funds industry. They are active on instruments like equities, fixed income, funds, and derivative.
Examples of actors are: 1) investment managers, distributors, transfer agents, fund administrators, 2) broker/dealers, 3) service bureaux, 3) custodians, 4) market data providers and 5) stock exchanges.

Examples of business areas are: 1a) initiation, 1b) pre-trade, 2) clearing & settlement and 3) security management. The latter include: account opening, standing orders, transaction and account information, advices and statements from account servicing institutions to account owners along the processing chain, queries and investigations.
According [r21] does UNIFI got a number of advantages compared with the precursor ISO 15022, which is only for securities messages. UNIFI:

- builds on the ISO 15022 data dictionary concept and registration infrastructure. It strengthens the monitoring by the industry;
- uses a more robust, syntax independent development methodology based on UML modeling of business processes and transactions;
- uses XML as the syntax for the actual physical messages;
- is in line with directions taken by other industries (UN/CEFACT).

FX SEG
[r21] The FX (Foreign eXchange) SEG evaluates candidate messages that support foreign exchange transactions. Examples of actors are: 1) investment managers, 2) trading portals, matching services providers, 3) custodians, 4) dealers, 5) money brokers, 6) industry associations (ISDA) and 7) application providers. The FX SEG includes business areas such as: 1) pre-trade: indication of interest, quotes, 2) trade: ordering, execution, allocation, affirmation, etc., 3) post-trade: confirmation, matching, assignment, novation, etc., 4) notification of trades to third parties, 5) trigger events, option exercises and 6) clearing and settlement, including netting and related reporting.

Trade Service SEG
[r21] The Trade Services SEG evaluates messages supporting transactions and business processes related to the trade finance business and financial supply chain management. Examples of services are: 1) open account trading (OAT), 2) reconciliation (A/R, A/P) and remittance data, 3) e-invoicing / EBPP (Electronic Bill Presentment and Payment), 4) purchase order, transport documents, 5) invoice financing and 6) pre/post-shipment and financing & factoring. The actors are: 1) private and corporate customers (treasurers), 2) financial Institutions and the 3) associations providing rules and master agreements, like IFSA and ICC.
Two proposals for development of messages have been made by the RMG.

- The 'Invoice Financing Request' from the Associazone per il Coporate Banking Interbancario (ACBI).
- The 'Trade Services Management' (also known as the 'TSU') from SWIFT.

ISTH (International Standards Team for Harmonization)
[r29] ISTH, consists of IFX (Interactive Financial eXchange Forum), TWIST, OAGi (Open Application Group) and SWIFT. They exist since 2003. The goal of ISTH was to define a single 'core payment kernel' for corporate-to-bank payment initiation and status messages. Thereby, it was important to avoid divergence and duplication of standards that already exist. The ISTH organizations agreed to use ISO 20022 and they committed to include the messages in their own existing set of standards. The core payment kernel was approved as the first ISO 20022 message standard in 2005. This kernel could be used by banks, corporates and vendors to streamline payment applications in order to take advantage of the straight through processing (STP) benefits. ISTH is also active in developing bank-to-corporate advices and statements.

Omgeo
This is a provider of post-trade and pre-settlement solutions. One of their projects was to enable Omgeo Central Trade Manager (Omgeo CTM) to send settlement notification messages via the SWIFTnet to custodians, sub-custodians and clearing agents for domestic and cross-border trades.

SWIFT (Society for Worldwide Interbank Financial Telecommunication)

[r29] SWIFT is a community for financial institutions since 1973. Their mission is to be the leader in communications solutions enabling interoperability between its members, their market infrastructures and their end-user communities.
This organization provides an equally named industry-owned standardized messaging service and interface software to approximately 8100 financial institutions.

SWIFT went live in 1977. They received a Computerworld Smithsonian Information technology Award for its work in the field of standardized financial telecommunication in 1991. The Interbank File Transfer (IFT) service went live on 1 July 1992. The service handles a range of bulk data transfers including mass payments, cheque truncation and internal reporting. The SWIFT Board transformed the Securities Board Task Force into the Securities Steering Council in 1998. Non-banks like investment managers and securities brokers could join the Council. SWIFT announced plans for two services that extend the reputation of financial institutions for trust and payments into the business-to-business domain in 2000. SIPN (SWIFT's secure IP network), SWIFTNet Link, SWIFTNet PKI and SWIFTNet Interact where deployed while new XML standards methodology would be developed. SWIFTNet went live in 2001, but the first SWIFTNet FIN message was sent on 15 August 2002. In the same year they where migrated to ISO 15022 standard. The community migrated to SWIFTNet IP platform in 2004. SAP joined over 300 solution providers including IBM, Microsoft and Oracle and announced that they will SWIFT-enable it's ERP. The SWIFTNet Trade Services Utility entered the pilot phase in 2006.
From the second half of 2007 are participants of the TARGET2 system able to use this Single Shared Platform (SSP), which gives access to a set of SWIFTNet messaging services: SWIFTNet FIN, FIN Copy, SWIFTNet InterAct, SWIFTNet FileAct and SWIFTNet Browse.

TWIST (Transaction Workflow Innovation Standards Team/ Treasury Workstation Integration Standards Team)
[r30] TWIST is a not-for-profit industry group that exists of corporate treasurers, fund managers, banks, system suppliers, electronic trading platforms, market infrastructures and professional services firms. The goal of this organization is to "connect the financial supply chain to the physical supply chain to release the value locked up in disjointed paper-based processes".
TWIST creates XML-based standards for 1) financial market transaction processing, 2) order management, 3) e-invoicing and payment processing, 4) bank account opening and closing, 5) billing of bank services, 6) credit management, 7) supply chain financing plus identity management and 8) security. E-invoicing and supply chain financing are examples of value added services 'on top of SEPA'. The standards of TWIST make straight through processing (STP) from end to end of financial processes possible. Thereby it does not matter in which way the processes are transacted, the service providers that are involved and the system infrastructure that is used.
TWIST also contributes in the management of the ISO 20022 standards for financial markets. They aim to make this the umbrella for its collection of standards.

UN/CEFACT (United Nations / Centre for trade facilitation and electronic-business)
[r21, r31, r47] This UN body was created in 1997 by the UN/ECE (Economic Commission for Europe) to improve worldwide coordination of trade facilitation across all industries. The focus of UN/CEFACT is on national and international transactions, through harmonization and simplification of processes, information flows and procedures.
They promote a 'technology neutral' business modeling with a central library of 'core components' and the use of the XML syntax. Almost similar to ISO 20022, but in this case it would encompass all industries.
ISO and UN/CEFACT came to an agreement to work together on harmonizing their specifications. The ultimate goal is that UNIFI provides the financial portion of the UN/CEFACT repository.
The International Trade and Business Procedures Group (TBG5) is a part UN/CEFACT. This group is responsible for finance standardization and trade facilitation. One of the goals TBG5 is to achieve interoperability between ISO 20022 (UNIFI) and ISO15000 (ebXML).

In 2004, TC68, TBG5 and SWIFT signed a Memorandum of Understanding (MoU) about e-business, which is a cooperation agreement between four standardization organizations (ISO, International Electrotechnical Commission, UN/ECE and International Telecommunication Union).
SWIFT made an analyses of the differences between the UNIFI and UN/CEFACT's CCTS (Core Component Technical Specification and made a proposal to bridge these differences in 2005.
In 2006, UNIFI and CCTS were transferred to the newly created TC68/WG and in charge of reviewing technical specifications of UNIFI.

2.1.2 Kinds of transactions

Although there are roughly three kinds of transactions, i.e. for credit, direct debit and card payment, Interpay [r69] lists the following kinds of transactions:

- POS (Point-of-sale) terminal transactions;
- ATM (Automated Teller Machine[4]) transactions;
- collections (direct debits) and reversals;
- business payments;
- converted transfers;
- acceptgiro (giro transfers);
- standing orders;
- telegiro (express payments);
- mass Payments.

[r43] The Bank for International Settlement (BIS) divides transactions in types of payment instruments and types of terminals.
Transactions per type of payment instrument are:

- credit transfer, which can be paper based or non-paper;
- direct debits;
- card payments with cards issued in the country, whereby cards can have a debit function, delayed debit function or credit function;
- e-money payment transactions, which can be by cards with an e-money function or through other e-money storages;
- cheques;
- other payment instruments.

Transactions can also be made from terminals in the country by cards issued in the country or outside the country. Furthermore, transactions can be made from terminals outside the country by cards issued in the country. Transactions per type of terminal are:

- cash transactions, which can be ATM cash withdrawals or ATM cash deposits;
- POS payment transactions;
- e-money card loading/unloading transactions;
- e-money card payment transactions.

2.1.3 Systems

This section describes systems that are responsible for financial transactions in Europe. The EU contains 25 different giro payment systems with national legislation, standards and access constrains [r80].

[4] in Dutch: "geldautomaat".

TOP

[r67] TOP is the domestic giro payment system of 'De Nederlandsche Bank' (DNB). It is a real-time gross settlement (RTGS) system, which means that payments are carried out immediately, irreversibly and individually, giving both transferer and transferee immediate certainty. TOP is used by financial institutions operating in the Netherlands. Examples of these financial institutions are banks and companies that offer clearing and settlement services, such as Equens and Euronext. The Dutch government and several of its institutions also use TOP. Although foreign central banks outside the euro area are not account-holders, they do have the option to channel their payment orders through the system of DNB. Table 2 lists RTGS systems of countries in Europe.

Member State	Name of system	Settlement agent	Location
National RTGS Systems			
Austria	Austrian Real-time Interbank Settlement System (ARTIS)	Oesterreichische Nationalbank	Vienna
Belgium	Electronic Large-value Interbank Payment System (ELLIPS)	Banque Nationale de Belgique/ Nationale Bank van België	Brussels
Denmark	KRONOS	Danmarks Nationalbank	Copenhagen
Estonia	Eesti Pank Real-Time Gross Settlement System (EP RTGS)	Eesti Pank	Tallinn
Germany	RTGSplus	Deutsche Bundesbank	Frankfurt
Greece	Hellenic Real-time Money Transfer Express System (HERMES)	Bank of Greece	Athens
Finland	Bank of Finland (BoF-RTGS)	Suomen Pankki	Helsinki
France	Transferts Banque de France (TBF)	Banque de France	Paris
Ireland	Irish Real-time Interbank Settlement System (IRIS)	Central Bank and Financial Services Authority of Ireland	Dublin
Italy	Sistema di regolamento lordo (BIREL)	Banca d'Italia	Rome
Luxembourg	Luxembourg Interbank Payment Systems (LIPS-Gross)	Banque centrale du Luxembourg	Luxembourg
The Netherlands	TOP	De Nederlandsche Bank	Amsterdam
Poland	SORBNET-EURO	Narodowy Bank Polski	Warschau
Portugal	Sistema de Pagamentos de Grandes Transacções (SPGT)	Banco de Portugal	Lisbon
Spain	Servicios de Liquidación del Banco de España (SLBE)	Banco de España	Madrid
Sweden	Riksbank's system (ERIX)	Sveriges Riksbank	Stockholm
United Kingdom	CHAPS euro	APACS, the UK payments association,	London
International RTGS Systems			
All	TARGET	ECB	

Table 2: RTGS Systems; source [r55, r12]

TARGET

[r11 r12,r51] TARGET stands for Trans European Automated Real-time Gross settlement Express Transfer system and consists of the national real-time gross settlement (RTGS) systems of the 12 euro area countries and of the ECB payment mechanism (EPM). Furthermore it contains the national euro RTGS systems of Denmark, Estonia, Poland, Sweden and the United Kingdom. These 18 systems (see the table above) are all interlinked in order to provide a uniform platform for the processing of euro payments.

The ECB created the TARGET platform to realize an efficient euro payment system for banks and central banks. The first operations of the system began on 4 January 1999, which is the same date as

the date of the launch of the euro. TARGET is with 1.9 trillion euros settled every day (2006), one of the three largest wholesale payment systems in the world (see table 3 for more figures). The other two are Fedwire (United States) and Continuous Linked Settlement (CLS), which is the international system for settling foreign exchange transactions.

What	When	# Payments	Value in €
processed national and cross-border payments	1 year (2003)	> 66 million	> 420 trillion
	1 year (2006)	> 83 million	> 533 trillion
payments processed in TARGET as a whole, i.e. domestic and cross-border payments taken together,	1 day (2003)	> 261 000[5]	Avg 1 650 billion
	1 day (2006)	> 326 000[6]	Avg 2 092 billion

Table 3: TARGET figures; source [r11]

TARGET has been developed because it:

- provides a safe and reliable mechanism for the settlement of euro payments,
- increases the efficiency of cross-border payments in euro, and
- serves the needs of the monetary policy of the ECB and to promote the integration of the euro money market.

Domestic and cross-border activities of TARGET are managed by the Governing Council of the ECB, who are assisted by the Payment and Settlement Systems Committee (PSSC) and its sub-group, the TARGET Management Working Group (TWMG). The responsibility of TWMG is to asses the performance of TARGET and also to review and propose the possible enhancements to the technical and organizational features.

TARGET2

[r51] TARGET2 is the successor of TARGET. It will go live on 19 November 2007. With this new system, the Eurosystem is aiming to:

- provide a harmonized level of service on the basis of a common technical platform;
- achieve a high level of cost recovery and have a single price structure applicable to both intra-national and cross-border payments;
- and to meet the new demands of users, as well as those resulting from the future connection to TARGET of the countries that joined the European Union on 1 May 2004.

Features of TARGET2 are:

- a single technical platform, also known as Single Shared Platform (SSP), in stead of a decentralized platform;
- TARGET-wide liquidity management services, e.g. prioritization of payments, liquidity reservation and active queue management;
- support to submit transactions with a debit time indicator;
- pooling of liquidity by grouping a number of accounts, e.g. settle a payment order if the amount is smaller than the sum of the liquidity available on all accounts in the group;
- six procedures for the settlement of ancillary systems (ASs) via a standardized interface;
- an online information and control module (ICM) to enable users to choose what information they want to receive and when;
- a TARGET2 directory to support system participants in their routing of payment instructions;
- compared with TARGET, a longer operational day (see table below).

[5] Note that 66 000 000 / 365 = 180 822 payments per day and 66 000 000 / 261 000 = 253 days
[6] Note that 83 000 000 / 365 = 227 397 payments per day and 83 000 000 / 326 000 = 255 days

	Time	Description
Daytime	6.45 a.m. - 7 a.m.	Preparations for daytime operations.
	7 a.m. - 6 p.m.	Day trade phase.
	5 p.m.	Cut-off for customer payment.
	6 p.m. + 15 min.	General cut-off for the use of standing facilities.
End of day	6 p.m. + 30 min.	Cut-off for the use of standing facilities on the last day of a minimum reserve period.
Start of day	6.45 p.m. - 7 p.m. (1)	Start-of-day processing.
	7 p.m. - 7.30 p.m. (1)	Provisioning of liquidity (from standing facilities, intraday credit, home accounts) until start of procedure for Ass.
Night-time window for AS	7.30 p.m. (1) -10 p.m.	Automated start of procedure message to set aside liquidity until start of cycle message of ASs, and ancillary system night-time processing (ancillary system settlement procedure 6.
	10 p.m. – 1 a.m.	Technical maintenance period of 3 hours. The system is shut down.
	1 a.m.-6.45 a.m.	Night-time processing (ancillary system settlement procedure 6).

Table 4: Operational day for TARGET2

Access to TARGET2 can be retrieved by direct and indirect participation, 'addressable BICs' and 'multi-addressee access'.
Direct participants have a RTGS account in the payment module (PM) of the SSP. This enables them to submit and receive payments directly to and from the system. Furthermore they could settle directly with their national central bank. Direct participants need a direct connection to SWIFT's secure IP network via their own SWIFT interface or via a SWIFT Service Bureau. SWIFTNet FIN service will be used for the exchange of payment information. The SWIFTNet services 'InterAct', 'Browse' and 'FileAct' are used for information and control service.
Indirect participants exist of supervised credit institutions that are established within the EEA (European Economic Area). Members of the area are Norway, Iceland and Liechtenstein. Indirect participants can only send and receive their payment orders from the system via direct participants.
Addressable BIC's exists of any direct participant's correspondent or branch that has a BIC. This group could send and receive payment orders to and from the system via direct participant. Their payment will be settled in the account of that direct participant in the PM of the SSP.
Multi-addressee access means that direct participants could authorize branches and other credit institutions belonging to their group, located in EEA countries, to channel payments through the direct participant's main account without its involvement by submitting and receiving payments themselves directly to and from the system.

SWIFTNet
SWIFTNet is an IP-based messaging platform of SWIFT. It includes the core store-and-forward SWIFTNet FIN service and three additional messaging services: SWIFTNet InterAct, SWIFTNet FileAct and SWIFTNet Browse. These services enable messaging between financial institutions and their industry counterparts, their end-customers or their market infrastructures.

EURO1
[r5] EURO1 is a system that has been developed by EBA Clearing that launched it in 1998. The system can be used for domestic and cross-border euro transactions between European banks. SWIFT supplied the messaging infrastructure and computing facilities where EURO1 is based on. Settlement takes place via a settlement account at the ECB at the end of the each day. The system meets the ten core principles of Systematically Important Payment Systems.

STEP1
[r5] STEP1 is a low-value payment system that has also been developed by EBA Clearing. It was launched in November 2000. The system has been designed to process a single cross-border payment in euros between STEP1 participants as well as the community of EURO1 banks.

STEP2
[r5] A third system of EBA Clearing, SIA S.p.A as a technology partner and SWIFT as a messaging partner, is STEP2, which is operational since April 2003. It is the first pan-European automated clearing house (PE-ACH) for bulk payments in euro. STEP2 is able to validate payment instructions and to route these instructions to the beneficiary banks as an automated settlement in EURO1/STEP1 at the beginning of the day. Participants in EURO1 and STEP1 can be a direct STEP2 participant. The system processes CREDEURO-compliant credit transfers, which means transactions up to 50,000 euro. It also meets the requirements of EC Regulation 2560/2001. This implies the use of International Bank Account Numbers (IBAN) and Bank Identifier Codes (BIC) to comply with the STEP2's straight-through processing (STP) criteria.

STEP2 SEPA Credit Transfer (SCT) Service
EBA Clearing completed the implementation of a SEPA Credit Transfer (SCT) Service on its STEP2 platform at the beginning of the third quarter of 2006. This service will provide generic functionality for sending payment instructions to any bank in Europe from the first of January 2008. Features are:

- that returns can be sent or received mixed with other payments or in separate files;
- that payment files will be sent on the settlement day (same-day cycle);
- that failed settlement instructions can be resend;
- that payments or files that are send up to n days ahead of the value date can be stored and further processed on the value date;
- that stored payments can be canceled or prioritized by using a web service;
- that participants could check online the status of payments.

STEP 2 SEPA Direct Debit (SDD) Service
This service is able to process direct debits in euro in SEPA and therefore uses the corresponding schemes of UNIFI. It runs also on the STEP2 platform and is planned to be tested half 2007 and ready in 2008. The service is part of EBA Clearing's Multi-Purpose Pan-European Direct Debit (M-PEDD) project. This means that the service will be extendible and may include the processing of payments of current domestic direct debit schemes or 'Electronic Bill Presentment and Payments schemes'.

CLS (Continuous Linked Settlement)
[r6] CLS is a multi currency system for interbank funds transfer, i.e. a RTGS for multiple currencies. The system exists since 2002 and can only be used by the CLS Bank, the central banks in whose currencies CLS settles, and members of CLS Bank. The CLS Bank is owned by 71 large financial groups throughout the US, Europe and Asia Pacific. The average daily turnover in global foreign exchange transactions is US$ 2 trillion.

EPM (ECB Payment Mechanism)
EPM is a payment-processing tool that operates as an integral part of TARGET and is connected to national systems of the EU. It operates technically in the same way as the other RTGS systems within TARGET. EPM will be used by 1) the ECB, but also by their customers: 2) the non-EU central banks, 3) European and international institutions and 3) clearing and settlement organizations like CLS (settlement for multiple currencies) and EBA (clearing).

2.1.4 Payment process

This paragraph describes several payment processes.

Front end

The payment process of a financial transaction can be triggered by several kinds of transactions, which can have different interfaces. Examples of interfaces are the menus of POSs, ATMs, phones or web interfaces. Figure 3 shows an interface of the latter, where a payer can choose between three possibilities for online payment. The credit transfer can be domestic (per bank/accept giro) or cross border. In case of the latter a currency should be selected, one of the payers accounts, the account number/IBAN of the beneficiary and how the costs will be divided. These can be shared, for the payer or for the beneficiary.

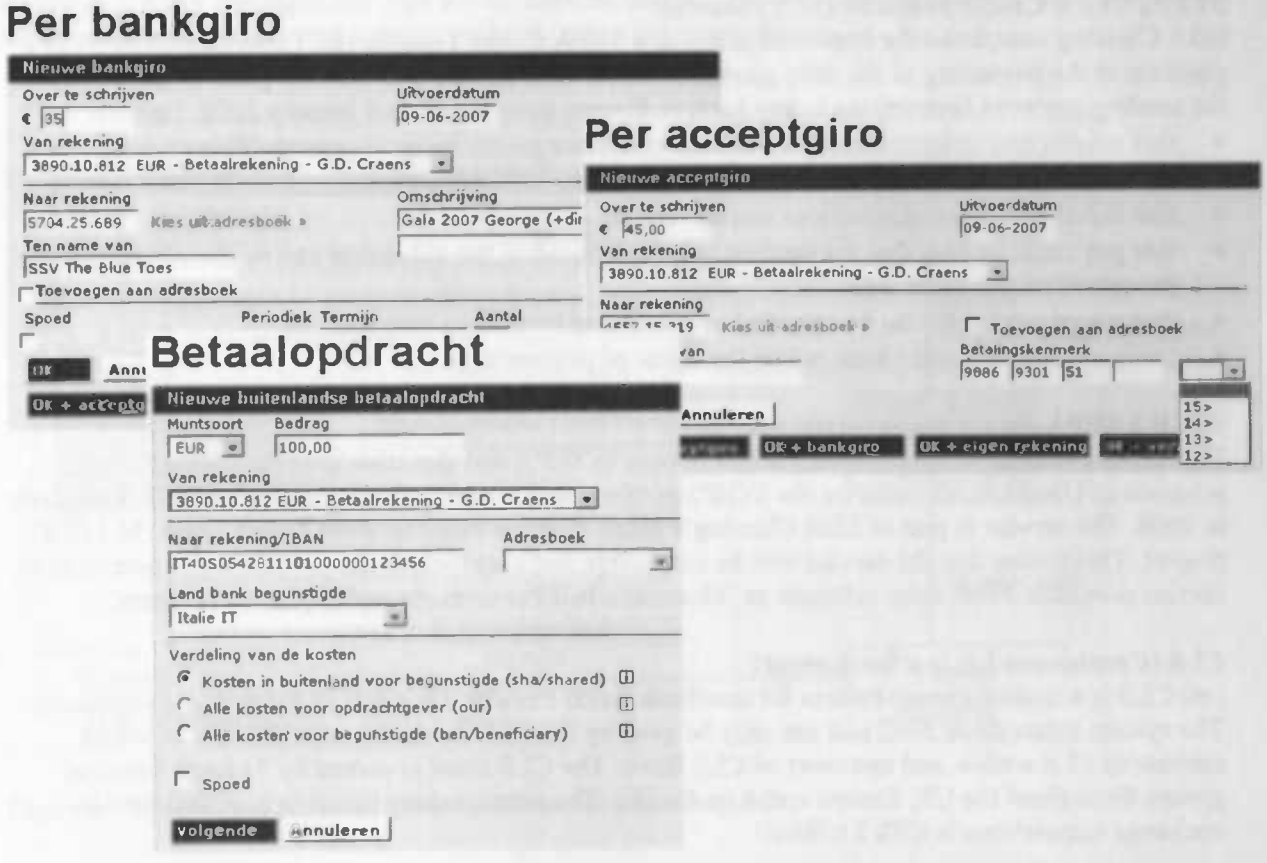


Figure 3: Front end for bank clients; source [r42]

National

As described above, the first step of the payment process could start with an order of client A to a bank (e.g. ING) to pay client B (see figure 4).
The second step is that bank A will sent mutation information to an Automated Clearing House (ACH) like for example Equens. The mutation information will be sent further to bank B. Bank B will inform the beneficiary about the transaction and the ACH will provide process information to bank A.
The third step is clearing and settlement on a regular basis. [r61] Clearing is the process of transmitting, reconciling and confirming payment orders, and establishing a final position for settlement (either based on individual transactions or bundles of transactions). Settlement means the transfer of funds between the payer and the payee (and thus between the payer's bank and the payee's bank). Thus during the settlement process the money will be transferred and during the clearing process it will not.

If for example ING has to pay Fortis 3 million euro's and Fortis has to pay 2 million euro's to ING, only 1 million will actual transferred from ING to Fortis. In this example will Equens send a settlement instruction for a value of 1 million to the settlement system TOP of DNB.
The fourth step is that the system of DNB will inform the banks with settlement information.

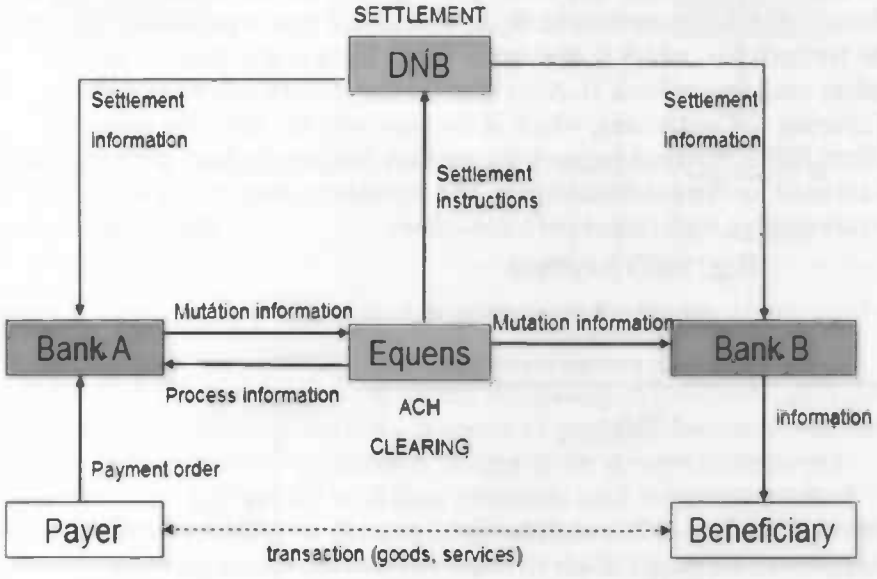


Figure 4: Domestic payment process [r80]

The transaction time is relative short and the costs are relative low (cents per transaction). Every country has its own standards, rules, infrastructure, etc. for domestic payment.

The figure below represents a more detailed version of the payment process. An example of a not unimportant detail is that the solvability of the payer has to be checked before a payment could be accomplished.

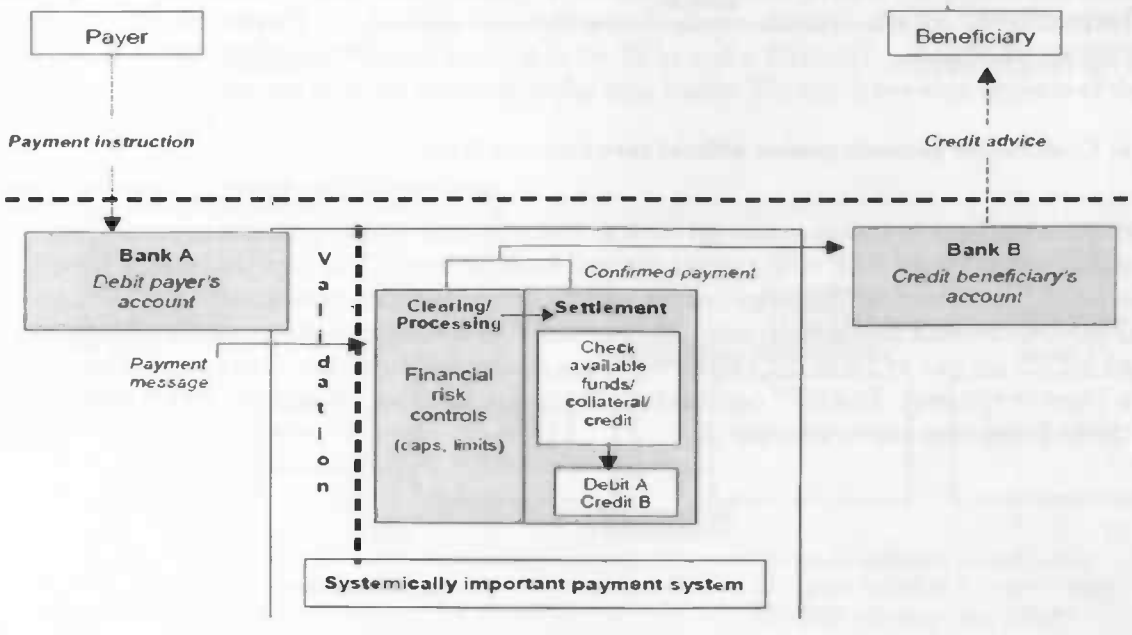


Figure 5: Lifecycle of a payment (Credit Transfer) [r44]

International

International payment can be divided in payment in euros and payment in another currency. The payment process of the latter will not change with the introduction of SEPA.

Figure 6 illustrates the scenario that goods or services from the Netherlands will be supplied to Austria and there will be paid in United States Dollars. First the payer sends a payment order to bank B. This bank will send a message to a correspondent bank in a country where they use the concerned currency and to the bank of the beneficiary. The correspondent bank of Austria will send a payment message to the correspondent bank of the Netherlands, which is also in the USA. Bank A will process the messages from its correspondent bank and of bank B. After that the beneficiary will be informed. Note that there is no central clearing and settlement, which is the case with the domestic payment process. There are minimal three banks involved for each transaction, but mostly there are more banks involved. SWIFT messages are used for the communication. The transaction time is relative long (a number of days) and the cost are relative high (euros per transaction).

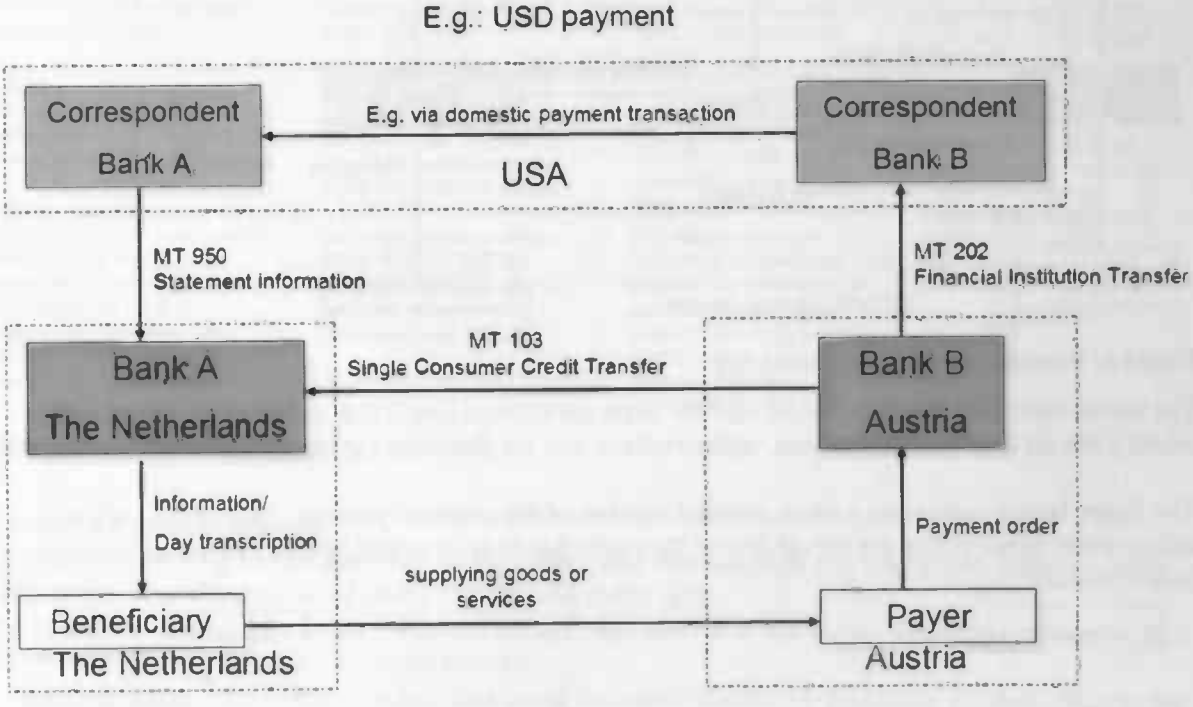


Figure 6: Cross border payment process without euro currency [r80]

The correspondent bank is always a national bank in case of a cross border payment in euros. This bank communicates via the ECB with another national bank. In figure 7 the clearing house will be represented by EBA Clearing. This organization uses STEP2 as pan-European automated clearing house (PE-ACH) for bulk payments in euro. The national *real-time gross* settlement (RTGS) systems: TOP and ARTIS are part of TARGET (Trans European Automated Real-time Gross settlement Express Transfer systems). TARGET consists (as already described) of the national RTGS systems and of the ECB payment mechanism (EPM).

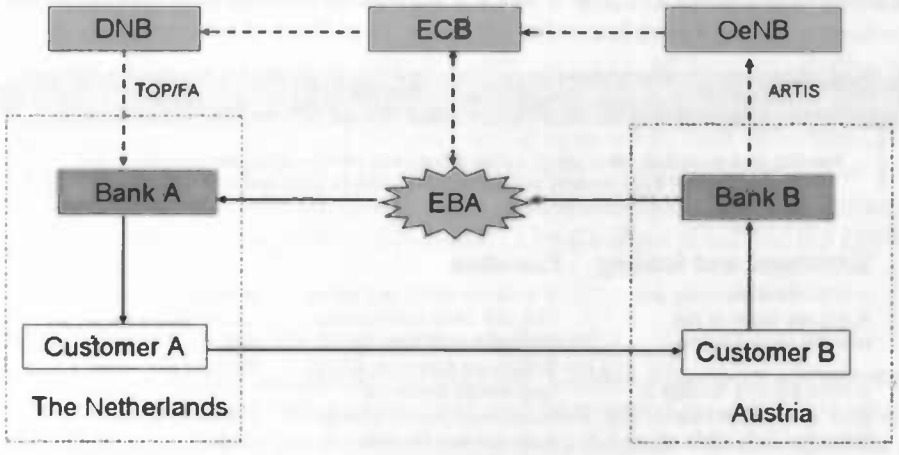


Figure 7: Cross border payment process in euros via EBA [r80]

[r53] EPM provides features for real-time processing of payments (see figure 8). It contains a Central Accounting System (CAS) for the settlement of payments between customers. Furthermore it can send required confirmation and payment messages to the concerned customer.
Customers of the ECB could send their payments (and inquiries) via the SWIFT FIN service. Moreover they could receive debit or credit advices and end-of-day statements via SWIFT. Communication for payment orders can also be made via another pre-agreed channel, in case an account holder does not have a SWIFT connection.
[r53] The ECB is through the EPM also a settlement service provider for the Euro1 system. EBA has a settlement account in EPM, because they are responsible for Euro1. The settlement process uses time tables and operational procedures.
Final balances are calculated and clearing banks are informed periodical. Banks with a debit position, also known as short banks, send a cross border TARGET payment order to their national central bank. The ECB sends a confirmation to EBA. EBA monitors the received messages from the ECB to verify that all expected payments have been credited to its accounts. EBA notifies the ECB to pay the long banks after the short banks have paid in and sufficient funds are available for the 'long' banks. This instruction debits the EBA settlement account in the EPM and a TARGET payment is sent to the national central banks that held the accounts of the long banks. The result is a zero balance of the settlement account.

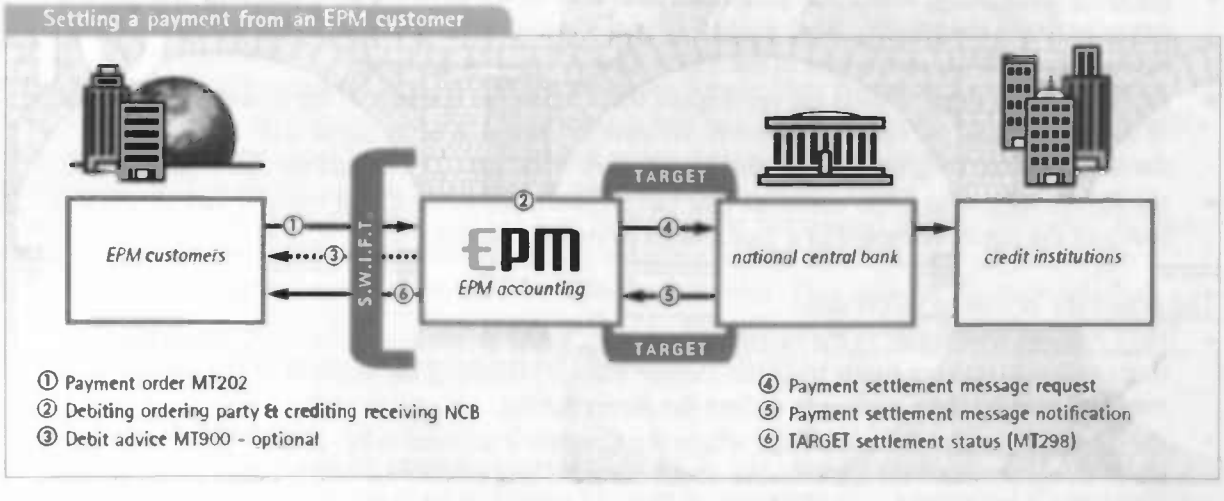


Figure 8: ECB payment mechanism (EPM) transaction; source [r53]

As already mentioned is CLS one of the users of EPM. The CLS settlement process is described in the figure below.

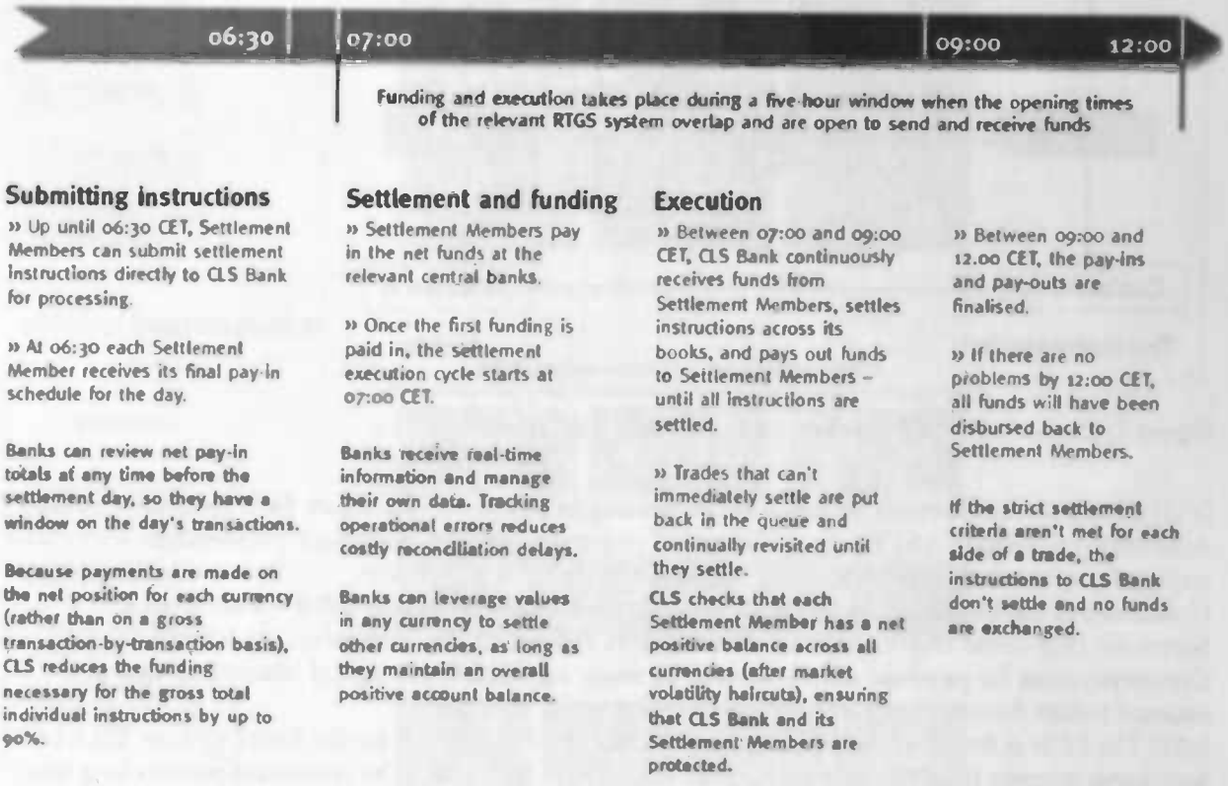


Figure 9: CLS settlement process; source [r6]

The SEPA payment process

[r62, r63] Figure 10 gives an overview of the contractual relationships and interaction between main actors in the payment process of credit transfer and direct debit.
This overview, also known as the "4-corner model", has similar to the earlier described processes, four main actors. These are in case of credit transfer:

- *the Originator*, who is the customer that starts the credit transfer by providing an instruction to the *Originator Bank*;
- *the Originator Bank*, who is the participant that acts on the payment instruction by making the payment to the *Beneficiary Bank* according to the information provided in the instruction and in accordance with the provisions of the SEPA Credit Transfer *Scheme*;
- *the Beneficiary Bank*, who is the participant that credits the account of the *Beneficiary*, according to the information provided in the received instruction and in accordance with the provisions of the *Scheme*. The *Originator Bank* and *Beneficiary Bank* may be one and the same participant;
- *the Beneficiary*, who is the customer that can be identified in the credit transfer instruction and receives the funds by means of a credit to its payment account.

The actors are in case of direct debit:

- *the Creditor*, who receives the mandate from the *Debtor* to initiate collections, which are instructions to receive funds from the *Debtor Bank* by debiting the account of the *Debtor*. The mandate enables the *Creditor* to collect the direct debits;
- *the Creditor Bank*, who is the bank where the *Creditor's* account is held and it is the bank that made an agreement with the *Creditor* about the rules and conditions of a product based on the SEPA Direct Debit *Scheme*. The bank receives and executes instructions from the *Creditor* (that are based on this agreement) to initiate the Direct Debit Transaction by forwarding the collection to the *Debtor Bank* in accordance with the *Rulebook* (see source [r62, r63]);

- *the Debtor Bank*, who is the bank where the account to be debited is held and which has concluded an agreement with the Debtor about the rules and conditions of a product based on the *Scheme*. On basis of this agreement, it executes each collection of the direct debit originated by the *Creditor* by debiting the *Debtor's* account, in accordance with the *Rulebook*;
- *the Debtor*, who gives the mandate to the *Creditor* to start collections. The *Debtor's bank* account is debited in accordance with the collections initiated by the *Creditor*. If the customer of the *Creditor* in the underlying contract is not the same person as the *Debtor*, then this person is called in the *Rulebook* the '*Direct Customer*'. The *Creditor Bank* and the *Debtor Bank* are participants in the *Scheme*.

Indirectly other parties are involved too, these are:

- *the Clearing and Settlement Mechanisms* (CSMs). Examples are automated clearing houses or other mechanisms such as intra-bank and intra-group arrangements and agreements between participants. Appendix 6 contains an activity diagram that describes the process of a single credit transfer between two payment clearing agents and a payment settlement agent;
- *the Intermediary Bank*, who offers intermediary services to *Debtor Banks* and/or *Creditor Banks*. This occurs for example in cases where they are not direct participants in a CSM.

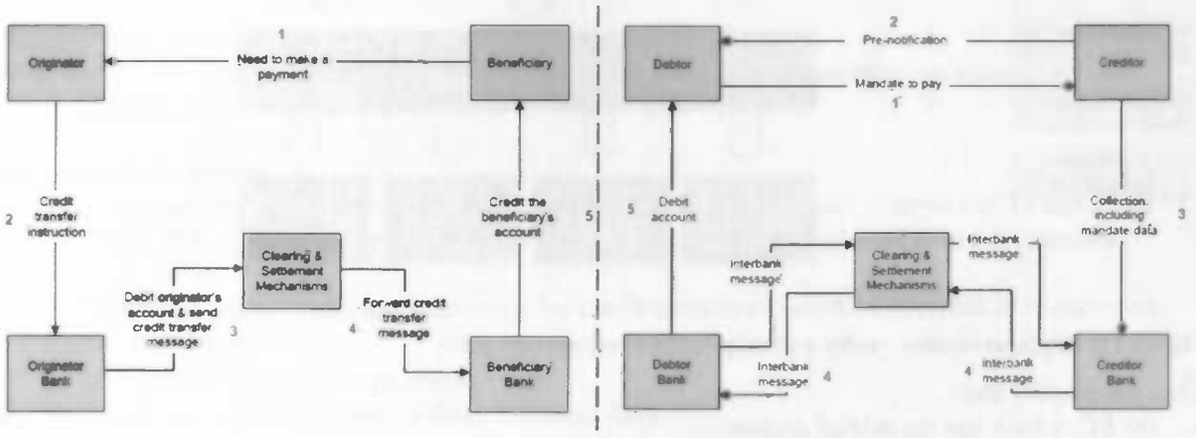


Figure 10: The four corner model. Left SCT and right SDD (contractual); source [r60]

2.2 SEPA

2.2.1 Introduction

[r60] The European Payments Council (EPC) defines the Single Euro Payment Area (SEPA) as an area where citizens, companies and other economic actors will be able to make and receive payments in euro, within Europe, whether between or within national boundaries under the same basic conditions, rights and obligations, regardless of their location.
The vision of the European Commission (EC) on SEPA can be formulated as: "the integrated market for payment services, which is subject to effective competition and where there is no distinction between cross-border and national payments within the euro area. This calls for the removal of all technical, legal and commercial barriers between the current national payment markets so that these become a single 'domestic' payments market for the whole euro area. [r56]".

2.2.2 SEPA players

[r18] To realize SEPA it will be important that the EC, ECB, EPC and all other involved parties in Europe will give their commitment on it. Figure 11 gives an overview of the roles and responsibilities.

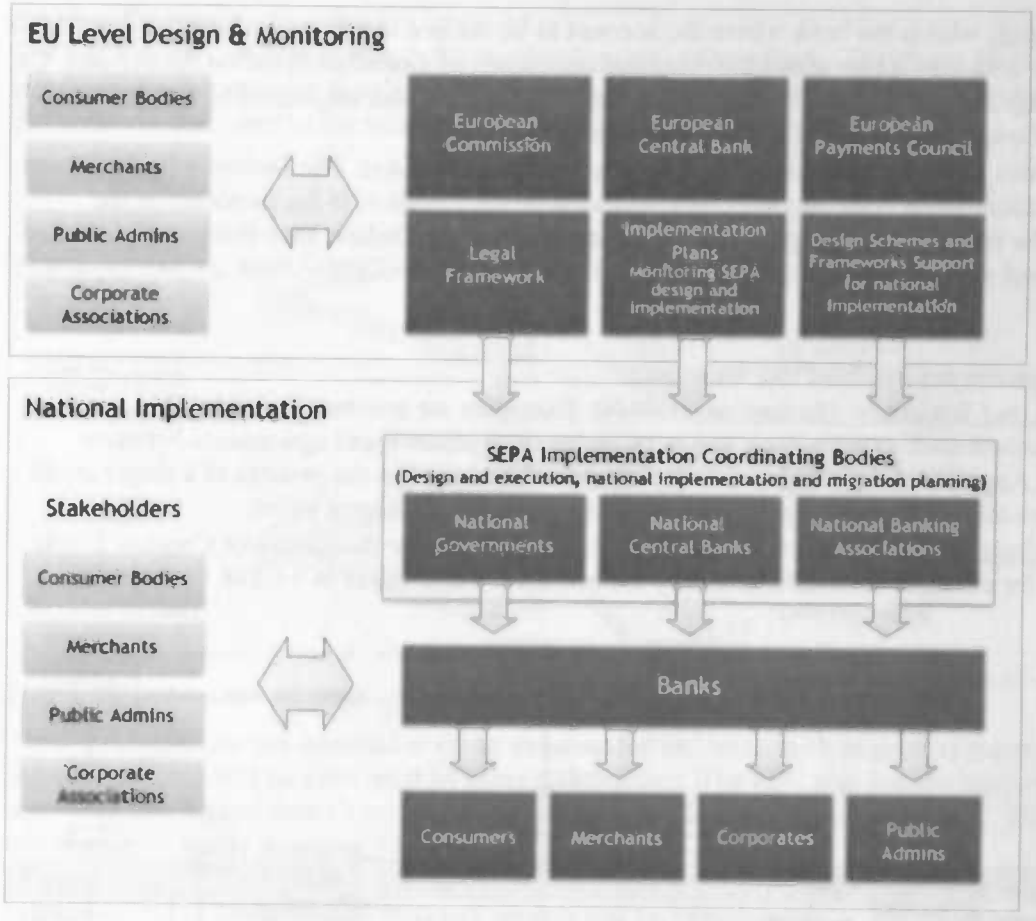


Figure 11: Implementation - roles and responsibilities, source: [r18]

Thus the players are:

- the EC, which has the role of decider;
- the ECB, which has the role of regulator;
- the EPC, which is created by the European banking community, has the role of designer. They delivered specifications (rulebooks) and principles (frameworks);
- banks and financial institutions. Banks from 28 countries are now associated with the work on SEPA;
- non-euro zone banks, which are indirectly concerned through their euro zone subsidiaries;
- automated clearing houses and card processors, which must provide the compliant clearing and processing platforms. Table 5 lists a number of clearing house of Europe;
- banks wholesale clients (consumers);
- banks retail clients (corporates);
- external service providers like consulting firms, IT companies, etcetera.

	Clearing house	Country
bacs	BACS (Bankers Automated Clearing Services)	United Kingdom
BBS	BBS (Bankenes BetalingsSentral)	Norway
bgc	BGC (Bank Giro Centralen)	Sweden
CEC	CEC (Centre for Exchange and Clearing)	Belgium
DIAS	Dias Interbanking Systems	Greece
EMZ	EMZ (Elektronische Massenzahlungsverkehr)	Germany
equens	Equens (Interpay and Transaktionsinstitut, since 21-09-06)	The Netherlands
GSIT	GSIT (Groupement pour un Système Interbancaire de Télécompensation)	France

	Iberpay	Spain
	KIR (Krajowa Izba Rozliczeniowa)	Poland
	PBS (Pengeinstituter BetalingsServices / Payment Business Service)	Denmark
	SIA	Italy
	SIC (Swiss Interbank Clearing)	Switzerland
	STET (Interbank Technology Services / Technological Systems for Exchange and Processing)	France
	VOCA	United Kingdom
	Fin Force	Cross border
	EBA Clearing (Euro Banking Association). It counts 70 shareholder banks and, through its EURO1, STEP1 and STEP2 systems, it offers both high-value and low-value clearing and settlement services to a wide community of banks in the European Union.	Cross border

Table 5: Clearing house

2.2.3 Components

[r54] The SEPA project addresses four major components: payment instruments, infrastructures, standardization, and a legal framework.

Payment instruments

The EPC created two new payment schemes: 1) the SEPA credit transfer schema (SCT) and 2) the SEPA direct debit schema (SDD). Beside these schemas they also developed a card framework.

The SCT schema defines rules and processes for credit transfers in euros. Examples of features are:

- SEPA-wide reachability;
- no limit on the value of the payment;
- the maximum settlement time is three business days;
- the scheme is separated from the processing infrastructure;
- the International Bank Account Number (IBAN) and the Bank Identifier Code (BIC[7]) are used as account identifiers.

The SDD schema also defines rules and processes for transfers in euros, but this time for direct debit. In case of SCT the payment is initiated by the payer and in case of SDD the payment is initiated by the payee. This means that the payer permits the payee to take money directly from his bank account. The SEPA card framework (SCF) refers to a set of principles about card payment, which issuers, acquirers, card schemes and operators will have to adapt to.

Infrastructures

Principles about which infrastructure provider will support the SEPA credit transfers and direct debit schemes are defined with the SEPA clearing and settlement framework. The aim of this framework is to ensure 1) reachability of all euro area banks and 2) a separation of scheme and infrastructure. TARGET2 is one of the systems that will be responsible for payment in SEPA. Other systems that are responsible for the payments in SEPA are the systems that provide the STEP2 SEPA Credit Transfer (SCT) Service and STEP 2 SEPA Direct Debit (SDD) Service.

[7] BIC is also known as SWIFT address or SWIFT code, which is defined in ISO 9362.

Standardization
The EPC uses a common approach for the development of standards. They have identified requirements that describe the data elements to be exchanged between financial intermediaries. These elements are described in the rulebooks for SCT and SDD (see figure 12).

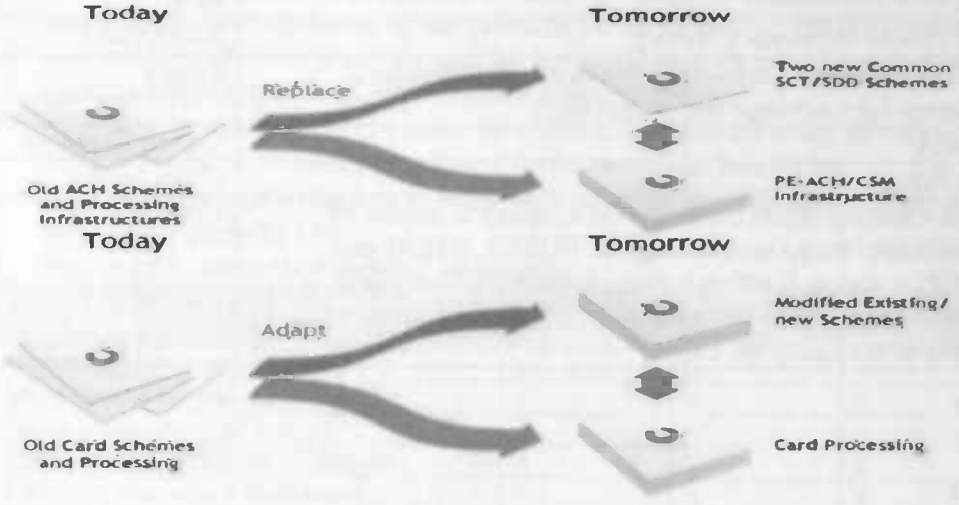


Figure 12: Old vs. new schemas

These rulebooks are not intended to cover value-added products, such as priority payments and e-invoicing, which will be developed by other stakeholders. The EPC also has the business requirements translated into logical data elements, which can be found in the SEPA Data Model. This model contains three layers (see table 6).

Layer	Describes	File
Busines process layer	Describes the business process and communication needs between parties, i.e. business rules, requirements and related data elements.	Rulebook documents
Logical data layer	Describes the data flow and logical messages, i.e. dataset, their attributes and inter-relationship	Implementation guideline documents
Physical data layer	Describes the physical representation of logical messages in a chosen syntax (XML) and chosen message standards. SEPA uses the UNIFI Direct Debit and the UNIFI Credit Transfer XML standard.	Repository of www.iso20022.org

Table 6: SEPA data model; source [r62,r59]

Legal framework
The proposed Payment Service Directive (PSD) has been founded to ensure that the same legal framework will be applied to all payments made within Europe. The proposal consists of three parts:

- the right to provide payments services to the public;
- transparency and information requirements;
- the right and obligation of users and providers of payment services.

2.2.4 Time line

[r54] The SEPA project can be divided into three phases: 1) design & preparation, 2) implementation & deployment and 3) co-existence and migration. The first phase involves the design of 1) credit transfer and direct debit schemes, 2) the frameworks for cards and clearing and 3) settlement of infrastructures. Although it is not visible in figure 13, it began in 2004. The second phase began in 2006 and is planned to continue until the begin of 2009. During this phase the SEPA instruments, standards and infrastructure will be prepared to roll-out and tests will be executed. In the third phase the national payment schemes will coexists with the new SEPA schemes. The critical mass of

transactions should be migrated before the end of 2010. However, according [r80] the migration phase will end in '201X', thus between 2010 and 2020.

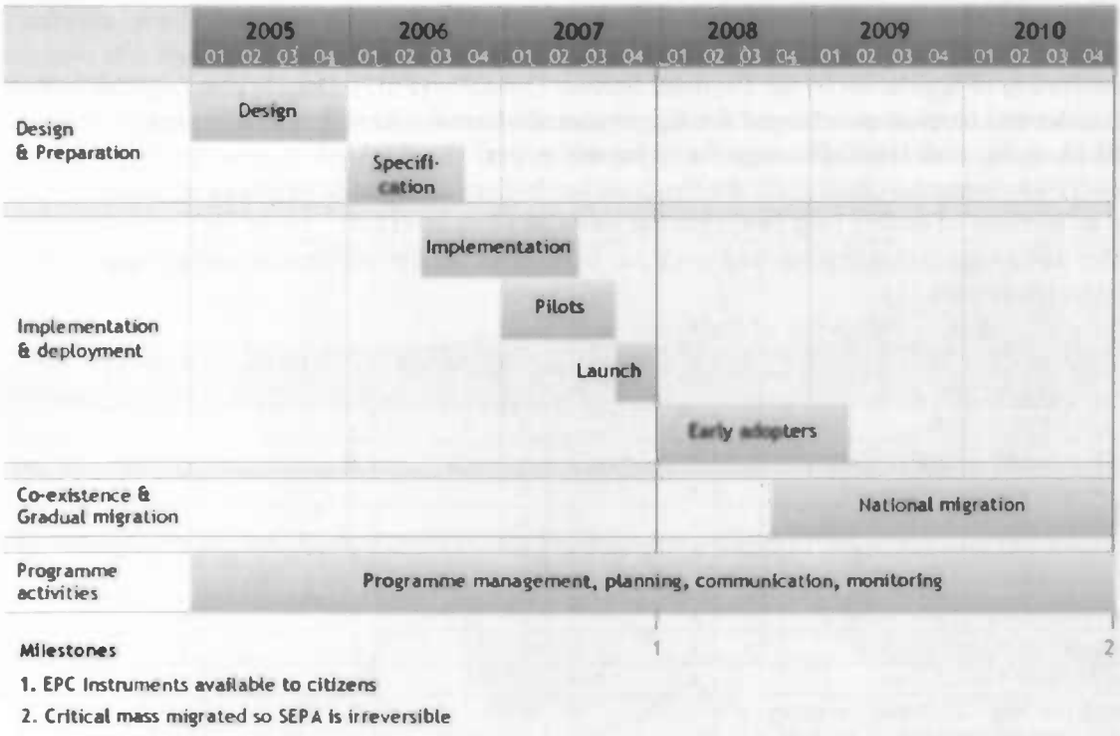


Figure 13: Timeline for SEPA; source: [r18]

2.2.5 Advantages

According to the EPC will SEPA enhance competition, promote more efficient payment systems and systemically strengthen the euro [r18].

- [r56, r80] Introduction of SEPA will approximately save the European market 50 – 100 billion euros per year. These potential savings will be caused through dematerialization of a wider transaction processing chain.
- Source [r85] states that product standardization and consolidation of payment infrastructures will maximize economies of scale. If for example all unit cost will be cut to 20% above the current best practice in the EU, it would generate 10 billion euros of additional profit.
- [r18] Another advantage of SEPA is the possibility for consumers to reach all accounts SEPA-wide from one home country account. Payment cards will be more widely accepted, displace cash and improve customer safety and security.
- Merchants will be able to simplify there back office processes, because they could accept more often payment cards.
- Faster settlement and simplified processing will improve cash flows and reduce costs. This will be an advantage for small and medium enterprises.
- Large merchants and corporates can save money, because they only have to use a single file in a common format for financial transactions throughout SEPA.
- Standardized schemes make it possible for governments and public administrations to provide better services to citizens at home and abroad.
- Banks could develop innovative products, enter new markets and win new relationships.
- Suppliers of payment products have an opportunity to build low cost technology products and services which will serve the SEPA market.
- [r46] Because the transaction time will be limited, it will be harder for banks to do unfair practices such as value dating, which is an advantage for consumers. Value dating means that a financial institution does not make received money from a transaction in its system directly available. A disadvantage is that consumers do not know when the transmitted money will be accessible and

that prices are opaque. A reason for value dating is that banks earn interest on the money transferred, paid, withdrawn or received. For example in 2005 it took 3 to 5 days for electronic money transfers to move money from one account to another. Due to these delayed payments, banks earned £30 million of interest in the UK.

- [r46] It will be harder or impossible to 'double-charge' in cross-border transfers, due harmonization of legislation by the Payment Service Directive (PSD). Double charging means that both sender and receiver are charged for the same credit transfer. Less double charging or no double charging at all is an advantage for payee and payer.
- Appendix 11 names the opportunity for financial institutions to lay in at large extent, which could mean an increase of quality (e.g. better performance and less costs).
- Another advantage is that less tailored work has to be done for the implementation of payment products. (Appendix 11).
- Source [r68] states the following advantages:

New SEPA Benefit	Consumers	Merchants	Corporates	Public Administrations
Common SEPA standards	x	x	x	x
More options for making payments	x	x	x	x
One account serves all Europe	x	x	x	x
Transparency and clarity of charges	x	x	x	x
Common processes with reduced costs/complexity	x	x	x	x
Predictable execution times	x	x	x	x
New SEPA Direct Debit Scheme	x		x	x
Common Scheme Governance	x	x	x	x
Transaction pooling / STP[8] multi country operations, better liquidity management		x	x	
End to end automated processes		x	x	x
EU wide acceptance of payment cards		x	x	x
Increased card transactions through cash displacement	x	x	x	x
Reduced fraud	x	x		x
Lower cost terminals		x		x
Single terminal footprint		x		x
Increased choice of processing services		x	x	x
Lower cost standard software and services		x	x	x

Table 7: SEPA benefits to bank customers; source [r68]

2.2.6 Disadvantages

This paragraph mentions disadvantages of the introduction of SEPA. A disadvantage for consumers is the use of a longer account number (IBAN). A longer number also means a higher chance to type a wrong symbol. This means 1) more work and cost for banks, 2) higher charges and wrong payments for the ordering of a customer and 3) delayed payment or no payment at all for the beneficiary customer. However, if a person uses a contact list with these account numbers on e.g. a website of a bank, than there is no risk of typing a wrong symbol anymore.
[r61] Financial institutions have to invest. They have to adapt their treasury/liquidity management to the new strategic situation. Furthermore they have to adapt their front/middle/back offices and they have to adapt their payments processes for compliance.

[r56] SEPA payment products should according the EC be competitive with the current best of breed products. Best of breed means in this context the economically most efficient product/service design taking into account all stakeholders' cost and benefits and also future development needs. However,

[8] STP stands for straight-through-processing (can be applied to the bank-to-bank chain only or on an end-to-end basis). It means: 1) value-added services offered before payment 2) processing the payment and 3) value-added services offered after payment.

NIBESVV (Nederlands Instituut voor het Bank-, Verzekerings- en Effectenbedrijf) states that UNIFI will not be a collection of best practices. An example is that Central Bank Reports (CBR[9]) might disappear.
Furthermore will countries with a low level of costs, like the Netherlands, have to encounter an increase of costs. Existing investments will be written off earlier, which means a waste of capital (see appendix 11).
The introduction of SEPA will also have the consequence that small banks have to outsource the processing of payments to bigger banks or have to quit (see figure 14). This means in case of the former that for example the SNS bank has a dependency on the ING bank. Europe counts approximately 7000 banks[10] that process 64 billion payments per year.

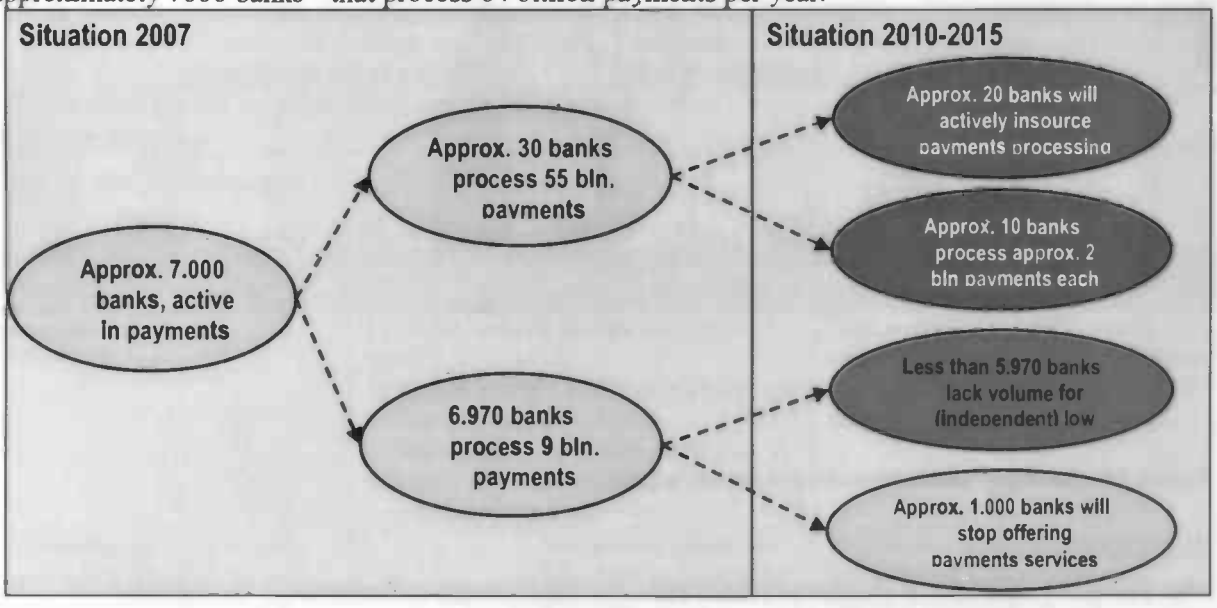


Figure 14: More competition means less banks [r61]

The increase of competition between financial organizations, in case of the soft and hard competition scenario, could be a disadvantage for most countries in Europe, because it could reduce the revenues. Figure 15 shows that EU12 banks could end up with 18 – 29 billion (38%-62%) less in the revenue of direct payments for the period after 2010. Without SEPA the estimated baseline of payment revenues will include an increase in the EU12 from €31 billion to €47 billion until 2010. Soft competition means in this context that payment prices across countries should converge by 2010 on the average price level of the three cheapest countries for each payment instrument. The scenario of hard competition means that the price level will be based on the cheapest country. Figure 15 contains also estimates of countries in the non-eurozone, because enterprises and citizens of this area will be impacted by SEPA competition if they open up accounts in the eurozone. The United Kingdom (UK) and Austria are included because of their dynamic market. Besides, the UK has the most important non-euro market in Europe and Austria considers itself to be a gateway to Eastern Europe. Sweden and Poland are added to diversify the geographical coverage. It stands out that the introduction of SEPA has only a positive effect for the Netherlands in case of the soft competition scenario compared with the baseline.

[9] More about CBR can be read in the next chapter.
[10] This number has been retrieved from source [r61]. Source [r83] lists a total of 6,313 institutions that offer payment services to non-MFI's (Monetary Financial Institution) in the euro area and for the whole EU this number is 8,738. Both counted for 2005. Each institution is counted once, irrespective of the number of offices it maintains in a country. Examples of institutions are: the central bank, credit institutions, branches of euro area and EEA-based credit institutions and branches of non-EEA-based banks.

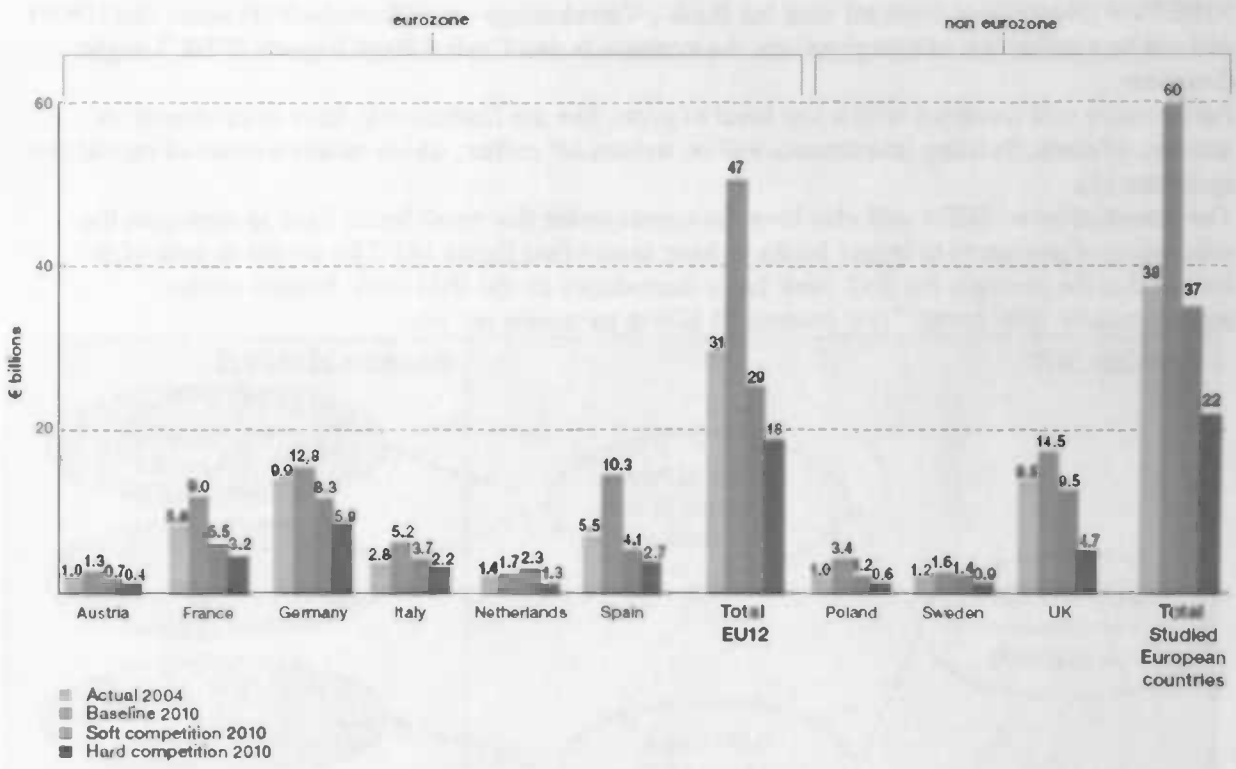


Figure 15: Total payments revenue per scenario [r48]

2.3 Summary

The first paragraph of this chapter provided financial background information. It described organizations that are related to SEPA and financial standardization in general. These were: EACT, EBA, ECB, ECBS, EPC, ISO, SWIFT and TWIST. Furthermore it gave an overview of kinds of transactions, which could roughly be divided into transactions for credit, direct debit and card payment. Also other classifications are described. After that, the systems that are responsible for financial transactions in Europe were described. These are: TOP, other RTGS systems of Europe, TARGET, TARGET2, SWIFTNet, EURO1, STEP1, STEP2, CLS, EPM and the systems for the services 'STEP2 SEPA SCT/SDD'. The paragraph ends with a description of payment processes, which are divided into national, international and SEPA payment processes.
The second paragraph described SEPA. The players, components, time line, advantages and disadvantages were treated. The tables on the next page summarize the advantage and disadvantages of SEPA.

Advantage	Consumers	Corporates / merchants	Banks
Cost savings	SEPA will approximately save the European market 50 – 100 billion euros per year.		
	More competition between banks and price transparency will lead to a reduction of transaction costs.	1) Faster settlement and simplified processing will improve cash flow and reduce costs. 2) Lower terminals cost.	Lower cost of standard software and services.
Increased interoperability, e.g. one account serves all Europe	1) All accounts SEPA-wide from one home country account. 2) More widely accepted payments cards will displace cash improving customer safety and security.	Merchants will be able to accept payment cards from all SEPA countries and back office processes will be simplified.	Less tailored work has to be done for the implementation of payment products.
Reduced fraud / unfair practices	Less value dating and double charging.		

Table 8: SEPA advantages

Disadvantages	Consumers	Corporates / merchants	Banks
Longer account number (IBAN).	1) Higher chance to type a wrong symbol. 2) Higher charges and wrong payments for the ordering of customer. 3) Wrong IBAN will cause delayed payment or no payment at all for the beneficiary customer.		Higher chance to type a wrong symbol by clients could mean more work and cost for banks in case of more incorrect account numbers.
Investments have to be made and existing investments will be written off earlier, which means a waste of capital. Countries like the Netherlands with a low level of costs will encounter an increase of costs.		Presumably Points Of Sale devices have to be replaced.	1) Treasury/liquidity management has to be adapted to the new strategic situation. 2) Front/middle/back offices have to be adapted. 3) Payments processes have to be adapted for compliance.
UNIFI will not be a collection of best practices (source: NIBESVV).			Central Bank Reports (CBR) might disappear (source: Innopay).
More competition.			1) Small banks have to outsource the processing of payments to bigger banks or have to quit; 2) EU12 banks could end up with 18 – 29 billion (38%-62%) less in the revenue of direct payments for the period after 2010.

Table 9: SEPA disadvantages

3 The UNIFI (ISO 20022) Standard

This chapter describes what UNIFI is, why it has been developed, which kind of messages do exist and how these messages are implemented. After the introduction an explanation of XML and XSD (XML Schema Definition) will follow first, because the financial messages of the physical layer are described in XSD.

3.1 Introduction

[r21] ISO 20022 is also known as the UNIversal Financial Industry message scheme (UNIFI). It can be used by the financial industry as a platform for the development of messages in a standardized XML syntax. To realize this platform it uses UML (Unified Modeling Language) to describe the financial areas, business transactions and associated message flows. One advantage of UML is that it is syntax-independent. After the messages are described in UML, they will be converted into XML schemas with aid of XML design rules.
UNIFI is aimed to improve the communication interoperability between financial institutions, their market infrastructures and their communities.
The document title of the standard is: "ISO 20022 Financial Services - UNIversal Financial Industry message scheme" and has been divided into five parts (see table 10 and [r72 – r76]).

Part	Name	Describes
1	ISO 20022-1: International Standard	Overall methodology and format specifications for inputs to and outputs from the ISO 20022 Repository
2	ISO 20022-2: International Standard	Roles and responsibilities of the registration bodies
3	ISO/TS 20022-3: Technical Specification	ISO 20022 modeling guidelines
4	ISO/TS 20022-4: Technical Specification	ISO 20022 XML design rules
5	ISO/TS 20022-5: Technical Specification	ISO 20022 reverse engineering

Table 10: UNIFI document parts

The UNIF platform is available to all developers of message standards. Organizations that already proposed development projects to UNIFI are SWIFT, TWIST ACBI, CLS, EPAS, FIX, Euroclear, OAGI and Omgeo. Market infrastructures like automatic clearing houses, real time gross settlement systems and stock exchanges, are adopting the standard and they stimulate developers to use the UNIFI standard. Industry initiatives and regulatory requirements, such as SEPA, Giovannini, MiFID and FATF use UNIFI to support their harmonization of business practices.

3.2 XML

XML stands for eXtensible Markup Language and is like HTML a markup language. Where HTML was designed to describe how data should be presented, XML was designed to describe the semantic (ontology) of the data. To do this, an XML author should define tags, because they are not predefined. The language can be used to structure, store and send information. The official definition is: "XML is a simple, very flexible text format, derived from Standard Generalized Markup Language (SGML; ISO 8879)" [r37]. Another definition is: "XML is a cross-platform, software and hardware independent tool for transmitting information"[r39]. [r75] Despite XML describes the semantics of data, it does not describe how it could be used. In case of UNIFI, this has been described in the ISO 20022 standard and in particular part four.
Although a lot can be written about XML, this paper has been constrained to the following points of interest.

Validation
[r39] If XML has got correct syntax, it is 'well formed' XML. Examples of syntax rules are:

- the XML document must have a root element;
- elements must have a closing tag and must be properly nested;
- tags are case sensitive;
- attribute values must always be quoted.

XML is 'valid' if it is has correct syntax and also conforms to the rules of a Document Type Definition (DTD) or XML Schema Definition (XSD). DTD and XSD are described in the next paragraph in more detail. According to the W3C specifications an XML processor should stop if it finds an error in the XML input. In contrast to this, web browsers try to continue if they process incomplete HTML.

Namespace
[r39] Namespaces can be used to avoid conflicts in element names. A name conflict will arise in case two different elements are defined with the same name in an XML document (see table 11).

Information in a table	Information about a table
<table> <tr> <td>Apples</td> <td>Bananas</td> </tr> </table>	<table> <name>African Coffee Table</name> <width>80</width> <length>120</length> </table>

Table 11: Name conflict

This problem can be solved by using a prefix (see table 12).

Information in a table	Information about a table
<h:table> <h:tr> <h:td>Apples</h:td> <h:td>Bananas</h:td> </h:tr> </h:table>	<f:table> <f:name>African Coffee Table</f:name> <f:width>80</f:width> <f:length>120</f:length> </f:table>

Table 12: Using a prefix

In stead of "<f:table>" it is also possible to add an xmlns attribute, e.g. "<f:table xmlns:f="http://www.w3schools.com/furniture">". Although this is only applied to a start tag of an element, all child elements with the same prefix will also be associated to it. The attribute has the following syntax: "xmlns:namespace-prefix="namespaceURI". The address is meant to give the namespace a unique name and will not used by the parser to look up information. However, authors can use the namespace as a web reference for more information about the namespace.

3.3 XSD (XML Schema Definition)

[r34, r38] XSD consists of a schema element and a variety of sub elements, such as elements of simple type, and elements of complex type. These XSD elements with their attributes determine the appearance of XML instance elements, their content and attributes. XSD makes it possible to define the order and the number of these elements and whether an element is empty or includes text. Furthermore it defines data types for elements and attributes. It is also possible to define default or fixed values for elements and attributes. See also appendix 5 for a list of default attributes, elements and simple types.
[r34] Thus, schemas are designed to define a class of XML documents (XML instances). In stead of documents, they may exist for example as streams of bytes sent over the Internet or as fields in a database record.

XSD differences and benefits
XSD can be seen as the successor of Document Type Definition (DTD). An important difference between those is the possibility of XSD to describe the type of data, which is not possible with a DTD. [r38] Advantages of this feature are that it is easier to 1) describe allowable document content, 2) validate the correctness of data, 3) work with data from a database, 4) define data restrictions on data, 5) define data patterns (data formats) and 6) convert data between different data types.
A second difference between XSD and DTD is the fact that XSD is also described in XML and DTD not. A benefit of XML Schemas written in XML is that authors don't have to learn a new language. Also the same editor, parser, Document Object Model (DOM) technology (to manipulate) and XSLT[11] technology (to transform schemas) can be used.
A third difference is the possibility to extend XSD. This means that other schemas can be reused by referring to one or more of them. Beside this, it is possible to create data types derived from the standard types.
A fourth difference is the support of namespaces through XSD, which is not the case with DTD.

The <schema> element
Every XSD contains a root element, which could have several attributes, like the attributes 'targetNamespace', 'elementFormDefault' and 'xmlns'.
The text: xmlns:xs="http://www.w3.org/2001/XMLSchema", indicates that the elements and data types used in the schema come from the "http://www.w3.org/2001/XMLSchema" namespace.
The prefix xs: is used to symbolize the XML schema namespace, although any prefix can be used. The same prefix, and hence the same association, also appears on the names of built-in simple types, e.g. xs:string. This can for example be used to define that the element city is of the type string, e.g. <xs:element name="city" type="xs:string"/> (see also example 1). The purpose of the association is to identify the elements and simple types as belonging to the vocabulary of the XML Schema language rather than the vocabulary of the schema author.
The fragment: xmlns=" http://tempuri.org/SimpleIBAN", refers to a default namespace.
An author can use the attribute: elementFormDefault="qualified" to indicate that any element used by the XML instance, which were declared in a schema, must be namespace qualified.
The target namespace indicates which elements belong to a particular namespace. This enables authors to distinguish definitions and declarations of different schemas (see example 1). More importantly, it allows XML validators to check whether an instance document conforms to one or more schemas. To do this it is necessary to identify which element declarations, attributes declarations and type definitions of a schema should be used to verify which elements and attributes are valid in the XML instance.

XSD

```
<?xml version="1.0" encoding="utf-8" ?>
<xs:schema id="SimpleIBAN"
    targetNamespace="http://tempuri.org/SimpleIBAN"
    elementFormDefault="qualified"
    xmlns="http://tempuri.org/SimpleIBAN"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">

   <xs:element name="Document" type="AccountIdentification3Choice"/>

    <xs:complexType name="AccountIdentification3Choice">
      <xs:sequence>
        <xs:choice>
          <xs:element name="IBAN" type="IBANIdentifier"/>
        </xs:choice>
      </xs:sequence>
    </xs:complexType>

    <xs:simpleType name="IBANIdentifier">
```

[11] XSLT stands for XSL Transformations. See also chapter 6 or the glossary.

```
      <xs:restriction base="xs:string">
        <xs:pattern value="[a-zA-Z]{2,2}[0-9]{2,2}[a-zA-Z0-9]{1,30}"/>
      </xs:restriction>
    </xs:simpleType>
</xs:schema>
```

XML

```
<Document xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://tempuri.org/SimpleIBAN">
  <IBAN>FR1420041010050500013M02606</IBAN>
</Document>
```

Example 1: XSD fragment with corresponding XML

Simple vs. complex types

The difference between simple types and complex types is that the latter does allow the use of elements in their content and it may carry attributes, while this is not the case with the former. A simple type may only contain text, but this text can be of many types. The syntax looks like: <xs:element name="Y" type="Z"/>, where Y is a element name and Z a type, which can be a built in type e.g. xs:string, xs:integer, xs:Boolean or a type defined by an author.
In the context of XSD there is a difference between definitions and declarations. The first one creates new types (both simple and complex), and the second one enables elements and attributes with specific names and types (both simple and complex) to appear in document instances.

Facets

A definition of a facet is: "a single defining aspect of a set of values for a given datatype. This set of values is also known as 'value space'. The facets of a datatype serve to distinguish those aspects of one datatype which differ from other datatypes. There are two types of facets, i.e. fundamental facets that define the datatype and non-fundamental or constraining facets that constrain the permitted values of a datatype.

Restrictions

So, it is possible to add restrictions (facets) to a data type in order to limit its content, or to match the data to a specific pattern (example 2). Table 13 gives a list of restrictions.

```
<xs:simpleType name="BBANIdentifier">
  <xs:restriction base="xs:string">
    <xs:pattern value="[a-zA-Z0-9]{1,30}"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="IBANIdentifier">
  <xs:restriction base="xs:string">
    <xs:pattern value="[a-zA-Z]{2,2}[0-9]{2,2}[a-zA-Z0-9]{1,30}"/>
  </xs:restriction>
</xs:simpleType>
```

Example 2: Prefix, simple type and restriction

Restriction	Description
enumeration	Defines a list of acceptable values.
fractionDigits	Specifies the maximum number of decimal places allowed. Must be equal to or greater than zero.
length	Specifies the exact number of characters or list items allowed. Must be equal to or greater than zero.
maxExclusive	Specifies the upper bounds for numeric values (the value must be less than this value).
maxInclusive	Specifies the upper bounds for numeric values (the value must be less than or equal to this value).
maxLength	Specifies the maximum number of characters or list items allowed. Must be equal to or greater than zero.
minExclusive	Specifies the lower bounds for numeric values (the value must be greater than this value).
minInclusive	Specifies the lower bounds for numeric values (the value must be greater than or equal to this value).
minLength	Specifies the minimum number of characters or list items allowed. Must be equal to or greater than zero.
pattern	Defines the exact sequence of characters that are acceptable.
totalDigits	Specifies the exact number of digits allowed. Must be greater than zero.
whiteSpace	Specifies how white space (line feeds, tabs, spaces, and carriage returns) is handled.

Table 13: Restriction for data types: source [r38]

Regular expressions

Regular expressions can be defined as expressions that describe a set of strings. In XSD the element 'pattern' is used, which is a synonym for a regular expression.

An example of a pattern can be found in the 'BBANIdentifier' (Basic Bank Account Number) element (example 2): '[a-zA-Z0-9]{1,30}' to allow a strings like NL12345678901234567890123456789012345678. The BBAN is a part of IBAN (International Bank Account Number) (see table 15).

[r26] The symbols '[' and ']' of a regular expression define a character class (also known as character set), which matches a single character out of all the possibilities offered by the character class. The symbol '-' specifies a range of characters or a hyphen if it is placed immediately after '['. The part: '{n,m}' means to repeat the previous item between n and m times where n >= 1 and m >= n.

Another example of a pattern can be found in the 'BICIdentifier' element (example 3). BIC stands for Bank Identification Code, thus a number to label a financial institution in order to facilitate the automated processing of payments.

```
<xs:simpleType name="BICIdentifier">
  <xs:restriction base="xs:string">
    <xs:pattern value=
                    "[A-Z]{6,6}[A-Z2-9][A-NP-Z0-9]([A-Z0-9]{3,3}){0,1}"/>
  </xs:restriction>
</xs:simpleType>
.....
    <xs:element name="BIC" type="BICIdentifier" minOccurs="0"
maxOccurs="1"/>
.....
```

Example 3: Regular expression

The BIC pattern is based on the agreement listed in table 14. Examples of BIC elements of the type BICIdentifier in an XML instance are for example 'BIC>AAAAJPJT</BIC>' and <BIC>AAAAGB2L</BIC>.

Part	Length	Description	RE
Bank Code	4	Alphabetic characters for identifying an individual bank, for example "DEUT" identifies Deutsche Bank.	[A-Z]{6,6}
Country Code	2	Two letter ISO country code such as DE for Germany.	
Location Code	2	Two alphanumerical characters (except zero) for identifying the location of the institution within the specific country such as FF for Frankfurt.	[A-Z2-9] [A-NP-Z0-9]
Branch Code	3	Three alphanumeric optional characters for identifying the specific office or branch. Branches that are identified by an 11-character BIC don't have a SWIFT interface and a direct connection to the SWIFT network. They use the SWIFT interface and SWIFT network connection of the institution with the corresponding 8-character BIC. In this case, BANKCCLLMAR is sending and receiving SWIFT messages through BANKCCLL.	([A-Z0-9]{3,3}) {0,1}
Total	8 - 11		

Table 14: Construction of BIC; source [r28]

Part	Length	Description	RE
Country Code	2	The country codes are specified in ISO 3166.	[a-zA-Z]{2,2}
check digits	2	Two digits calculated as specified in clause 6 of [r57].	[0-9]{2,2}
Basic Bank Account Number (BBAN)	up to 30	The BBAN has a fixed length per country. The BBAN includes an explicit identification code of the bank/branch servicing the account at fixed positions within the BBAN, which are specified per country.	[a-zA-Z0-9]{1,30}
	5 - 34		

Table 15: Construction of IBAN; source [r57]

3.4 Interoperability within the financial industry

[r21] UNIFI supports the interoperability between different message standards. For example in the pre-trade domain of the security industry or in the customer-to-bank payment domain.

Pre-trade trade domain

The FIX protocol can be used for the first steps of the securities trade life cycle from pre-trade to post-trade, while ISO 15022 can be used for the later steps starting from trade through settlement and reconciliation (figure 16).

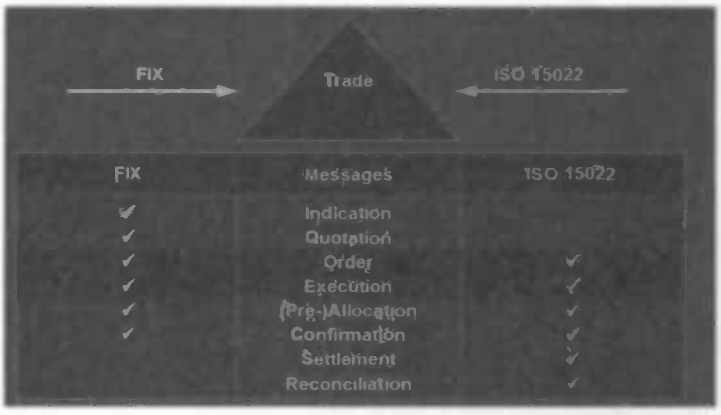


Figure 16: Two overlapping standards to support the end-to-end transaction

The table above illustrates two interoperability problems. The first problem is that the information has to 'flow' throughout two different standards from the beginning to the end of the transaction life cycle. The second problem is the overlap in the scope of the standards which may require the translation of a message in one syntax into the equivalent message in another syntax.
An example of overlap areas are "order" and "execution". Figure 17 shows that the executing party has to interpret messages from several parties. Conversions have to be made from the FIX standard into the ISO 15022 standard and vice versa.

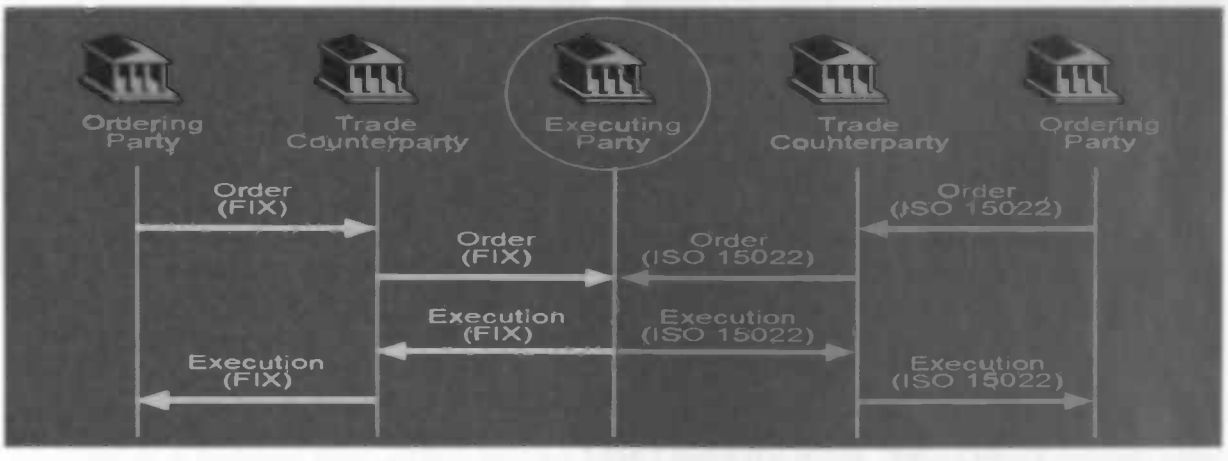


Figure 17: Messages from several parties

Although these standards use a different syntax, the messages are semantically equivalent. UNIFI reverse engineering leads to the identification of business concepts and a business model that is based on the messages of the two different standards.
Figure 18 demonstrates the principle of convergence and co-existence. The former means that the old standards can be migrated to the UNIFI equivalent and the latter means that for example the FIX message can be translated into an ISO 20022 message.

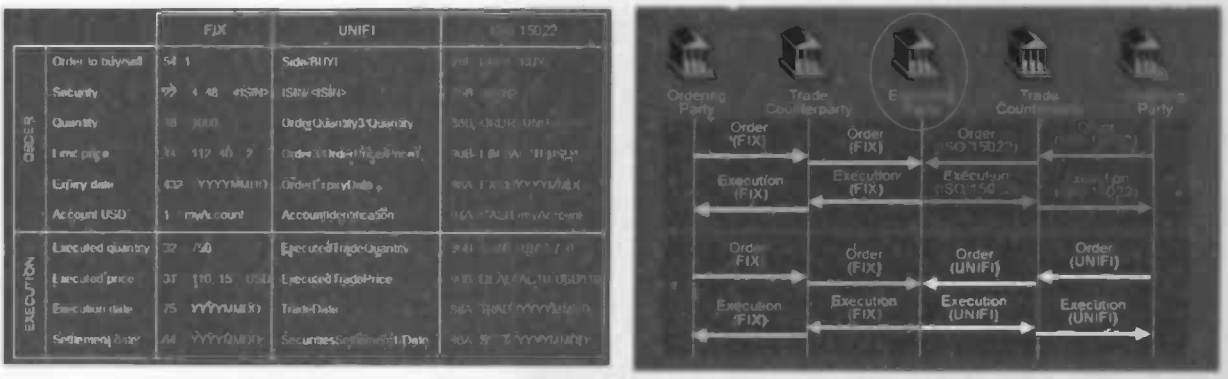


Figure 18: Working towards interoperability and convergence (1/2)

Customer-to-bank payment domain

Figure 19 lines out a scenario that one corporate customer communicates with multiple banks and uses therefore different formats for a payment request. Another scenario is that one bank uses different formats to adapt to the different formats of its customers.

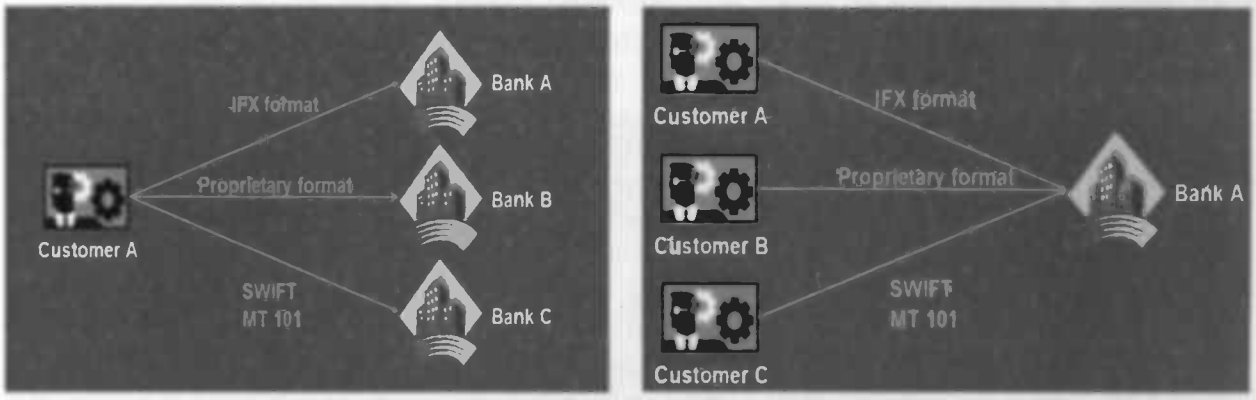


Figure 19: Working towards interoperability and convergence (2/2)

The International Standard Team for Harmonization (ISTH[12]) applied reverse engineering to the existing formats and came with the single UNIFI message model to cover the customer-to-bank payment initiation. The resulting 'CustomerCreditTransferInitiation' message is also known as the 'core payment kernel model', which is the first UNIFI message that has been approved in September 2005 (see figure 20).

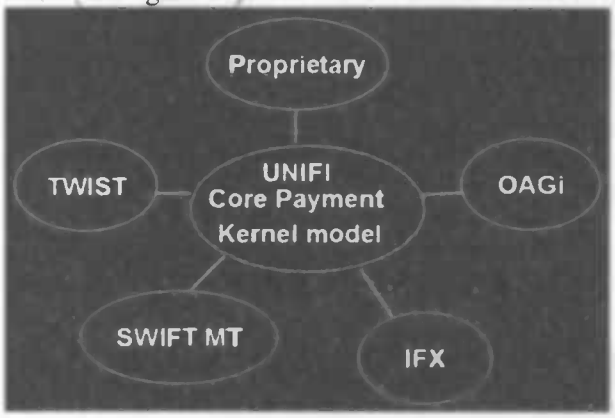


Figure 20: The 'core payment kernel'

[12] ISTH has been formed by IFX, OAGi, TWIST and SWIFT.

Convergence tables can be used to translate each standard into the model and vice versa, as long as other standards than UNIFI standards are used. This reduces the number of interfaces from n*(n-1) to n*2, where n is a standard.

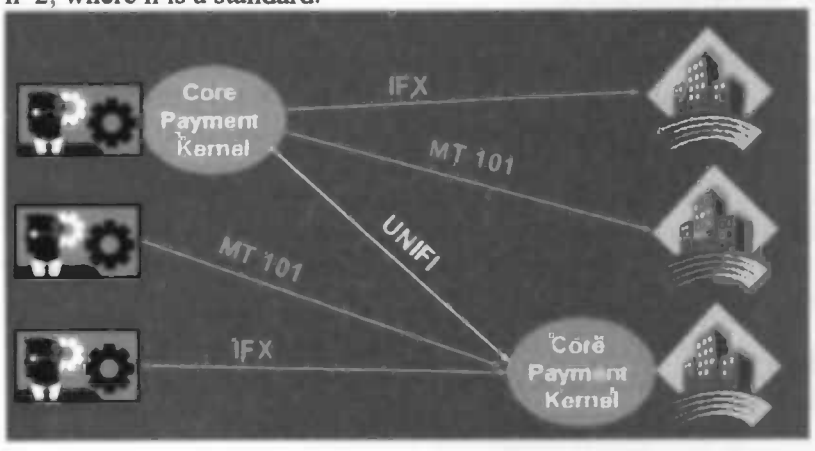


Figure 21: Adopting UNIFI facilitates convergence and co-existence

3.5 Cross industry harmonization

[r21] *"Interoperability within the financial industry is good, but interoperability with all industry sectors is even better"*. Financial communication can be divided into communication between financial institutions and communication between their clients from other industries. Various pieces of a financial message are the same for both parts. Examples are: 'address', 'buyer' and 'seller', 'invoice' and 'amount'. The UN/CEFACT (United Nations / Centre for trade facilitation and e-business) plays a role to achieve this.

3.6 Extend possibilities

Two questions of the thesis where: "are organizations who implement UNIFI still able to add new functionality/wishes, without being incompatible with UNIFI?" and "is it for example possible to add extra attributes to the standard as defined in the ISO 20022?". The answer is yes, but new messages should first be approved by the standard evaluation groups of UNIFI to avoid incompatibility.
The rulebooks published by the EPC confirm this: "The Scheme has been designed to be capable of evolution to permit the development of features and improvements to satisfy future needs" [r62, r63]. Source [r84] describes: *"UNIFI allows (...) organizations involved in financial message development to register their business requirements and define message sets according to an international agreed approach (...)"*. Furthermore appendix 12 states that attributes can be defined, proposed and finally get added to the repository of ISO. For example an attribute that represent the energy value for financial transactions of oil companies. Users of UNIFI could use schemas with these attributes, but it will not be obligatory. Financial institutions are free to create messages to offer their customers added services like payment scheduling. However, for existing messages (e.g. messages for inter bank transactions) they get incompatible with UNIFI if they deviate from it. Thus an agreement about addition of UNIFI should be made in the SEPA first. [r63] Examples of possible additions for direct debit are:

- an additional optional process for mandate submission through the Debtor Bank;
- support for business-to-business direct debits;
- the inclusion of features such as electronic signature and full connection to e-invoicing services.

[r62] E-invoicing is an example of an Additional Optional Service (AOS) and as well an example of a possible addition for credit transfer schema.

3.7 Kinds of messages

[r22, r72, r77,r78] The following kinds of UNIFI messages can be inventoried:

- messages for the funds industry (validated and approved by the Security SEG);
- foreign exchange transaction messages (evaluated by the FX SEG);

- messages supporting transactions and business processes related to trade finance business and financial supply chain management (evaluated by the Trade Service SEG);
- payment messages (evaluated by the Payment SEG);

These messages are or will be developed according the rules of the ISO 20022 standard. A difference between published and not published UNIFI messages has been made to answer the question: "what kinds of messages do exist?". At the moment the Security SEG, the FX SEG and the Payment SEG have published messages and the Trade Service SEG did not.

The Securities SEG has validated a set of 45 candidate UNIFI messages developed by SWIFT for the Funds industry in June 2005 and where approved in November 2005. These messages can be divided into:

1) cash management (6 messages);
2) reference data (3 messages);
3) security management (7 messages);
4) security settlement (11 messages);
5) security trade (18 messages).

The FX SEG published 15 'forex notifications' messages.

The Payment SEG approved four UNIFI messages that cover the area of 'customer to bank credit transfer initiations', also known as the 'core payment kernel' that has been developed by ISTH in September 2005. Twelve messages covering a new version of the payment initiation messages and the direct debit and interbank credit transfer payments were approved in June 2006. These interbank debit and credit transfer messages were developed by SWIFT and will be used for clearing and settlement in the SEPA environment.
Messages for the cash management business area can be divided into 1) bank-to-customer cash management and 2) exception & investigations. A set of 14 messages that cover exceptions and investigations of cash management were approved in July 2006. Three messages that cover bank-to-customer cash management were published half 2007. A set of 28 cash management messages were planned to be developed in 2007.

Table 16,17,18 and 19 lists the names and files of the 'UNIFI Payments messages', including the related messages that are approved by the Payments SEG (Standards Evaluation Group) as UNIFI messages.

Nr	Message name	XSD file	# pages[13]
1	Customer Credit Transfer Initiation	pain.001.001.02	17
2	Customer Direct Debit Initiation	pain.008.001.01	16
	Related messages:		
3	Customer Payment Reversal	pain.007.001.01	16
4	Payment Cancellation Request	pain.006.001.01	16
5	Payment Status Report	pain.002.001.02	17
		AVG: 82 / 5 =	16,4

Table 16: XSDs of payments initiation (pain); source [r77] (Last updated on: 27 October 2006)

	Message name	XSD file	# pages
6	Financial Institution To Financial Institution Customer Credit Transfer[14]	pacs.008.001.01	17
7	Financial Institution To Financial Institution Customer	pacs.003.001.01	17

[13] The number of pages are rounded to above; character type is Courier New; Size is 10pt;
AVG = (82 + 113 + 73+ 45) / (5 + 7 + 3 + 14) = 313 / 29 = 10,8 pages per XSD
[14] See appendix for this schema.

	Direct Debit		
8	Financial Institution Credit Transfer	pacs.009.001.01	10
	Related messages:		
9	Financial Institution To Financial Institution Payment Reversal	pacs.007.001.01	17
10	Payment Return	pacs.004.001.01	18
(4)	Payment Cancellation Request	pacs.006.001.01	16
(5)	Payment Status Report	pacs.002.001.02	18
		AVG: 113 / 7 =	16,1

Table 17: XSDs of payments clearing & settlement (pacs); source [r78] (Last updated on: 27 October 2006)

Two related messages of the tables above are the same messages. Nevertheless the specific usage rules defined for using these messages in the two different legs (payments initiation and payments clearing & settlement) of the end-to-end payment chain are not the same.

	Message name	XSD file	# pages
11	BankToCustomerAccountReportV01	camt.052.001.01	25
12	BankToCustomerStatementV01	camt.053.001.01	25
13	BankToCustomerDebitCreditNotificationV01	camt.054.001.01	23
		AVG: 73 / 3 =	24,3

Table 18: XSDs of cash management – bank-to-customer cash management (camt a); source [r72] (Last updated on: 20 April 2007)

	Message name	XSD file	# pages
14	AdditionalPaymentInformation	camt.028.001.01	11
15	CancelCaseAssignment	camt.032.001.01	1
16	CaseStatusReport	camt.039.001.01	2
17	CaseStatusReportRequest	camt.038.001.01	1
18	ClaimNonReceipt	camt.027.001.01	2
19	DebitAuthorisationRequest	camt.037.001.01	2
20	DebitAuthorisationResponse	camt.036.001.01	2
21	NotificationOfCaseAssignment	camt.030.001.01	2
22	RejectCaseAssignment	camt.031.001.01	2
23	RequestForDuplicateInstruction	camt.033.001.01	1
24	RequestToCancelPayment	camt.008.002.01	2
25	RequestToModifyPayment	camt.007.002.01	11
26	ResolutionOfInvestigation	camt.029.001.01	3
27	UnableToApply	camt.026.001.01	3
		AVG: 45/14=	3,2

Table 19: XSDs of cash management – exceptions and investigations (camt b); source [r72] (Last updated on: 11 August 2006)

3.8 Implementation

"How are (UNIFI) XSD messages implemented?" is the next question. Despite there is a lot documented about these messages, there is no space and time to mention them all in full detail. However, to give a clear picture of these messages, one of the payment messages has been described in more detail.

3.8.1 Customer Credit Transfer (pacs.008.001.01.xsd)

[r78] The *'Financial Institution To Financial Institution (FIToFI) Customer Credit Transfer'* is a schema that can be used for inter-bank movement of money from a party bank account (the Debtor[15])

[15] Debtor means in Dutch: 'schuldenaar'.

to beneficiary party (the creditor[16]). Transfers of money can be exchanged as single instructions or grouped according common characteristics to exchange in a batch mode.

Message flow
Figure 22 shows how the schema can be used in several scenarios.

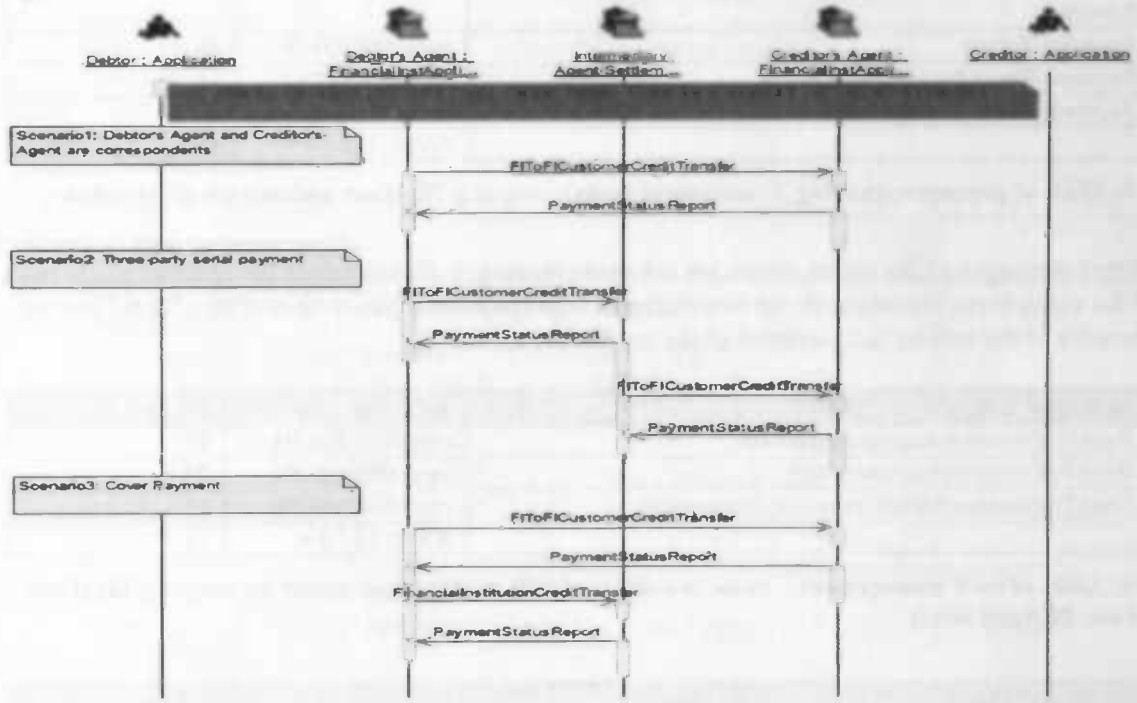


Figure 22: Sequence diagram of FIToFICustomer Credit Transfer (pacs.008.001.01.xsd); source [r78]

At the first scenario the debtor agent and the creditor agent communicate with each other. Firstly the debtor agent sends the FIToFICustomerCreditTransfer message to the creditor agent. Secondly the creditor agent optionally confirms the processability of the FIToFICustomerCreditTransfer instruction by sending a positive PaymentStatusReport message to the debtor agent.

At the second scenario the debtor agent and the creditor agent are, in contradiction of the first scenario, not communicating directly with each other. For communication there is an intermediary agent between them (3-party serial payment). Instead of an intermediary agent, it can also be a clearing and/or settlement agent in case a payment clearing system will be used. This scenario can be divided into four steps. Firstly the debtor agent sends the FIToFICustomerCreditTransfer message to the intermediary agent. After that the intermediary agent optionally confirms the processability of the FIToFICustomerCreditTransfer by sending a positive PaymentStatusReport message to the debtor agent.
The third step is that the intermediary agent forwards the FIToFICustomerCreditTransfer message to the creditor agent. Lastly the creditor agent optionally confirms the processability of the FIToFICustomerCreditTransfer instruction by sending a positive PaymentStatusReport message to the intermediary agent.

At the third scenario the debtor agent and the creditor agent are not correspondents in the currency of the transaction: there is a settlement agent between them (3-party cover payment). To keep this scenario simple, the debtor agent and the creditor agent have the same settlement agent. The first step is that the debtor agent sends the FIToFICustomerCreditTransfer message to the creditor agent. Secondly the creditor agent optionally confirms the processability of the FIToFICustomerCreditTransfer instruction by sending a positive PaymentStatusReport message to the

[16] Creditor means in Dutch: 'schuldeiser'

debtor agent. The third step is that the debtor agent sends a FinancialInstitutionCreditTransfer message to the settlement agent. Note that another schema than the FIToFICustomerCreditTransfer has been used. Finally the settlement agent optionally confirms the processability of the FinancialInstitutionCreditTransfer by sending a positive PaymentStatusReport message to the debtor agent.

Usage

The FIToFICustomerCreditTransfer message can be used in domestic and cross-border scenarios. The message can contain one or more customer credit transfer instructions. It is possible to use the message in different ways.

It is possible to include the funds for the customer transfer(s) and the payment details. This will be done if the instructing and instructed agents want to use their direct account relationship in the currency of the transfer.

A second possibility is to use the 'cover method', which is a payment method where the message contains only the payment details and the instructing agent must cover the customer transfer by sending a FinancialInstitutionCreditTransfer to a reimbursement agent. This option could be chosen if the instructing agent and the instructed agent have no direct account relationship in the currency of the transfer or do not wish to use their account relationship. A consequence is that other agents will be involved to cover the customers transfer(s).

A third possibility is to use the 'serial method', which is a payment method where more then two financial institutions are involved in transfering the FIToFICustomerCreditTransfer message to another financial institution.

Outline

The FIToFICustomerCreditTransfer can be divided into two building blocks, namely a group header and a block that consist of credit transfer transaction information. Both are mandatory. The former appears once and the latter appears repetitive. The header contains for example elements like MessageIdentification and CreationDateAndTime, while the second building block could contain elements like Creditor, CreditorAgent, Debtor and DebtorAgent.

Rules

The schema also contains a set of rules, which are summarized in the table below.

Rule	Description	When
InstructedAgent Rule	CreditTransferTransactionInformation/ InstructedAgent is not allowed.	GroupHeader/InstructedAgent is present.
InstructingAgent Rule	CreditTransferTransactionInformation/ InstructingAgent is not allowed.	GroupHeader/InstructingAgent is present.
TotalInterbank Settlement Amount1 Rule	All occurrences of CreditTransferTransactionInformation/ InterbankSettlementAmount must have the same currency as the currency of GroupHeader/ TotalInterbankSettlementAmount.	GroupHeader/ TotalInterbankSettlementAmount is present.
TotalInterbank Settlement Amount2 Rule	It must equal the sum of all occurrences of CreditTransferTransactionInformation/ InterbankSettlementAmount.	GroupHeader/ TotalInterbankSettlementAmount is present.
Interbank SettlementDate Rule	CreditTransferTransactionInformation/ InterbankSettlementDate is not allowed.	GroupHeader/ InterbankSettlementDate is present.
	CreditTransferTransactionInformation/ InterbankSettlementDate must be present.	GroupHeader/InterbankSettlementDate is not present.
PaymentType Information Rule	CreditTransferTransactionInformation/ PaymentTypeInformation is not allowed.	GroupHeader/PaymentTypeInformation is present.

Table 20: Common rules of FIToFICustomerCreditTransfer

3.8.2 XSD example

The examples below are code snippets of the 16 page long 'FIToFI Customer Credit Transfer' XSD. These examples illustrate how a UNIFI message has been described in XSD. See appendix 7 for a (partial) tree view and appendix 8 for the whole XSD document.

```
<xs:complexType name="pacs.008.001.01">
  <xs:sequence>
    <xs:element name="GrpHdr" type="GroupHeader2"/>
    <xs:element name="CdtTrfTxInf"
type="CreditTransferTransactionInformation2" minOccurs="1"
maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

Example 4: Two building blocks

```
<xs:complexType name="GroupHeader2">
  <xs:sequence>
    <xs:element name="MsgId" type="Max35Text"/>
    <xs:element name="CreDtTm" type="ISODateTime"/>
    <xs:element name="BtchBookg" type="BatchBookingIndicator"
minOccurs="0" maxOccurs="1"/>
    <xs:element name="NbOfTxs" type="Max15NumericText"/>
    <xs:element name="CtrlSum" type="DecimalNumber" minOccurs="0"
maxOccurs="1"/>
    <xs:element name="TtlIntrBkSttlmAmt" type="CurrencyAndAmount"
minOccurs="0" maxOccurs="1"/>
    <xs:element name="IntrBkSttlmDt" type="ISODate" minOccurs="0"
maxOccurs="1"/>
    <xs:element name="SttlmInf" type="SettlementInformation1"/>
    <xs:element name="PmtTpInf" type="PaymentTypeInformation3"
minOccurs="0" maxOccurs="1"/>
    <xs:element name="InstgAgt"
type="BranchAndFinancialInstitutionIdentification3" minOccurs="0"
maxOccurs="1"/>
    <xs:element name="InstdAgt"
type="BranchAndFinancialInstitutionIdentification3" minOccurs="0"
maxOccurs="1"/>
  </xs:sequence>
</xs:complexType>
```

Example 5: First building block: group header

```
<xs:complexType name="CreditTransferTransactionInformation2">
  <xs:sequence>
    <xs:element name="PmtId" type="PaymentIdentification2"/>
    <xs:element name="PmtTpInf" type="PaymentTypeInformation3"
minOccurs="0" maxOccurs="1"/>
    .....
```

Example 6: Part of the second building block: credit transfer transaction information

3.8.3 XML example

The text below represents an example of an XML instance of the 'FIToFI Customer Credit Transfer' XSD.

```
<?xml version = "1.0" encoding = "UTF-8"?>
<Document xmlns = "urn:iso:std:iso:20022:tech:xsd:pacs.008.001.01"
xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance">
      <pacs.008.001.01>
            <GrpHdr>
                  <MsgId>BBBB/060928-CCT/JPY/123</MsgId>
                  <CreDtTm>2006-09-28T16:00:00</CreDtTm>
                  <NbOfTxs>1</NbOfTxs>
                  <SttlmInf>
```

```
                <SttlmMtd>COVE</SttlmMtd>
                <InstgRmbrsmntAgt>
                    <FinInstnId>
                        <BIC>CCCCJPJT</BIC>
                    </FinInstnId>
                </InstgRmbrsmntAgt>
                <InstdRmbrsmntAgt>
                    <FinInstnId>
                        <BIC>AAAAJPJT</BIC>
                    </FinInstnId>
                </InstdRmbrsmntAgt>
            </SttlmInf>
            <InstgAgt>
                <FinInstnId>
                    <BIC>BBBBUS33</BIC>
                </FinInstnId>
            </InstgAgt>
            <InstdAgt>
                <FinInstnId>
                    <BIC>AAAAGB2L</BIC>
                </FinInstnId>
            </InstdAgt>
        </GrpHdr>
        <CdtTrfTxInf>
            <PmtId>
                <InstrId>BBBB/060928-CCT/JPY/123/1</InstrId>
                <EndToEndId>ABC/4562/2006-09-08</EndToEndId>
                <TxId>BBBB/060928-CCT/JPY/123/1</TxId>
            </PmtId>
            <PmtTpInf>
                <InstrPrty>NORM</InstrPrty>
            </PmtTpInf>
            <IntrBkSttlmAmt Ccy = "JPY">10000000</IntrBkSttlmAmt>
            <IntrBkSttlmDt>2006-09-29</IntrBkSttlmDt>
            <ChrgBr>SHAR</ChrgBr>
            <Dbtr>
                <Nm>ABC Corporation</Nm>
                <PstlAdr>
                    <StrtNm>Times Square</StrtNm>
                    <BldgNb>7</BldgNb>
                    <PstCd>NY 10036</PstCd>
                    <TwnNm>New York</TwnNm>
                    <Ctry>US</Ctry>
                </PstlAdr>
            </Dbtr>
            <DbtrAcct>
                <Id>
                    <PrtryAcct>
                        <Id>00125574999</Id>
                    </PrtryAcct>
                </Id>
            </DbtrAcct>
            <DbtrAgt>
                <FinInstnId>
                    <BIC>BBBBUS33</BIC>
                </FinInstnId>
            </DbtrAgt>
            <CdtrAgt>
                <FinInstnId>
                    <BIC>AAAAGB2L</BIC>
                </FinInstnId>
```

```
                </CdtrAgt>
                <Cdtr>
                      <Nm>DEF Electronics</Nm>
                      <PstlAdr>
                            <AdrLine>Corn Exchange 5th Floor</AdrLine>
                            <StrtNm>Mark Lane</StrtNm>
                            <BldgNb>55</BldgNb>
                            <PstCd>EC3R7NE</PstCd>
                            <TwnNm>London</TwnNm>
                            <Ctry>GB</Ctry>
                      </PstlAdr>
                </Cdtr>
                <CdtrAcct>
                      <Id>
                            <PrtryAcct>
                                  <Id>23683707994215</Id>
                            </PrtryAcct>
                      </Id>
                </CdtrAcct>
                <Purp>
                      <Cd>GDDS</Cd>
                </Purp>
                <RmtInf>
                      <Strd>
                            <RfrdDocInf>
                                  <RfrdDocTp>
                                        <Cd>CINV</Cd>
                                  </RfrdDocTp>
                                  <RfrdDocNb>4562</RfrdDocNb>
                            </RfrdDocInf>
                            <RfrdDocRltdDt>2006-09-08</RfrdDocRltdDt>
                      </Strd>
                </RmtInf>
          </CdtTrfTxInf>
    </pacs.008.001.01>
</Document>
```

Example 7: XML instance of 'FIToFI Customer Credit Transfer' XSD.

3.9 E-invoicing

The aim of this paragraph is to give answers to the questions:
"Does the ISO 20022 define schemas for e-invoicing? If not, how could an e-invoice schema look like? What are the benefits of e-invoicing? What are the disadvantages of e-invoicing?"
Firstly a definition of e-invoicing will be given. Secondly the main stakeholders are mentioned. Thirdly the benefits and disadvantages are described and the fourth part gives an overview of the status of e-invoicing.

3.9.1 Definition

A definition of e-invoicing is: "sending and storing invoices 'by electronic means'. This means the transmission or making available to the recipient and storage using electronic equipment for processing (including digital compression) and storage of data, and employing wires, radio transmission, optical technologies or other electromagnetic means [r49]". Another formulation of e-invoicing is: "sending invoices through an electronic medium, like Internet or e-mail. [2, page 21]". [r62] E-invoicing is an example of an Additional Optional Service (AOS) and as well an example of a possible addition for credit transfer schema.

3.9.2 Stakeholders

[r47] The main stakeholders of e-invoicing related activities are:

- the European Commission, with their project to deliver e-invoicing cross border within the EU to lever SEPA and contribute significantly to the Lisbon Agenda;
- the EBA Association with their project to investigate e-invoicing and to look into the case for the creation of a common EU e-invoicing scheme, rulebook, and standards;
- ACBI in regard to the invoice financing work under ISO 20022;
- the Nordic region that adopt the e-invoice formats, including Finvoice, UBL, and UN/CEFACT;
- SWIFT in relation to the TSU phase 2 activity.

3.9.3 Advantages

[r82 page 12] The main benefits of e-invoicing and e-archiving are increased efficiency, cost reduction (table 21) and faster customer payments (see also table 21).
Source [r56, page 21] describes that these benefits go even beyond just payments. Through enhanced application of information technology is invoicing the link between internal company processes and the payment system. The potential economic advantages are so large, that they could make a significant contribution to the Lisbon process, also known as Lisbon Strategy or Lisbon Agenda, to make the EU a more competitive economy.

Total annual EU invoices	Exceeds 20 billion
Percentage of invoices which are B2B or (business to government) B2G	Exceeds 50%
Current cost to manually process an invoice	€30-80
Cost saving from electronic processing	60-90%
Conservative estimate of total annual saving	More than €100bn a year

Table 21: Estimates of the European Association of Corporate Treasures (EACT); source [r64]

According to [r65], the standardization and automation of invoices might reduce the number of mistakes they contain. A study by Atradius mentions that 82% of the invoices in the UK include a wrong address. Two-third of the companies send the wrong invoice back to the supplier and one-tenth refuses to pay them. Invoices with mistakes are paid on average 15 to 30 days later as a consequence of this.
According source [r47] will corporate customers be able to 1) retrieve money faster and regularly; 2) have a bigger chance to get early payment and volume discounts and 3) sell more by offering credit terms to customers. Banks and other financial institutions will be able to offer their clients more products, higher service levels, better financing rates and they will be able to reduce their operating costs.

Also software vendors will have advantages, because they could upgrade accounting systems that are not e-invoicing enabled yet.

3.9.4 Disadvantages

[r82 page 12] Temporary disadvantage associated with e-invoicing and e-archiving are the lack of readiness of internal, customer and supplier systems, cost and complexity. E-invoicing can also lead to problems during inspections of tax authorities, which may result in a denial of VAT/GST deduction or administrative penalties. According [r82] only few companies consider regulation and legislation as a barrier to e-invoicing and e-archiving.

3.9.5 Status

Does the ISO 20022 define schemas for e-invoicing? Innopay explained that ISO 20022 is not a standard for payment transactions only, but also for related processes like: invoicing, foreign exchange and Documentary Trade[17]. Although it is possible to define e-invoicing schemas with ISO 20022, it does not mean that these schemas are already published. The following fragments confirm this.

- [r65] On 29th of January 2007, Gtnews published an article that states that the European Association of Corporate Treasurers (EACT) is working on a project to harmonize the e-invoicing standards.
- [r83] UN/CEFACT could submit the cross industry invoice standard to ISO 20022 as a logical step to secure alignment and interoperability between ISO 20022 and ISO15000 (ebXML).
- [r62] The vision of the EPC is that automated reconciliation of invoices will become much simpler as banks commit themselves to use the scheme to pass remittance reference information unchanged throughout the complete banking system on behalf of the originating customer to the intended payment beneficiary. This information may be structured or unstructured at the discretion of the person making the transfer. [r21]
  The RMG has approved two proposals for development of messages. These are: 1) The 'Invoice Financing Request' and 2) the 'Trade Services Management'. [r47] The purpose of the invoice financing request messages is to create possibilities for banks to offer invoice financing services through information exchange between them and the account owners like for example companies. An invoice financing request contains chiefly common information like customer and supplier information, the amount to pay and the payment terms and conditions.

3.9.6 Example

[r47] Existing standards for electronic invoicing are for example from Finvoice, Visa, EDI/EDIFACT and OASIS. [r85] Universal Business Language (UBL) can be used to answer the question: "How could an e-invoice schema look like?". UBL is developed by OASIS Technical Committee with participation from a variety of industry data standards organizations. It is designed to plug into existing business, legal, auditing, and records management practices, eliminating the re-keying of data in existing fax- and paper-based supply chains and providing an entry point into electronic commerce for small and medium-sized businesses. UBL can operate within a standard business framework such as ISO 15000 (ebXML).
UBL schemas are modular, reusable, and extensible. UBL contains a royalty-free library of standard electronic XML business documents such as invoices and purchase orders. It also contains a library of XSDs for reusable data components such as "Address," "Item," and "Payment", which are the common data elements of these or other business documents.
UBL 2.0 contains 31 document types. One of them is 'invoice', which is meant for billing between supplier accounting party and customer accounting party. It describes the payment for goods or services supplied under conditions agreed by both parties. The 18 pages long XSD specification can be found at xsd/maindoc/UBL-Invoice-2.0.xsd of source [r85].

[17] Document Trade means the exchange of documents for secure business with high risc companies.

3.10 Advantages

[r45] Standardization in the financial world saves costs, because financial institutions speak the same language for their financial communications. Other advantages of ISO 20022 are that it:

- Improves quality
  - Focus on end-to-end business requirements
  - Industry validation of requirements & solutions
  - Formal business modeling
  - Dictionary with reusable components
- Reduces development and implementation costs
  - Decouple business and physical message standard
  - Reusable components
  - Use well-supported open standards (UML, XML, XMI[18])
  - Increase automation capabilities ("e-products")
  - Facilitate interoperability (coexistence)
- Improves time-to-market
  - Automated generation of physical message standards
  - Faster implementation ("injection")

3.11 Disadvantages

For this research there are several people approached, to find out what the disadvantages of UNIFI are.

ISO 20022 can be seen as the answer on the question: "what has been standardized for financial transactions in SEPA?"
Despite ISO 20022 is the result of a lot of conversations between people/organizations[19], it could be possible that the specification doesn't meet the whishes of all users (financial institutions). This could be caused by technical constraints, political reasons or something else.
The former leads to the question: "Is the expression strength of XSD sufficient?".
To find an answer to this question, other questions can be asked, like:
"what should be standardized for financial transactions in SEPA?".
Examples are 'from' and 'to' of the type string and 'account number' and 'amount' of the type integer.
This question is hard to answer, because there are a lot of different kinds of transactions, each with a lot of attributes. Therefore the question should be more specific. Like for example:
"what should be standardized different, than now the case is with ISO 20022?" (What do financial institutions like to have standardized in the context of financial transactions?).

SEPA Credit Transfer
Innopay answered that the Central Bank Reports (CBR) might disappear. These CBRs are used to get more insight in financial transactions. It provides countries with information for the balance of payments. If for example a payment above the 50,000 euros from Germany to the Netherlands took place, than Germany would like to know why the payment has been executed and banks would like to know what the euro amount for goods is. For Germany there is a maximum of 8 x 256 characters to indicate why a transaction took place. This feature does not exist in SEPA Credit Transfer. It's indecisive if CBR will be phased out with the introduction of SEPA.

SEPA Direct Debit
Innopay described that the case for SDD is much more complex than SCT. This is caused by the direct debit process. In Europe there are two possibilities: the mandate can be send to the supplier and the supplier will send this mandate to the bank or the mandate can be send to the bank directly. The ECB is figuring out how these processes exactly work at the moment. Although the options for SDD are still open, it will be sure that local processes will change.

[18] The XML Metadata Interchange (XMI) is an Object Management Group (OMG) standard for exchanging metadata information via XML.
[19] The SEG members are the representatives of the future end-users of UNIFI messages.

From the answer of Innopay can be concluded that there are no disadvantages for SDD identified yet.

SEPA Cards Framework
Although there are differences between the card frameworks of countries, there are no disadvantages known for the introduction of SCF.

3.12 Summary

This chapter described what UNIFI is, why it has been developed, which kind of messages do exist and an example of the implementation of these.
The UNIFI standard provides a common platform for the development of financial industry messages in an XML syntax. It captures in a syntax-independent way financial areas, business transactions and associated messages flows. XML design rules are used to convert UML messages into XML schemas. Because this technology has been used, also an explanation of XML and XSD has been given.
UNIFI has been developed because it enables the communication interoperability between financial institutions, their market infrastructure and their communities.
At the moment there exist 45 messages that are related to the funds industry (evaluated by the Security SEG), 15 messages for foreign exchange transactions (evaluated by the FX SEG) and 27 messages that are related to payment (evaluated by the Payment SEG). Payment messages can be divided into payment initiation, clearing & settlement and cash management. The Trade Service SEG did not published messages yet.
An example of a payment message has been given to answer the question about the implementation of these messages. As example the *'Financial Institution To Financial Institution (FIToFI) Customer Credit Transfer'* message has been used. This message can be used for inter-bank movement of money from a party bank account (the Debtor) to beneficiary party (the creditor). The chapter continued with a description about e-invoicing. It is possible to define e-invoicing messages with UNIFI, but these are not published yet.
Advantages of UNIFI are 1) the improvement of quality, 2) a reduction of the development & implementation costs and 3) that standardized messages can be produced and implemented faster. Disadvantages of UNIFI are that it will not be a collection of best practices, although it should be according the vision of the EPC. Germany counts for example a maximum of 8 x 256 characters to indicate why a transaction took place, but SCT does not have such a feature. This means that the Central Bank Reports (CBR) might disappear. More advantages and disadvantages can be read in the next chapter, because it will focus more on the technical aspects.

4 Theory & Technology

This chapter describes the theory and technology of standardization, languages and notations of these languages. It continues with a description of inheritance and in the last paragraph a UNIFI message has been described in a programming language.

4.1 Standardization aspects

This paragraph treats the question: "What is standardization and which aspects are related to standardization?".

4.1.1 Standardization

[r32] According to the World Wide Web Consortium is a standard the product of any community after a process of development and agreement. However, standards in the formal sense of the word are those produced according to the so-called formal standardization process, typified by organizations such as the International Standards Organization (ISO). [r70] ISO and IEC (International Electrotechnical Commission) use the following definition: a standard is a document, established by consensus and approved by a recognized body that provides, for common and repeated use, rules, guidelines or characteristics for activities or their results, aimed at the achievement of the optimum degree of order in a given context.
Standards are in the context of SEPA according to source [4 page 25], the rules that govern technology, behavior and interactions. Furthermore they add to this that technical standards are necessary to allow interaction and interoperability between IT systems and to foster automation of the payment process.
Examples of several aspects of standardization are the chosen language and thus the syntax, semantics and the compatibility / interoperability.

4.1.2 Interoperability

Definitions interoperability

What is interoperability actually? Instead of one definition, there are many. Although there is similarity between these definitions, they differ due the context they are used. Source [r23] lists the following definitions:

1) *Interoperability is the ability of a system or a product to work with other systems or products without special effort on the part of the customer. Products achieve interoperability with other products using either or both of two approaches: By adhering to published interface standards, or by making use of a "broker" of services that can convert one product's interface into another product's interface "on the fly" – adapted from Whatis.com.*
   An example of the first approach is the set of standards that have been developed for the World Wide Web. These standards include TCP/IP, HTTP, and HTML. The second kind of interoperability approach is exemplified by the Common Object Request Broker Architecture (CORBA) and its Object Request Broker (ORB).
   Compatibility is a related term. A product is compatible with a standard but interoperable with other products that meet the same standard (or achieve interoperability through a broker).

2) *Interoperability is the ability of two or more systems or components to exchange information and to use the information that has been exchanged – [IEEE 90].*

3) *Interoperability means the easy integration of products from multiple vendors without the need for custom hardware or software – Lonmark Association.*

4) *Interoperability is the ability to use documents created with one DTD for a particular purpose within another environment. For example, any DTD that uses TEI Extended Pointers for linking creates documents which are interoperable, as regards linking, with TEI-encoded documents. This*

interoperability applies even if the DTD uses no tags in common with TEI at all, since the extended pointer mechanism is controlled by a particular use of attributes, not by specific element types. Panorama Pro actually supports this particular type of interoperability, by offering built-in support for TEI Extended Pointers – CIMI briefing paper on DTD interoperability.

5) *Interoperability is defined as the ability of two or more systems or components to exchange and use information and the ability of systems to provide and receive services from other systems and to use the services so interchanged to enable them to operate effectively together – Police Information Technology Organisation.*

6) *Interoperability is the ability of computers on a network to fully share application software – Product Data Management Information Center glossary.*

7) *Interoperability is the ability of two or more systems or components to exchange information and use the exchanged information without special effort on either system.*

8) *Interoperability is the ability of two or more systems or components to exchange and to use information, the ability to provide and receive services from other systems, and the ability to use the services so interchanged to enable them to operate effectively together.*

Semantic interoperability
[r16] Another word for meaning is semantic. The relationship between data and what the data stand for is also known as data semantics. Models can be made to get a good understanding of what the data represents that could be interchanged. Semantic interoperability goes about how to achieve this (mutual) understanding.

Conclusion
Interoperability is the ability to exchange and use information. In the context of this paper, interoperability could mean the ability to use XML messages, which can be validated according the XSDs of UNIFI, within financial institutions of SEPA. Thus XSD improves interoperability between systems of financial institutions, because XSD is platform and program language independent.

4.2 Languages

[r79] A language can be defined as a set of character strings, where a string is a sequence of symbols of a finite-length. More formally described: a language is a set of strings all of which are chosen from some Σ^*, where Σ is a particular alphabet. Thus a language could have an infinite number of strings created from a finite alphabet.
Figure 23 represents a venn diagram that classifies languages. Interest goes mainly to the context free languages, because it gives a better understanding of XML. For completeness, the other languages are mentioned in short as well.

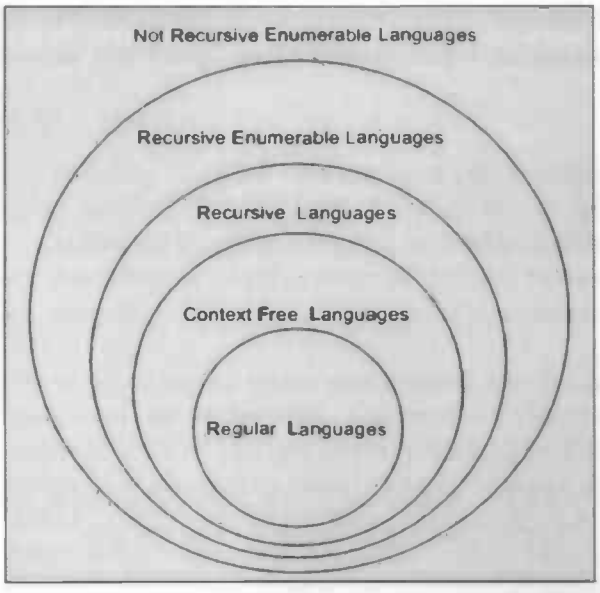


Figure 23: Venn diagram of languages

4.2.1 Regular languages

Regular languages are described by regular expressions, which are algebraic notations that describe exactly the same languages as finite automata. A finite automaton can be described with a transition diagram or a transition table and has:

1) a finite set of states;
2) a finite set of input symbols (Σ);
3) a transition function. This function needs as arguments a state and an input symbol. It returns one state in case of a deterministic finite automaton (DFA) and a set of states in case of a non-deterministic finite automaton (FNA);
4) a start state;
5) a set of final or accepting states.

4.2.2 Context Free Grammar and languages

[15, r33] A context-free grammar (CFG) is a recursive notation for a context-free language (CFL). Thus they contain a set of rules that describe the set of all possible 'sentences' in a language, such that any valid sentence has a single interpretation. Principally, all programming languages can be described by a CFG. CFG's have been used for parsers[20] since the 1960's, but more recently, they have been used to describe formats of XML documents via DTD and XSD.

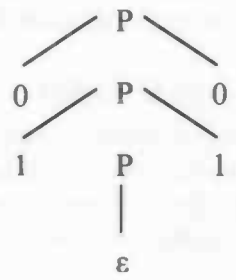
A CFG contains four parts:

1) a finite set of symbols that form the strings of the language defined. This set is also known as alphabet, terminals or terminal symbols, e.g. {0,1};
2) a finite set of variables also known as non terminals or syntactic categories that represent a language, which means a set of strings;
3) a start symbol, which is one of the variables that represents the language defined;
4) a finite set of productions or rules that represent the recursive definition of the language. These rules contain a) a variable (head) b) the production symbol '→' and c) a string of zero or more terminals and variables (body).

A CFG G can be defined as (V,T,P,S), where V stands for the set of variables, T for the terminals, P for the set of productions and S for the start symbol.

[20] Functions that discover the structure of a program.

An example of a CFG that defines a palindrome[21] is Gpal = (P, {0,1}, {P → ε | 0 | 1 | 0P0 | 1P1},P). The '|' can be interpreted as 'or'. Examples of derivations that belong to the CFL of Gpal are 0110, 010 and 11011 (see example 8).



Example 8: A parse tree for derivation of 0110

[r79] Although the first example gives a clear picture of a CFG, it does not give a clear picture how CFG is related to e.g. a DTD. The second example will clarify this (see example 9). The left side describes a DTD and the right side the CFG rules.

Document Type Definition (DTD)	CFG rules
<!DOCTYPE PcSpecs [<!ELEMENT PC (MODEL, PRICE, PROCESSOR, RAM, DISK+)> <!ELEMENT MODEL (#PCDATA)> <!ELEMENT PRICE (#PCDATA)>	PC → Model Price Processor Ram Disks Disks → Disk | Disk Disks
<!ELEMENT PROCESSOR (MANF, MODEL, SPEED)> <!ELEMENT MANF (#PCDATA)> <!ELEMENT MODEL (#PCDATA)> <!ELEMENT SPEED (#PCDATA)> <!ELEMENT RAM (#PCDATA)>	Processor → Manf Model Speed
<!ELEMENT DISK (HARDDISK | CD | DVD)> <!ELEMENT HARDDISK (MANF, MODEL, SIZE)> <!ELEMENT MANF (#PCDATA)> <!ELEMENT MODEL (#PCDATA)> <!ELEMENT SIZE (#PCDATA)> <!ELEMENT CD (SPEED)> <!ELEMENT DVD (SPEED)>]>	Disk → HardDisk | Cd | Dvd HardDisk → Manf Model Size

Example 9: A DTD and CFG that describe a language; (source [r79]

'!DOCTYPE PcSpecs' defines that the root element of this document is 'PcSpecs'. PCDATA means parsed character data. The text will be examined by the parser for entities and markup.
When children (e.g. MODEL, PRICE are children of PC) are declared in a sequence separated by commas, the children must appear in the same sequence in the document. The + sign declares that the child element 'DISK' must occur one or more times inside the 'PC' element.

4.2.3 Recursive and Recursive Enumerable languages

A Turing machine (TM) is an abstract computing machine with power of both real computers and other mathematical definitions of what could be computed.
Recursive Enumerable languages are languages that are accepted by TMs. There can be assumed that a TM halts if it is in an accepting state. If a string is not in the language the TM will reject, halt or loop forever. Recursive languages are a subset of these languages. A language is recursive if there exists a TM that will accept (and thus halt) if the input string is in the language, but will halt and reject otherwise. Thus the TM always halts. A synonym for recursive is decidable. Instead of language, the

[21] A palindrome is a string that reads the same forward and backward, e.g. 010, 11011.

word problem can be used. A problem is undecidable if it is not a recursive language or with other words: if it can not be solved by using a computer.

4.3 BNF

[r1, r19, r71] The BNF (Backus Naur Form) notation is a formal notation to describe the syntax of a given (context free) language. A formal syntax definition:

- names the syntactic parts (i.e. non-terminal symbols) of the language;
- describes the possible order of symbols that make valid sentences of the language;
- describes the syntactic structure of any sentence of the language.

The notation begins with a start symbol, which can be replaced by alternatives. An example of such an production rule is: "symbol := alternative1 | alternative2 ...". The language defined by a BNF grammar is the set of all strings that can be produced by following these rules. The notation for defining the syntax of a language by rules is also known as a syntactic metalanguage [r71]. The symbol on the left side of := or ::= will be replaced by one of the alternatives on the right side. The symbol '|' means or. Alternatives could contain symbols (non terminals) and terminals. A terminal is a piece of the language defined. They terminate the production process, because there are no productions rules for them. Thus, non terminal can be split in smaller components of the language and terminals can not. The example below is a grammar for the language of all numbers, which could be negative and fractional.

```
S  := '-' FN |FN
FN := DL | DL '.' DL
DL := D | D DL
D  := '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'
```

Example 10: BNF grammar where FN = Fractional Number, DL = Digit List and D = Digit

The example above can also be described in Extended BNF (EBNF), which contains three operators that make recursion superfluous.

- ? means that the symbol on the left is optional, thus it could appear zero or one time.
- + means that something could appear one or more times
- * means that something could appear zero or more times.

```
S := '-'? D+ ('.' D+)?
D := '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'
```

Example 11: EBNF

Other extensions are:

- that terminal symbols could be quoted. This makes it possible that any character, including the characters of EBNF itself, can be used for the language being defined;
- [and] to indicate optional symbols;
- { and } to indicate repetition;
- use of an explicit final character to eliminate ambiguity where a rule ends;
- use of brackets to group items together.

EBNF is used for programming language standards, definitions of protocol formats, data formats and markup languages, such as XML and SGML. Use of EBNF or a variant of this has the advantage that there can be no disagreement on what the syntax of the language is. Another advantage is that it makes it easy to build a compiler, because a parser for the compiler can be generated automatically with a compiler-compiler like for example YACC (Yet Another Compiler-Compiler).

[r71] The ISO 14977 standard describes the EBNF syntax, such that not every textbook and standard has to define its own BNF variant. A standard syntactic metalanguage should be:

1) concise, which makes the language compact and more easily to understand;
2) precise, which makes the rules unambiguous;
3) formal, which means that rules can be parsed or processed by a computer;
4) natural, which makes the format and notation easy to understand;

5) general, which makes it suitable for multiple purposes including the description of many different languages;
6) self describing, thus the notation could describe itself;
7) linear, which makes it possible to express the syntax as a single stream of characters. This results in easier printing and computer processing of the syntax.

4.4 Inheritance

This part is related to questions that are defined in the introduction:
"What are the possibilities around inheritance in XML/XSD? Are the schemas of UNIFI generic enough to apply (more) inheritance? How is inheritance applied in Object Oriented languages like Java? Give a description of the generics and inheritance of this language."

Inheritance can be seen as the ability of a new class to be created from an existing class. The new class, derived class or subclass, *inherits* the members of the existing class, base class or superclass, and may extend the functionality of the base class by adding new types and methods and by overriding existing ones.

4.4.1 Inheritance in XML/XSD

[r34] As already mentioned it is possible to extend XSD. Thus other schemas can be reused by referring to one or more of them. Beside this, it is possible to create data types derived from existing types. This section describes more about inheritance related features of XML/XSD.

Schema in multiple documents
Large schemas are mostly undesirable, because access control, readability and thus maintenance become harder.

Figure 24 demonstrates how the include element makes definitions and declarations of 'address.xsd', 'Y.xsd' and 'Z.xsd' available in PurchaseOrder.xsd. The target namespace of the included components must be the same as the target namespace of the including schema (e.g. http://www.example.com/IPO). During processing of an XML file it will make no difference if an XSD includes other XSD(s) or if everything is located in one XSD. The processor will handle the gathering of definitions that are described in the included documents. The XML file refers only to the topmost document and the common namespace.

Deriving types
The complexContent element makes it possible to restrict or extend the content model of a complex type. A type (e.g. Address) can be extended by using the value of the 'base' attribute on the 'extension' element (see figure 24). The attribute 'restriction' limits the declarations of a (complex) base type, which means that values represented by a new type becomes a subset of the values of the base type.

```
<!--PurchaseOrder.xsd -->
<schema targetNamespace="http://www.example.com/IPO"
xmlns="http://www.w3.org/2001/XMLSchema" xmlns:ipo="http://www.example.com/IPO">

<include schemaLocation="http://www.example.com/schemas/address.xsd"/>
<include schemaLocation="http://www.example.com/schemas/Y.xsd"/>
<include schemaLocation="http://www.example.com/schemas/Z.xsd"/>

<element name="purchaseOrder" type="ipo:PurchaseOrderType"/>

<complexType name="PurchaseOrderType">
  <sequence>
    <element name="shipTo"    type="ipo:Address"/>
    <element name="billTo"    type="ipo:Address"/>
    .....
```

```
<!-- address.xsd-->
<schema targetNamespace="http://www.example.com/IPO"
xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:ipo="http://www.example.com/IPO">

<complexType name="Address">
  <sequence>
    <element name="name"   type="string"/>
    <element name="street" type="string"/>
    <element name="city"   type="string"/>
  </sequence>
 </complexType>

 <complexType name="USAddress">
  <complexContent>
    <extension base="ipo:Address">
      <sequence>
        <element name="state" type="ipo:USState"/>
        <element name="zip"   type="positiveInteger"/>
      </sequence>
    </extension>
  </complexContent>
 </complexType>

 <complexType name="UKAddress">
  <complexContent>
    <extension base="ipo:Address">
      <sequence>
        <element name="postcode" type="ipo:UKPostcode"/>
      </sequence>
      <attribute name="exportCode" type="positiveInteger" fixed="1"/>
    </extension>
  </complexContent>
 </complexType>

<simpleType name="USState">
  <restriction base="string">
    <enumeration value="AK"/>
    <enumeration value="AL"/>
    <enumeration value="AR"/>
    <!-- and so on ... -->
  </restriction>
 </simpleType>
```

```
<!--Y.xsd-->
```

```
<!--Z.xsd-->
```

Figure 24: Schema in multiple documents

Using derived types in instance documents
In example 12, both billTo and ShipTo are of the type 'ipo:Address'. The advantage is that not every combination of international address (e.g. UK and USA address) for billing and shipping has to be specified in the XSD. The attribute 'xsi:type' in an XML instance makes it possible to define which type is intended.

```
<?xml version="1.0"?>
<ipo:purchaseOrder
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:ipo="http://www.example.com/IPO"
  orderDate="1999-12-01">

  <shipTo exportCode="1" xsi:type="ipo:UKAddress">
    <name>Helen Zoe</name>
    <street>47 Eden Street</street>
    <city>Cambridge</city>
    <postcode>CB1 1JR</postcode>
  </shipTo>

  <billTo xsi:type="ipo:USAddress">
    <name>Robert Smith</name>
    <street>8 Oak Avenue</street>
    <city>Old Town</city>
    <state>PA</state>
    <zip>95819</zip>
  </billTo>
.....
```

Example 12: XML instance

Redefining types & groups
Redefine can be used instead of include to include definitions and declarations. The difference between these is that in case of redefine a new type can have the same name as the base type. An example is that a new type 'Address' will be defined that uses '*<extension base="ipo:Address">*' to add '*<element name="country" type="string"/>*' (see example 13).

```
  <redefine
    schemaLocation="http://www.example.com/schemas/address.xsd">

    <!-- redefinition of Address -->
    <complexType name="Address">
      <complexContent>
        <extension base="ipo:Address">
          <sequence>
            <element name="country" type="string"/>
          </sequence>
        </extension>
      </complexContent>
    </complexType>

  </redefine>
```

Example 13: Redefine Address

Substitution groups
Elements can be substituted for other elements with the substitution group mechanism. An XML instance may contain for example <ipo:shipComment> and <ipo:customerComment> tags instead of the 'head element': <ipo:comment>, if the PurchaseOrderType got the elements:
<element ref="ipo:comment" minOccurs="1"/>'
<element name="shipComment" type="string" substitutionGroup="ipo:comment"/>

<element name="customerComment" type="string" substitutionGroup="ipo:comment"/>

The attribute 'abstract' forces substitution of an element or type and the use of substitution groups. For example the following line can be used to forbid the use of the comment element in an XML instance of example 12: <element name="comment" type="string" abstract="true"/>.

Controlling the creation & use of derived types

Derivation can be controlled for complex types by 1) restriction[22], 2) extension or 3) both (e.g. <complexType name="Address" final="restriction">). Another option is to add the 'finalDefault' attribute to the schema element to do this for the whole document. The attribute FinalDefault has got just as 'final' one of the three values 'restriction',' extension', or '#all'.

Also for simple types there is an attribute for the type-derivation mechanism, namely the 'fixed' attribute. Example 14 represents a simple type for a postcode that must have a length of 7. Although this length criterion may not be changed in derivations of this type, a derived type can still make restrictions like for example which symbols are allowed.

```
<simpleType name="Postcode">
  <restriction base="string">
    <length value="7"
fixed="true"/>
  </restriction>
</simpleType>
```

```
<simpleType name="UKPostcode">
  <restriction base="ipo:Postcode">
    <pattern value=
       "[A-Z]{2}\d\s\d[A-Z]{2}"/>
  </restriction>
</simpleType>
```

Example 14: Derivation of simple types; \d = digits 0-9; \s = whitespace; {n} = repeats the previous item

Besides these mechanisms to control type derivations, there is also a mechanism that controls which substitution groups and derivations may be used in XML instances. The attribute 'block' can be used to block any derivation-by-restriction from being used instead of e.g. 'Address' (see example 15).

```
<complexType name="Address" block="restriction">
  <sequence>
    <element name="name"   type="string"/>
    <element name="street" type="string"/>
    <element name="city"   type="string"/>
  </sequence>
</complexType>
```

Example 15: The block attribute

Instead of the value 'restriction', also 'extension' and '#all' can be used. To do this for the whole document, the blockDefault attribute with one of the block values can be added to the schema element.

Importing Types

A disadvantage of the include element is the demand that the included XSD's must have the same target namespace. By using the include element, only definitions and declarations of the same target namespace could be used. The include element requires a schema location, because multiple documents use the same target namespace.

The import mechanism enables global named components from different target namespaces to be used together. Import elements must appear as the first children of the schema element. They also must have a prefix and can have a schema location optionally. The prefix will be used to qualify references to any schema component that belongs to that namespace.

[22] See also 'facets' of the previous chapter.

Atomic, list and union types
Simple types can be divided into atomic types, list types and union types. An atomic type is indivisible from the XML Schema perspective. The NMTOKEN 'US' is for example indivisible, because 'U' or 'S' does not have a meaning by itself. An example of an element of a list type is "US UK FR". Atomic types and list types enable an element or an attribute value to be one or more instances of one atomic type. In contrast, union types enables an element or attribute value to be one or more instances of one type drawn from the union of multiple atomic and list types. Example 16 creates a union type for representing American states as singleton letter abbreviations or lists of numeric codes. The zipUnion union type is built from one atomic type and one list type:

```
<xsd:simpleType name="zipUnion">
  <xsd:union memberTypes="USState listOfMyIntType"/>
</xsd:simpleType>
```

Example 16: Union type for zipcodes

Conclusion
An XSD can be extended with other XSDs and data types can be derived from existing types. The import mechanism enables global named components from different target namespaces to be used together. Furthermore it is possible to redefine types and groups. Redefine makes it possible to create a new type that can have the same name as the base type. The attribute 'abstract' forces substitution of an element or type and the use of substitution groups. Derivation can be controlled for complex types by 1) restriction, 2) extension or 3) both. Simple types can be divided into atomic types, list types and union types. An atomic type is indivisible form XML Schema perspective. Atomic types and list types enable an element or an attribute value to be one or more instances of one atomic type. In contrast, union types enables an element or attribute value to be one or more instances of one type drawn from the union of multiple atomic and list types.

4.4.2 Inheritance in UNIFI

This section treats the questions: "How much of the inheritance related features of XSD are applied on the XSDs of UNIFI?" and "are the schemas of UNIFI generic enough to apply (more) inheritance?" This question could be answered by:

1) building further on research of this topic that have been carried out;
2) examining the creation process of XSDs, because it could contain rules and purposes with reference to inheritance;
3) examining samples of the UNIFI schemas that represent the approximately 313 pages of 29 schemas for only the UNIFI Payment part;
4) examining samples of the UNIFI documentation.

As far it could be affirmed, this question is not already answered by others.

The creation process of XSDs
[r74] Part three of the ISO 20022 standard contains modeling guidelines and provides thus information about the creation process of XSDs. The ISO 20022 methodology can be divided into the phases: 1) business analysis, 2) requirements analysis, 3) logical analysis, 4) logical design also known as message design and 5) technical design.
Phase four is aimed to redefine the logical model of phase three in order to get it formal. Furthermore it is aimed to find reusable message components.

The following lines describe a number of definitions to clarify 'message components' (see figure 25).

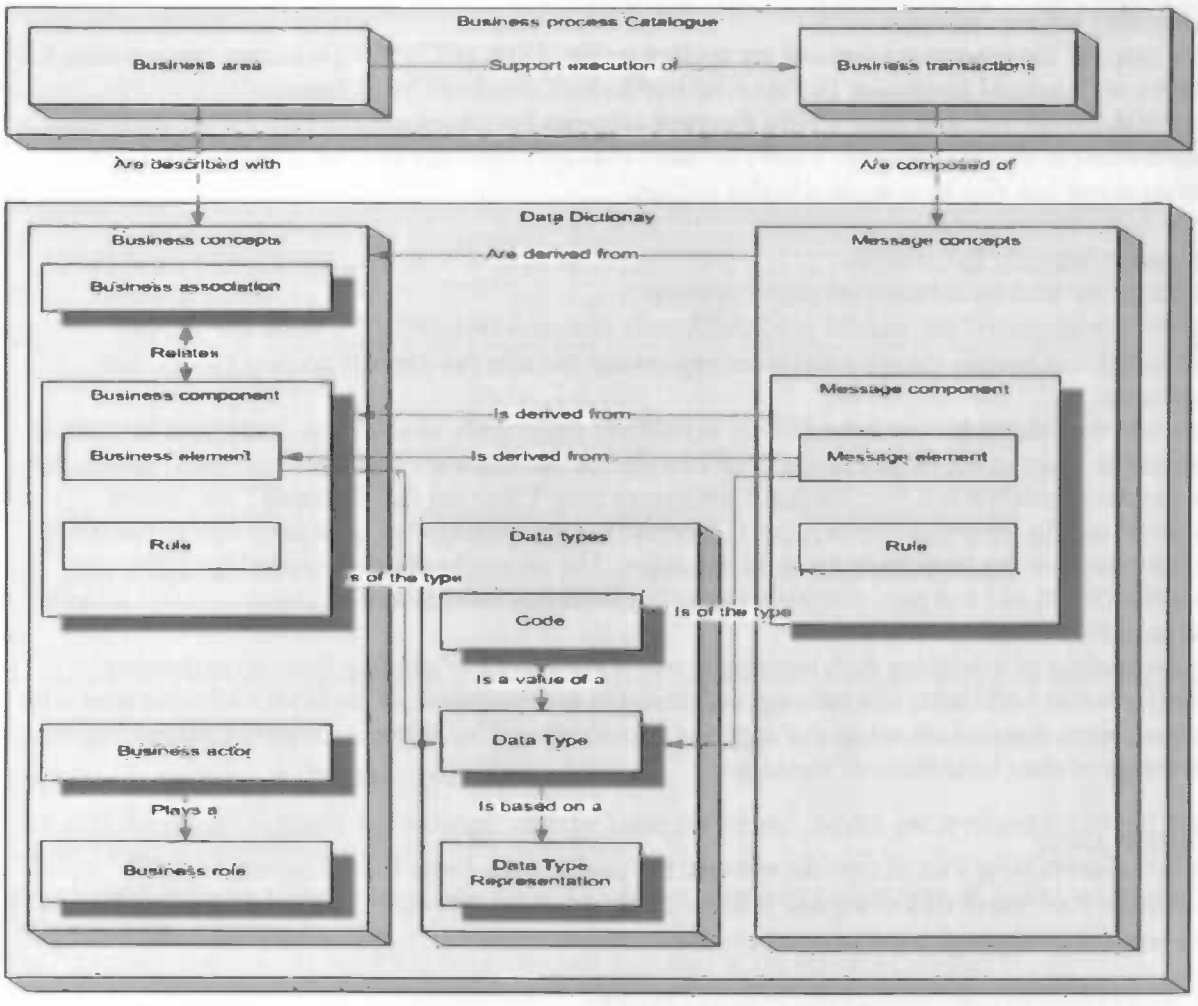


Figure 25: The UNIFI repository

The definition of a message in the context of ISO 20022 is: a set of set of structured information exchanged between *business actors* or *business roles*, in the scope of a *business transaction.* A business actor means a physical business user playing one or more business roles in a particular *business process* or *business transaction.*
A message component is a re-usable dictionary item that can be used as a building block for assembling message definitions. A message component exists of a set of message elements and rules. Examples of message elements are 'Trade' 'Date' and 'Time' as part of the message component 'Trade Details'. Dictionary items are stored in the data dictionary, which is a part of the ISO 20022 repository that contains all items that can be re-used during business process modeling and message definition activities. Message definitions are formal descriptions of the structure of a message.

The output of phase four exists of hierarchical class diagrams, which are called message definition diagrams, and textual rules that are written in a formal language.
Phase five consist of mapping logical descriptions of e.g. messages and collaborations to the target technology like for instance XSD[23], for the message syntax and programming code for some validation rules. The mapping specifications are described in part four (XML design rules) of the ISO 20022 standard [r75]. This standard states that there is one schema per message definition diagram.
After reading this specification can be concluded that the mapping is a one to one process.
Furthermore can be concluded that decisions about the structure (including inheritance) of messages and as a consequence the structure of XSDs are made during phase four.

[23] XSD is at the moment the only used target technology.

Examining schema samples

How much of the inheritance features are applied on the XSDs of UNIFI? To answer this question the schema of 'Financial Institution To Financial Institution Customer Credit Transfer' (pacs.008.001.01.xsd) and other UNIFI Payment schemas have been used to verify if the inheritance related features of XSD appear in UNIFI schemas (see also appendix 8). The same structure as the section above has also been applied to this section.

Schema in Multiple Documents

Although the schema contains the targetnamespace "targetNamespace="urn:iso:std:iso:20022:tech:xsd:pacs.008.001.01", there are no other documents that contain the same targetnamespace and the schema does not contain any include statements.

However the schema has not been split up in multiple documents, one schema does occur in multiple documents. Two of the twenty seven UNIFI Payments messages are used multiple times; namely for the payment initiation and the clearing & settlement part. These are the 'Payment Cancellation Request' and the 'Payment Status Report'. Normally, each message has been described in one XSD, but an exception has been made for these messages. The former has been described in pain.006.001.01.xsd and pacs.006.001.01.xsd. The latter has been described in pain.002.001.02.xsd and pacs.002.001.02.

One advantage of describing each message in one XSD instead of splitting them up in multiple schemas is that boundaries of a message are clear. On the other hand, if the UNIFI schemas have a lot of equal parts, than the advantages of applying (more) inheritance/reference could be bigger than the advantage of clear boundaries of messages.

Deriving Types

The schema contains a lot of type derivations. For example the Basic Bank Account Identifier, which is a simpleType based on a string and restricted with a regular expression. The identifier has been used as a type for an element, e.g. "`<xs:element name="BBAN" type="BBANIdentifier"/>`".

```
<xs:simpleType name="BBANIdentifier">
  <xs:restriction base="xs:string">
    <xs:pattern value="[a-zA-Z0-9]{1,30}" />
  </xs:restriction>
</xs:simpleType>
```

Example 17: restriction of a string type

Another example of a derivation is the extension of the complexType 'currency and amount' (see example 18, part A). This complexType inherits the properties of "`CurrencyAndAmount_SimpleType`" (part B) and adds an attribute to it of the type "`CurrencyCode`" (part C).

```
<xs:complexType name="CurrencyAndAmount">                                   A
  <xs:simpleContent>
    <xs:extension base="CurrencyAndAmount_SimpleType">
      <xs:attribute name="Ccy" type="CurrencyCode" use="required"
/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
<xs:simpleType name="CurrencyAndAmount_SimpleType">                         B
  <xs:restriction base="xs:decimal">
    <xs:minInclusive value="0" />
    <xs:fractionDigits value="5" />
    <xs:totalDigits value="18" />
  </xs:restriction>
</xs:simpleType>
```

<xs:simpleType name="CurrencyCode"> <xs:restriction base="xs:string"> <xs:pattern value="[A-Z]{3,3}" /> </xs:restriction> </xs:simpleType>	C
<xs:element name="InstdAmt" type="CurrencyAndAmount" minOccurs="0" maxOccurs="1" />	D
<xs:element name="IntrBkSttlmAmt" type="CurrencyAndAmount" />	E
<xs:element name="ChrgsAmt" type="CurrencyAndAmount" />	F
<xs:element name="Amt" type="CurrencyAndAmount" minOccurs="0" maxOccurs="1" />	G
<xs:complexType name="ReferredDocumentAmount1Choice"> <xs:sequence> <xs:choice> <xs:element name="DuePyblAmt" type="CurrencyAndAmount" /> <xs:element name="DscntApldAmt" type="CurrencyAndAmount" /> <xs:element name="RmtdAmt" type="CurrencyAndAmount" /> <xs:element name="CdtNoteAmt" type="CurrencyAndAmount" /> <xs:element name="TaxAmt" type="CurrencyAndAmount" /> </xs:choice> </xs:sequence> </xs:complexType>	H
<xs:element name="TtlIntrBkSttlmAmt" type="CurrencyAndAmount" minOccurs="0" maxOccurs="1" />	I

Example 18: extension of CurrencyAndAmount

Note that the former is based on 'decimal' and the latter on 'string', which are prefixed with 'xs:" This prefixes refer to an attribute "xmlns:xs="http://www.w3.org/2001/XMLSchema" of the root element 'schema'. The value of this attribute refers to a default simple type of XSD (see appendix for more of these types).
The 'currency and amount' type has been invoked ten times in the schema (see part D – I).

Using Derived Types in Instance Documents
The sample XML instances from the UNIFI documentation did not contain derived types. The attribute type or xsi:type did not occur.

Redefining Types & Groups
The redefine tag has not been used in the UNIFI Payment messages. Thus, apparently there was no need to include definition and declaration in another way than with the 'include' statement.

Substitution Groups
Also this feature has not been used in the UNIFI Payment messages.

Controlling the creation & use of derived types
The attributes fixed, block and finalDefault have not been used.

Importing Types
The import statement does not occur.

Atomic, list and union types
Of course the atomic types do occur, but that is not the case with the union types and list types. The built-in list types NMTOKENS, IDREFS, and ENTITIES are not used and the key word 'list' for defining lists are not used either. However, this does not mean that the UNIFI schemas don't contain lists, because these can be defined with an enumeration.

Overview of used attributes and elements

The table below gives an overview of occurrences of inheritance related default attributes and elements. The table is a subset of the table that is listed in appendix 5. The first column contains default attributes and elements, which are (mostly) discussed earlier in this thesis. The second column indicates if the attribute or element occurs in one of the UNIFI Payment schemas. The third column gives a short description of the first column.

Attributes	Used	Description
abstract	No	If the value of this attribute is true, than the element can not be used in an instance document. It forces to use a member of the element's substitution group to appear in the instance document.
attributeFormDefault	No	Can be added to the schema element to indicate if local declared attributes must be qualified or unqualified.
base	**Yes**	A type definition used as the basis for an extension or restriction is known as the base type definition of that definition
block	No	Can be used to block any derivation-by-restriction from being used. Possible values are 'restriction', 'extension' and '#all'.
blockDefault	No	Can be added to the schema element to apply the block option for the whole document.
default	No	Default attribute values are applied when attributes are missing. Not to confuse with default element values that will be applied when elements are empty.
elementFormDefault	**Yes**	Can be added to the schema element to indicate if local declared elements must be qualified or unqualified.
final	No	No further derivations are possible if the complex type is final.
finalDefault	No	Can be added to the schema element and has the same effect as specifying the final attribute on every type definition and element declaration in the schema.
fixed	No	Can be applied to any of its facets to prevent a derivation of that type from modifying the value of the fixed facets.
memberTypes	No	The datatypes that participate in the definition of a union datatype are known as the memberTypes of that union datatype.
mixed	No	If this attribute is set on true, than it is allowed to put data between child-elements of a complexType.
ref	No	An element that refers to a global element enables the referenced element to appear in the instance document in the context of the referencing declaration.
schemaLocation	No	Can be used in a schema, because the include and import element requires this attribute to refer to another schema document.
xsi:schemaLocation	No	Can be used in an XML instance to provide hints from the author to a processor, to refer to the location of the schema document.
substitutionGroup	No	This attributes can be used at elements that substitute other elements. The value of this attribute refers to the element that will be substituted.
targetNamespace	**Yes**	Several kinds of component have a target namespace, which is either absent or a namespace name. XML namespaces provide a method for qualifying element and attribute names used in XML documents by associating them with namespaces identified by URI references.
xsi:type	No	To identify which derived type must be used in an XML instance document.
use	**Yes**	Indicate whether the attribute is required, optional, or prohibited.
Elements		
all	No	This is an option to constrain a group of elements. All the elements of the group may appear once or not at all. The elements may also appear in any order.
any	No	This element specifies that any well-formed XML is permissible in a type's content model.
extension	**Yes**	This allows element or attribute content in addition to that is allowed by another specified type definition.
group	No	It provides for naming a model group for use by reference in the XML representation of complex type definitions and model groups.
import	No	Can be used to refer to a schema with another target namespace.
include	No	Can be used to refer to a schema with the same target namespace.

redefine	No	Can be used to include definitions and declarations. A new type can have the same name as the base type.
restriction	**Yes**	This is a type definition whose declarations or facets are in a one-to-one relation with those of another specified type definition, with each in turn restricting the possibilities of the one it corresponds to.
union	No	This element can be used to declare union types.
unique	No	Makes it possible to indicate that any attribute or element value must be unique within a certain scope.

Example 19: Occurrences of inheritance related default attributes and elements

Although not all inheritance related features of XSD have been used in the XSDs of UNIFI, the most important ones like base, restriction and extension have been applied on the XSDs of UNIFI.

Examining documentation samples

The documentation used for this research part exists of web pages that have been extracted automatically from a Rational Rose model with the Rose Web Publisher. It contains a logical view that is divided into 1) business models, 2) requirements, 3) system description, 4) messages and 5) 'Dictonary'. The dictionary is divided into 5a) data types, 5b) business actors and roles, 5c) business components and 5d) message components.

A statement has been made to answer the question: "are the schemas of UNIFI generic enough to apply (more) inheritance?" This statement is:
"schemas of UNIFI are generic enough to apply (more) inheritance if message components of UNIFI payment messages are multiple times defined in 1) payment initiation (pain), 2) clearing & settlement (pacs) 3a) cash management - bank-to-customer cash management (camt a) and 3b) cash management - exceptions and investigations (camt b).
As a result of table 22, but also the examination of 5a, 5b and 5c, can be assumed that the complete content of the 'Dictionaries' of the Rational Rose model documentation are not meant for one part of the UNIFI messages. This means that another way has to be found to validate/invalidate the statement or the statement should be changed.

	1 pain	**2 pacs**	**3a camt**	**3b camt**
# components	187	187	194	186
1 pain	x	equal	overlap	overlap
2 pacs	x	x	overlap	overlap
3a camt	x	x	x	overlap
3b camt	x	x	x	x

Table 22: payment message components of 5) the 'Dictionary'

[r77] In the message definition reports of the UNIFI message they don't use the term 'message component'. The documents describe for each message, like for example the FIToFICustomerCreditTransferV01 message (pacs.008.001.01.xsd) the:
1) message scope and usage including the rules,
2) the message structure,
3) the message items description
4) and a number of examples.

The message structure has been described in a table, which lists message items, XML tags and their type (see table 23).
An XML tag is a specific name assigned to a 'Message Element'. This name will appear as an XML tag in XML instances and XML schemas.
Note that message items could contain other message items. For example 'GroupHeader' contains e.g. 'MessageIdentification' and 'CreationDateTime'.

Message Item	**<XML Tag>**	**Mult.**	**Represent./Type**
GroupHeader	<GrpHdr>	[1..1]	

MessageIdentification	<MsgId>	[1..1]	Text
CreationDateTime	<CreDtTm>	[1..1]	DateTime
BatchBooking	<BtchBookg>	[0..1]	Indicator
NumberOfTransactions	<NbOfTxs>	[1..1]	Text
Etc.			

Table 23: Message structure of pacs.008.001.01.xsd (FiToFiCustomerCreditTransferV01); source [r78]

As a result of this knowledge the statement can be changed into:
"schemas of UNIFI are generic enough to apply (more) inheritance if message items of UNIFI payment messages are multiple times defined in 1) payment initiation, 2) clearing & settlement 3a) cash management - bank-to-customer cash management and 3b) cash management - exceptions and investigations".
The message definition reports contain indexes of message items, XML tags and message item types. The former has been used to make a comparison between the UNIFI Payment message categories. Table 24 shows a method how to count and allocate the payment messages. The ascending numbers from 1 to 27 represent the XSDs of the UNIFI payment messages, which are classified in pain, pacs, camt a and camt b.

Message item	Pain					Pacs					Camt a			Camt b													Total
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	
AccptncDtTm	0	0	0	0	1	1	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5
Acct	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	4
AcctSvcr	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
AcctSvcrRef	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
ActlDt	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
AddtlCxlRsnInf	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
AddtlInf	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
AddtlNtfctnInf	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
AddtlNtryInf	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
AddtlNtryInfInd	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
AddtlRptInf	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
AddtlRmtInf	1	1	1	1	1	1	1	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12
AddtlRtrRsnInf	0	0	1	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5
AddtlRvslRsnInf	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
AddtlStsRsnInf	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
AddtlTxInf	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
Adr	1	1	0	0	0	1	1	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7
Agt	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
AmdmntInd	0	1	1	1	1	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7
AmdmntInfDtls	0	1	1	1	1	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7
Amt	1	1	1	1	1	1	1	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	13
AmtDtls	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
AmtToDbt	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0	3
AnncdPstngAmt	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
AnyInf	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
Assgne	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	13
AssgneInstrId	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0	0	0	1	1	1	1	7
Assgnmt	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	1	1	1	1	1	1	1	1	12
AssgnmtCxlConf	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
Assgnr	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	13
AssgnrInstrId	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0	0	0	1	1	1	1	7
Authstn	1	1	1	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6
Authrty	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4
AuthrtyCtry	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4
AuthrtyNm	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4
Avlbty	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
Etc.																												

Table 24: Payment message items of the message definition reports

176	is total number of occurrences of message items in the sample
27	is total number of schemas
29	is total number of messages
36	is total number of message items of this sample
4,89	is the average number of occurrences of one message item

Table 25: Statistics of the table above this one

Due to time constraints the table is not complete. Nonetheless it is already possible to anticipate and to define expectations. For this research can be expected that the statement above will be true. That the message items appear multiple times (avg # 4,89: see table 25) is not surprising, because that is part of the concept of the UNIFI repository. Although this says in some degree about the balance between 'clear boundaries' and 'use of more inheritance and reference', the ISO has decided to create for every message a different schema. However, an alternative is the use of XSDs with a generic character. Use of these generic XSDs will reduce the length of the content of the XSDs that include these generic XSDs.

4.4.3 Inheritance & generics in Java

Inheritance

[r2] Inheritance in the Java language means deriving classes from other classes. A subclass (or derived class or extended class or child class) is a class that is derived from one and only one (single inheritance) superclass (or base class or parent class). Only 'Object' has no superclass.
A subclass inherits all the public and protected members from its superclass, which are fields[24], methods, nested classes, but no constructors. Although a constructor of a superclass is not a member, it can be invoked from a subclass. A subclass also inherits the package-private members of a parent, if a subclass is in the same package as its parent. A subclass doesn't inherit the private members of its parent class.
Inherited members can be used as is, replaced, hided or supplemented with new members:

- The inherited fields can be used directly.
- A field can be hided by declaring a field in the subclass with the same name as in the superclass.
- New fields can be declared in the subclass that doesn't exist in the superclass.
- A new *instance* method can be written in the subclass that has the same signature as the one in the superclass, thus *overriding* it.
- A new *static* method in the subclass can be written in case it has the same signature as the one in the superclass, thus *hiding* it.
- New methods can be declared in the subclass if they are not declared in the superclass.
- A subclass constructor can be written that invokes the constructor of the superclass, either implicitly or by using the keyword super.

Overriding and hiding methods

Overriding occurs if the instance method in the subclass and the superclass use 1) the same signature, thus the name, the number and the type of parameters, and 2) the same return type (e.g. testInstanceMethod of example 20). The return type of an overriding method that returns a subtype of the type returned by the overridden method is also know as a covariant return type.
Hiding occurs if a subclass defines a class method with the same signature (e.g. testClassMethod in example 20) as a class method in the superclass (see table 26). Note that this method uses the word static, which makes it a 'class method'. This means that the method should invoked by the class name, which implies that it is not necessarily to create an instance of the class.

[24] A *field* is a member variable in a class. Other kinds of variables are the *local variable*, which is a variable in a method or block of code and the *parameter*, which is a variable in a method declaration.

	Superclass Instance Method	Superclass Static Method
Subclass Instance Method	Overrides	Generates a compile-time error
Subclass Static Method	Generates a compile-time error	Hides

Table 26: Defining a Method with the Same Signature as a Superclass's Method

```
public class Animal {
    public static void testClassMethod() {
        System.out.println("The class method in Animal.");
    }
    public void testInstanceMethod() {
        System.out.println("The instance method in Animal.");
    }
}
//The second class, a subclass of Animal, is called Cat:
public class Cat extends Animal {
    public static void testClassMethod() {
        System.out.println("The class method in Cat.");
    }
    public void testInstanceMethod() {
        System.out.println("The instance method in Cat.");
    }

    public static void main(String[] args) {
        Cat myCat = new Cat();
        Animal myAnimal = myCat;
        Animal.testClassMethod();
        myAnimal.testInstanceMethod();
    }
}
```

Example 20: Overriding versus hiding

The main method in the example above creates an instance of Cat and calls testClassMethod() on the class and testInstanceMethod() on the instance.
The version of the hidden method that gets invoked depends on whether it is invoked from the superclass or the subclass. The output of the example above is: "The class method in Animal. The Instance method in Cat".
The access specifier for an overriding method can allow more, but not less access than the overridden method. Thus a protected instance method in the superclass can be made public, but not private, in the subclass.

Hiding fields and the keyword super

Although it is not recommend, it is possible to hide fields of a superclass. This happens when a field in a subclass has the same name as a field in a superclass, even if their types are not the same. A field in a superclass can be accessed from a subclass by prefixing it with the keyword 'super' followed with a dot. This works also for methods. The syntax for calling a constructor of a superclass is 'super();' or 'super(parameter list);'. The compiler automatically inserts a call to the former if there is no explicit invocation in a constructor to a superclass. A compile-time error will arise if the superclass does not have a no-argument constructer. Such an error will not occur if Object is the only superclass, because Object does have a no-argument constructor.

Writing Final Classes and methods

The final keyword in a method declaration can be used to indicate that the method can not be overridden by subclasses. If a class has been made final, it is not possible to make sub classes of the class anymore.

Interfaces

Although it is in Java not possible to inherit from multiple classes, it is possible to implement multiple interfaces. In the Java context, an interface is a reference type, which is similar to a class, but can only contain constants, method signatures and nested types. An example of a method declaration is: "int turn(Direction direction, double radius, double startSpeed, double endSpeed);". Thus without a body and ending with a semicolon. Java does not allow making instances of an interface. They can be implemented by classes or extended by other interfaces. Objects can have multiple types, namely, the type of their own class and the type of all the interfaces that they implement. If a variable is declared to be the type of an interface, its value can reference any object that is instantiated from any class that implements the interface.

Abstract Methods and Classes

Abstract classes can not be instantiated. They should be subclassed. An abstract method is a method without a body, e.g. "abstract void moveTo(double deltaX, double deltaY);". A class must be declared abstract if it contains abstract methods. Methods of an interface are implicitly abstract. This means that it is unnecessary to make interfaces abstract.

Differences between abstract classes and interfaces are that abstract classes can contain fields that are not static and final. Furthermore an abstract class can contain implemented methods, which is not the case with interfaces. An abstract class with only abstract method declarations should be declared as an interface. Another difference is that multiple interfaces can be implemented by classes on every place in the class hierarchy.

Generics

[r2] Use of generics will lead to software which is more stable, because more bugs can be detected at compile time instead of runtime.

The example below illustrates a class in a non-generic way.

```
public class Box {

        private Object object;

        public void add(Object object) {
            this.object = object;
        }

        public Object get() {
            return object;
        }
    }
```

Example 21: A simple box class; source [r2]

If the class will be used in a proper way, it will not lead to errors. Thus by adding and casting of the same type:

```
integerBox.add(new Integer(10));
Integer someInteger = (Integer)integerBox.get();
```

However, the example below illustrates code that will give no compile errors, but will lead to an error at runtime: 'java.lang.ClassCastException: java.lang.String cannot be cast to java.lang.Integer'

```
public class BoxDemo2 {

    public static void main(String[] args) {

        // ONLY place Integer objects into this box!
        Box integerBox = new Box();

        // Imagine this is one part of a large application
        // modified by one programmer.
```

```
        integerBox.add("10"); // note how the type is now String

        // ... and this is another, perhaps written
        // by a different programmer
        Integer someInteger = (Integer)integerBox.get();
        System.out.println(someInteger);
    }
```

Example 22: Cast Exception in non-genic code; source [r2]

Generic Types

The example below will prevent such cast exception. By adding a 'type variable' (formal type parameter): '<T>' after 'public class Box', a 'generic type declaration' has been made. This variable can have 'values' of any class type, interface type or any other type, but not a primitive data type.

```
public class Box<T> {

    private T t; // T stands for "Type"

    public void add(T t) {
        this.t = t;
    }

    public T get() {
        return t;
    }
}
```

Example 23: Generic version of the Box class; source [r2

The class can be instantiated by a 'generic type invocation, e.g.
'Box<Integer> integerBox = new Box<Integer>();' Use of the get method can be done without casting:
integerBox.add(new Integer(10));
Integer someInteger = integerBox.get(); // no cast!

Adding an incompatible type will cause a compilation error.
A generic type could have multiple type parameters, as long as each parameter is unique within the declaration (e.g: Box<T,U>). Commonly used parameters are:
1) E = Element (used extensively by the Java Collections Framework); 2) K = Key; 3) N = Number; 4) T = Type; 5) V = Value; 6) S,U,V etc. = second, third, fourth types.

Generic Methods and Constructors

To limit the scope of a type parameter, it can also be declared within a method or a constructor. For example: 'public <U> void inspect(U u){…'.

Bounded Type Parameters

Parameters can be restricted by using the type parameter's name, than 'extends' and thirdly the upper bound, e.g. Number (see example below).

```
public <U extends Number> void inspect(U u){
    System.out.println("T: " + t.getClass().getName());
    System.out.println("U: " + u.getClass().getName());
}
```

Example 24: Bounded type parameter

In case a string will be passed to the method 'inspect' an error will raise:
<U>inspect(U) in Box<java.lang.Integer> cannot be applied to (java.lang.String)
integerBox.inspect("10");

The & character can be used to indicate that extra interfaces must be implemented, e.g. '<U extends Number & MyInterface>'.

Subtyping

If types are compatible with each other, than an object of one type can be assigned to one of another type. An integer and a double are *a kind of* Number (see example below).

```
public void someMethod(Number n){ // method body omitted  }

    someMethod(new Integer(10));  // OK
    someMethod(new Double(10.1)); // OK
```

Example 25: Kind of relationship

This principle works also with generics:

```
    Box<Number> box = new Box<Number>();
    box.add(new Integer(10));  // OK
    box.add(new Double(10.1)); // OK
```

Example 26: Sub typing

However, it is not allowed to invoke e.g.
'public void boxTest(Box<Number> n){…}' with 'Box<Integer>' or 'Box<Double>', because the latter two are no subtypes of 'Box<Number>'. A similar example will also result in a compile error: 'public void cageTest(Cage<Animal> n){…}' invoked with 'Cage<Lion>' or 'Cage<Butterfly>'.This example illustrates that there is no cage capable to hold both lions and butterflies.

Wildcards

To build further on the example above, it is possible to specify a cage that could hold a certain kind of animal, including Animal itself (see the example below). Cage<Lion> and Cage<Butterfly> are subtypes of someCage.

```
Cage<? extends Animal> someCage;
someCage = lionCage; // OK
someCage = butterflyCage; // OK
```

Example 27: Use of a bounded wildcard where Animal forms the upper bound of the expected type

Instead of 'extends' it is also possible to use 'super', to hold a supertype of Animal (including Animal itself). The unbounded wildcard '<?>' can be used to specify an unknown type, which expresses the same as <? extends Object>.
Although, 'someCage.add(new Lion())' will result in an error at compiler-time, this method can be used to read from (see example below).

```
void feedAnimals(Cage<? extends Animal> someCage) {
    for (Animal a : someCage)
        a.feedMe();
}
feedAnimals(lionCage);
feedAnimals(butterflyCage);
feedAnimals(animalCage);
```

Example 28: Use of wildcards

Type Erasure

This is a technique where the compiler removes all information related to type parameters and type arguments within a class or method. Type erasure enables applications that use generics to maintain binary compatibility with libraries and applications that were created before generics. Box<Integer> will for example be translated to the raw type: 'Box'. The example below illustrates meaningless operations at runtime.

```
public class MyClass<E> {
    public static void myMethod(Object item) {
        if (item instanceof E) {  //Compiler error
```

```
        ...
    }
    E item2 = new E();    //Compiler error
    E[] iArray = new E[10]; //Compiler error
    E obj = (E)new Object(); //Unchecked cast warning
  }
}
```

Example 29: Bold text illustrates impossible operations

Conclusion

Java provides a lot of features with respect to inheritance. Inherited members can be used as is, replaced, hided or supplemented with new members.

- The inherited fields can be used directly.
- A field can be hided by declaring a field in the subclass with the same name as in the superclass.
- New fields can be declared in the subclass that doesn't exist in the superclass.
- A new *instance* method can be written in the subclass that has the same signature as the one in the superclass, thus *overriding* it. If a class has been made final, it is not possible to make sub classes of the class anymore.
- A new *static* method in the subclass can be written in case it has the same signature as the one in the superclass, thus *hiding* it.
- New methods can be declared in the subclass if they are not declared in the superclass.

Although it is in Java not possible to inherit from multiple classes, it is possible to implement multiple interfaces. Methods of an interface are implicitly abstract.

Use of generics will lead to software which is more stable, because more bugs can be detected at compile time instead of runtime. Examples of features are:

- generic types, to prevent cast exception at runtime;
- generic methods and constructors;
- bounded type parameters;
- subtyping;
- use of Wildcards
- type erasure.

4.5 UNIFI described in a programming language

Size can be seen as a parameter of complexity. For example, it could be less complex if a message of 10 pages with XSD syntax could also be expressed with Java syntax in 2 pages.

It is possible to describe UNIFI message in other languages than XSD. For this research an XML Schema Definition Tool (Xsd.exe) has been used. This command line tool generates XML schema or Common Language Runtime (CLR) classes from XDR[25], XML, and XSD files, or from classes in a runtime assembly. The classes can be of the language C#, Visual Basic or Visual J#. The latter uses the Java-language syntax and has been chosen. The input was the UNIFI message 'pain.008.001.01.xsd' (17 pages) and the output 'pain_008_001_01.jsl' (66 pages). The output file contains a lot of unnecessary comments and white space and therefore a parser has been made for this research project that could compress it to 34 pages. It juts out that it is still twice the size of the original UNIFI message.

File		Size (pages)
pain.008.001.01.xsd		*17*
# ComplexTypes	94	
# SimpleTypes	94	
pain_008_001_01.jsl		*66*
# classes	47	

[25] XDR means XML-Data-Reduced schema file. XDR is an early XML-based schema format.

# enum	21	
pain_008_001_01_Parsed.jsl	34	

Table 27: XSD vs J# Statistics

In the example below complexType ' pain.008.001.01' of the XSD, is described as a class in the jsl. Also the types (e.g. 'GroupHeader1', 'PaymentInstructionInformation2') are described as classes in the jsl. For both examples are two methods made: one for setting a value and one for getting a value of a certain type.

Fragment of pain.008.001.01.xsd

```
  <xs:complexType name="pain.008.001.01">
    <xs:sequence>
      <xs:element name="GrpHdr" type="GroupHeader1" />
      <xs:element name="PmtInf" type="PaymentInstructionInformation2"
minOccurs="1" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
```

Fragment of pain_008_001_01.jsl

```
public class pain00800101
{
    private GroupHeader1 grpHdrField;
    private PaymentInstructionInformation2[] pmtInfField;

    public GroupHeader1 get_GrpHdr()
    {
        return this.grpHdrField;
    }
    public void set_GrpHdr(GroupHeader1 value) { this.grpHdrField =
value; }

    /** @attribute System.Xml.Serialization.XmlElementAttribute
("PmtInf")*/
    public PaymentInstructionInformation2[] get_PmtInf()
    {
        return this.pmtInfField;
    }
    public void set_PmtInf(PaymentInstructionInformation2[] value) {
this.pmtInfField = value; }
}
```

Example 30: XSD vs J#

Although the example below uses also a complexType, there are some differences compared with the example above. Firstly the example below uses choice tags, which is not the case by the example above. Secondly the example below has got a 'get' and 'set' method of the type 'Object' instead of a reference to the enumeration 'CashAccountType4Code' and the string variant 'Max35Text'. Thirdly the xsd.exe tool added comment, which could be used to guide the Xml Serializer classes in properly reading and writing the XML. They are basically hints to the parser [r4].
Besides the complexType the example below contains also a simpleType. Enumeration in both languages is described in a similar way.

Fragment of pain.008.001.01.xsd

```
  <xs:complexType name="CashAccountType2">
    <xs:sequence>
      <xs:choice>
        <xs:element name="Cd" type="CashAccountType4Code" />
        <xs:element name="Prtry" type="Max35Text" />
      </xs:choice>
    </xs:sequence>
  </xs:complexType>

<xs:simpleType name="CashAccountType4Code">
```

```
    <xs:restriction base="xs:string">
      <xs:enumeration value="CASH" />
      <xs:enumeration value="CHAR" />
      <xs:enumeration value="COMM" />
      <xs:enumeration value="TAXE" />
      <xs:enumeration value="CISH" />
      <xs:enumeration value="TRAS" />
      <xs:enumeration value="SACC" />
      <xs:enumeration value="CACC" />
      <xs:enumeration value="SVGS" />
      <xs:enumeration value="ONDP" />
      <xs:enumeration value="MGLD" />
      <xs:enumeration value="NREX" />
      <xs:enumeration value="MOMA" />
      <xs:enumeration value="LOAN" />
      <xs:enumeration value="SLRY" />
      <xs:enumeration value="ODFT" />
    </xs:restriction>
  </xs:simpleType>
```

Fragment of pain_008_001_01.jsl

```
public class CashAccountType2
{
    private Object itemField;
    /** @attribute System.Xml.Serialization.XmlElementAttribute("Cd",
CashAccountType4Code.class)*/
    /** @attribute System.Xml.Serialization.XmlElementAttribute("Prtry",
String.class)*/
    public Object get_Item()           { return this.itemField;  }
    public void set_Item(Object value) { this.itemField = value; }
}

public enum CashAccountType4Code {  CASH,  CHAR,  COMM,  TAXE,  CISH,
TRAS,  SACC,  CACC,  SVGS,  ONDP,  MGLD,  NREX,  MOMA,  LOAN,  SLRY,  ODFT
}
```

Example 31: XSD vs J#

4.6 Summary

This chapter described the theory and technology of standardization, languages and notations of these languages. Furthermore it described inheritance and a UNIFI message in a programming language. Standards are in the context of SEPA rules that govern technology, behavior and interactions. Technical standards are necessary to allow interaction and interoperability between IT systems and to foster automation of the payment process. Interoperability is the ability to exchange and use information. XSD improves interoperability between systems of financial institutions, because XSD is platform and program language independent.

A language can be defined as a set of character strings, where a string is a sequence of symbols of a finite-length. Languages can be regular, context free, recursive, recursive enumerable and not recursive enumerable. A context-free grammar (CFG) is a recursive notation for a context-free language (CFL). CFG's have been used for parsers since the 1960's, but more recently, they have been used to describe formats of XML documents via DTD and XSD. The BNF (Backus Naur Form) notation is a formal notation to describe the syntax of a given (context free) language.

The chapter continued with a description of the possibilities around inheritance. Both Java and XSD provide a lot of features around inheritance. Although not all inheritance related features of XSD have been used in the XSDs of UNIFI, the most important ones like base, restriction and extension have been applied on the XSDs of UNIFI. The UNIFI messages are generic enough to apply more inheritance. Although this will reduce the length of the XSDs, because message items can be used multiple times, ISO has decided to create for every message a different schema.

XSD messages can also be described with Java syntax, but this will not reduce the size of the messages. An example showed that an XSD of 17 pages contains 66 pages in J#. Because the output file contains a lot of unnecessary comments and white space, a parser has been made that could compress it to 34 pages. But this is still twice the size of the original UNIFI message.

5 Available tools

The purpose of this part of the thesis is to answer the following research question: "What are the current tools to develop and view (UNIFI) XSDs" and their sub questions:

- What are the extra features of the concerned tool compared with the features of XSD?
- What are the constraints and problems of the tool? With which functionality could the concerned tools be extended?

The indicated missing functionalities are a reason to build an own tool: the SEPA UNIFI Test Tool.

This paragraph is divided into three different categories of tools: firstly the SEPA UNIFI tools, secondly e-invoicing tools and the third category exists of common XML tools.

5.1 SEPA UNIFI tools

The following tools of this category are known during the research.

UNIFI Message Processor - Java Class - V1.0
The UNIFI Message Processor is a Java library to build a UNIFI payment or security handling system. It provides functionality to validate UNIFI messages and parse them into Java objects. A trial can be downloaded from [r7]. One site wide license (allows unlimited developers at a single physical address) with 60 days support for the Payments (UNIFIprocessorPMT) V1.0 part will cost € 4,560.00 and for the Securities (UNIFIprocessorSEC) V1.0 part it will cost € 7,695.00

IBM WebSphere TX
[r41] IBM has made the WTX as a solution for SEPA. The tool is built for the validation and transformation of complex, semi-structured, interdependent, and disparate data without the need to program. The company expects to deliver this industry pack by the end of 2nd quarter 2007.

SAP Payment Engine
[r27] This engine supports standards for international payment transactions such as IBAN and BIC as well as enhancements specific to the bank or country. The tool enables banks to choose to process payment transactions by it self, outsource them or a combination of these. It runs on the so called 'NetWeaver' technology platform, but other platform technologies can also be used. SAP builds this tool together with Accenture and a charter client for tailoring the tool to the SEPA requirements.

Equens payment service
[r15] In stead of build or buy your own software to create SEPA services, it is also possible to use existing services. Equens provides services for payment and card processing with SEPA compliant services. For several kinds of transactions they can arrange validation, clearing, settlement preparation, booking information and notifications for direct debit transactions.

5.2 E-invoicing tools

Several tools for e-invoicing are already on the market. Examples of companies who deliver this are Parabill [r25], Esker [r17], Xign, [r40] and Mindstone [r24].

5.3 XML tools

Table 28 lists and classifies a number of XML tools that have been described in appendix 10 (which is mainly based on source [r36].

Tool	API	Plug in	Web based	Standalone
Content Checker			yes	
Quality of Design (QOD) Tool			yes	
Castor	yes			
EDIFIX 5.5R				yes
ITCWorks	yes			
Java-X		yes		
Javax.xml of J2SE	yes			
JaxFront	yes			
JBind	yes			
KLEEN				yes
LINQ to XSD		yes		
NetBeans Schema support				yes
MSXML	yes			
Schema-Form				yes
Schema Viewer				yes
Syntext Serna				yes
SQC				yes
Stylus Studio				yes
Visual Studio 2005's XML Tools				yes
X2U			yes	
Xerces-J	yes			
XML Architect				yes
XML Beans	yes			
XMLFox				yes
XML Infoset Browser	yes			
XML Schema Object Model	yes			
XML Schema Validator			yes	
XML Spy		yes		yes
Xsbrowser			yes	
XSV			yes	

Table 28: Overview of XML validation tools

5.4 Conclusion

The paragraph above summarizes a list of available tools that work with (UNIFI) XSD.
Although there are SEPA UNIFI tools on the market, it is hard to give a complete list of extra features they have, compared with these of XSD and the constraints they have. The reason for this is that these tools are not available for free plus they must be tailored and integrated at e.g. an infrastructure of a customer.
If the scope of the question: "What are the current tools to develop and view (UNIFI) XSDs?" would be extended to XSD only, than the list of tools becomes a lot longer (see also [r20]). These tools provide in general functionality to validate, manipulate and display XML and XSD files.
The questions: "what are the constraints and problems of the tool? With which functionality could the concerned tool be extended?" could be answered with: "functionality for validation of IBAN and BIC". An example of a feature of XSD is the possibility to define the format of e.g. IBAN. Although the format can be right, i.e. conforms to a regular expression, the number still could be invalid, because it does not match to the country specific rules or it does not exist. One rule is that the first two symbols must represent the country where the account holder has his/her account (e.g. NL). An example of an extra feature of the concerned tool is that it also checks on such a rule during a validation.

6 SEPA UNIFI Test Tool

This chapter describes the requirements design, implementation, validation and verification of the SEPA UNIFI Test Tool (SUTT).

Tools for validation of XML and other functionality can be standalone, web based, provided as plug in or available as API's (Application Programming Interface).
To build SEPA systems that use UNIFI standards for the banking industry, it is important to do as much as possible without human intervention. For this reason only the latter (API) has been used for the development of the SUTT. Because SEPA systems like for example SAP Payment Engine, IBM WebSphere TX and the system for the Equens payment service are huge projects far beyond the available time of this final research project, only a part of the technology of a SEPA system will be applied. Although some functionality of the SUTT is already on the market, there are still good reasons to build the tool.
One of them is to get familiar with these technologies instead of buying it. The tool can be used to demonstrate functionality that uses these technologies. Another purpose of the tool is that it can be used as a central platform for testers. These testers could for example generate large XML files that are based on an XSD, to test systems that use UNIFI.
To make it easy accessible for users, the SUTT has got a web interface.

6.1 Requirements

Table 29 list the functional requirements, priority and status of SUTT (see also Appendix 2). Priorities are made because the list of requirements could be extended, but the project time could not. The method MoSCoW has been used to do this. It originated as part of the Dynamic Systems Development Method (DSDM). MoSCoW stands for:

- must haves, these are fundamental to the projects success;
- (on time);
- should haves, these are important but the projects success does not rely on it;
- could haves, these can easily be left out without impacting on the project;
- (on budget);
- won't have this time round, these requirements can be left out this time and done at a later date.

FR	Description	Pri*	Done
FR1a	The Test Tool should be able to **validate** XML against XSD.	M	100%
FR1b	Users should be able to upload XML files of any arbitrary size.	M	100%
FR1c	Users should be able to **change** the XML file to let a test fail or succeed.	M	100%
FR1d	The SEPA UNIFI Test tool should be smarter than XSD "out of the box", like for example XX123456789 is valid bank account according to the regular expression [A-Z]{2}[0-9]{1,32}, but an existing account as XX is not a country code and 123456789 not likely to be given out by a bank.	M	100%
FR1e	The system should check the International Bank Account Number (IBAN)	M	100%
FR1f	The system should check the Bank Identifier Code (BIC)	M	100%
FR1g	The corresponding XSD should automatically load after the user adds an XML file. Beside this, the user should be able to choose an XSD of a dropdown list.	M	100%
FR1h	It should be possible to upload multiple files.	C	
FR2a	The Test Tool should be able to **generate XML** based on an XSD of UNIFI. This means that an XML file will be produced according to the XSD UNIFI format.	M	95%

FR2b	The content will exist of dummy data. The tool should be prepared of content that should exist (of random) real data from a database.	C	95%
FR3a	The Test Tool should be able to **transform** CSV and XLS to an XML format.	C	
FR3b	The Test Tool should be able to transform CLIEOP to XML.	W	
FR3c	It should be possible for the user to complete the missing data.	W	
FR4a	The Test Tool should be able to **generate user entry form** based on XSD for user friendly entry.	S	25%
FR4b	After a user filled in the form, a view of the generated XML will be given by the system.	S	
FR5a	Users should be able to create a new log in account, log in and change their password.	S	
FR5b	Users should be able to upload, save and manage XML files in a directory structure.	C	
FR5c	A description of XML and XSD files should be attached.	C	

Table 29: Requirements; * Priority: M = Must have; S = Should have; C = Could have; W = Won't have this time

All 'must haves' are implemented except FR2a. The code is present for FR4a that transforms XML to HTML, but the XSLT(s) that is able to generate HTML for the UNIFI standards has still to be made or a more generic implementation that works for every XSD. The data for FR2b does not come from a database, but will also be generated. Unfortunately the generator has some limitations and therefore does not generate valid XML.

6.2 Design

Several designs are made to model the requirements that are described in the requirement document. These are:

- a model for input and output of the tool;
- use cases and activity diagrams;
- a high level design with a use case diagram and a global architecture;
- a detailed design with class diagrams.

The first can be found in appendix 2, the second in appendix 3 and the latter two in appendix 4.

The tool contains an interface for end users and computers. The former is described as the presentation layer and the second as web services (see figure 26). Both communicate with the logic layer, which is divided in the components: validate, generate XML, generate form, transform, manage files and common logic.

A web service is a programmatic interface/software component for application to application communication over the Internet. Interfaces of web services are described in WSDL (Web Service Description Language). They can be found via e.g. UDDI (Universal Description, Discovery and Integration), specifications in WSIL (Web Services Inspection Language) and DISCO files. Web services can be accessed by other applications via standard network protocols like SOAP (Simple Object Access Protocol). Advantages of web services are:

- the possibility of interoperability between software applications of different platforms and programming languages;
- their extensibility;
- their machine-processable descriptions in XML;
- the possibility to reuse existing logic.

The SUTT web services could for example be used by third parties in order that they can make their own presentation layer on top of it or as part of a system that processes XSD messages. Furthermore is could be used as an intermediate layer between the logic layer and the presentation layer. However, that's not the case, because the layer is not necessary if the tool will be installed on one machine and omitting it will improve performance.
At last, the tool contains two databases. The BIC database contains Bank Identifier Codes, which are needed for the validation process.

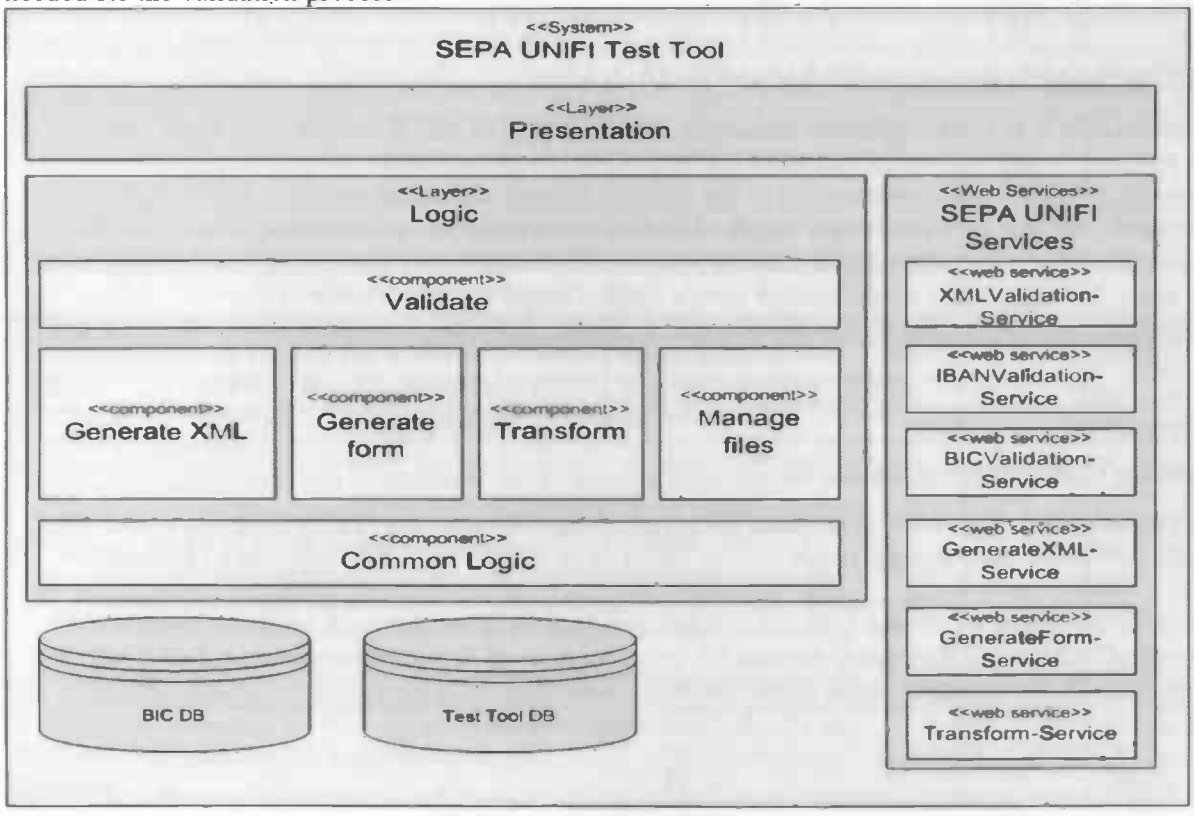


Figure 26: Global architecture

6.3 Implementation

Appendix 4 shows the relations between classes. In appendix 3 can be read what the basic and alternative scenarios of the SUTT components are. This paragraph describes how these scenarios are implemented. Although, validate, generate XML and generate form are implemented, transform and manage files are not. SUTT has been developed in C#.NET with aid of Visual Studio 2005. The solution of SUTT exists of four projects:

- a web project for the presentation layer;
- a library project for the logic layer;
- a web service project to offer SEPA UNIFI webservices;
- a test project that contains JUnit tests for the logic layer.

6.3.1 Validate

Basically, a user can choose an XML and XSD file to generate a validation report. Validator.cs is responsible for checking the syntax of the XML as well as the XSD. The class uses the System.Xml library.

IBANValidator is responsible for filtering IBANs of the concerning XML document. Subsequently, a check will be done, to be sure that the input contains only alphanumeric symbols. After that the numbers will be checked according rules of [r58]. These rules are divided in three steps. The first step is to move the first four characters of the IBAN to the right of the number, e.g. BE62510007547061

becomes 5100075470 61BE62. The second step is to convert letters into numerics according a conversion table (see table below).

A = 10	G = 16	M = 22	S = 28	Y = 34
B = 11	H = 17	N = 23	T = 29	Z = 35
C = 12	I = 18	O = 24	U = 30	
D = 13	J = 19	P = 25	V = 31	
E = 14	K = 20	Q = 26	W = 32	
F = 15	L = 21	R = 27	X = 33	

Table 30: Alpha to numeric conversion table for IBAN check

The third step is to evaluate that the remainder of the number of step 2 modulo 97 is equal to 1. For the previous example it would be 510007547061111462 modulo 97 = 1.
According to [r58] it is recommended to use integers instead of floating points, because of precision reasons. IBAN numbers can have a length of 34 characters and even more after application of the conversion table. The validation calculation has been implemented by splitting up the number, because the types that represent a whole number have a limited range in C# (see table 31).

Type	Min value	max value	Max value length
Int (Int32)	-2,147,483,648	2,147,483,647	10
Long (int64)	–9,223,372,036,854,775,808	9,223,372,036,854,775,807	19
Ulong (int64)	0	18,446,744,073,709,551,615	20

Table 31: Limited range of types in C#

An example of an IBAN that is too long for a type of the table above is "FR14 2004 1010 0505 0001 3M02 606", after step one and two it will be "2004 1010 0505 0001 3220 2606 1527 14" (length = 30).
The example below represents how this number will be split up in parts of 9 symbols, including the answer of 'strNumericIBANpart' modulo 97. An advantage of this implementation is that it will also work if the IBAN standard would allow IBANs of more than 34 symbols.

```
* strNumericIBAN       = 200410100505000132202606152714
* strNumericIBANpart   = 200410100
* answer                      = 49
* strToDivide                 = 49505000132202606152714
* strNumericIBANpart          = 495050001
* answer                             = 25
* strToDivide                       = 2532202606152714
* strNumericIBANpart                = 253220260
* answer                                   = 14
* strNumericIBANpart                      = 146152714
* answer                                          = 1
* conclusion: IBAN is valid according digit check
```

Example 32: Modulo check by splitting up the digit

The BICValidator collects, similar to the IBANValidator, BICs of the concerning XML document. Unfortunately there is no algorithm to validate BICs. This means that these codes must be looked up in a database. SWIFT has the authority to assign, register and publish BICs. BICs are stored in a 'BIC Directory'. There are several kinds of BIC products (see table 32).

Product	Description	Format	Update frequency
For FIN messaging (integration)			
BIC Directory	Contains BIC's BEIs (Business Entity Identifier to identify non-financial institutions in financial messages) and physical address details.	txt, dat, ebcdic[26], dos	Monthly
BIC Database Plus	Combines information from the BIC directory with the national clearing codes of different countries. It contains references to more than 270,000 financial institutions.	txt, ebcdic	Monthly
Bank File	For SWIFT Alliance only; packaged in ZIP format or tar.Z format.	txt, dat	Monthly
For FIN messaging (Consultation)			
BIC Online	Search options of this web tool are: BIC code, institution keyword, city, country and branch name (free of charge). A search for 'The Netherlands' resulted in 183 records.		Monthly
BIC Online Professional	Additional search options and related data are: institution address, market infrastructure participation, SWIFT user category. Furthermore it has the ability to trace the change history of a BIC from a given date.		Daily
BIC Enquiry Tool	It contains a local version of the BIC Directory (full and delta), BIC Database Plus (full and delta), and files with currency and country information. The costs of annual subscription fee are 200 – 8,500 USD.		Ad-hoc
BIC Directory paper	Also known as International Bank Identifier Code directory. Contains a geographical listing of 1) BICs and BEIs per country and city; 2) a list of restricted participants to market infrastructures; 3) a list of alphabetic BICs and BEIs and 4) operational information like payment instruction cut off time. The unit price is 90 – 190 USD. Every month a free complement appears with changes (addition, deletion, modification).	pdf	Quarterly
For SWIFTNet messaging (integration and consultation)			
SWIFTNet Services Directory	Contains a list of registered financial institutions, which is only available for registered users.		

Table 32: BIC products

The BIC Directories can be 'full' or 'delta'. The latter means an update for a certain period. The on-line ordering service for these downloads may only be used by registered users or on behalf of a business customer, but not as an individual customer.
The BIC database of SUTT contains BICs of the Directory paper for demonstration purpose. For integration the BIC Directory or BIC Database is recommend.

The tool is also smarter than XSD "out of the box" like for example XX123456789 is a valid bank account according to the regular expression [A-Z]{2}[0-9]{9}, but not likely to be an existing account as XX is not a country code. After the tool has collected the IBANs as described above, the tool will check if the first two characters from each IBAN do exist in the SUTT database. The country and in some cases multiple countries that match, will be displayed in the validation report. If the first two characters don't match, than the user will be notified about this.

Figure 27 is an example of a validation report that could be generate after a user pushes the validate button.

[26] Extended Binary Coded Decimal Interchange Code

Validation report

Standard validation:

Standard validation completed. The XML is wel-formed and valid.

IBAN validation:	
IBAN	*Status*
BE30001216371411	IBAN is valid. Found country is /countries are: BE Belgium
ZE30001216371455	IBAN is not valid.
FR1420041010050500013M02606	IBAN is valid. Found country is /countries are: FR France or French Guiana FR Guadeloupe or Martinique FR Runion or French Polynesia FR Guernsey or Jersey FR Mayotte FR New Caledonia FR New Caledonia FR Saint Pierre and Miquelon FR French Southern Territories FR Wallis and Futuna

BIC validation:	
BIC	*Is in DB*
AEIBPKK1ISL	True
AEIBPKK1ISL	True
ANTUFRP1	True
BBBBUS66	False
ALLAINBBCHD	True
KOBBMEP2	True
DDDDBEBB	False
BBBBUS66	False

Figure 27: Example of a validation report

6.3.2 Generate XML

The content of XML files can be divided in tags that describe data and the data itself, i.e. the values. The tags of an XML file can thus be generated, because XSD describes which tags are allowed in an XML file. Valid values of an XML file can also be generated, because XSD describes the format, e.g. the type and length.

[r3] For this requirement a component of the shelf (COTS) has been used, which is packaged in 'Microsoft.Xml.XMLGen'. The input is an XSD document and the output is an XML document. The property 'MaxThreshold' of the class 'XmlSampleGenerator' overrides the value of the 'maxOccur' attribute in the schema to the default value '5', if maxOccurs="unbounded" or in case of a 'high number'. Example 33 and example 34 illustrates this. Note that the numbering starts with 21, because '<RmtInf>' appeared earlier in the document for four times.

```
<xs:element name="Ustrd" type="Max140Text" minOccurs="0"
maxOccurs="unbounded"/>
```

Example 33: Snippet of pain.001.001.02.xsd with an unbounded attribute

```
.....
<RmtInf>
    <Ustrd>Ustrd21</Ustrd>
    <Ustrd>Ustrd22</Ustrd>
    <Ustrd>Ustrd23</Ustrd>
    <Ustrd>Ustrd24</Ustrd>
    <Ustrd>Ustrd25</Ustrd>
.....
```

Example 34: Snippet of the generated XML

The property 'ListLength' of the class 'XmlSampleGenerator' defines the number of items that should be generated for a simple type of variety list. The default value is three.

All facets are supported, except xs: pattern. Furthermore, all datatypes [r35] are supported, except xs:entity, xs;entities and xs:notation.
Rules to generate values are:

- *The values of type xs:string are generated by adding a numeric counter to the element. The length of such a generated string is modified based on the presence of Length, minLength, and maxLength facets.*
- *Values of specific string-derived types like xs:NCName, xs:token, and others, have the type name appended with a numeric counter to differentiate them from xs:string.*
- *Numeric values are generated by selecting values between the maximum and minimum values for the corresponding CLR[27] type. Every type has an associated default start value (For example, the start value is 1 for xs:integer and 1900-01-01 for xs:date) that is modified based on the presence of certain facets like minInclusive, minExclusive and so on.*
- *If the element or attribute has a default values, like for example in case of an enumeration, than the first value of these default values will be used in the generated file.*
- *If the element or attribute has a fixed value, than only this value will be used in the generated file.*

Limitations of the 'Microsoft.Xml.XMLGen' are:

- *that the W3C Identity constrains: xs:key, xs:keyref, xs:unique, are not supported;*
- *if xs:pattern facets exist on simple types, values generated may not conform to the pattern. Enumerations of the xs:QName type may not work as expected since this requires the prefixes in the schema to be preserved;*
- *xs:ENTITY, xs:ENTITIES, and xs:NOTATION types are not supported;*
- *xs:base64Binary content is generated only if enumerations exist in the schema for that type.*

6.3.3 Generate form

Several implementation options are found and considered for this requirement. One of these options was using 'CodeDOM' namespace of the .NET framework, which contains classes to generate code in any .NET language. A disadvantage of this option is that it does not support in interpreting XML to define code for code generation. Another option is the use of XSLT technology [r9].
XSLT stands for XSL Transformations. XSL stands for Extensible Stylesheet Language Family, which is a family of recommendations for defining XML document transformation and presentation. It consists of three parts:

- XSL Transformations (XSLT), a language for transforming XML;
- the XML Path Language (XPath), an expression language used by XSLT to access or refer to parts of an XML document;
- XSL Formatting Objects (XSL-FO) an XML vocabulary for specifying formatting semantics.

Figure 28 illustrates how the XSLT has been applied. Because XSD itself is described in XML, it can be transformed with aid of XSLT and an XSLT processor into formats like XML, HTML, PDF, Wireless Markup Language and in case of SUTT: C#.Net Components.

[27] CLR stands for Common Language Runtime, which is the core runtime engine in the .NET Framework

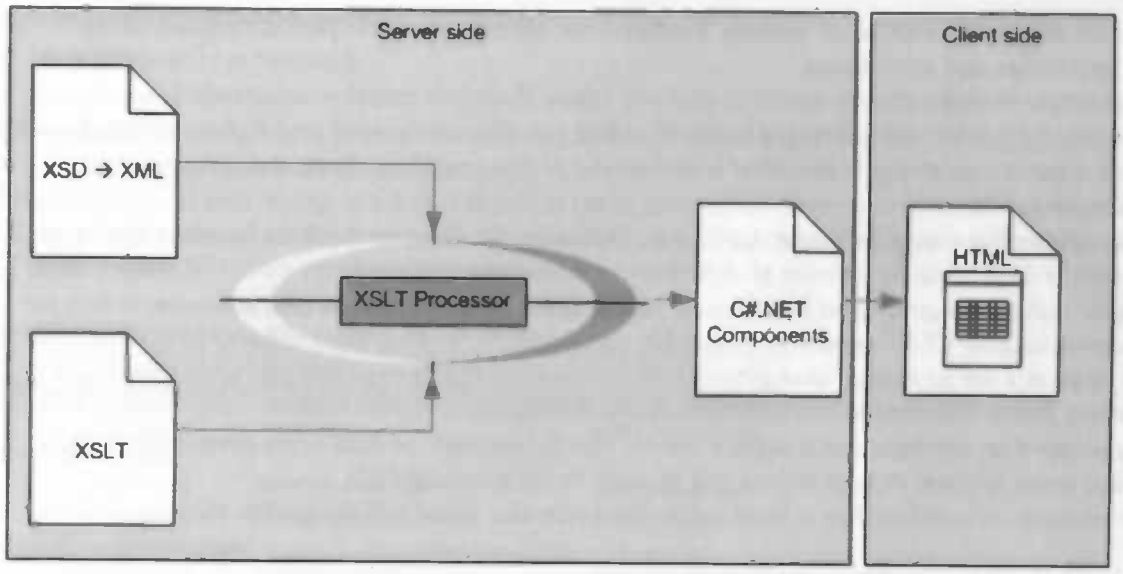


Figure 28: Application of XSLT in SUTT

Example 35 illustrates a part of an XSD that serves together with the XSLT of example 36, as input for the XSLT processor.

```
  <simpleType type='AccountIdentification3Choice'
name='AccountIdentificationChoice'>
    <option value ='IBAN' selected='yes'>IBAN</option>
    <option value ='BBAN' selected='no'>BBAN</option>
    <option value ='UPIC' selected='no'>UPIC</option>
  </simpleType >
```

Example 35: part of XSD input

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:asp="remoye">
  <xsl:output method="xml" indent="yes" encoding="utf-8" omit-xml-
declaration="yes"/>
  <xsl:template match="/">
  <table border ="0">

  <xsl:for-each select="//simpleType">
    <xsl:if test="@type='AccountIdentification3Choice'"><br />
      <asp:Label ID="Label{@name}" runat="server"
Text="{@name}"></asp:Label>
      <asp:TextBox ID="TextBox{@name}" runat="server"></asp:TextBox>
    </xsl:if>

    <xsl:if test="@type='AccountIdentification3Choice'">
      <asp:ListBox ID="{@name}" SelectionMode='multiple' runat="server">
        <xsl:for-each select="option">
          <asp:ListItem Value="{@value}" Selected="False"></asp:ListItem>
        </xsl:for-each>
      </asp:ListBox>
    </xsl:if>

    <xsl:if test="@name='BICIdentifier'"><br />
      <asp:Label ID="Label{@name}" runat="server"
Text="{@name}"></asp:Label>
      <asp:TextBox ID="TextBox{@name}" runat="server"></asp:TextBox>
    </xsl:if>
  </xsl:for-each>

  </table>
```

```
  <br /><asp:button id="submit" runat="server" Text="Submit" />
  </xsl:template>
</xsl:stylesheet>
```

Example 36: XSLT

Example 37 shows the output, which will be translated in HTML by the .NET framework. Figure 29 represents the HTML form that a user will see, after the 'generate form' button has been pushed, in case of the input just mentioned. For demonstrations purpose a textbox as well as a list box has been made for the 'AccountIdentificationChoice' element.

```
<table border="0" >
  <br />
  <asp:Label ID="LabelBICIdentifier" runat="server" Text="BICIdentifier" />
  <asp:TextBox ID="TextBoxBICIdentifier" runat="server" />
  <br />
  <asp:Label ID="LabelAccountIdentificationChoice" runat="server"
Text="AccountIdentificationChoice" />
  <asp:TextBox ID="TextBoxAccountIdentificationChoice" runat="server" />
  <asp:ListBox ID="AccountIdentificationChoice" SelectionMode="multiple"
runat="server">
    <asp:ListItem Value="IBAN" Selected="False" />
    <asp:ListItem Value="BBAN" Selected="False" />
    <asp:ListItem Value="UPIC" Selected="False" />
  </asp:ListBox>
</table>
<br  />
<asp:button id="submit" runat="server" Text="Submit"  />
```

Example 37: C#.NET components

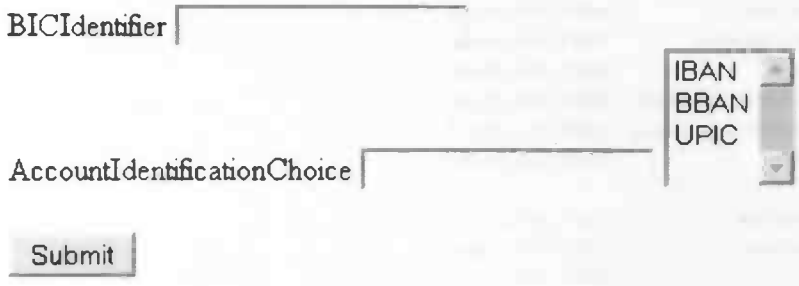


Figure 29: HTML form of the corresponding example above

Although the SUTT is able to transform XSD to a form, it is not able to do this for all UNIFI XSDs or every XSD in general. A recommended source for further development is [r81, which lead to an article that describes the transformation of XSD to XForm.

6.4 Validation and Verification

Validation means, in the context of software engineering, the process of checking if the specification corresponds to the agreed wishes of the user. Verification is the process of testing if the software does what the specification states. Testing helps to identify the correctness, completeness, the security and quality of developed computer software. There are four levels of testing. These are unit testing, integration testing, system testing and acceptance testing. Black box testing can be applied to these four levels and white box testing to the first three. Black box testing takes an external point of view of the test object to derive test cases, which can be functional or non-functional. White box testing, also known as clear box testing, glass box testing or structural testing, takes an internal point of view of the system to design test cases.

Unit tests have been made for the SUTT code. The unit tests compare expected data with the actual data of class methods.

Advantages of unit testing are that:

- it is commonly automated;
- it provides information whether code still works properly if changes are made;
- it simplifies integration, because tested parts makes it easier to test the sum of these parts;
- it provides up to date documentation.

Disadvantages of unit testing are that it does not catch integration errors, performance problems or other system-wide issues. Furthermore unit tests show only the presence of errors and not the absence of error. Figure 30 shows results of these tests.

Figure 30: Unit test results

Another classification of testing is scripted versus exploratory. Scripted testing means that learning and test design take place before test execution, and often the learning has to be done again during test execution. Exploratory testing means simultaneous test design and test execution with an emphasis on learning. During this test method a tester walks through the product, find out what it is and test it. Both scripted and exploratory testing has been applied on SUTT.

6.5 Summary

This chapter described the requirements design, implementation, validation and verification of the SEPA UNIFI Test Tool (SUTT).

The tool has been build to get familiar with several technologies around XSD. The tool can be used to demonstrate functionality that uses these technologies. Another purpose of the tool is that it can be used as a central platform for testers. Several designs are made during the design phase. These are: 1)

a model for input and output of the tool, 2) use cases and activity diagrams, 3) a high level design with 3a) use a case diagram and 3b) a global architecture and 4) a detailed design with class diagrams. SUTT has been developed in C#.NET with aid of Visual Studio 2005. The solution of SUTT exists of projects for the 1) presentation layer, 2) a library project for the logic layer, 3) a web service project to offer SEPA UNIFI web services and 4) a test project that contains JUnit tests for the logic layer. The unit tests compare expected data with the actual data of class methods. Furthermore scripted and exploratory testing has been applied on SUTT.

7 Conclusion

This chapter contains a summary of answers on the defined research questions, a conclusion, a short reflection and suggestions for future work.

7.1 Research answers

The main research question was: "is XSD a suitable solution for standardization of financial messages?" To answer this question sub questions were defined. This section summarizes the answers for each question.

1) What is standardization and which aspects are related to standardization?
→ A standard is the product of any community after a process of development and agreement. Examples of several aspects of standardization are the chosen language and thus the syntax, semantics and the compatibility / interoperability.

2) What are the current tools to develop and view (UNIFI) XSDs?
→ These have been described in the chapter about available tools.

a) What are the extra features of the concerned tool compared with the features of (UNIFI) XSD?
→ Although there are SEPA UNIFI tools on the market, it is hard to give a complete list of extra features they have, compared with these of XSD and the constraints they have. The reason for this is that these tools are not available for free plus they must be tailored and integrated at e.g. an infrastructure of a customer.
If the scope of the question: "What are the current tools to develop and view (UNIFI) XSDs?" would be extended to XSD tools only, than the list of tools becomes a lot longer. These tools provide in general functionality to validate, manipulate and display XML and XSD files.

b) What are the constraints and problems of the tool? With which functionality could the concerned tool be extended?
→ These are for example validation of IBAN and BIC.

c) How does a SEPA UNIFI Test tool look like?
→ This has been described in the SUTT chapter and appendixes.

3) What are the possibilities and constraints of (UNIFI) XSD?
a) What kinds of UNIFI messages do exist?
→ These are:
- messages for the funds industry (validated and approved by Security SEG);
- messages for foreign exchange transactions (evaluated by the FX SEG)
- payment messages (evaluated by the Payment SEG);

b) How are (UNIFI) XSD messages implemented?
→ Despite there is a lot documented about these messages, there is no space and time to mention them all in full detail. However, to give a clear picture of these messages, one of the payment messages has been described in more detail. [r78] The 'Financial Institution To Financial Institution (FIToFI) Customer Credit Transfer' is a schema that can be used for inter-bank movement of money from a party bank account (the Debtor) to beneficiary party (the creditor) (see also appendix 6, 7 and 8).

c) Are organizations who implement UNIFI still able to add new functionality/wishes, without being incompatible with UNIFI? Is it for example possible to add extra attributes to the standard as defined in the ISO 20022?
→ Yes.

d) Does the ISO 20022 define schemas for e-invoicing?
   i. If not, how could an E-invoice schema look like?

→ Innopay explained that ISO 20022 is not a standard for payment transactions only, but also for related processes like: invoicing, foreign exchange and Documentary Trade. Schemas for e-invoicing (defined according the ISO 20022 standard) are not published yet. UBL-Invoice-2.0.xsd of OASIS is an example of how an e-invoice scheme could look like.

   ii. What are the benefits of e-invoicing?

→ The main benefits are increased efficiency, cost reduction and faster customer payments.

   iii. What are the disadvantages of e-invoicing?

→ Temporary disadvantage associated with e-invoicing and e-archiving are the lack of readiness of internal, customer and supplier systems, cost and complexity.
E-invoicing can also lead to problems during inspections of tax authorities, which may result in a denial of VAT/GST deduction or administrative penalties.

e) What are the possibilities around inheritance in XML/XSD?

→ An XSD can be extended with other XSDs and data types can be derived from existing types. The import mechanism enables global named components from different target namespaces to be used together. Furthermore it is possible to redefine types and groups. The attribute 'abstract' forces substitution of an element or type and the use of substitution groups. Derivation can be controlled for complex types by 1) restriction, 2) extension or 3) both. Simple types can be divided into atomic types, list types and union types.

f) How much of the inheritance related features of XSD are applied on the XSDs of UNIFI?

→ Although not all inheritance related features of XSD have been used, the most important ones like base, restriction and extension have been applied on the XSDs of UNIFI.

g) Are the schemas of UNIFI generic enough to apply (more) inheritance?

→ A statement has been made that states that the schemas of UNIFI will be generic enough to apply (more) inheritance if message items are multiple times defined in the UNIFI payment messages. A table has been made to judge about this. For this research can be expected that the statement above will be true. That the message items appear multiple times (avg # 4,89) is not surprising, because that is part of the concept of the UNIFI repository. Although this says in some degree about the balance between 'clear boundaries' and 'use of more inheritance and reference', the ISO has decided to create for every message a different schema. However, an alternative is the use of XSDs with a generic character. Use of these generic XSDs will reduce the length of the content of the XSDs that include these generic XSDs.

h) How is inheritance applied in Object Oriented languages like Java? Give a description of the generics and inheritance of this language.

→ Java provides a lot of features with respect to inheritance, which are mentioned earlier in this thesis. Use of generics will lead to software which is more stable, because more bugs can be detected at compile time instead of runtime. Examples of features are:

- generic types, to prevent cast exception at runtime;
- generic methods and constructors;
- bounded type parameters;
- subtyping;
- use of wildcards;
- type erasure.

i) How will a UNIFI message with Java syntax look like?

→ This has been described in paragraph 4.5 and an example has been placed in appendix 9.

7.2 Conclusion

After answering several sub questions can be concluded that the answer on the main research question: "is XSD a suitable solution for standardization of financial messages?" is 'yes'. Reasons are the advantages of XSD:

- The possibility of XSD to describe the type of data. Advantages of this feature are that it is easier to 1) describe allowable document content, 2) validate the correctness of data, 3) work with data from a database, 4) define data restrictions on data, 5) define data patterns (data formats) and 6) convert data between different data types.
- that XSD is also described in XML. A benefit of XML Schemas written in XML is that authors don't have to learn a new language. Also the same editor, parser, Document Object Model (DOM) technology (to manipulate) and XSLT technology (to transform schemas) can be used.
- The possibility to extend an XSD. This means that other schemas can be reused by referring to one or more of them. Beside this, it is possible to create data types derived from the standard types. The inheritance features of XSD are not completely applied at the UNIFI messages. For example generic XSDs could be made instead of copying message items in multiple schemas. However, ISO has decided to create for every message a different schema. Another example is that the same related messages could be defined once, instead of storing them in multiple files.
- Furthermore XSD improves interoperability between systems of financial institutions, because it is platform and program language independent.
- Another advantage is that XSD can be build from the UML models that describe in this case the business processes and transactions.

7.3 Reflection and future work

To find more examples of constraints of UNIFI, more people should be interviewed with a financial background and knowledge of UNIFI.
The SUTT tool could also be extended with the 'would have' and other missing requirement.

References

[r1]	<http://cui.unige.ch/db-research/Enseignement/analyseinfo/AboutBNF.html>
[r2]	<http://java.sun.com/docs/books/tutorial/java>
[r3]	<http://msdn2.microsoft.com/en-us/library/aa302296.aspx>
[r4]	<http://samples.gotdotnet.com/quickstart/howto/doc/xmlserialization/xsdtocls.aspx>
[r5]	<http://www.abe.org>
[r6]	<http://www.cls-group.com>
[r7]	<http://www.componentsource.com/products/unifi-message-processor/index-eur.html>; (March 2007)
[r8]	<http://www.consilium.europa.eu/cms3_fo/showPage.asp?id=250&mode=g&lang=en>
[r9]	<http://www.dnzone.com/ShowDetail.asp?NewsId=151>
[r10]	<http://www.eact-group.com>
[r11]	<http://www.ecb.int/paym/target/current/view/html/index.en.html>
[r12]	<http://www.ecb.int>
[r13]	<http://www.ecbs.org>
[r14]	<http://www.epcaconference.com/index.php/2007/2007/about_epca>
[r15]	<http://www.equens.com/eng/Services/giro_payments.asp>
[r16]	<http://www.ercim.org/publication/Ercim_News/enw51/solvberg.html>
[r17]	<http://www.esker.co.uk/deliveryware_solutions/businesse_invoicing.asp>
[r18]	<http://www.europeanpaymentscouncil.eu; February 2007>
[r19]	<http://www.garshol.priv.no/download/text/bnf.html>
[r20]	<http://www.garshol.priv.no/download/xmltools/search.html>
[r21]	<http://www.iso20022.org >
[r22]	<http://www.iso20022.org/index.cfm?item_id=59950> (UNIFI Payments messages)
[r23]	<http://www.libraries.psu.edu/tas/jca/ccda/tf-meta6.html>
[r24]	<http://www.mindstone.be/standard.asp?pageID=28&parMen=11>
[r25]	<http://www.parabill.net>
[r26]	<http://www.regular-expressions.info/reference.html>
[r27]	<http://www.sap.com/netherlands/industries/banking/sepa/index.epx>
[r28]	<http://www.swift.com/biconline/index.cfm?fuseaction=display_aboutbic>
[r29]	<http://www.swift.com>
[r30]	<http://www.twiststandards.org/twiststandards/tiki-index.php?page_ref_id=28>
[r31]	<http://www.unece.org/cefact>
[r32]	<http://www.w3.org/DSig/reports/Final1.html>

[r33]	<http://www.w3.org/TR/2001/WD-xmlschema-formal-20010320>
[r34]	<http://www.w3.org/TR/2004/REC-xmlschema-0-20041028/> or <http://www.w3.org/TR/xmlschema-0/>
[r35]	<http://www.w3.org/TR/xmlschema-2/>
[r36]	<http://www.w3.org/XML/Schema>
[r37]	<http://www.w3.org/XML>
[r38]	<http://www.w3schools.com/schema>
[r39]	<http://www.w3schools.com/xml>
[r40]	<http://www.xign.com/>
[r41]	<http://www-5.ibm.com/be/events/en/middleware_appdev.html#394>
[r42]	<www.rabobank.nl>
[r43]	Bank for International Settlement (BIS), Committee on Payment and Settlement Systems; Statistics on payment and settlement systems in selected countries; March 2007
[r44]	Banks for International Settlement; Core Principles For Systemically Important Payment Systems; January 2001
[r45]	Benoit Van Daele; UNIFI Standards Convergence and Coexistence; 4 April, 2006 <http://isotc.iso.org/livelink/livelink>
[r46]	BEUC; Tell me what I am paying: if we want a Single Market, we need better payment systems; September 2006; <http://www.beuc.eu>; 5 June, 2007
[r47]	Business Justification for the update of the UNIFI (ISO 20022) Financial Repository; 22 March 2006; page 1; <www.iso20022.org/index.cfm?item_id=59858>
[r48]	Capgemini, ABN Amro, EFMA (European Financial Management & Marketing Association; World payment report 2006; page 35
[r49]	Council of the European Union (C. Picqué); CouncilDirective2001_115_ECl; 17 January, 2002; art 2.2e
[r50]	Dr. J. Jonker en drs. B.J.W. Pennink; De kern van methodologie; 2000; page 32
[r51]	ECB, The Eurosystem, the central bank system of the euro area; From TARGET to TARGET2, Innovation and transformation; 2006
[r52]	ECB/Eurosystem; Correspondent central banking model (CCBM) - procedure for Eurosystem counterparties; December 2006; page 5
[r53]	ECB; EPM: the ECB payment mechanism <http://www.ecb.int/paym/target/current/epm/html/index.en.html>
[r54]	European Central Bank, Euro system; The Single Euro Payment Area (SEPA): An integrated Retail Payment Market; 2006
[r55]	European Central Bank; Guideline of ECB on TARGET/Official Journal of the European Union; 23 January, 2006
[r56]	European Commission, Internal Market and Services DG, Financial Institutions, Retail issues, consumer policy and payment systems; Consultive paper on SEPA incentives; 13 February 2006; <http://ec.europa.eu/internal_market/payments/sepa/index_en.htm>
[r57]	European Committee for banking standards (ECBS); IBAN: International Bank Account Number (EBS204 V3.2); AUGUST 2003
[r58]	European Committée for banking standards; IBAN: International bank account number EBS204V3.2; August 2003
[r59]	European Payment Council; SEPA data model,version 2.2; 13 December 2006; page 4,5
[r60]	European Payments Council; EPC066/06 version 1.3 Making SEPA a Reality; 9 January 2007; <http://www.europeanpaymentscouncil.eu/knowledge_bank_list.cfm?documents_category=1>
[r61]	European Payments Council; Framework for the evolution of the clearing and settlement of payments in SEPA (PE-ACH/CSM Framework) version 1.1 Approved; 5 January 2007; page 10

[r62]	European Payments Council; SEPA Credit Transfer Schema Rulebook; 8 March, 2006; page 2, 6
[r63]	European Payments Council; SEPA Direct Debit Schema Rulebook; 8 March, 2006; page 2
[r64]	Gtnews; How Will SEPA Deliver Working Capital Benefits in Accounts Receivable?; 6 October, 2006; <http://www.gtnews.com/article/6505.cfm>
[r65]	Gtnews; The SME's Perspective on SEPA; 29 Jan 2007; <http://www.gtnews.com/feature/167.cfm>
[r66]	http://www.consilium.europa.eu/cms3_fo/showPage.asp?id=242&lang=en&mode=g
[r67]	http://www.dnb.nl/dnb/home/payments
[r68]	http://www.europeanpaymentscouncil.eu/content.cfm?page=sepa_benefits_and_implications
[r69]	Interpay; Annual Report 2005; http://www.equens.com/eng/Publications/Annual_reports.asp
[r70]	ISO/IEC Directives, Part 2; Rules for the structure and drafting of International Standards; 2004 page 8.
[r71]	ISO/IEC; ISO 14977 (EBNF); 1996; <http://www.cl.cam.ac.uk/~mgk25/iso-14977.pdf>
[r72]	ISO; UNIFI (ISO 20022) Message Definition Report, Payments Standards - Exceptions and Investigations; October 2006
[r73]	ISO; 20022-1: International Standard - Overall methodology and format specifications; 15 December 2004
[r74]	ISO; 20022-3: Technical Specification - ISO 20022 modeling guidelines; 15 December 2004
[r75]	ISO; 20022-4: Technical Specification - ISO 20022 XML design rules; 15 December 2004
[r76]	ISO; 20022-5: Technical Specification - ISO 20022 reverse engineering; 15 December 2004
[r77]	ISO; UNIFI (ISO 20022) Message Definition Report, Payments Standards - Initiation; October 2006
[r78]	ISO; UNIFI (ISO 20022) Message Definition Report, Payments Standards - Clearing and Settlement; October 2006; page 42, 43
[r79]	John E. Hopcroft, Rajee Motwani, Jeffrey D. Ullman; Introduction to Automata Theory, Languages, and Computation; 2007; page 30, 45,46, 57, 171, 173, 200, 335, 371, 384
[r80]	NIBESVV (Nederlands Instituut voor het Bank-, Verzekerings- en Effectenbedrijf); Handout SEPA; 5 June, 2007
[r81	P. Garvey, B. French; Generating User Interfaces from Composite Schemas; 12 December, 2003; <http://www.idealliance.org/papers/dx_xml03/papers/03-03-04/03-03-04.html>
[r82]	PricewaterhouseCoopers; e-Invoicing and e-Archiving - taking the next step; 2005 <http://www.pwc.com>
[r83]	TBG5; Finance minutes; 17 November 2006
[r84]	ABN Amro; the 2006 guide to the Single Euro Payment Area; 2006; page 21
[r85]	TWIST Innovations Centre; White Paper; version 1.0; 15 May 2006
[r85]	Universal Business Language (UBL) v2.0 <http://www.oasis-open.org/specs/index.php#ublv2.0>

Glossary

Term	Description
ACH	Automated Clearing House
AES	Advanced Electronic Signature This is any kind of electronic authentication that is attached to or logically associated with the data to be signed and that meets the following requirements (Council Directive 2001/115/EC, art 2(2)© and Council Directive 1999/93/EC, art 2) • it is uniquely linked to the signatory; • it is capable of identifying the signatory; • it is created using means that the signatory is able to maintain under his sole control; and • it is linked to the data to which it relates in such a manner that any subsequent change to the data is detectable.
AOS	Additional Optional Services Complementary features and services based on the schemes of SDD and SCT.
ATM	Automated Teller Machine This is a computerized telecommunications device that provides a financial institution's customers a secure method of performing financial transactions in a public space without the need for a human clerk or bank teller. Using an ATM, customers can access their bank accounts in order to make cash withdrawals (or credit card cash advances) and check their account balances. Many ATMs also allow people to deposit cash or checks, transfer money between their bank accounts, pay bills, or purchase goods and services.
B2B	Business to Business
B2C	Business to Consumer
BBAN	The BBAN (Account Number) is the identifier used by financial institutions which identifies uniquely in an individual country the account of a customer at a financial institution.
BEUC	Bureau Européen des Unions de Consummateurs (The European Consumers'Organisation) is the representative organization of more than 40 independent national consumer organizations from almost 30 European countries.
BGC	Bankgirocentrale
BIC	Bank Identifier Code (ISO 9362) This is developed by SWIFT to ensure error-free identification of parties in automated systems. The BIC is a unique address which, in telecommunication messages, identifies precisely the financial institutions involved in financial transactions.
Cash WG	Cash Working Group
CAST	Corporate Action on Standards CAST includes projects to encourage greater standardization and process automation in bank-to-corporate and corporate-to-corporate communications across the whole financial supply chain. CAST is looking specifically at business models, best practices and standardization in the area of remittance information (and e-reconciliation), digital identity and e-invoicing. The objective is to define end-user requirements in these areas, compare them with any existing standards to detect shortcomings and identify 'best of breed' solutions whenever a single solution is not practical or feasible. In addition, CAST addresses the specific issue of interoperability between EBPP operators and Certification Authorities in Europe.
CCBM	Correspondent Central Banking Model, which is a solution to ensure that all assets eligible for use in the monetary policy operation and to obtain liquidity in TARGET are made available to all the Eurosystem's counterparties;
CESR	Committee of European Securities Regulators

CIT	Cash-In-transit company
Clearing	This is the process of transmitting, reconciling and confirming payment orders, and establishing a final position for settlement (either based on individual transactions or bundles of transactions) [r61].
Clearing house	A central collection place where banks exchange checks or drafts; participants maintain an account against which credits or debits are posted.
Credeuro/ICP	Interbank Convention on Payments is a convention, which establishes the pan-European interbank principles for basic cross-border straight through processing (STP) credit transfers in euros.
CSM	Clearing and Settlement Mechanisms This is the separated processing infrastructure, which is described in the SEPA ECT and EDD Rulebooks.
CT	Credit Transfer A payment initiated by the payer. In the case of a credit transfer, a payment instruction is sent to the payer's bank (the sender's bank), which moves the funds to the payee's bank (the receiver's bank), possibly via several intermediaries.
CWG	Cards Working Group
DD	Direct Debit A transfer initiated by the payee (the receiver) via the payee's bank after agreement between the payee and payer (the sender). Direct debits are often used for recurring payments (such as utility bills) with a pre-authorised agreement being put in place with the payer. Direct debits are also used for one-off payments where the payer authorizes an individual payment.
DTD	Document Type Definition
EACT	European Association of Corporate Treasurers The Purpose of EACT is to group the National Associations of Corporate Treasurers and Financiers of the countries belonging to the European Union. EACT's aims are: • to develop and strengthen relations with European authorities and institutions; • to share experiences, express common points of view, undertake joint actions on financial and treasury matters as well as relationships with financial partners; • to carry out and publish joint surveys and working papers.
EBA	European Banking Association
EBPP	Electronic Bill Presentment and Payment, the process by which companies bill customers (the business-to-consumer or B2C) and receive payments electronically over the Internet. There are two types of presentment models: • direct model: a biller delivers the bill to customers via its own Web site, or via a third-party's site; • consolidator model: bills from multiple billers are delivered to a single Web site, to be presented in aggregate to the consumer for viewing and payment.
ebXML	electronic business XML A global framework for the standardized use of XML in e-business on internet-based networks.
EC	European Commission
ECB	European Central Bank
ECBS	The European Committee for Banking Standard
ECOFIN	Economic and Finance Council
ECSAs	European Credit Sector Associations
ECTWG	Electronic Credit Transfer Working Group
EDDWG	Electronic Direct Debit Working Group
EDI	Electronic Data Interchange, this means • an electronic transfer of data; • from computer to computer;

	• using an agreed structured format; • that can be read by a computer; • and can be processed automatically and unambiguously; • there should be an interchange agreement between the EDI trading partners making provision for the use of procedures that guarantee the authenticity of the origin and the integrity of the data.
EEA	European Economic Area (Norway, Iceland, Liechtenstein)
E-invoicing	Electronic invoicing is the sending of invoices 'by electronic means', i.e. transmission or making available to the recipient and storage using electronic equipment for processing (including digital compression) and storage of data, and employing wires, radio transmission, optical technologies or other electromagnetic means (Council Directive 2001/115/EC, art 2(2)(e)).
EIPP	Electronic Invoice Presentment and Payment is the process by which companies present invoices and make payments to one another (B2B) through the internet.
EMV	Europay MasterCard Visa programme to implement CHIP & PIN security for card transactions. EMV contains a functional specification of a payment application on a chip and specifications for the card and terminal.
EPC	European Payments Council The decision making and coordination body of the European banking industry in relation to payments. Its purpose is to support and promote the creation of SEPA. EPC defines common positions for core payment services within a competitive market place, provides strategic guidance for standardization, formulates best practices and supports and monitors implementation of decisions taken.
EPCA	European Payments Consulting Association
EPOS	An integrated POS device which incorporates EftPos, scanner and merchant applications.
ESCB	The European System of Central Banks comprises the ECB and the National Central Banks (NCBs) of all EU Member States whether they have adopted the euro or not.
e-SEPA	This regards to the extension of SEPA with the standardization of end-to-end processes for corporations and the development of 'forward-looking' features.
ETS	Electronic Transfer Scheme
Eurosystem	The Eurosystem comprises the ECB and the NCBs of those countries that have adopted the euro.
FTM	Financial Transfer Message
IBAN	International Bank Account Number (ISO 13616 and EBS 204) An expanded version of the Basic Bank Account Number (BBAN) used to uniquely identify the account of a customer at a financial institution. The aim of the IBAN is to facilitate the automatic processing of cross-border credit transfers.
ICS	International Card Schemes (e.g. Visa Mastercard)
IFT	Interbank File Transfer
IFX	Interactive Financial Exchange
IPR	Intellectual Property Rights
IRD	Image Replacement Document
ISO	International Organization for Standardization
ISTH	International Standards Team Harmonization
M&A	Mergers and Acquisitions
MoU signature	Memorandum of Understanding
M-PEDD	Multi-purpose pan-European direct debit
MTU	Mobile Top Up
NBAs	National Banking Associations
NBC	Nationaal Betalingscircuit

NCB	National Central Bank
NFC	Near Field Communication
NGC	Nominating and Governance Committee
NLF	New Legal Framework is a body of legislation (Directives and Regulations) from the European Parliament and Council. It governs payments services throughout the European Union.
OAGi	Open Applications Group Inc.
OASIS	Organisation for the Advancement of Structured Information Standards
OCR	optical character recognition
OITS	SG Operations Infrastructure Technology & Standards Support Group
pain	payments initiation (used at UNIFI)
payment scheme	The rules and practices for the provision and operation of a SEPA payment instrument agreed at interbank level in a competitive environment.
PE-ACH	Pan European Automated Clearing House
PEDD	Pan European Direct Debit
PIWG	Payment Instrument Working Group
POS	Point of Sale
PSD	Payment Services Directive (proposal sent to European Parliament on 01/12/2005). This Directive will ensure that the same legal framework applies to all payments made within Europe.
RA	Registration Authority
RMG	Registration Management Group
ROC	Roll Out Committee Part of the EPC. The purpose of ROC is to prepare and support the successful roll-out phase of the SEPA CT & DD Schemes, by coordinating and guiding national communities steering EPC activities.
RTGS	Real Time Gross Settlement
SCF	SEPA Cards Framework
SCF	SEPA Cards Framework
Scheme	A set of rules, practices and standards agreed between providers of payment services.
SCT	SEPA Credit Transfer
SDD	SEPA Direct Debit
SECA	Single Euro Cash Area
SEGs	Standards Evaluation Groups
SEPA	Single Euro Payment Area
Settlement	This is the transfer of funds between the payer and the payee (and between the payer's bank and the payee's bank) [r61].
SGML	Standard Generalized Markup Language (ISO 8879) A meta language in which one can define markup languages for documents.
SME	Small Medium Enterprises
SMPG	Securities Market Practice Group
SSP	Single Shared Platform
standards	Standards are rules that govern technology, behavior and interactions. Technical standards are necessary to allow interaction and interoperability between IT systems and to foster automation of the payment process [r54 page 25].
STEP2	Europe's first pan-European ACH managed by EBA clearing company
STP	Straight-through-processing (can be applied to the bank-to-bank chain only, or on an end-to-end basis). It means: • value-added services offered before payment; • processing the payment; • value-added services offered after payment.
SWIFT	Society for Worldwide Interbank Financial Telecommunication

TARGET2	The Euro system's replacement for TARGET (Trans European Automated Real-time Gross settlement Express Transfer system)
TC68	Technical Committee 68 - Financial Services (TC68) of the International Organization for Standardization (ISO)
TOP	TOP is the domestic large-value giro payment system of De Nederlandsche Bank (DNB). TOP ensures that transactions are processed correctly and in time and that adequate information is provided about them. TOP is a real-time gross settlement system, which means that payments are carried out immediately, irreversibly and individually, giving both transferer and transferee immediate certainty as to each amount transferred [r67].
TWIST	Treasury Workstation Integration Standards Team. TWIST is focused on the innovation and standardization of the processes to deliver the requirements of corporates.
UN/CEFACT	United Nations / Centre for trade facilitation and electronic-business
UN/CEFACT	United Nations / Centre for trade facilitation and e-business
UNIFI	UNIversal Financial Industry message scheme It's the nickname of 'ISO 20022 - the platform proposed by ISO to develop all financial messages. UNIFI does not describe the messages themselves; it is a 'recipe' to develop message standards. The main ingredients of this recipe are a development methodology, a registration process and a central repository.
VAT	Value Added Tax (in Dutch BTW)
web service	This is a programmatic interface/software component for application to application communication over the Internet. Interfaces of web services are described in WSDL (Web Service Description Language) and can be accessed by other applications via standard network protocols like SOAP (Simple Object Access Protocol).
XSD	XML Schema Definition, the successor of DTD. XSD is an instance of an XML schema written in XML Schema.
XSLT	XSLT stands for XSL Transformations. XSL stands for Extensible Style sheet Language Family, which is a family of recommendations for defining XML document transformation and presentation. It consists of three parts: • XSL Transformations (XSLT)a language for transforming XML; • the XML Path Language (XPath), an expression language used by XSLT to access or refer to parts of an XML document; • XSL Formatting Objects (XSL-FO) an XML vocabulary for specifying formatting semantics.

Appendices

Appendix 1: Project plan

G. Craens
Date: 12-2-2007
Version: 01
Status: final

Table of content

1 Scope of this document

This document contains the major project choices and plans, which are not already mentioned in other documents like the planning document.
The goal of this project plan is that it has to support with managing the project.

2 Document changes

Date	Modification/added	Version

3 Introduction

The goal of the final research project is to extend experience of scientific research in Software and Systems Engineering. Besides the practical part of the assignment, the project also includes a theoretical part.

4 General Information

Name	George Craens gcraens@gmail.com
Student number	s1450522
Period	3 and 4
Start date	12-02-07
End date	06-07-07
Duration	20 weeks, 100 days
Supervisor RuG	Ir. S. Achterop
Supervisor company	drs. Hans Stevens
Location	Pecoma Friesche straatweg 211c 9743 AD Groningen 050 520 1888

5 Deliverables

Deliverables of this final research project are:

1 Requirement document
2 Design document
3 Software
4 Research paper
5 Presentation

6 Assignment description

Banks and other financial organizations are using a lot of different formats to realize services (e.g. payments by direct debit or credit card) of their customers today. Over 26 XSD formats with different number of attributes could be used for a single type of transaction.
Also the difference in law of a country causes different formats. The rules for an automatically direct debit are for example different in Italy compared with the Netherlands.

The Single Euro Payments Area (SEPA) initiative for the European financial infrastructure involves the creation of a zone for the Euro in which all electronic payments are considered domestic, and where a difference between national and international payments does not exist. The project aims to improve the efficiency of international payments and also to develop common financial instruments, standards, procedures, and infrastructure to enable economies of scale. This will replace the complex and costly international infrastructures that are currently in operation (such as SWIFT[28]), thus reducing the overall cost to the European economy of moving capital around the region.
SEPA is based on ISO20022[29] and uses the XML and XSD format. The agreement of standardization was made in December 2006.

It seems that XSD will be the solution for standardization in the future. But is XSD suitable? What are the constraints of XSD?
Design and implement an architecture that will result in a SEPA Unifi Test tool. This tool should:
1. be able to validate XML against XSD
2. be able to generate XML based on an XSD
3. be able to transform CSV, XLS, ... to XML format
4. be able to generate user entry form based on XSD for user friendly entry
5. SEPA Unifi Test tool should be smarter than XSD "out of the box" like for example XX123456789 is valid bank account according to [A-Z]{2}[0-9]{1,32} regular expression but not likely to be an existing account as XX is not a country code and 123456789 not likely to be given out by a bank.

[28] The SWIFT (Society for Worldwide Interbank Financial Telecommunication) operates a worldwide financial messaging network. Messages are exchanged between banks and other financial institutions.
Basically, SWIFT provides a centralized store-and-forward mechanism, with some transaction management. For bank A to send a message to bank B, it formats the message according to standard, and securely sends it to SWIFT

[29] The UNIFI standard (UNIversal Financial Industry message scheme) provides the financial industry with a common platform for the development of messages in a standardized XML syntax, using:
- a modelling methodology (based on UML) to capture in a syntax-independent way financial business areas, business transactions and associated message flows;
- a set of XML design rules to convert the messages described in UML into XML schemas.

This flexible framework allows communities of users and message development organizations to define message sets according to an internationally agreed approach and to migrate to the use of common XML-based syntax.
Source: http://www.iso20022.org/

7 Planning

7.1 Global

Description	Date	Project week
Start	12-02-07	1
Main research question and sub questions defined	23-02-07	2
Prototype Test Tool version 01	26-03-07	7
Evaluation Final Research Project with RuG	16-04-07	10
Final version Test Tool	28-05-07	16
Presentation document ready	25-06-07	20
End	06-07-07	21

7.2 Detailed

The content of the detailed version of the planning can be found in "Planning_XX.xls".

8 Risk management

At the moment the only relevant risk of this project is the obscurity of the main research question and sub questions.
The risk can be presented with a name, the impact it has got on the project and the chance that a problem will arise. The impact will be defined on a scale of 0 – 100.

Risk name	Impact	Chance in %	Total	Possible solution
No research question defined before 23-02-07 (project week 2)	60	25	1500	Contact supervisor(s)
Unknown technology	5	80	400	Retrieve information of the Internet to get familiar with the new technology

Appendix 2: Requirements document

SEPA UNIFI Test Tool

G. Craens
Date: 09-04-2007
Version: 007
Status: final

Table of content

1 Scope of this document

This document describes the system services and constraints of the SEPA UNIFI Test Tool.

2 Document changes

Date	Modification / added	Version
19-02-2007	Picture scope replaced, Priority added, overlap deleted, FR1 + FR4 extended	004
02-03-2007	4.2 + 4.4 + 6.2 updated	005
09-04-2007	FR 6 (IBAN) renamed to 5a; 5b added (BIC) FR7 renamed to FR6b + c; 6a added (log in)	006
14-06-2007	FR 1d renamed to 1h; 5a renamed to 1d; 5b renamed and split into 1e and 1f; 1g (load XSD) has been added and 6 became 5.	007

3 Project preliminaries

3.1 Purpose and scope of the project

The goal of the final research project is to extend experience of scientific research in Software and Systems Engineering. The research goes about standardization in the financial world.
The realization of the goal exists of a practical part and a theoretical part. For the practical part a SEPA UNIFI Test Tool will be made. The research questions will be defined during the first weeks of the project.

3.2 General assignment description

It seems that XSD will be the solution for standardization in the future. But is XSD suitable? What are the constraints of XSD?
Design and implement an architecture that will result in a SEPA UNIFI Test Tool.

3.3 Business Context

Banks and other financial organizations are using a lot of different formats to realize services (e.g. payments by direct debit or debit card) of their customers today. Over 26 XSD formats with different number of attributes could be used for a single type of transaction. Also the difference in law of a country causes different formats. The rules for direct debit are for example different in Italy compared with the Netherlands.

The Single Euro Payments Area (SEPA) initiative for the European financial infrastructure involves the creation of a zone for the euro in which all electronic payments are considered domestic, and where a difference between national and international payments does not exist. The project aims to improve the efficiency of international payments and also to develop common financial instruments, standards, procedures, and infrastructure to enable economies of scale. This will replace the complex and costly international infrastructures that are currently in operation (such as SWIFT), thus reducing the overall cost to the European economy of moving capital around the region. SEPA is based on ISO20022 and uses the XML and XSD format. The agreement of standardization was made in december 2006.

3.4 Stakeholders

Name	Role /responsibility
G. Craens	Student, will do the research and realize the system
S. Achterop	Supervisor (RuG)
H. Stevens	Supervisor (Pecoma)
D. Rinkes	Coördinator (Pecoma)
E. Wildschut	Technical supervisor (Pecoma)

3.5 Ideas for Solutions

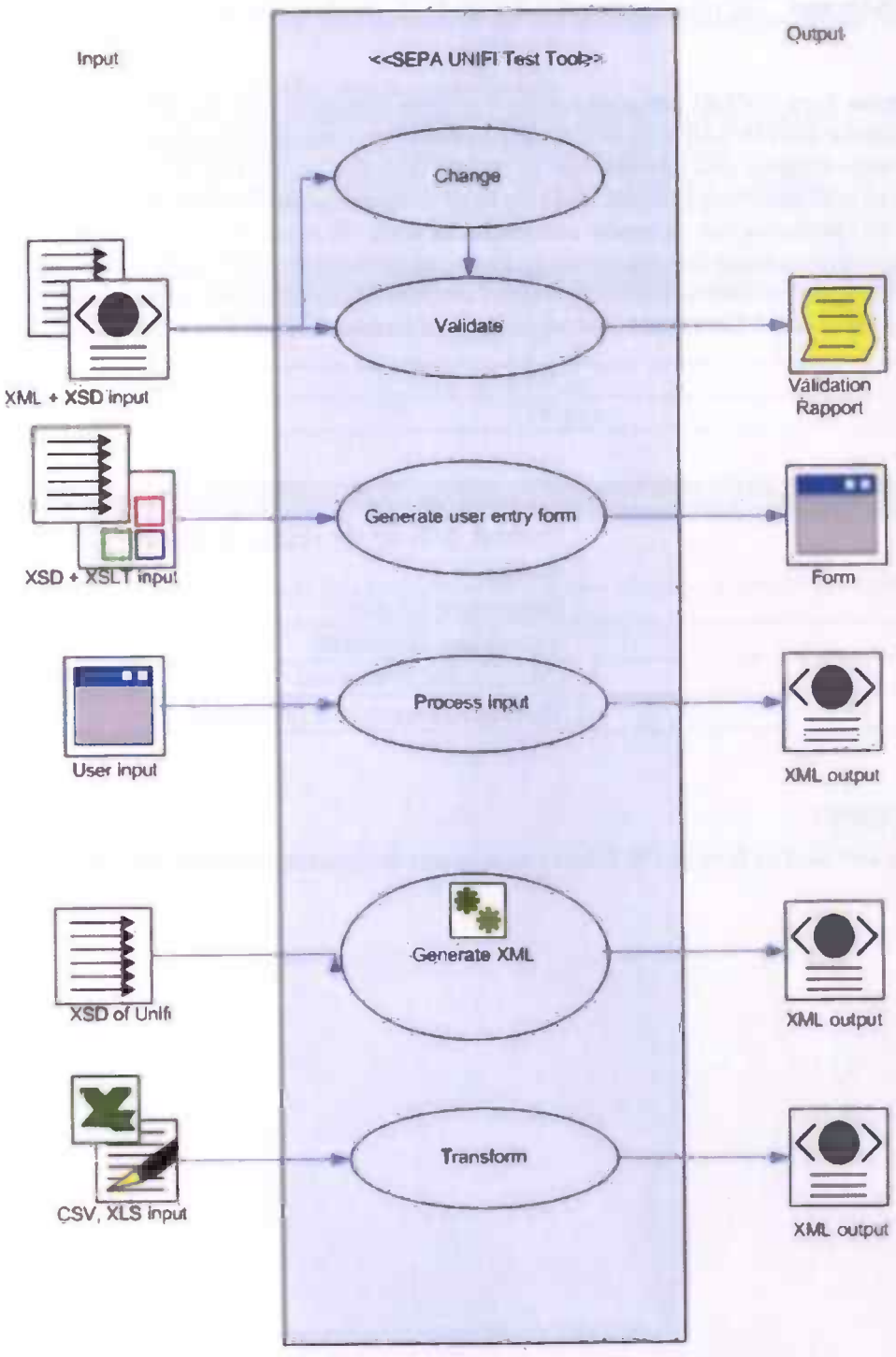
The system will be implemented in Java or C#. Users could access the system via a web interface.

4 System services

This chapter describes what the system must accomplish.

4.1 The scope of the system

The scope of the system is summarized with the model below, which will be described in more detail in the next paragraph.



4.2 Function requirements

The table of requirements has been moved to the SEPA UNIFI Test Tool chapter of the thesis.

4.3 Data requirements

Standard data formats like XML will be used.

4.4 Priority

Priorities should be made because the list of requirements could be extended, but the project time could not. The method MoSCoW will be used to do this. It originated as part of the Dynamic Systems Development Method.

- Must Haves are fundamental to the projects success
- Should Haves are important but the projects success does not rely on these
- Could Haves can easily be left out without impacting on the project
- Won't Have this time round can be left out this time and done at a later date

Must have	Should have	Could have	Won't have this time
FR1a,b,c,d,e,f,g	FR4a,b	FR1h	FR3b,c
FR2a		FR2b	
		FR3a	
		FR5b,c	

5 System constraints

5.1 Interface requirements

Req. #	Description
NF1	Users should have simple access to the Test Tool. For example via a web interface.

5.2 Performance requirements

Not defined.

5.3 Security requirements

Not defined.

5.4 Other constraints

Not defined.

6 Projects matters

6.1 Open Issues

6.1.1 Choice of architecture vs. performance

Although the performance requirements are not defined yet, the table below can help to make a choice between a client server solution and a stand alone solution.
The table is related to the first case and gives an overview of the time a user has to wait minimal if he or she wants to upload an XML file to validate.

Time (sec)		File size (MB)		
		1	10	100
Speed (Mbits/sec)	1	8	80	800
	10	0,8	8	80
	100	0,08	0,8	8
	1000	0,008	0,08	0,8

Table 1: case 1 client/server; File size in MB * 8 / Mbits per sec = time in sec

6.1.2 Security

Of course security is important and definitely in case of transactions of money. If a simple encryption of the data can be combined with the technologies around UNIFI has to be investigated. Because it goes about a test tool, this issue will be assumed not to have a 'must have' priority.

6.1.2 Preliminary Schedule

	End data	Project week
Prototype version 1	27-04-2007	11
Final version	28-05-2007	16

More details are described in Planning_002.xls.

7 Glossary

Acronym	Description
BIC	Bank Identifier Code
ClieOp	Stands for 'cliëntopdrachten', which is the name of an electronic file format that can be used by business clients to deliver their direct debit orders to a bank.
IBAN	International Bank Account Number
SEPA	Single Euro Payment Area
SWIFT	Society for Worldwide Interbank Financial Telecommunication
UNIFI	UNIversal Financial Industry message scheme It's the nickname of 'ISO 20022 - the platform proposed by ISO to develop all financial messages. UNIFI does not describe the messages themselves, it is a 'recipe' to develop message standards. The main ingredients of this recipe are a development methodology, a registration process and a central repository.

8 References

http://www.iso20022.org
http://www.europeanpaymentscouncil.eu/index.cfm
http://www.swift.com

Appendix 3: Use case document

SEPA UNIFI Test Tool

G. Craens
Date: 09-03-2007
Version: 002
Status: final

Table of content

1 Scope of this document

This document describes the functionality of the SEPA UNIFI Test Tool, further in this document referred as 'tool' or 'system'. For each use case the actor will be an end user on the website.

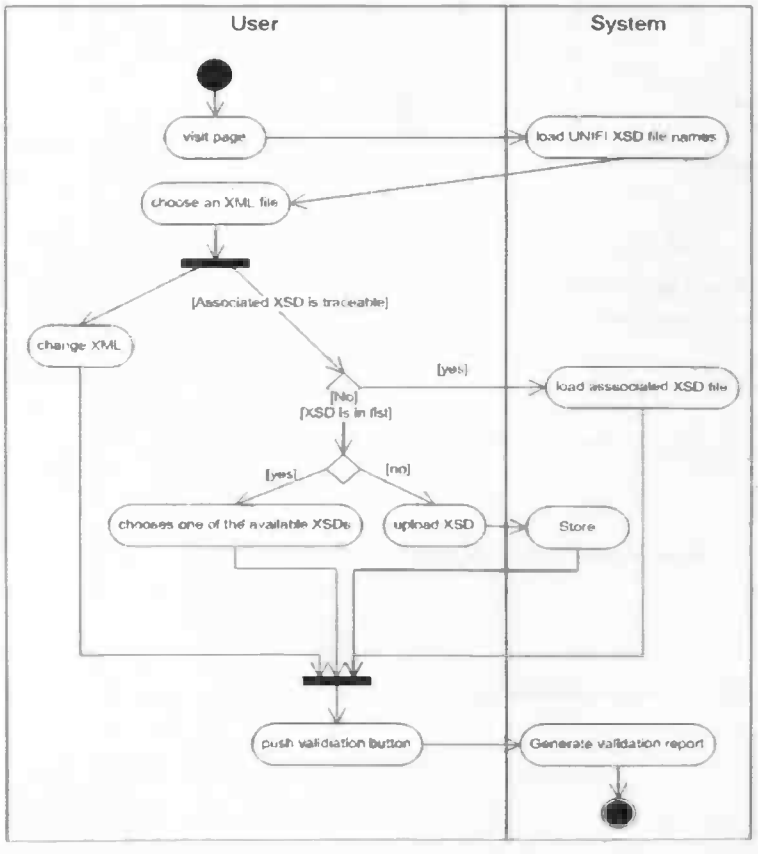
2 Document changes

Date	Modification / added	Version
09-03-07	Activity diagrams added + validation (could have)	002

3 Use cases & activity diagrams

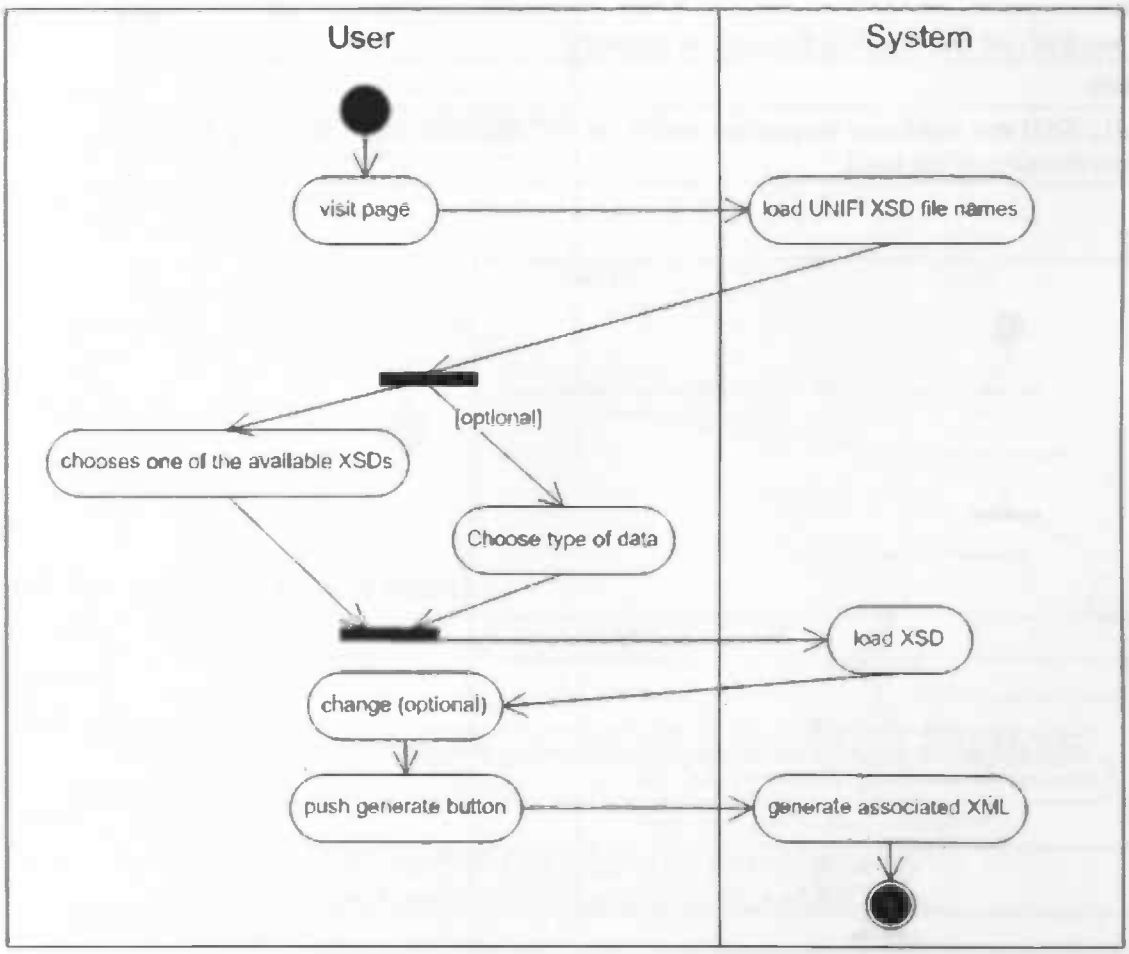
3.1 Validate

Use Case Number	UC001
Actors	
Start state	
A user navigates to the validation page.	
Basic scenario	
1. A user will choose an XML file of his or her computer (to up load). 2. The system tries to load the associated XSD file 3. (optional) The user chooses one of the available XSDs or will choose an XSD file of his or her computer (to up load). The system will display the content of these files. 4. (optional) The user may change the content 5. The user triggers the system to validate. 6. The system presents a validation rapport. The system will announce the user that the XML is valid or points to the user what is invalid.	
Alternative scenario	
• In stead of step 1 and/or 2, the user pastes the content of an XML and XSD file in a text area. • In stead of one file, multiple files can be uploaded.	
End state	
An XML, XSD and validation rapport are visible on the validation page (including links to other functionality of the tool)	



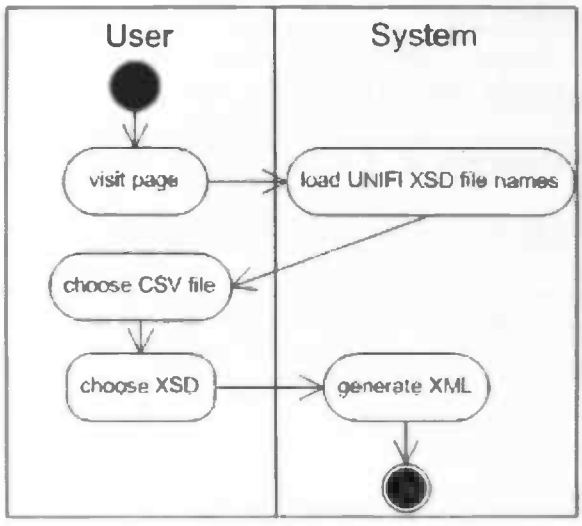
3.2 Generate XML

Use Case Number	UC002
Actors	
Start state	
A user navigates to the generate XML page. Names of a set of UNIFI XSD's are loaded.	
Basic scenario	
1. The user chooses one of the available XSD's. 2. The system generates and displays XML including dummy data in a certain appropriate format.	
Alternative scenario (could have)	
1. The user chooses an XSD and makes a choice between generating dummy data or data from a database e.g. BIC or IBAN codes. 2. The system generates and displays XML including data in a certain appropriate format, which depends on the user's choice.	
End state	
The above described XML output is visible on the generate XML page. This could be saved to a storage area on the server	



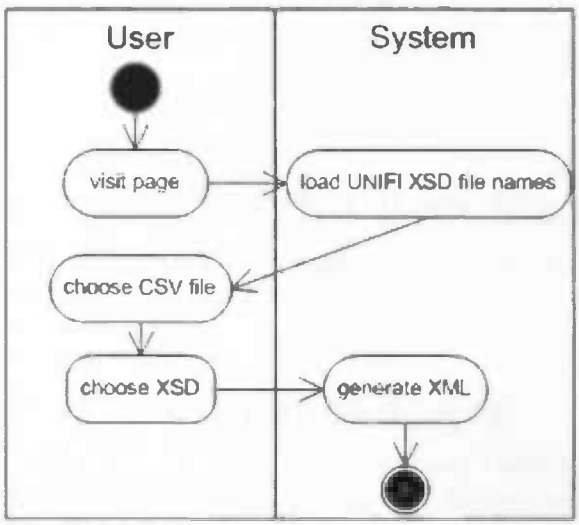
3.3 Generate form

Use Case Number	UC003
Actors	
Start state	
A user navigates to the generate form page. Names of a set of UNIFI XSD's are loaded.	
Basic scenario	
1. The user chooses one of the available XSDs. 2. The system generates and displays a form	
Alternative scenario	
3. The basic scenario will be followed up with the scenario that the user fills in the form 4. The system creates a report of the filled in form.	
End state	
Basic scenario: a generated form Alternative scenario: a report of user input	



3.4 Transform

Use Case Number	UC004
Actors	
Start state	
A user navigates to the transform page. Names of a set of UNIFI XSD's are loaded.	
Basic scenario	
1. The user chooses a CSV* file of his or her computer (to up load). 2. The user chooses one of the available XSD's. 3. The system tries to generate XML according to the format as described in a chosen XSD with content of the uploaded file. The system will inform the user and displays the results	
Alternative scenario	
* Maybe the user can choose other formats in the future.	
End state	
The above described XML output is visible on the transform page.	



3.5 Manage files (could have)

Use Case Number	UC005
Actors	
Start state	
A user has been logged in and navigates to the manage page. A list of XML and XSD files are displayed.	
Basic scenario	
A user can add or delete XML and XSD files to his or her personal directory. A user can attach, remove or change a description of the XML and XSD files.	
Alternative scenario	
End state	
A list of XML and XSD files are displayed.	

3.6 Validation (could have)

Use Case Number	UC006
Actors	
Start state	
1. A user has been logged in and navigates to the validation page. The system will load a list of available personal XML files.	
Basic scenario	
2. A user will choose a one or more XML files of his or her computer (to up load) or will select one or more personal XML files of a list. 3. The system tries to load the associated XSD file, which could be available in the common UNIFI directory or the personal directory. 4. (optional) The user chooses one of the available XSDs or will choose an XSD file of his or her computer (to up load). The system will display the content of these files. 5. (optional) The user may change the content 6. The user triggers the system to validate. 7. The system will validate the syntax, but also some parts of the content, like for example if an IBAN number exist. 8. The system presents a validation rapport. The system will announce the user that the XML is valid or points to the user what is invalid.	
Alternative scenario	
• In stead of step 1 and/or 2, the user pastes the content of an XML and XSD file in a text area. • In stead of one file, multiple files can be uploaded.	
End state	
AN XML, XSD and validation rapport are visible on the validation page (including links to other functionality of the tool)	

Appendix 4: Design document

SEPA UNIFI Test Tool

G. Craens
Date: 06-04-2007
Version: 004
Status: final

Table of content

1 Scope of this document

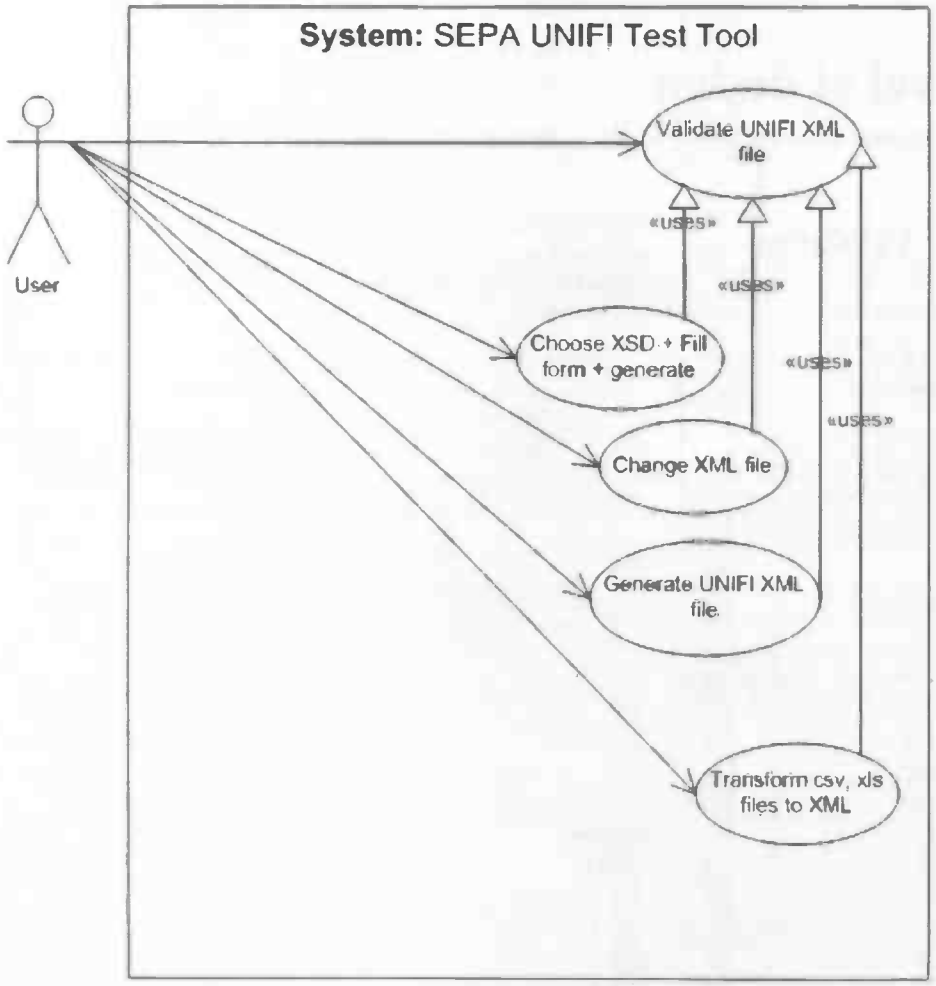
The purpose of this document is to describe the design of the SEPA UNIFI Test Tool at different levels of abstraction.

2 Document changes

Date	Modification / added	Version
19-02-2007	High level design synchronized with req. doc version 004	002
06-04-2007	Class Diagram added: Validation.	003
24-04-2007	Class Diagram added: GenerateXML, GenerateForm, Transform, Manage files	004

3 High level design

3.1 Use case diagram



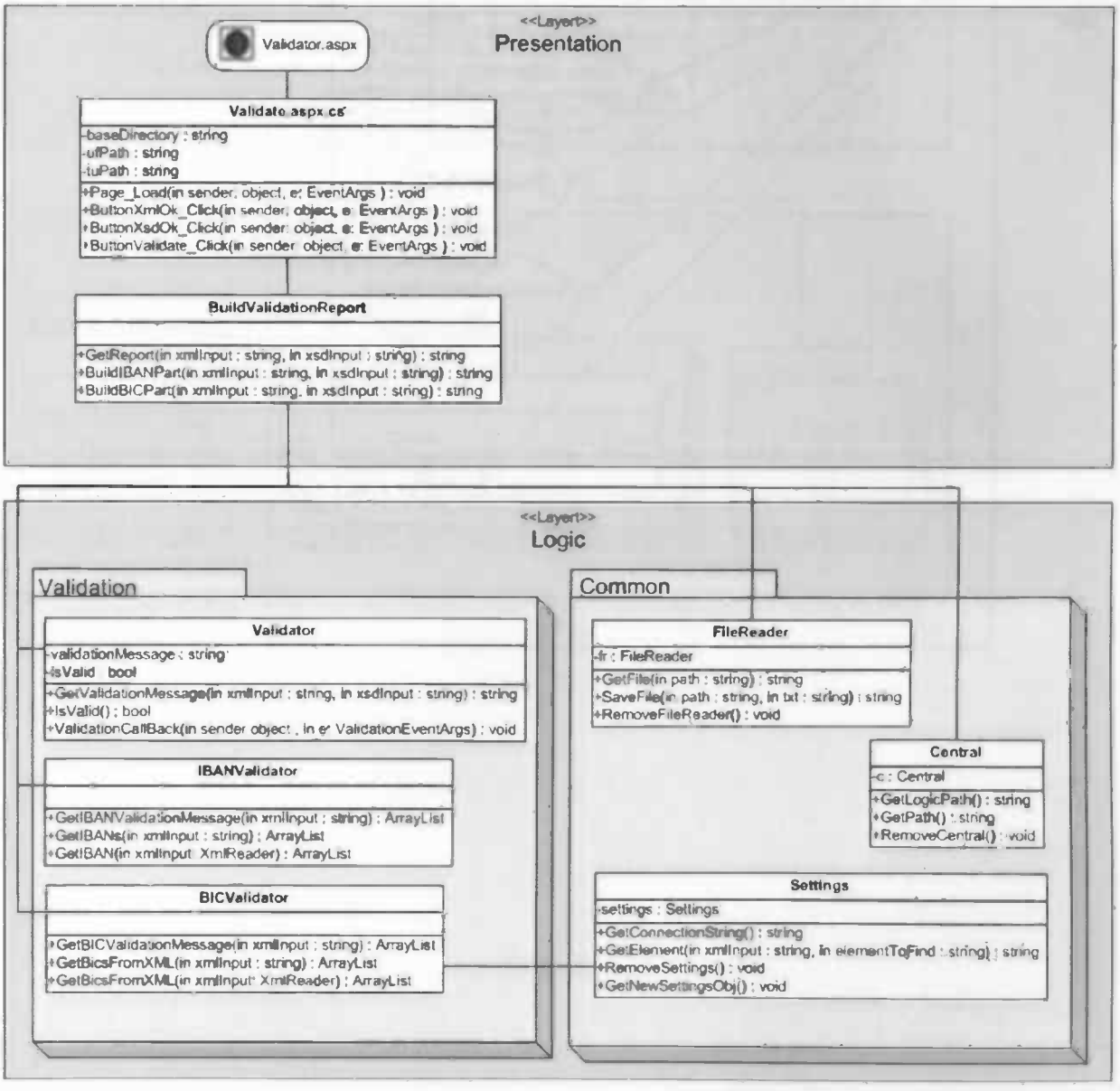
3.2 Architecture of SEPA UNIFI Test Tool components

See chapter: SEPA UNIFI Test Tool

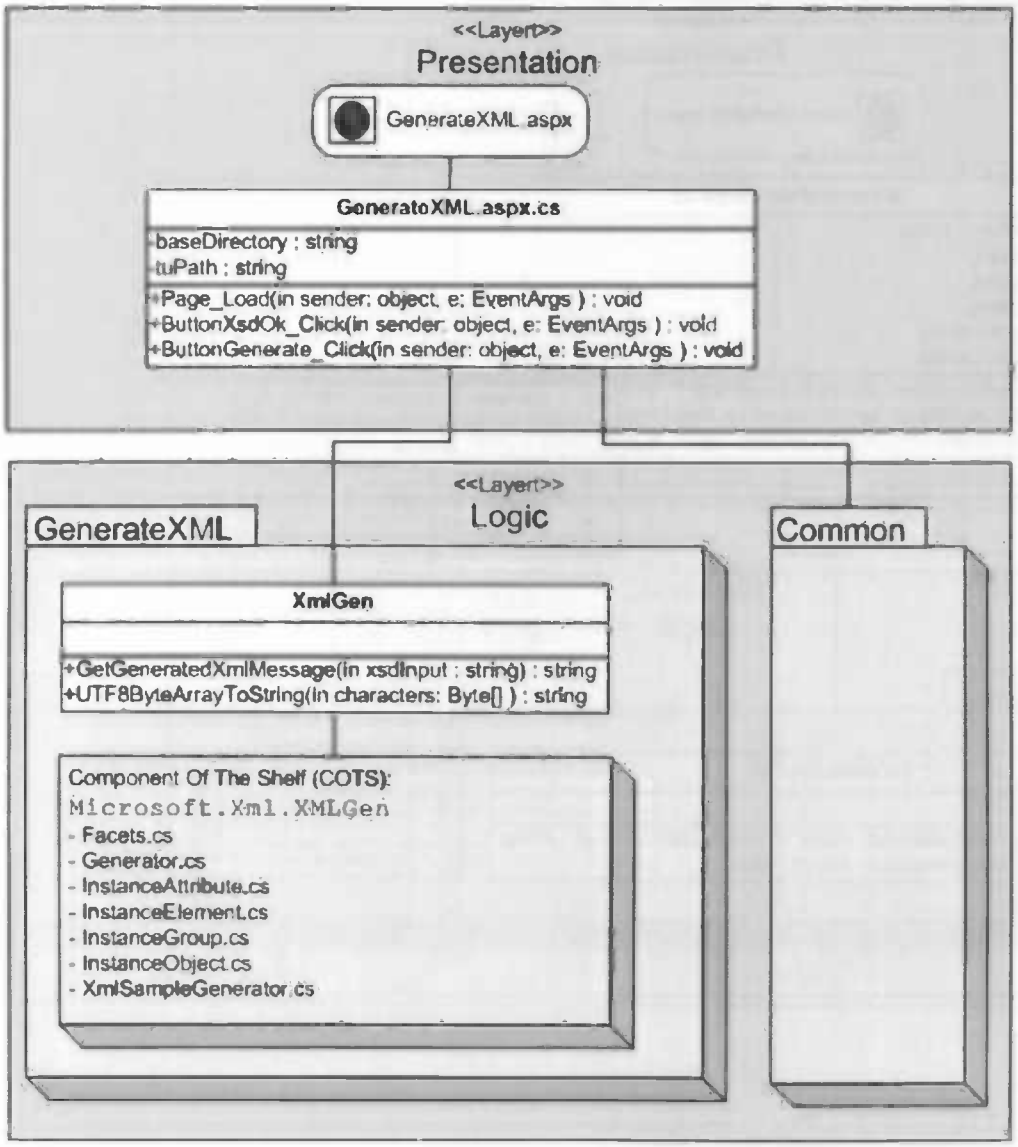
4 Detailed level of design

The following diagrams present class diagrams of the system per functionality of the presentation layer and the logic layer.

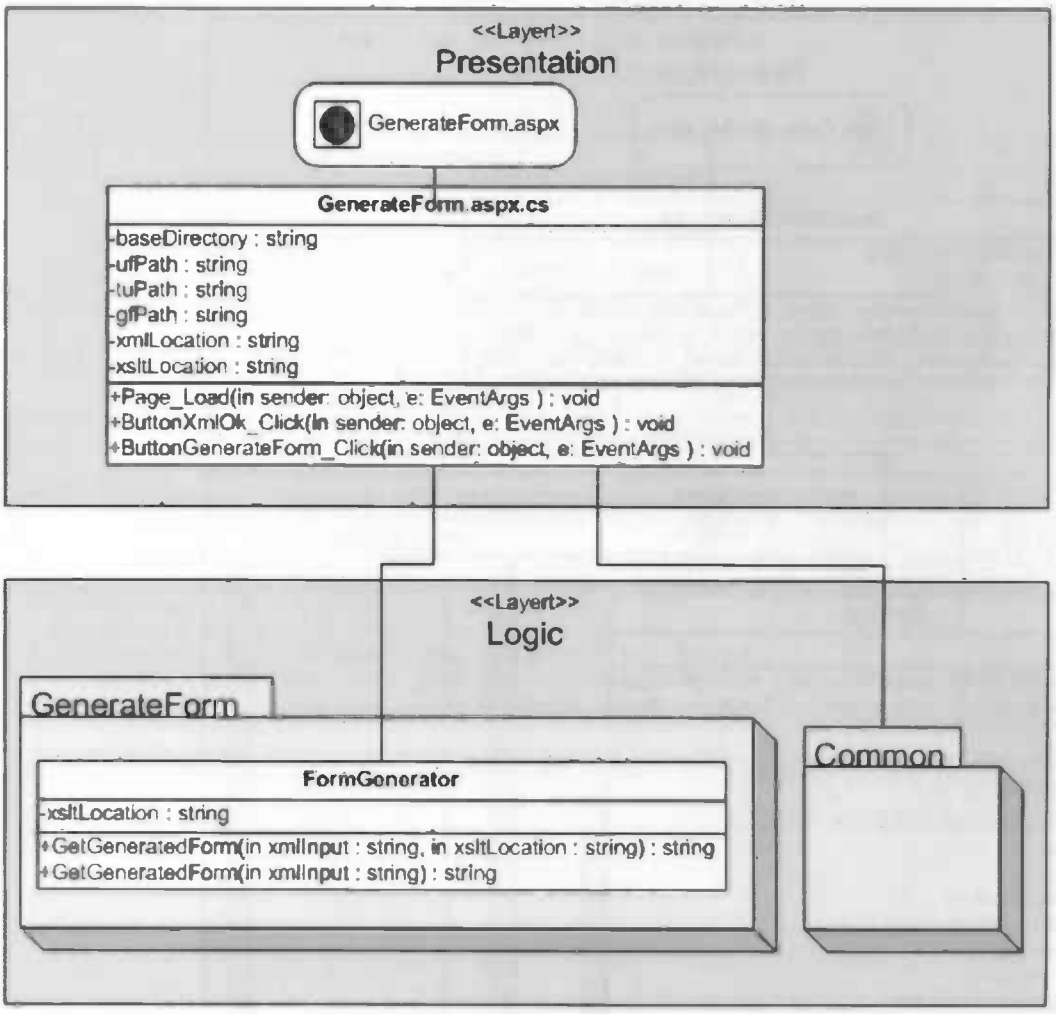
4.1 Class diagram: Validation



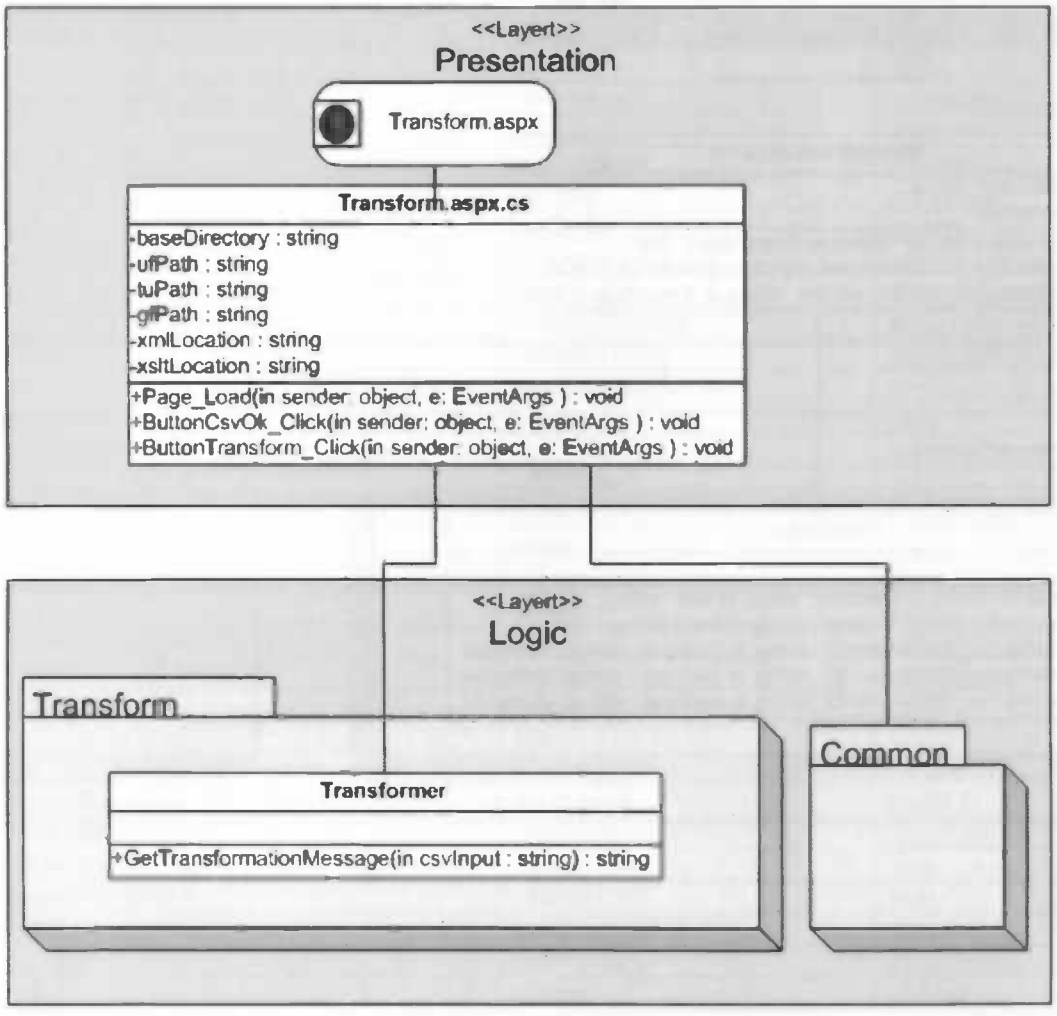
4.2 Class diagram: Generate XML



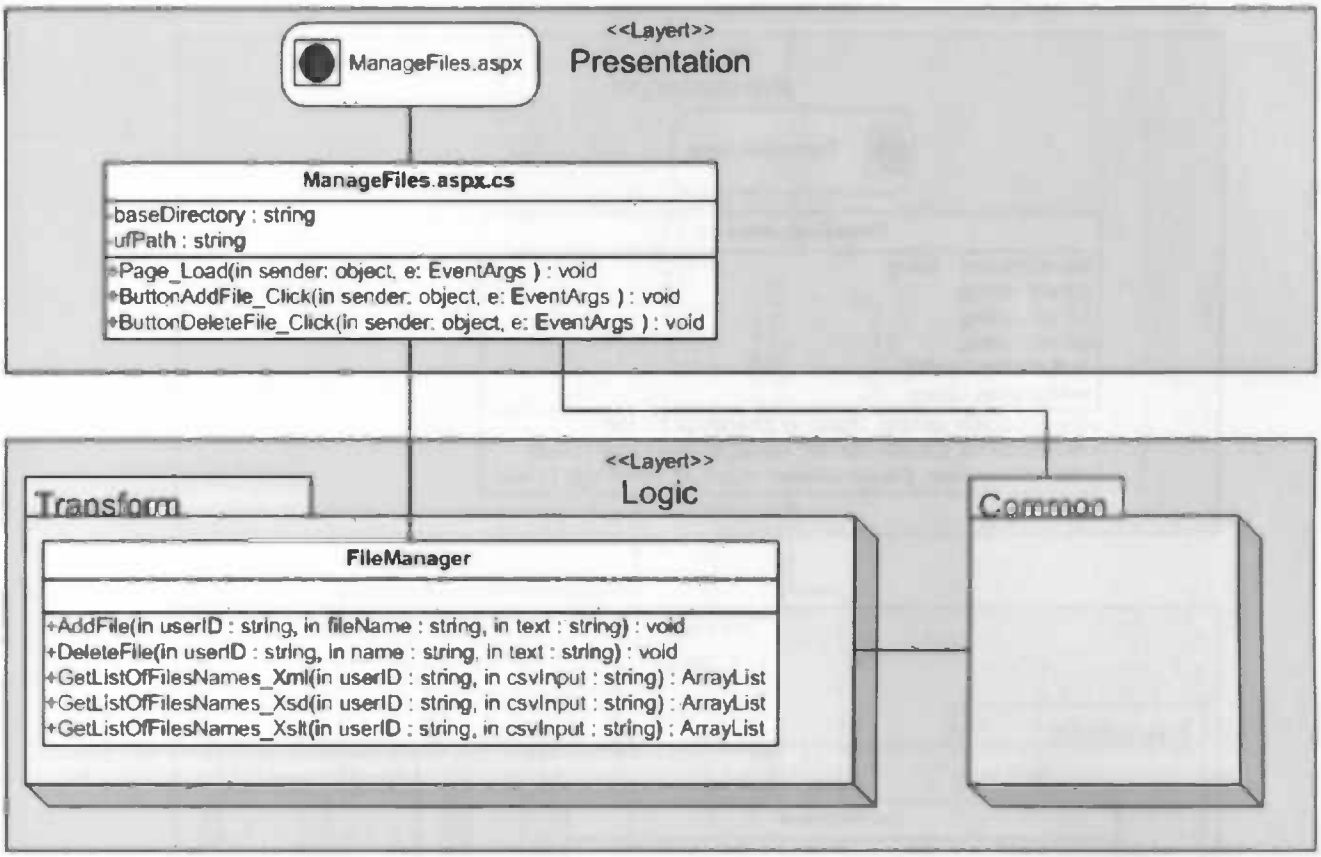
4.3 Class diagram: Generate form



4.4 Class diagram: Transform



4.5 Manage files

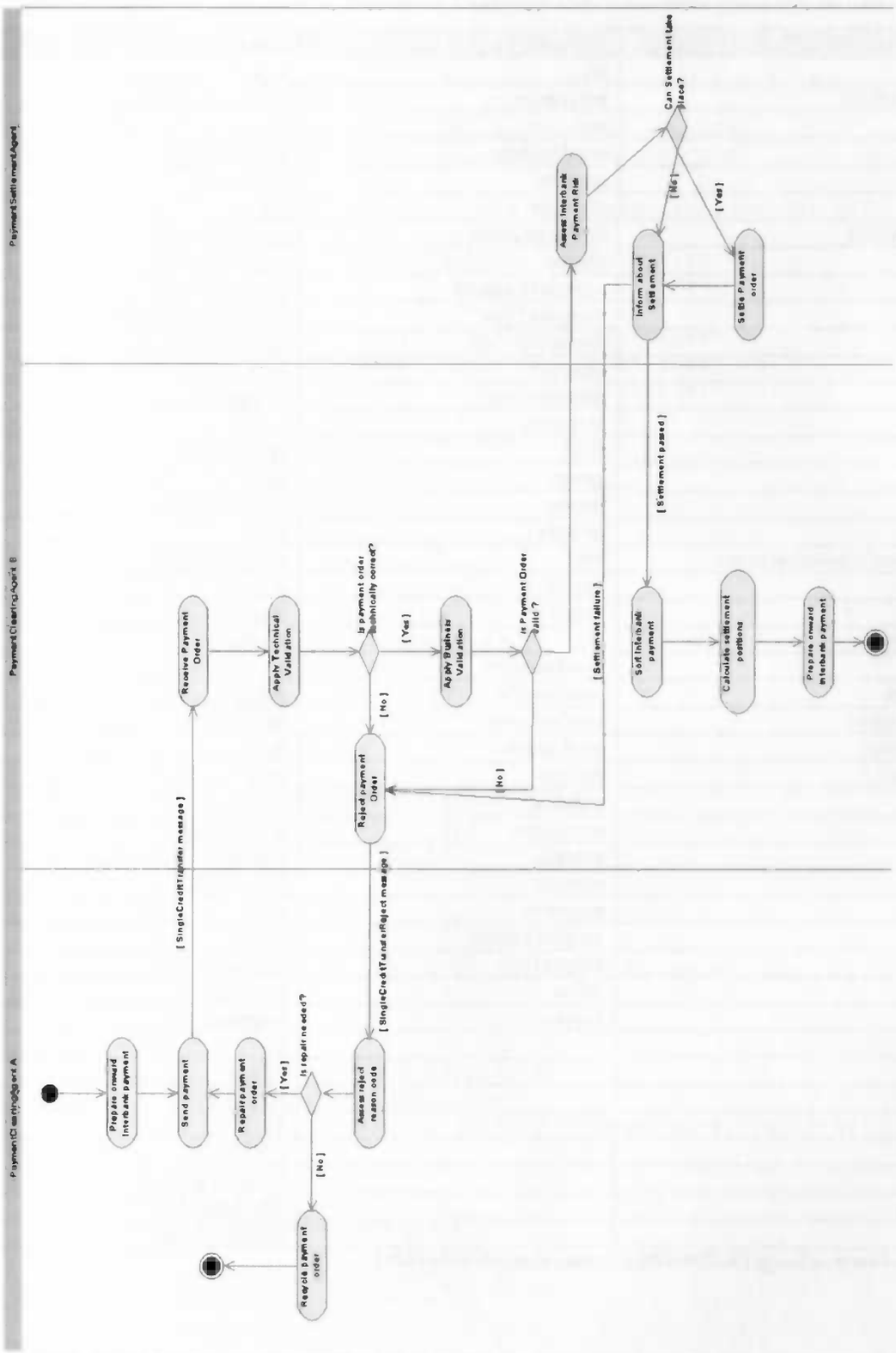


Appendix 5: XSD Attributes, elements and types

Attributes	Elements	(simple) types
abstract	all	string
attributeFormDefault	annotation	normalizedString
base	any	token
block	anyAttribute	base64Binary
blockDefault	appinfo	hexBinary
default	attribute	integer
elementFormDefault	attributeGroup	positiveInteger
final	choice	negativeInteger
finalDefault	complexContent	positiveInteger
fixed	complexType	nonNegativeInteger
form	documentation	nonPositiveInteger
itemType	element	long
memberTypes	enumeration	unsignedLong
maxOccurs	extension	int
minOccurs	field	unsignedInt
mixed	group	short
name	import	unsignedShort
namespace	include	byte
xsi:noNamespaceSchemaLocation:	key	unsignedByte
xsi:nil:	keyref	decimal
nillable	length	float
processContents	list	double
ref	maxInclusive	boolean
schemaLocation	maxLength	duration
xsi:schemaLocation:	minInclusive	dateTime
substitutionGroup	minLength	date
targetNamespace	pattern	time
type	redefine	gYear
xsi:type	restriction	gYearMonth
use	schema	gMonth
xpath	selector	gMonthDay
	sequence	gDay
	simpleContent	Name
	simpleType	NCName
	union	anyURI
	unique	language
		ID
		IDREF
		IDREFS
		ENTITY
		ENTITIES
		NOTATION
		NMTOKEN
		NMTOKENS

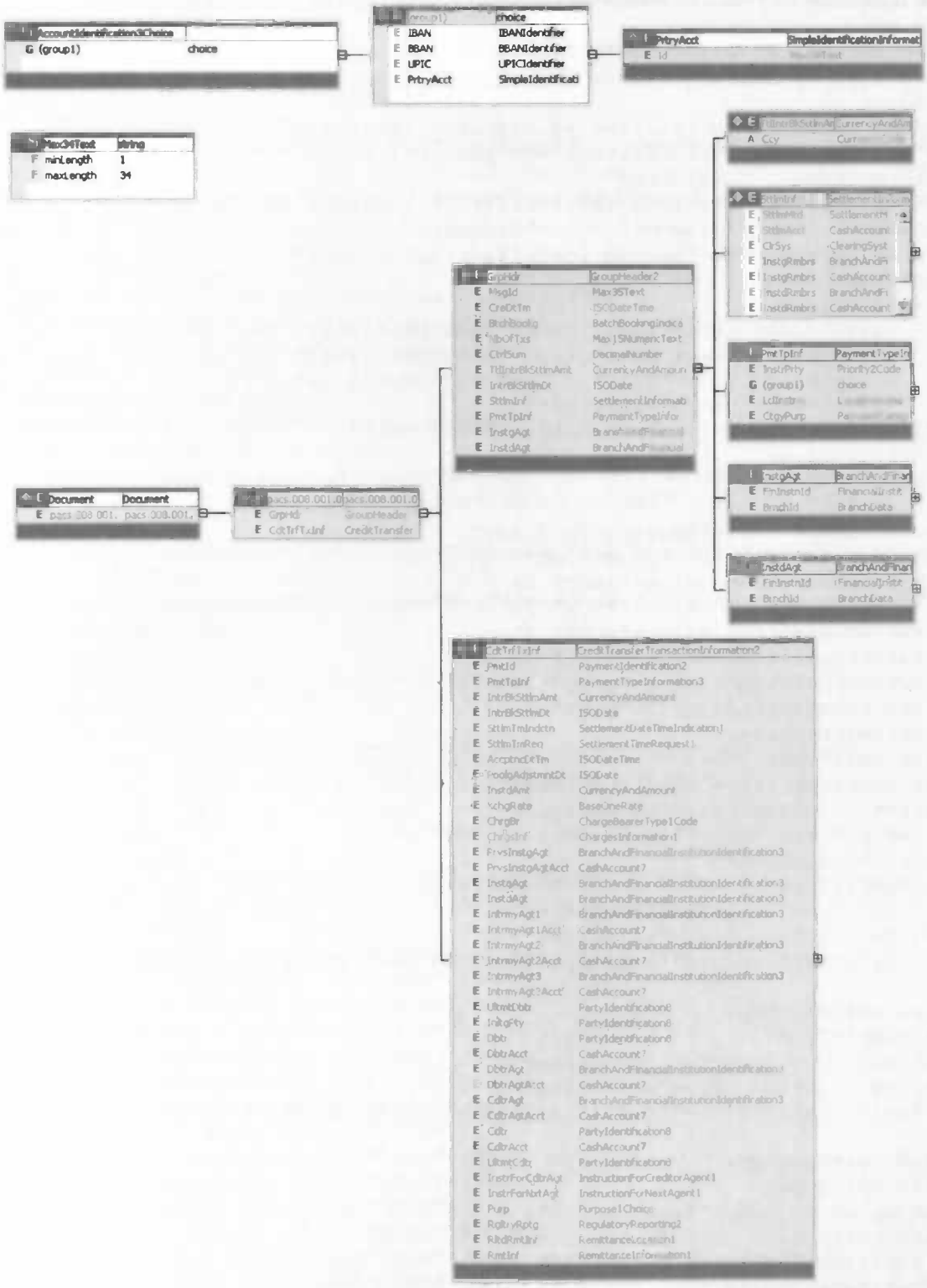
Source: [http://www.w3.org/TR/2004/REC-xmlschema-0-20041028]

Appendix 6: Activity diagram SingleCreditTransfer



The process of a single credit transfer between two payment clearing agents and a payment settlement agent. Source: [http://www.iso20022.org]

Appendix 7: UNIFI example (partial tree view)



Financial Institution To Financial Institution Customer Credit Transfer = pacs.008.001.01.xsd

Appendix 8: UNIFI example in XSD

Financial Institution To Financial Institution Customer Credit Transfer (pacs.008.001.01)

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Generated by SWIFTStandards Workstation (build:R5.1.0.4) on 2006 Sep 08
11:21:07-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="urn:iso:std:iso:20022:tech:xsd:pacs.008.001.01"
elementFormDefault="qualified"
targetNamespace="urn:iso:std:iso:20022:tech:xsd:pacs.008.001.01">
  <xs:element name="Document" type="Document" />
  <xs:complexType name="AccountIdentification3Choice">
    <xs:sequence>
      <xs:choice>
        <xs:element name="IBAN" type="IBANIdentifier" />
        <xs:element name="BBAN" type="BBANIdentifier" />
        <xs:element name="UPIC" type="UPICIdentifier" />
        <xs:element name="PrtryAcct"
type="SimpleIdentificationInformation2" />
      </xs:choice>
    </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="AddressType2Code">
    <xs:restriction base="xs:string">
      <xs:enumeration value="ADDR" />
      <xs:enumeration value="PBOX" />
      <xs:enumeration value="HOME" />
      <xs:enumeration value="BIZZ" />
      <xs:enumeration value="MLTO" />
      <xs:enumeration value="DLVY" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="BBANIdentifier">
    <xs:restriction base="xs:string">
      <xs:pattern value="[a-zA-Z0-9]{1,30}" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="BEIIdentifier">
    <xs:restriction base="xs:string">
      <xs:pattern value="[A-Z]{6,6}[A-Z2-9][A-NP-Z0-9]([A-Z0-9]{3,3}){0,1}"
/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="BICIdentifier">
    <xs:restriction base="xs:string">
      <xs:pattern value="[A-Z]{6,6}[A-Z2-9][A-NP-Z0-9]([A-Z0-9]{3,3}){0,1}"
/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="BaseOneRate">
    <xs:restriction base="xs:decimal">
      <xs:fractionDigits value="10" />
      <xs:totalDigits value="11" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="BatchBookingIndicator">
    <xs:restriction base="xs:boolean" />
  </xs:simpleType>
  <xs:complexType name="BranchAndFinancialInstitutionIdentification3">
    <xs:sequence>
```

```
      <xs:element name="FinInstnId"
type="FinancialInstitutionIdentification5Choice" />
      <xs:element name="BrnchId" type="BranchData" minOccurs="0"
maxOccurs="1" />
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="BranchData">
    <xs:sequence>
      <xs:element name="Id" type="Max35Text" minOccurs="0" maxOccurs="1" />
      <xs:element name="Nm" type="Max35Text" minOccurs="0" maxOccurs="1" />
      <xs:element name="PstlAdr" type="PostalAddress1" minOccurs="0"
maxOccurs="1" />
    </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="CHIPSUniversalIdentifier">
    <xs:restriction base="xs:string">
      <xs:pattern value="CH[0-9]{6,6}" />
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="CashAccount7">
    <xs:sequence>
      <xs:element name="Id" type="AccountIdentification3Choice" />
      <xs:element name="Tp" type="CashAccountType2" minOccurs="0"
maxOccurs="1" />
      <xs:element name="Ccy" type="CurrencyCode" minOccurs="0"
maxOccurs="1" />
      <xs:element name="Nm" type="Max70Text" minOccurs="0" maxOccurs="1" />
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="CashAccountType2">
    <xs:sequence>
      <xs:choice>
        <xs:element name="Cd" type="CashAccountType4Code" />
        <xs:element name="Prtry" type="Max35Text" />
      </xs:choice>
    </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="CashAccountType4Code">
    <xs:restriction base="xs:string">
      <xs:enumeration value="CASH" />
      <xs:enumeration value="CHAR" />
      <xs:enumeration value="COMM" />
      <xs:enumeration value="TAXE" />
      <xs:enumeration value="CISH" />
      <xs:enumeration value="TRAS" />
      <xs:enumeration value="SACC" />
      <xs:enumeration value="CACC" />
      <xs:enumeration value="SVGS" />
      <xs:enumeration value="ONDP" />
      <xs:enumeration value="MGLD" />
      <xs:enumeration value="NREX" />
      <xs:enumeration value="MOMA" />
      <xs:enumeration value="LOAN" />
      <xs:enumeration value="SLRY" />
      <xs:enumeration value="ODFT" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="CashClearingSystem3Code">
    <xs:restriction base="xs:string">
      <xs:enumeration value="ABE" />
      <xs:enumeration value="ART" />
```

```
<xs:enumeration value="AVP" />
<xs:enumeration value="AZM" />
<xs:enumeration value="BAP" />
<xs:enumeration value="BEL" />
<xs:enumeration value="BOF" />
<xs:enumeration value="BRL" />
<xs:enumeration value="CAD" />
<xs:enumeration value="CAM" />
<xs:enumeration value="CBJ" />
<xs:enumeration value="CHP" />
<xs:enumeration value="DKC" />
<xs:enumeration value="RTP" />
<xs:enumeration value="EBA" />
<xs:enumeration value="ELS" />
<xs:enumeration value="ERP" />
<xs:enumeration value="XCT" />
<xs:enumeration value="HRK" />
<xs:enumeration value="HRM" />
<xs:enumeration value="HUF" />
<xs:enumeration value="LGS" />
<xs:enumeration value="LVL" />
<xs:enumeration value="MUP" />
<xs:enumeration value="NOC" />
<xs:enumeration value="PCH" />
<xs:enumeration value="PDS" />
<xs:enumeration value="PEG" />
<xs:enumeration value="PNS" />
<xs:enumeration value="PVE" />
<xs:enumeration value="SEC" />
<xs:enumeration value="SIT" />
<xs:enumeration value="SLB" />
<xs:enumeration value="SPG" />
<xs:enumeration value="SSK" />
<xs:enumeration value="TBF" />
<xs:enumeration value="TGT" />
<xs:enumeration value="TOP" />
<xs:enumeration value="FDW" />
<xs:enumeration value="BOJ" />
<xs:enumeration value="FEY" />
<xs:enumeration value="ZEN" />
<xs:enumeration value="DDK" />
<xs:enumeration value="AIP" />
<xs:enumeration value="BCC" />
<xs:enumeration value="BDS" />
<xs:enumeration value="BGN" />
<xs:enumeration value="BHS" />
<xs:enumeration value="BIS" />
<xs:enumeration value="BSP" />
<xs:enumeration value="EPM" />
<xs:enumeration value="EPN" />
<xs:enumeration value="FDA" />
<xs:enumeration value="GIS" />
<xs:enumeration value="INC" />
<xs:enumeration value="JOD" />
<xs:enumeration value="KPS" />
<xs:enumeration value="LKB" />
<xs:enumeration value="MEP" />
<xs:enumeration value="MRS" />
<xs:enumeration value="NAM" />
<xs:enumeration value="PTR" />
<xs:enumeration value="ROL" />
```

```
      <xs:enumeration value="ROS" />
      <xs:enumeration value="SCP" />
      <xs:enumeration value="STG" />
      <xs:enumeration value="THB" />
      <xs:enumeration value="TIS" />
      <xs:enumeration value="TTD" />
      <xs:enumeration value="UIS" />
      <xs:enumeration value="MOS" />
      <xs:enumeration value="ZET" />
      <xs:enumeration value="ZIS" />
      <xs:enumeration value="CHI" />
      <xs:enumeration value="COP" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="ChargeBearerTypelCode">
    <xs:restriction base="xs:string">
      <xs:enumeration value="DEBT" />
      <xs:enumeration value="CRED" />
      <xs:enumeration value="SHAR" />
      <xs:enumeration value="SLEV" />
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="ChargesInformationl">
    <xs:sequence>
      <xs:element name="ChrgsAmt" type="CurrencyAndAmount" />
      <xs:element name="ChrgsPty"
type="BranchAndFinancialInstitutionIdentification3" />
    </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="ClearingChannel2Code">
    <xs:restriction base="xs:string">
      <xs:enumeration value="RTGS" />
      <xs:enumeration value="RTNS" />
      <xs:enumeration value="MPNS" />
      <xs:enumeration value="BOOK" />
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="ClearingSystemIdentificationlChoice">
    <xs:sequence>
      <xs:choice>
        <xs:element name="ClrSysId" type="CashClearingSystem3Code" />
        <xs:element name="Prtry" type="Max35Text" />
      </xs:choice>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="ClearingSystemMemberIdentification3Choice">
    <xs:sequence>
      <xs:choice>
        <xs:element name="Id" type="ExternalClearingSystemMemberCode" />
        <xs:element name="Prtry" type="Max35Text" />
      </xs:choice>
    </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="CountryCode">
    <xs:restriction base="xs:string">
      <xs:pattern value="[A-Z]{2,2}" />
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="CreditTransferTransactionInformation2">
    <xs:sequence>
      <xs:element name="PmtId" type="PaymentIdentification2" />
```

```
      <xs:element name="PmtTpInf" type="PaymentTypeInformation3"
minOccurs="0" maxOccurs="1" />
      <xs:element name="IntrBkSttlmAmt" type="CurrencyAndAmount" />
      <xs:element name="IntrBkSttlmDt" type="ISODate" minOccurs="0"
maxOccurs="1" />
      <xs:element name="SttlmTmIndctn" type="SettlementDateTimeIndication1"
minOccurs="0" maxOccurs="1" />
      <xs:element name="SttlmTmReq" type="SettlementTimeRequest1"
minOccurs="0" maxOccurs="1" />
      <xs:element name="AccptncDtTm" type="ISODateTime" minOccurs="0"
maxOccurs="1" />
      <xs:element name="PoolgAdjstmntDt" type="ISODate" minOccurs="0"
maxOccurs="1" />
      <xs:element name="InstdAmt" type="CurrencyAndAmount" minOccurs="0"
maxOccurs="1" />
      <xs:element name="XchgRate" type="BaseOneRate" minOccurs="0"
maxOccurs="1" />
      <xs:element name="ChrgBr" type="ChargeBearerTypeCode" />
      <xs:element name="ChrgsInf" type="ChargesInformation1" minOccurs="0"
maxOccurs="unbounded" />
      <xs:element name="PrvsInstgAgt"
type="BranchAndFinancialInstitutionIdentification3" minOccurs="0"
maxOccurs="1" />
      <xs:element name="PrvsInstgAgtAcct" type="CashAccount7" minOccurs="0"
maxOccurs="1" />
      <xs:element name="InstgAgt"
type="BranchAndFinancialInstitutionIdentification3" minOccurs="0"
maxOccurs="1" />
      <xs:element name="InstdAgt"
type="BranchAndFinancialInstitutionIdentification3" minOccurs="0"
maxOccurs="1" />
      <xs:element name="IntrmyAgt1"
type="BranchAndFinancialInstitutionIdentification3" minOccurs="0"
maxOccurs="1" />
      <xs:element name="IntrmyAgt1Acct" type="CashAccount7" minOccurs="0"
maxOccurs="1" />
      <xs:element name="IntrmyAgt2"
type="BranchAndFinancialInstitutionIdentification3" minOccurs="0"
maxOccurs="1" />
      <xs:element name="IntrmyAgt2Acct" type="CashAccount7" minOccurs="0"
maxOccurs="1" />
      <xs:element name="IntrmyAgt3"
type="BranchAndFinancialInstitutionIdentification3" minOccurs="0"
maxOccurs="1" />
      <xs:element name="IntrmyAgt3Acct" type="CashAccount7" minOccurs="0"
maxOccurs="1" />
      <xs:element name="UltmtDbtr" type="PartyIdentification8"
minOccurs="0" maxOccurs="1" />
      <xs:element name="InitgPty" type="PartyIdentification8" minOccurs="0"
maxOccurs="1" />
      <xs:element name="Dbtr" type="PartyIdentification8" />
      <xs:element name="DbtrAcct" type="CashAccount7" minOccurs="0"
maxOccurs="1" />
      <xs:element name="DbtrAgt"
type="BranchAndFinancialInstitutionIdentification3" />
      <xs:element name="DbtrAgtAcct" type="CashAccount7" minOccurs="0"
maxOccurs="1" />
      <xs:element name="CdtrAgt"
type="BranchAndFinancialInstitutionIdentification3" />
      <xs:element name="CdtrAgtAcct" type="CashAccount7" minOccurs="0"
maxOccurs="1" />
```

```
      <xs:element name="Cdtr" type="PartyIdentification8" />
      <xs:element name="CdtrAcct" type="CashAccount7" minOccurs="0"
maxOccurs="1" />
      <xs:element name="UltmtCdtr" type="PartyIdentification8"
minOccurs="0" maxOccurs="1" />
      <xs:element name="InstrForCdtrAgt"
type="InstructionForCreditorAgent1" minOccurs="0" maxOccurs="unbounded" />
      <xs:element name="InstrForNxtAgt" type="InstructionForNextAgent1"
minOccurs="0" maxOccurs="unbounded" />
      <xs:element name="Purp" type="Purpose1Choice" minOccurs="0"
maxOccurs="1" />
      <xs:element name="RgltryRptg" type="RegulatoryReporting2"
minOccurs="0" maxOccurs="10" />
      <xs:element name="RltdRmtInf" type="RemittanceLocation1"
minOccurs="0" maxOccurs="10" />
      <xs:element name="RmtInf" type="RemittanceInformation1" minOccurs="0"
maxOccurs="1" />
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="CreditorReferenceInformation1">
    <xs:sequence>
      <xs:element name="CdtrRefTp" type="CreditorReferenceType1"
minOccurs="0" maxOccurs="1" />
      <xs:element name="CdtrRef" type="Max35Text" minOccurs="0"
maxOccurs="1" />
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="CreditorReferenceType1">
    <xs:sequence>
      <xs:choice>
        <xs:element name="Cd" type="DocumentType3Code" />
        <xs:element name="Prtry" type="Max35Text" />
      </xs:choice>
      <xs:element name="Issr" type="Max35Text" minOccurs="0" maxOccurs="1"
/>
    </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="CurrencyAndAmount_SimpleType">
    <xs:restriction base="xs:decimal">
      <xs:minInclusive value="0" />
      <xs:fractionDigits value="5" />
      <xs:totalDigits value="18" />
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="CurrencyAndAmount">
    <xs:simpleContent>
      <xs:extension base="CurrencyAndAmount_SimpleType">
        <xs:attribute name="Ccy" type="CurrencyCode" use="required" />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:simpleType name="CurrencyCode">
    <xs:restriction base="xs:string">
      <xs:pattern value="[A-Z]{3,3}" />
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="DateAndPlaceOfBirth">
    <xs:sequence>
      <xs:element name="BirthDt" type="ISODate" />
      <xs:element name="PrvcOfBirth" type="Max35Text" minOccurs="0"
maxOccurs="1" />
```

```
      <xs:element name="CityOfBirth" type="Max35Text" />
      <xs:element name="CtryOfBirth" type="CountryCode" />
    </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="DecimalNumber">
    <xs:restriction base="xs:decimal">
      <xs:fractionDigits value="17" />
      <xs:totalDigits value="18" />
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="Document">
    <xs:sequence>
      <xs:element name="pacs.008.001.01" type="pacs.008.001.01" />
    </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="DocumentType2Code">
    <xs:restriction base="xs:string">
      <xs:enumeration value="MSIN" />
      <xs:enumeration value="CNFA" />
      <xs:enumeration value="DNFA" />
      <xs:enumeration value="CINV" />
      <xs:enumeration value="CREN" />
      <xs:enumeration value="DEBN" />
      <xs:enumeration value="HIRI" />
      <xs:enumeration value="SBIN" />
      <xs:enumeration value="CMCN" />
      <xs:enumeration value="SOAC" />
      <xs:enumeration value="DISP" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="DocumentType3Code">
    <xs:restriction base="xs:string">
      <xs:enumeration value="RADM" />
      <xs:enumeration value="RPIN" />
      <xs:enumeration value="FXDR" />
      <xs:enumeration value="DISP" />
      <xs:enumeration value="PUOR" />
      <xs:enumeration value="SCOR" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="DunsIdentifier">
    <xs:restriction base="xs:string">
      <xs:pattern value="[0-9]{9,9}" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="EANGLNIdentifier">
    <xs:restriction base="xs:string">
      <xs:pattern value="[0-9]{13,13}" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="ExternalClearingSystemMemberCode">
    <xs:restriction base="xs:string">
      <xs:minLength value="1" />
      <xs:maxLength value="35" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="ExternalLocalInstrumentCode">
    <xs:restriction base="xs:string">
      <xs:minLength value="1" />
      <xs:maxLength value="35" />
    </xs:restriction>
```

```
  </xs:simpleType>
  <xs:simpleType name="ExternalPurposeCode">
    <xs:restriction base="xs:string">
      <xs:minLength value="1" />
      <xs:maxLength value="35" />
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="FinancialInstitutionIdentification3">
    <xs:sequence>
      <xs:element name="BIC" type="BICIdentifier" minOccurs="0"
maxOccurs="1" />
      <xs:element name="ClrSysMmbId"
type="ClearingSystemMemberIdentification3Choice" minOccurs="0"
maxOccurs="1" />
      <xs:element name="Nm" type="Max70Text" minOccurs="0" maxOccurs="1" />
      <xs:element name="PstlAdr" type="PostalAddress1" minOccurs="0"
maxOccurs="1" />
      <xs:element name="PrtryId" type="GenericIdentification3"
minOccurs="0" maxOccurs="1" />
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="FinancialInstitutionIdentification5Choice">
    <xs:sequence>
      <xs:choice>
        <xs:element name="BIC" type="BICIdentifier" />
        <xs:element name="ClrSysMmbId"
type="ClearingSystemMemberIdentification3Choice" />
        <xs:element name="NmAndAdr" type="NameAndAddress7" />
        <xs:element name="PrtryId" type="GenericIdentification3" />
        <xs:element name="CmbndId"
type="FinancialInstitutionIdentification3" />
      </xs:choice>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="GenericIdentification3">
    <xs:sequence>
      <xs:element name="Id" type="Max35Text" />
      <xs:element name="Issr" type="Max35Text" minOccurs="0" maxOccurs="1"
/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="GenericIdentification4">
    <xs:sequence>
      <xs:element name="Id" type="Max35Text" />
      <xs:element name="IdTp" type="Max35Text" />
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="GroupHeader2">
    <xs:sequence>
      <xs:element name="MsgId" type="Max35Text" />
      <xs:element name="CreDtTm" type="ISODateTime" />
      <xs:element name="BtchBookg" type="BatchBookingIndicator"
minOccurs="0" maxOccurs="1" />
      <xs:element name="NbOfTxs" type="Max15NumericText" />
      <xs:element name="CtrlSum" type="DecimalNumber" minOccurs="0"
maxOccurs="1" />
      <xs:element name="TtlIntrBkSttlmAmt" type="CurrencyAndAmount"
minOccurs="0" maxOccurs="1" />
      <xs:element name="IntrBkSttlmDt" type="ISODate" minOccurs="0"
maxOccurs="1" />
      <xs:element name="SttlmInf" type="SettlementInformation1" />
```

```
      <xs:element name="PmtTpInf" type="PaymentTypeInformation3"
minOccurs="0" maxOccurs="1" />
      <xs:element name="InstgAgt"
type="BranchAndFinancialInstitutionIdentification3" minOccurs="0"
maxOccurs="1" />
      <xs:element name="InstdAgt"
type="BranchAndFinancialInstitutionIdentification3" minOccurs="0"
maxOccurs="1" />
    </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="IBANIdentifier">
    <xs:restriction base="xs:string">
      <xs:pattern value="[a-zA-Z]{2,2}[0-9]{2,2}[a-zA-Z0-9]{1,30}" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="IBEIIdentifier">
    <xs:restriction base="xs:string">
      <xs:pattern value="[A-Z]{2,2}[B-DF-HJ-NP-TV-XZ0-9]{7,7}[0-9]{1,1}" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="ISODate">
    <xs:restriction base="xs:date" />
  </xs:simpleType>
  <xs:simpleType name="ISODateTime">
    <xs:restriction base="xs:dateTime" />
  </xs:simpleType>
  <xs:simpleType name="ISOTime">
    <xs:restriction base="xs:time" />
  </xs:simpleType>
  <xs:simpleType name="Instruction3Code">
    <xs:restriction base="xs:string">
      <xs:enumeration value="CHQB" />
      <xs:enumeration value="HOLD" />
      <xs:enumeration value="PHOB" />
      <xs:enumeration value="TELB" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="Instruction4Code">
    <xs:restriction base="xs:string">
      <xs:enumeration value="PHOA" />
      <xs:enumeration value="TELA" />
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="InstructionForCreditorAgent1">
    <xs:sequence>
      <xs:element name="Cd" type="Instruction3Code" minOccurs="0"
maxOccurs="1" />
      <xs:element name="InstrInf" type="Max140Text" minOccurs="0"
maxOccurs="1" />
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="InstructionForNextAgent1">
    <xs:sequence>
      <xs:element name="Cd" type="Instruction4Code" minOccurs="0"
maxOccurs="1" />
      <xs:element name="InstrInf" type="Max140Text" minOccurs="0"
maxOccurs="1" />
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="LocalInstrument1Choice">
    <xs:sequence>
```

```
    <xs:choice>
      <xs:element name="Cd" type="ExternalLocalInstrumentCode" />
      <xs:element name="Prtry" type="Max35Text" />
    </xs:choice>
  </xs:sequence>
</xs:complexType>
<xs:simpleType name="Max140Text">
  <xs:restriction base="xs:string">
    <xs:minLength value="1" />
    <xs:maxLength value="140" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="Max15NumericText">
  <xs:restriction base="xs:string">
    <xs:pattern value="[0-9]{1,15}" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="Max16Text">
  <xs:restriction base="xs:string">
    <xs:minLength value="1" />
    <xs:maxLength value="16" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="Max256Text">
  <xs:restriction base="xs:string">
    <xs:minLength value="1" />
    <xs:maxLength value="256" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="Max34Text">
  <xs:restriction base="xs:string">
    <xs:minLength value="1" />
    <xs:maxLength value="34" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="Max35Text">
  <xs:restriction base="xs:string">
    <xs:minLength value="1" />
    <xs:maxLength value="35" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="Max3Text">
  <xs:restriction base="xs:string">
    <xs:minLength value="1" />
    <xs:maxLength value="3" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="Max70Text">
  <xs:restriction base="xs:string">
    <xs:minLength value="1" />
    <xs:maxLength value="70" />
  </xs:restriction>
</xs:simpleType>
<xs:complexType name="NameAndAddress3">
  <xs:sequence>
    <xs:element name="Nm" type="Max70Text" />
    <xs:element name="Adr" type="PostalAddress1" />
  </xs:sequence>
</xs:complexType>
<xs:complexType name="NameAndAddress7">
  <xs:sequence>
```

```
      <xs:element name="Nm" type="Max70Text" />
      <xs:element name="PstlAdr" type="PostalAddress1" />
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="OrganisationIdentification2">
    <xs:sequence>
      <xs:element name="BIC" type="BICIdentifier" minOccurs="0"
maxOccurs="1" />
      <xs:element name="IBEI" type="IBEIIdentifier" minOccurs="0"
maxOccurs="1" />
      <xs:element name="BEI" type="BEIIdentifier" minOccurs="0"
maxOccurs="1" />
      <xs:element name="EANGLN" type="EANGLNIdentifier" minOccurs="0"
maxOccurs="1" />
      <xs:element name="USCHU" type="CHIPSUniversalIdentifier"
minOccurs="0" maxOccurs="1" />
      <xs:element name="DUNS" type="DunsIdentifier" minOccurs="0"
maxOccurs="1" />
      <xs:element name="BkPtyId" type="Max35Text" minOccurs="0"
maxOccurs="1" />
      <xs:element name="TaxIdNb" type="Max35Text" minOccurs="0"
maxOccurs="1" />
      <xs:element name="PrtryId" type="GenericIdentification3"
minOccurs="0" maxOccurs="1" />
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="Party2Choice">
    <xs:sequence>
      <xs:choice>
        <xs:element name="OrgId" type="OrganisationIdentification2" />
        <xs:element name="PrvtId" type="PersonIdentification3"
minOccurs="1" maxOccurs="4" />
      </xs:choice>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="PartyIdentification8">
    <xs:sequence>
      <xs:element name="Nm" type="Max70Text" minOccurs="0" maxOccurs="1" />
      <xs:element name="PstlAdr" type="PostalAddress1" minOccurs="0"
maxOccurs="1" />
      <xs:element name="Id" type="Party2Choice" minOccurs="0" maxOccurs="1"
/>
      <xs:element name="CtryOfRes" type="CountryCode" minOccurs="0"
maxOccurs="1" />
    </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="PaymentCategoryPurpose1Code">
    <xs:restriction base="xs:string">
      <xs:enumeration value="CORT" />
      <xs:enumeration value="SALA" />
      <xs:enumeration value="TREA" />
      <xs:enumeration value="CASH" />
      <xs:enumeration value="DIVI" />
      <xs:enumeration value="GOVT" />
      <xs:enumeration value="INTE" />
      <xs:enumeration value="LOAN" />
      <xs:enumeration value="PENS" />
      <xs:enumeration value="SECU" />
      <xs:enumeration value="SSBE" />
      <xs:enumeration value="SUPP" />
      <xs:enumeration value="TAXS" />
```

```
      <xs:enumeration value="TRAD" />
      <xs:enumeration value="VATX" />
      <xs:enumeration value="HEDG" />
      <xs:enumeration value="INTC" />
      <xs:enumeration value="WHLD" />
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="PaymentIdentification2">
    <xs:sequence>
      <xs:element name="InstrId" type="Max35Text" minOccurs="0"
maxOccurs="1" />
      <xs:element name="EndToEndId" type="Max35Text" />
      <xs:element name="TxId" type="Max35Text" />
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="PaymentTypeInformation3">
    <xs:sequence>
      <xs:element name="InstrPrty" type="Priority2Code" minOccurs="0"
maxOccurs="1" />
      <xs:choice>
        <xs:element name="SvcLvl" type="ServiceLevel2Choice" minOccurs="0"
maxOccurs="1" />
        <xs:element name="ClrChanl" type="ClearingChannel2Code"
minOccurs="0" maxOccurs="1" />
      </xs:choice>
      <xs:element name="LclInstrm" type="LocalInstrument1Choice"
minOccurs="0" maxOccurs="1" />
      <xs:element name="CtgyPurp" type="PaymentCategoryPurpose1Code"
minOccurs="0" maxOccurs="1" />
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="PersonIdentification3">
    <xs:sequence>
      <xs:choice>
        <xs:element name="DrvrsLicNb" type="Max35Text" />
        <xs:element name="CstmrNb" type="Max35Text" />
        <xs:element name="SclSctyNb" type="Max35Text" />
        <xs:element name="AlnRegnNb" type="Max35Text" />
        <xs:element name="PsptNb" type="Max35Text" />
        <xs:element name="TaxIdNb" type="Max35Text" />
        <xs:element name="IdntyCardNb" type="Max35Text" />
        <xs:element name="MplyrIdNb" type="Max35Text" />
        <xs:element name="DtAndPlcOfBirth" type="DateAndPlaceOfBirth" />
        <xs:element name="OthrId" type="GenericIdentification4" />
      </xs:choice>
      <xs:element name="Issr" type="Max35Text" minOccurs="0" maxOccurs="1"
/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="PostalAddress1">
    <xs:sequence>
      <xs:element name="AdrTp" type="AddressType2Code" minOccurs="0"
maxOccurs="1" />
      <xs:element name="AdrLine" type="Max70Text" minOccurs="0"
maxOccurs="5" />
      <xs:element name="StrtNm" type="Max70Text" minOccurs="0"
maxOccurs="1" />
      <xs:element name="BldgNb" type="Max16Text" minOccurs="0"
maxOccurs="1" />
      <xs:element name="PstCd" type="Max16Text" minOccurs="0" maxOccurs="1"
/>
```

```
      <xs:element name="TwnNm" type="Max35Text" minOccurs="0" maxOccurs="1"
/>
      <xs:element name="CtrySubDvsn" type="Max35Text" minOccurs="0"
maxOccurs="1" />
      <xs:element name="Ctry" type="CountryCode" />
    </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="Priority2Code">
    <xs:restriction base="xs:string">
      <xs:enumeration value="HIGH" />
      <xs:enumeration value="NORM" />
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="Purpose1Choice">
    <xs:sequence>
      <xs:choice>
        <xs:element name="Cd" type="ExternalPurposeCode" />
        <xs:element name="Prtry" type="Max35Text" />
      </xs:choice>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="ReferredDocumentAmount1Choice">
    <xs:sequence>
      <xs:choice>
        <xs:element name="DuePyblAmt" type="CurrencyAndAmount" />
        <xs:element name="DscntApldAmt" type="CurrencyAndAmount" />
        <xs:element name="RmtdAmt" type="CurrencyAndAmount" />
        <xs:element name="CdtNoteAmt" type="CurrencyAndAmount" />
        <xs:element name="TaxAmt" type="CurrencyAndAmount" />
      </xs:choice>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="ReferredDocumentInformation1">
    <xs:sequence>
      <xs:element name="RfrdDocTp" type="ReferredDocumentType1"
minOccurs="0" maxOccurs="1" />
      <xs:element name="RfrdDocNb" type="Max35Text" minOccurs="0"
maxOccurs="1" />
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="ReferredDocumentType1">
    <xs:sequence>
      <xs:choice>
        <xs:element name="Cd" type="DocumentType2Code" />
        <xs:element name="Prtry" type="Max35Text" />
      </xs:choice>
      <xs:element name="Issr" type="Max35Text" minOccurs="0" maxOccurs="1"
/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="RegulatoryAuthority">
    <xs:sequence>
      <xs:element name="AuthrtyNm" type="Max70Text" minOccurs="0"
maxOccurs="1" />
      <xs:element name="AuthrtyCtry" type="CountryCode" minOccurs="0"
maxOccurs="1" />
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="RegulatoryReporting2">
    <xs:sequence>
```

```
      <xs:element name="DbtCdtRptgInd" type="RegulatoryReportingType1Code"
minOccurs="0" maxOccurs="1" />
      <xs:element name="Authrty" type="RegulatoryAuthority" minOccurs="0"
maxOccurs="1" />
      <xs:element name="RgltryDtls" type="StructuredRegulatoryReporting2"
minOccurs="0" maxOccurs="1" />
    </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="RegulatoryReportingType1Code">
    <xs:restriction base="xs:string">
      <xs:enumeration value="CRED" />
      <xs:enumeration value="DEBT" />
      <xs:enumeration value="BOTH" />
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="RemittanceInformation1">
    <xs:sequence>
      <xs:element name="Ustrd" type="Max140Text" minOccurs="0"
maxOccurs="unbounded" />
      <xs:element name="Strd" type="StructuredRemittanceInformation6"
minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="RemittanceLocation1">
    <xs:sequence>
      <xs:element name="RmtId" type="Max35Text" minOccurs="0" maxOccurs="1"
/>
      <xs:element name="RmtLctnMtd" type="RemittanceLocationMethod1Code"
minOccurs="0" maxOccurs="1" />
      <xs:element name="RmtLctnElctrncAdr" type="Max256Text" minOccurs="0"
maxOccurs="1" />
      <xs:element name="RmtLctnPstlAdr" type="NameAndAddress3"
minOccurs="0" maxOccurs="1" />
    </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="RemittanceLocationMethod1Code">
    <xs:restriction base="xs:string">
      <xs:enumeration value="FAXI" />
      <xs:enumeration value="EDIC" />
      <xs:enumeration value="URID" />
      <xs:enumeration value="EMAL" />
      <xs:enumeration value="POST" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="ServiceLevel1Code">
    <xs:restriction base="xs:string">
      <xs:enumeration value="SEPA" />
      <xs:enumeration value="SDVA" />
      <xs:enumeration value="PRPT" />
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="ServiceLevel2Choice">
    <xs:sequence>
      <xs:choice>
        <xs:element name="Cd" type="ServiceLevel1Code" />
        <xs:element name="Prtry" type="Max35Text" />
      </xs:choice>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="SettlementDateTimeIndication1">
    <xs:sequence>
```

```
      <xs:element name="DbtDtTm" type="ISODateTime" minOccurs="0"
maxOccurs="1" />
      <xs:element name="CdtDtTm" type="ISODateTime" minOccurs="0"
maxOccurs="1" />
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="SettlementInformation1">
    <xs:sequence>
      <xs:element name="SttlmMtd" type="SettlementMethod1Code" />
      <xs:element name="SttlmAcct" type="CashAccount7" minOccurs="0"
maxOccurs="1" />
      <xs:element name="ClrSys" type="ClearingSystemIdentification1Choice"
minOccurs="0" maxOccurs="1" />
      <xs:element name="InstgRmbrsmntAgt"
type="BranchAndFinancialInstitutionIdentification3" minOccurs="0"
maxOccurs="1" />
      <xs:element name="InstgRmbrsmntAgtAcct" type="CashAccount7"
minOccurs="0" maxOccurs="1" />
      <xs:element name="InstdRmbrsmntAgt"
type="BranchAndFinancialInstitutionIdentification3" minOccurs="0"
maxOccurs="1" />
      <xs:element name="InstdRmbrsmntAgtAcct" type="CashAccount7"
minOccurs="0" maxOccurs="1" />
      <xs:element name="ThrdRmbrsmntAgt"
type="BranchAndFinancialInstitutionIdentification3" minOccurs="0"
maxOccurs="1" />
      <xs:element name="ThrdRmbrsmntAgtAcct" type="CashAccount7"
minOccurs="0" maxOccurs="1" />
    </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="SettlementMethod1Code">
    <xs:restriction base="xs:string">
      <xs:enumeration value="INDA" />
      <xs:enumeration value="INGA" />
      <xs:enumeration value="COVE" />
      <xs:enumeration value="CLRG" />
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="SettlementTimeRequest1">
    <xs:sequence>
      <xs:element name="CLSTm" type="ISOTime" />
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="SimpleIdentificationInformation2">
    <xs:sequence>
      <xs:element name="Id" type="Max34Text" />
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="StructuredRegulatoryReporting2">
    <xs:sequence>
      <xs:element name="Cd" type="Max3Text" minOccurs="0" maxOccurs="1" />
      <xs:element name="Amt" type="CurrencyAndAmount" minOccurs="0"
maxOccurs="1" />
      <xs:element name="Inf" type="Max35Text" minOccurs="0" maxOccurs="1"
/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="StructuredRemittanceInformation6">
    <xs:sequence>
      <xs:element name="RfrdDocInf" type="ReferredDocumentInformation1"
minOccurs="0" maxOccurs="1" />
```

```
      <xs:element name="RfrdDocRltdDt" type="ISODate" minOccurs="0"
maxOccurs="1" />
      <xs:element name="RfrdDocAmt" type="ReferredDocumentAmount1Choice"
minOccurs="0" maxOccurs="unbounded" />
      <xs:element name="CdtrRefInf" type="CreditorReferenceInformation1"
minOccurs="0" maxOccurs="1" />
      <xs:element name="Invcr" type="PartyIdentification8" minOccurs="0"
maxOccurs="1" />
      <xs:element name="Invcee" type="PartyIdentification8" minOccurs="0"
maxOccurs="1" />
      <xs:element name="AddtlRmtInf" type="Max140Text" minOccurs="0"
maxOccurs="1" />
    </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="UPICIdentifier">
    <xs:restriction base="xs:string">
      <xs:pattern value="[0-9]{8,17}" />
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="pacs.008.001.01">
    <xs:sequence>
      <xs:element name="GrpHdr" type="GroupHeader2" />
      <xs:element name="CdtTrfTxInf"
type="CreditTransferTransactionInformation2" minOccurs="1"
maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

Appendix 9: UNIFI example in J#

```
/**Part of Financial Institution To Financial Institution Customer Credit
Transfer (pacs.008.001.01)*/

import System.Xml.Serialization.*;
/** @attribute System.CodeDom.Compiler.GeneratedCodeAttribute("xsd",
"2.0.50727.42")*/
/** @attribute System.SerializableAttribute()*/
/** @attribute System.Diagnostics.DebuggerStepThroughAttribute()*/
/** @attribute System.ComponentModel.DesignerCategoryAttribute("code")*/
/** @attribute
System.Xml.Serialization.XmlTypeAttribute(Namespace="urn:iso:std:iso:20022:
tech:xsd:pain.008.001.01")*/
/** @attribute
System.Xml.Serialization.XmlRootAttribute(Namespace="urn:iso:std:iso:20022:
tech:xsd:pain.008.001.01", IsNullable=false)*/
public class Document
{
    private pain00800101 pain00800101Field;
    /** @attribute
System.Xml.Serialization.XmlElementAttribute("pain.008.001.01")*/
    public pain00800101 get_pain00800101()
    {
        return this.pain00800101Field;
    }
    public void set_pain00800101(pain00800101 value) {
this.pain00800101Field = value; }
}
/** @attribute System.CodeDom.Compiler.GeneratedCodeAttribute("xsd",
"2.0.50727.42")*/
/** @attribute System.SerializableAttribute()*/
/** @attribute System.Diagnostics.DebuggerStepThroughAttribute()*/
/** @attribute System.ComponentModel.DesignerCategoryAttribute("code")*/
/** @attribute
System.Xml.Serialization.XmlTypeAttribute(TypeName="pain.008.001.01",
Namespace="urn:iso:std:iso:20022:tech:xsd:pain.008.001.01")*/
public class pain00800101
{
    private GroupHeader1 grpHdrField;
    private PaymentInstructionInformation2[] pmtInfField;
    public GroupHeader1 get_GrpHdr()
    {
        return this.grpHdrField;
    }
    public void set_GrpHdr(GroupHeader1 value) { this.grpHdrField = value;
}
    /** @attribute System.Xml.Serialization.XmlElementAttribute("PmtInf")*/
    public PaymentInstructionInformation2[] get_PmtInf()
    {
        return this.pmtInfField;
    }
    public void set_PmtInf(PaymentInstructionInformation2[] value) {
this.pmtInfField = value; }
}
/** @attribute System.CodeDom.Compiler.GeneratedCodeAttribute("xsd",
"2.0.50727.42")*/
/** @attribute System.SerializableAttribute()*/
/** @attribute System.Diagnostics.DebuggerStepThroughAttribute()*/
/** @attribute System.ComponentModel.DesignerCategoryAttribute("code")*/
```

```
/** @attribute
System.Xml.Serialization.XmlTypeAttribute(Namespace="urn:iso:std:iso:20022:
tech:xsd:pain.008.001.01")*/
public class GroupHeader1
{
    private String msgIdField;
    private System.DateTime creDtTmField;
    private String[] authstnField;
    private boolean btchBookgField;
    private boolean btchBookgFieldSpecified;
    private String nbOfTxsField;
    private System.Decimal ctrlSumField;
    private boolean ctrlSumFieldSpecified;
    private Grouping1Code grpgField;
    private PartyIdentification8 initgPtyField;
    private BranchAndFinancialInstitutionIdentification3 fwdgAgtField;
    public String get_MsgId() { return this.msgIdField; }
    public void set_MsgId(String value) { this.msgIdField = value; }
    public System.DateTime get_CreDtTm()
    {
        return this.creDtTmField;
    }
    public void set_CreDtTm(System.DateTime value) { this.creDtTmField =
value; }
    /** @attribute
System.Xml.Serialization.XmlElementAttribute("Authstn")*/
    public String[] get_Authstn() { return this.authstnField; }
    public void set_Authstn(String[] value) { this.authstnField = value; }
    public boolean get_BtchBookg() { return this.btchBookgField; }
    public void set_BtchBookg(boolean value) { this.btchBookgField = value;
}
    /** @attribute System.Xml.Serialization.XmlIgnoreAttribute()*/
    public boolean get_BtchBookgSpecified() { return
this.btchBookgFieldSpecified; }
    public void set_BtchBookgSpecified(boolean value) {
this.btchBookgFieldSpecified = value; }
    public String get_NbOfTxs() { return this.nbOfTxsField; }
    public void set_NbOfTxs(String value) { this.nbOfTxsField = value; }
    public System.Decimal get_CtrlSum()
    {
        return this.ctrlSumField;
    }
    public void set_CtrlSum(System.Decimal value) { this.ctrlSumField =
value; }
    /** @attribute System.Xml.Serialization.XmlIgnoreAttribute()*/
    public boolean get_CtrlSumSpecified() { return
this.ctrlSumFieldSpecified; }
    public void set_CtrlSumSpecified(boolean value) {
this.ctrlSumFieldSpecified = value; }
    public Grouping1Code get_Grpg()
    {
        return this.grpgField;
    }
    public void set_Grpg(Grouping1Code value) { this.grpgField = value; }
    public PartyIdentification8 get_InitgPty()
    {
        return this.initgPtyField;
    }
    public void set_InitgPty(PartyIdentification8 value) {
this.initgPtyField = value; }
    public BranchAndFinancialInstitutionIdentification3 get_FwdgAgt()
```

```
    {
        return this.fwdgAgtField;
    }
    public void set_FwdgAgt(BranchAndFinancialInstitutionIdentification3
value) { this.fwdgAgtField = value; }
}
/** @attribute System.CodeDom.Compiler.GeneratedCodeAttribute("xsd",
"2.0.50727.42")*/
/** @attribute System.SerializableAttribute()*/
/** @attribute
System.Xml.Serialization.XmlTypeAttribute(Namespace="urn:iso:std:iso:20022:
tech:xsd:pain.008.001.01")*/
public enum Grouping1Code {  SNGL,  GRPD,  MIXD }
/** @attribute System.CodeDom.Compiler.GeneratedCodeAttribute("xsd",
"2.0.50727.42")*/
/** @attribute System.SerializableAttribute()*/
/** @attribute System.Diagnostics.DebuggerStepThroughAttribute()*/
/** @attribute System.ComponentModel.DesignerCategoryAttribute("code")*/
/** @attribute
System.Xml.Serialization.XmlTypeAttribute(Namespace="urn:iso:std:iso:20022:
tech:xsd:pain.008.001.01")*/
public class PartyIdentification8
{
    private String nmField;
    private PostalAddress1 pstlAdrField;
    private Party2Choice idField;
    private String ctryOfResField;
    public String get_Nm() { return this.nmField; }
    public void set_Nm(String value) { this.nmField = value; }
    public PostalAddress1 get_PstlAdr()
    {
        return this.pstlAdrField;
    }
    public void set_PstlAdr(PostalAddress1 value) { this.pstlAdrField =
value; }
    public Party2Choice get_Id()
    {
        return this.idField;
    }
    public void set_Id(Party2Choice value) { this.idField = value; }
    public String get_CtryOfRes() { return this.ctryOfResField; }
    public void set_CtryOfRes(String value) { this.ctryOfResField = value;
}
}
/** @attribute System.CodeDom.Compiler.GeneratedCodeAttribute("xsd",
"2.0.50727.42")*/
/** @attribute System.SerializableAttribute()*/
/** @attribute System.Diagnostics.DebuggerStepThroughAttribute()*/
/** @attribute System.ComponentModel.DesignerCategoryAttribute("code")*/
/** @attribute
System.Xml.Serialization.XmlTypeAttribute(Namespace="urn:iso:std:iso:20022:
tech:xsd:pain.008.001.01")*/
public class PostalAddress1
{
    private AddressType2Code adrTpField;
    private boolean adrTpFieldSpecified;
    private String[] adrLineField;
    private String strtNmField;
    private String bldgNbField;
    private String pstCdField;
    private String twnNmField;
```

```
    private String ctrySubDvsnField;
    private String ctryField;
    public AddressType2Code get_AdrTp()
    {
        return this.adrTpField;
    }
    public void set_AdrTp(AddressType2Code value) { this.adrTpField =
value; }
    /** @attribute System.Xml.Serialization.XmlIgnoreAttribute()*/
    public boolean get_AdrTpSpecified() { return this.adrTpFieldSpecified;
}
    public void set_AdrTpSpecified(boolean value) {
this.adrTpFieldSpecified = value; }
    /** @attribute
System.Xml.Serialization.XmlElementAttribute("AdrLine")*/
    public String[] get_AdrLine() { return this.adrLineField; }
    public void set_AdrLine(String[] value) { this.adrLineField = value; }
    public String get_StrtNm() { return this.strtNmField; }
    public void set_StrtNm(String value) { this.strtNmField = value; }
    public String get_BldgNb() { return this.bldgNbField; }
    public void set_BldgNb(String value) { this.bldgNbField = value; }
    public String get_PstCd() { return this.pstCdField; }
    public void set_PstCd(String value) { this.pstCdField = value; }
    public String get_TwnNm() { return this.twnNmField; }
    public void set_TwnNm(String value) { this.twnNmField = value; }
    public String get_CtrySubDvsn() { return this.ctrySubDvsnField; }
    public void set_CtrySubDvsn(String value) { this.ctrySubDvsnField =
value; }
    public String get_Ctry() { return this.ctryField; }
    public void set_Ctry(String value) { this.ctryField = value; }
}
/** @attribute System.CodeDom.Compiler.GeneratedCodeAttribute("xsd",
"2.0.50727.42")*/
/** @attribute System.SerializableAttribute()*/
/** @attribute
System.Xml.Serialization.XmlTypeAttribute(Namespace="urn:iso:std:iso:20022:
tech:xsd:pain.008.001.01")*/
public enum AddressType2Code {  ADDR,  PBOX,  HOME,  BIZZ,  MLTO,  DLVY }

public class CreditorReferenceTypel { /**...*/}
public enum DocumentType3Code {  RADM,  RPIN,  FXDR,  DISP,  PUOR,  SCOR }
public class CreditorReferenceInformationl { /**...*/}
public class ReferredDocumentAmountlChoice { /**...*/}
public class CurrencyAndAmount { /**...*/}
public enum ItemChoiceType5 {  CdtNoteAmt,  DscntApldAmt,  DuePyblAmt,
RmtdAmt,  TaxAmt }
public class ReferredDocumentTypel { /**...*/}
public enum DocumentType2Code {  MSIN,  CNFA,  DNFA,  CINV,  CREN,  DEBN,
HIRI,  SBIN,  CMCN,  SOAC,  DISP }
public class ReferredDocumentInformationl { /**...*/}
public class StructuredRemittanceInformation6 { /**...*/}
public class RemittanceInformationl { /**...*/}
public class NameAndAddress3 { /**...*/}
public class RemittanceLocationl { /**...*/}
public enum RemittanceLocationMethodlCode {  FAXI,  EDIC,  URID,  EMAL,
POST }
public class TaxType { /**...*/}
public class TaxDetails { /**...*/}
public class TaxInformation2 { /**...*/}
public class StructuredRegulatoryReporting2 { /**...*/}
public class RegulatoryAuthority { /**...*/}
```

```
public class RegulatoryReporting2 { /**...*/}
public enum RegulatoryReportingType1Code { CRED, DEBT, BOTH }
public class Purpose1Choice { /**...*/}
public enum ItemChoiceType4 { Cd, Prtry }
public class AmendmentInformationDetails1 { /**...*/}
public class BranchAndFinancialInstitutionIdentification3 { /**...*/}
public class FinancialInstitutionIdentification5Choice { /**...*/}
public class ClearingSystemMemberIdentification3Choice { /**...*/}
public enum ItemChoiceType1 { Id, Prtry }
public class FinancialInstitutionIdentification3 { /**...*/}
public class GenericIdentification3 { /**...*/}
public class NameAndAddress7 { /**...*/}
public class BranchData { /**...*/}
public class CashAccount7 { /**...*/}
public class AccountIdentification3Choice { /**...*/}
public class SimpleIdentificationInformation2 { /**...*/}
public enum ItemChoiceType3 { BBAN, IBAN, PrtryAcct, UPIC }
public class CashAccountType2 { /**...*/}
public enum CashAccountType4Code { CASH, CHAR, COMM, TAXE, CISH,
TRAS, SACC, CACC, SVGS, ONDP, MGLD, NREX, MOMA, LOAN, SLRY, ODFT
}
public enum Frequency1Code { YEAR, MNTH, QURT, MIAN, WEEK, DAIL,
ADHO, INDA }
public class MandateRelatedInformation1 { /**...*/}
public class DirectDebitTransaction1 { /**...*/}
public class PaymentIdentification1 { /**...*/}
public class DirectDebitTransactionInformation1 { /**...*/}
public class PaymentTypeInformation2 { /**...*/}
public enum Priority2Code { HIGH, NORM }
public enum ClearingChannel2Code { RTGS, RTNS, MPNS, BOOK }
public class ServiceLevel3Choice { /**...*/}
public enum ServiceLevel2Code { SEPA, SDVA }
public class LocalInstrument1Choice { /**...*/}
public enum ItemChoiceType2 { Cd, Prtry }
public enum SequenceType1Code { FRST, RCUR, FNAL, OOFF }
public enum PaymentCategoryPurpose1Code { CORT, SALA, TREA, CASH,
DIVI, GOVT, INTE, LOAN, PENS, SECU, SSBE, SUPP, TAXS, TRAD, VATX,
HEDG, INTC, WHLD }
public enum ChargeBearerType1Code { DEBT, CRED, SHAR, SLEV }
public class PaymentInstructionInformation2 { /**...*/}
public enum PaymentMethod2Code { DD }
public class GenericIdentification4 { /**...*/}
public class DateAndPlaceOfBirth { /**...*/}
public class PersonIdentification3 { /**...*/}
public enum ItemChoiceType { AlnRegnNb, CstmrNb, DrvrsLicNb,
DtAndPlcOfBirth, IdntyCardNb, MplyrIdNb, OthrId, PsptNb, SclSctyNb,
TaxIdNb }
public class OrganisationIdentification2 { /**...*/}
public class Party2Choice { /**...*/}
```

Appendix 10: XML tools

Content Checker
The National Institute of Standards and Technology (NIST) made an online tool to validate XML. Many content standards are emerging today based on XML Schema. These specifications define semantics and structure for data to be exchanged between systems. However, in the interest of creating reusable standards, the specifications often do not capture the full range of semantics which will be needed in individual transactions.
As industry standards are increasingly built with flexibility to support users in various industrial sectors such as automotive, healthcare, formal semantics and structure requirements have been placed into separate layers of specification and some are delayed until the standard implementation. In addition, most popular schema languages do not provide sufficient expressiveness to support accurate and precise semantic expression.
An extensively sophisticated schema specification will be too complex a specification for implementers to effectively use. A simplistic schema specification, on the other hand, will be too loose and will allow for imprecise specifications. Nevertheless, the separation of lexicons, structures, and semantics of a content specification into layers positively affects the adoption of the standard.
The Content Checker seeks to complement this structure by providing a facility to precisely specify, extend, and test for conformance data being exchanged based on the semantics defined in an XML schema, or content standard.
(http://www.mel.nist.gov/msid/XML_testbed/Content_Checker.html).

Quality of Design (QOD) Tool
Another online tool of NIST is QOD, which enables users to test XSD. The XML Schema Quality of Design Tool assists in consistently using XML Schema for the specification of information. Consistent design of XML schemas within an organization or single integration project can reduce the number and the severity of interoperability problems. In addition, this consistency makes the XML schema easier to extend, understand, implement, and maintain; and, it paves the way for automated testing and mapping.
The purpose of QOD is to provide a prototypical environment for checking the XML schema design quality in a collaborative environment. In this version, there is no intention to create and maintain an extensive list of test cases, although the environment allows the user to do so. In the next version QOD will be extended to better support sharing and publication of design rules with the intention of maintaining an extensive set of test cases. The site http://www.w3.org/XML/Schema contains al whole list of XML tools. Some of them are worthwhile to mention and described below.
(http://syseng.nist.gov/b2bTestbed/projects/XMLVT/html/xmlvtWelcome.jsp?session=qod)

Castor, *Castor, Arnaud Blandin 2002-06-06.*
Castor provides the only open-source Schema Object Model that loads your XML Schema in a Java representation. It also generates Java classes given an XML Schema and performs validation.
(http://www.castor.org)

EDIFIX 5.5R, *New CCTS/UBL Data Modeling and Schema Generation Software Sylvia Webb 2005-02-05.*
The scalable product line GEFEG EDIFIXR supports all processes involved in the preparation of electronic business applications: e.g. deriving compatible XML schemata from existing traditional EDI (Electronic Data Interchang) guidelines.
The XML-UML EDIFIX is targeted for users who design professional business messages with data modeling technology and as the next step represent and further edit these data models in XML schemas.
(http://www.gefeg.com/en/index.htm)

ITCWorks
ITCWorks provides Java classes for handling XML Schemas. (http://www.i-techcorp.com)

***Java-X**, Java-X John R. Snyder, 2005-12-27.*
This Eclipse plug-in project is a code generator that outputs a W3C validated XML Schema Definition from any Java class. The plug-in is designed for the Eclipse 3.1 Java perspective.
(http://www.java-x.us/downloads.htm)

Javax.xml of J2SE
The validation API decouples the validation of an XML document from the parsing of the document. This allows Java technology to support multiple schema language
(http://java.sun.com/developer/technicalArticles/xml/validationxpath)

JaxFront
JAXFront is a java technology to render electronic forms on multiple UI channels (Java Swing, HTML, PDF) on the basis of an xml schema that acts as a business model. The dynamically generated GUIs & Forms allow the user a sophisticated way of editing XML data without being exposed to the underlying XML technology. (http://:www.jaxfront.com)

JBind
A Java-XML databinding framework that supports the complete XML schema with the exception of external parsed entities, Stefan Wachter, 2002-09-13.
JBind consists of a schema compiler that generates Java sources corresponding to a supplied schema. In addition it contains a schema validating XML parser. (http://jbind.sourceforge.net)

***KLEEN**, Graphical authoring tool KLEEN Berthold Daum, 2003-09-26.*
Asset Oriented Modeling (AOM) is an open conceptual modeling method combining Higher Order Entity Relationship Modeling with Hedge-Regular Grammars. The graphical tool KLEEN supports the creation and validation of asset oriented models. It can generate code in various formats such as XML Schema, Java, XMI, and SQL. KLEEN supports various refactoring methods for models and generated code, including the translations of non-deterministic structures into deterministic schemata. An Eclipse 3.0 - 3.2 SDK is needed to install and run KLEEN. (http://www.aomodeling.org)

***LINQ to XSD**, LINQ to XSD Preview Alpha 0.1 released to the web Ralf Lämmel, 2006-11-27.*
LINQ to XSD is the code name of an incubation project that aims to provide .NET developers with support for typed XML programming on top of LINQ to XML. While the LINQ to XML programmer operates on generic XML trees, the LINQ to XSD programmer operates on typed XML trees -- instances of .NET types that model the XML types of a specific XML schema (XSD). LINQ to XSD can be used whenever you have an XML schema available, or you are willing to infer a schema from the XML data at hand. LINQ to XSD is integrated into Visual Studio; so you just tag an XML schema as an 'LINQ to XSD schema', build your project, and the automatically derived object model is then part of your solution -- just as if XML schemas were .NET types. The derived object model enforces various validation constraints imposed by the underlying XML schema.
(http://blogs.msdn.com/xmlteam/archive/2006/11/27/typed-xml-programmer-welcome-to-linq.aspx)

***NetBeans Schema support**, Chris Webster 2006-06-25*
The XML schema tools available in the NetBeans Enterprise Pack 5.5 Early Access release allow you to visualize and edit XML schemas. The XML schema tools focus on complex information display and real-world use issues (for example, scalable visualization and editing of massive XML schemas). In addition, the XML schema tools reduce the complexity of creating and editing XML schemas, thus allowing someone who is not a schema expert to create and modify XML schema and other XML documents. Using the XML schema tools, you can reference external schemas and run advanced queries in the Analysis view.
The tool contains functionality to:
- Visualize and edit an XML schema in a scalable fashion using the Schema view

- Edit a schema via an abstract instance view which provides an editable visualization of the instance document structure
- Query and Visualize the Schema
- Perform refactoring across XML Schema, WSDL, and BPEL
(http://www.netbeans.org/products)

***MSXML**, Microsoft XML Core Services 4.0 SP1 (formerly known as MSXML)*
Support of the World Wide Web (W3) Consortium final recommendation for XML Schema, with both DOM and SAX.
Substantially faster XSLT engine. Our tests show about x4, and for some scenarios x8, acceleration.

***SchemaAgent**, Altova SchemaAgent 2007 Erin Cavanaugh 2006-11-17*
Altova SchemaAgent 2007 represents a new paradigm for modeling and managing advanced schemas and their components in workgroups. It allows you to view and manage XML Schema relationships in a visual way, and construct complex schemas from distributed schema elements using drag-and-drop functionality. In addition to XML Schema files, you can also see XML Schemas used as sources or targets in Altova MapForce data mappings.
(http://www.altova.com/products/schemaagent/xml_schema_management.html for details).

***Schema-Form**, Schema-Forms - XML based eForms program released KK Aw 2002-06-11*
Multicentric Technology is pleased to announce the release of Schema-Forms, an XML Schema based eForms program. Schema-Forms generate an eForm based on the XML-Schema and the constraints specified in the schema are applied during data entry. The output can be transformed to HTML with an XSLT document. (http://www.multicentric.com).

***Schema Viewer**, SchemaViewer 1.0, Frank Kilkelly 2002-11-08*
SchemaViewer 1.0 is a Java application that virtually eliminates tedious browsing of XML Schema documents by representing them as a easily navigatable hierarchical tree. The application is a Swing-based GUI. (http://www.geocities.com/frakilk/software.html)

***Syntext Serna**, Syntext Serna - Schema Validating WYSIWYG XML Editor V2.5.0 Syntext 2006-01-26*
The second release candidate of V2.5.0 from the 11th of January is announced to be V2.5.0 release.
Some of the software is granted for *free* only for non-profit activities.
(http://www.syntext.com/products/serna/relnotes.htm)

***SQC**, IBM XML Schema Quality Checker. Achille Fokoue 2003-07-26*
XML Schema Quality Checker (SQC) is a program which takes as input documents containing XML Schemas written in the W3C XML schema language and diagnoses improper uses of the schema language. Where the appropriate action to correct the schema is not obvious, the diagnostic message may include a suggestion about how to make the fix. (http://www.alphaworks.ibm.com/tech/xmlsqc)

***Stylus Studio**, Stylus Studio 2006 Release 3 Tony Lavinio 2006-06-13*
The release includes new XML tools, enhancements to Stylus Studio XML Deployment Components, and support for many data processing components for working with XQuery, XSL:FO, EDI and XSLT technologies.
Stylus Studio 2006 XML is an XML Integrated Development Environment (XML IDE) for working with XML, XQuery, XSLT, EDI, XML Schema/DTD, XPath, SQL/XML, XHTML, and Web services. A Convert-to-XML utility allows you to design your own adapters to use these technologies to manipulate non-XML files with either pre-packaged or your own XML tools. Dozens of OASIS catalogs containing XML Schemas and related files for standards such as DocBook, UBL, SOAP/WSDL, SVG, XHTML are also included. Maps for all current versions of EDIFACT and X12 are included in the box.
Stylus Studio incorporates tools from Saxonica (Saxon 6 and 8), Apache (Xerces-C++ and Xerces-J, and Xalan), Microsoft (MSXML and .Net), XSV, DISA/X12, and others, so schemas can be tested and

transformed across multiple platforms. XSLT and XQuery can be debugged and profiled against multiple engines, and connections to native XML databases provide links to where the data resides. After designing, testing and profiling the code, Stylus Studio helps to deploy it, by generating the source code. A license for Saxon SA is included for testing XSLT 1, XSLT 2 and XQuery in a real-world setting, in addition to the many other tools above. And two-way editing means that changes to the diagrams affect your code, and changes to your code affect the diagrams, all in real-time. (http://www.stylusstudio.com/xml_download.html)

Visual Studio 2005's XML Tools
These are the XML Editor and the XSLT Debugger (formerly known by the codename "Whidbey"). The XML editor includes the following functionality:

- *Design time well-formedness and validation errors.*
- *Validation support for Schema, DTD, and XDR.*
- *Inferring an XSD Schema from an XML instance.*
- *Converting a DTD or XDR to XSD Schema.*
- *Context-sensitive Intellisense.*
- *XSLT editing, viewing the results of the transform.*
- *Standard Visual Studio code-editing, such as outlining and commenting or un-commenting.*

The XSLT debugger comprises the following functionality:

- *Invoking the debugger from the XML Editor.*
- *The ability to set and remove breakpoints.*
- *Standard Visual Studio debugger function key and menu bindings (F9 for setting/removing breakpoints, F11 for "Step In," and so on).*
- *Viewing the output of the transform as it is being generated.*
- *Locals, Watch, and Call stack windows.*
- *Stepping into the XSLT from a C# (or any other CLR language) program.*

(http://msdn2.microsoft.com/en-us/library/aa302298.aspx)

***X2U**, Announcement for availability of X2U, Joerg Rieger, 2003-10-15*
Short description:
Existing XML editors still ignore the fact that users don't want to read XML markup. Our view is: Not users have to align to XML, but XML has to align to users.
Users fill out simple web forms, which are automatically derived from XML models. The XML model provides the XML structure and an interface (man-machine, machine-machine).
X2U accepts an XML document, an XML Schema, a DTD or an XML form (XForms). All you need is a web browser. Forms often restrict the contents an author wants to put in. The X2U approach is more flexible: Simply change the XML model to customize the forms. (http://www.lumrix.net/x2u)

***Xerces-J**, Xerces 2 Java Parser released lmartin@ca.ibm.com 2001-12-20*
This release now contains, among other features, full XML Schema support (with complete constraint checking). XML Schema support was redesigned and reimplemented in Xerces 2.
(http://xerces.apache.org/xerces2-j/index.html)

***XML Architect**, Sysonyx's xmlArchitect, Bryce Nielsen, 2002-04-26*
xmlArchitect is an intuitive XML Schema editor, displaying a realtime treeview mockup of what the Instance XML Document should look like while editing the XML Schema. Also, direct edits can happen inside this XML Tree and the XML Schema will update itself to reflect the new XML structure. Product Information is at [http://www.sysonyx.com/Products/xmlDraft/index.asp]; Downloadable at (http://www.sysonyx.com/Products/xmlDraft/downloads.asp)

***XML Beans**, Apache XMLBeans v2.1.0 available Radu Preotiuc-Pietro 2005-12-09*
The Apache XMLBeans team is pleased to announce the availability of Apache XMLBeans v2.1.0. XMLBeans is a complete XML processing solution for Java, including XML Schema support, XML

Schema to Java binding, lightweight fast Infoset access, XML Schema information access, DOM, SAX and StAX implementations, XPath/XQuery integration. Everything is available in a package that is easy to use and fully integrated, allowing the user to seamlessly switch between all APIs.
Binary downloads for Windows and Unix are available from
(http://xmlbeans.apache.org/sourceAndBinaries)

***XMLFox**, XMLFox Advanced XML/XSD editor Russ Sabitov 2005-03-14*
XMLFox Advance is an intuitive xml and xml schema (XSD) editor, allows the xml developer to create schemas and show a visual representation of what the xml document will look like for that schema. XMLFox Advance supports Validation an XML against an XSD schema.
(http://www.XMLFox.com)

***XML Infoset Browser**, XML Infoset Browser for Java, Ed Merks, 2002-09-18*
org.eclipse.xsd is a Java reference library that implements the XML Schema Infoset Model as described in the W3C XML Schema specifications. It can be used for any code that examines, creates, or modifies XML Schemas (standalone or as part of other artifacts, such as XForms or WSDL documents). The library provides an API for manipulating the components of an XML Schema, as well as an API for manipulating the DOM-accessible representation of XML Schema as a series of XML documents, and for keeping these representations in agreement as schemas are modified.
(Eclipse: http://www.eclipse.org/xsd)

***XML Schema Object Model** Xml Schema Object Model (in the System.Xml.Schema namespace), Microsoft 2002-06-03*
System.Xml.Schema namespace contains the XML classes that provide standards-based support for XML Schemas definition language (XSD) schemas. This namespace is part of the .Net Framework SDK http://msdn.microsoft.com/downloads/default.asp?url=/downloads/sample.asp?url=/msdn-files/027/000/976/msdncompositedoc.xml&frame=true and implements the XML Schema Part 1: Structures (http://www.w3.org/TR/xmlschema-1/) and the XML Schema Part 2: Datatypes (http://www.w3.org/TR/xmlschema-2/) specifications.

***XML Schema Validator**, Online W3C XML Schema Validator, Microsoft, 2002-10-11*
An online HTML form-based interface for validating schemas (XSD & XDR) and instance documents using the System.Xml.XmlValidatingReader in the .NET framework

XML Spye
XMLSpy is an integrated development environment (IDE) for XML from Altova. XMLSpy allows computer programmers to create XML-based and Web services applications. XMLSpy is also available as a plug-in for Microsoft Visual Studio and Eclipse.

***Xsbrowser**, xsbrowser v2.0 released Joerg Rieger 15 October 2001*
xsbrowser allows to navigate a DTD- or XML-Schema (rec-xmlschema-1-20010502) using a web browser. xsbrowser hides the document model's representation syntax and shows only the knowledge that is significant for the construction of valid documents.

XSV, XSV, an Open Source XML Schema Validator, with *web-form access* from University of Edinburgh/W3C.

Appendix 11: UNIFI Message components

Message components of Cash Management - Exceptions and Investigations (camt b); source: http://www.iso20022.org

Packages	AmountRange_AmountRangeBoundary	MessageReferences	Party
AccountIdentification	MemberIdentification	Date_DateTime	Error
PaymentInstruction	Queue	PaymentStatusReasonCode	Report
Query	System	MaxText	Account_&_Specialisations
Entry_&_Specialisations	PaymentStatus	Balance_&_Specialisations	PaymentTransaction
IdentificationInformation	IndividualPerson	PersonIdentification	NonFinancialInstitution
NonFinancialInstitutionIdentification	FinancialInstitutionIdentification	RemittanceInformation	RemittanceAmount
StructuredPostalAddress	PostalAddress	PaymentReject	SettlementInstruction
ServiceLevel	StructuredRemittanceInformation	CommunicationNumber	CurrencyExchange
FinancialInstrumentQuantity	InvestmentPlan	Organisation	OrganisationIdentification
DeliveryParameters	RoundingParameters	PaymentCard	PhoneAddress
SecuritiesAccount	SecuritiesSettlementParameters	SecuritiesSubBalance	StructuredPhoneAddress
PriceValue	CommissionWaiving	Cheque	CreditTransfer
DirectDebit	SecurityIdentification	Price	Technical
Commission	Charge	Tax	InvestmentAccount
OrganisationOrIndividual	RedemptionExecution	SubscriptionExecution	InvestmentFundOrderExecution
SecuritiesBalance	PaymentInstrument	RedemptionOrder	SubscriptionOrder
SwitchExecutionSubscriptionLeg	SwitchExecutionRedemptionLeg	SwitchRedemptionLeg	SwitchSubscriptionLeg
SwitchExecution	InvestmentFundSwitchOrder	CashAccount	ProprietaryElement
PaymentTransfer	Interest	InterestEntry	CashEntry
RiskManagementLimit	Liquidity	CashBalance	InformationQualifier
Limit	SystemBusinessInformation	Member	StandingOrder
SystemEventInformation	SystemClosureInformation	SystemStatus	SystemIdentification

NetAssetValueValuation	InvestmentFundClass	ValuationStatistics	SecuritiesTransfer
SettlementChain	FundsCashFlow	Funds Ex-Construct	AlternateSecurityIdentification
InvestmentFundTax	RegulatoryReporting	GroupInformation	SecuritiesQuantity
InvestigationCase	Adjustment	Goods	GoodsQuantity
GoodsTrade	Incoterms	Location	Product
Tolerance	Transport	TransportByAir	TransportByRail
TransportByRoad	TransportBySea	ChainParty	Reservation
PEPISAInvestmentPlan	Transfer	CalculationBasis	CancellationRight
CountryAndResidentialStatus	EUCapitalGain	FundOrder	AccountOwner
Allocation	AmortisedBond	AssetBackedFixedRateBond	ConvertibleBond
CrossTrade	Curve	DepositaryReceipt	DisclosedListTrading
Discretion	Equity	ExchangeForPhysicalTradeParameters	FixedRateBond
Future	IndicationOfInterest	InterestBearingInstrument	ListTrading
MarketParameters	NonDisclosedListTrading	Option	PrePaymentSpeed
QuoteVariables	Rating	Quote	RepurchaseAgreement
SecuritiesOrder	SecuritiesOrderParameters	SecuritiesPostTradeBooking	SecuritiesTrade
SecuritiesTradeTransaction	Security	Spread	TradingStrategy
Warrant	FinancialInstrument	Market	OrganisationList
VariableRateBond	ClearingSystemMemberIdentification	DirectDebitMandate	FinancialInstitution
PaymentType	FixingConditions	ForeignExchangeTradeAgreement	InvestmentFund
PaymentAgreement	TreasuryTradeNegotiation	Eligibility	Meeting
PowerOfAttorney	Proxy	Vote	CurrencyOption
PremiumCalculation	CashClearingSystemIdentification	CashClearingSystem	FinancialInstitutionPaymentTransaction
InstructionForAgent	TransactionReason	ProductIdentification	

Appendix 12: Interview

with H. Voskuilen and G. Aussems of NIBESVV/Ereon | 05-06-07 | Amsterdam

Introduction

ISO 20022 can be seen as the answer on the question: "what has been standardized for financial transactions in SEPA?".
Despite ISO 20022 is the result of a lot of conversations between people/organizations, it could be possible that the specification doesn't meet the whishes of all users (financial institutions). This could be caused by technical constraints, political reasons or something else.
The former leads to the question: "is the expression strength of XSD sufficient?".
To find an answer to this question, other questions can be asked, like:
"what should be standardized for financial transactions in SEPA?". Examples are 'from' and 'to' of the type string and 'account number' and 'amount' of the type integer. This question is hard to answer, because there are a lot of different kinds of transactions, each with a lot of attributes. Therefore the question should be more specific. The first three questions are UNIFI related and the other two questions are more related to SEPA in general.

Questions

1) What should be standardized different, than now the case is with ISO 20022?
   → Everything that is defined in the current situation comes back in the new situation. With ISO 20022 it has not been defined worse, but probably better.

2) What should be standardized different, than now the case is with the current situation?"
   → Someone of Equens or the Standard Evaluation Groups of ISO 20022 should know more about this.

3) Is it possible for organizations that implement the UNIFI standard to add new functionality/wishes, without being incompatible with UNIFI? Is it for example possible to add new attributes to the UNIFI standard?
   → Attributes can be defined, proposed and finally get added to the repository of ISO. For example attributes for financial transactions of oil companies (e.g. energy value). Users of UNIFI could use schemas with these attributes, but it will not be obligatory.

4) What are the biggest disadvantages of SEPA and for who is this a disadvantage?
   → It will not be a collection of best practices. Countries like the Netherlands with a low level of costs will encounter an increase of costs.
   →Existing investments will be written off earlier, which means a waste of capital.

5) Do you know more disadvantage of SEPA that are not listed in the table below?

Disadvantages	Consumers	Corporates / merchants	Banks
Longer account number (IBAN).	1) Higher chance to type a wrong symbol; 2) Higher charges and wrong payments for the ordering of customer; 3) Wrong IBAN will cause delayed payment or no payment at all for the beneficiary customer.		Higher chance to type a wrong symbol by clients could mean more work and cost for banks in case of more incorrect account numbers.
Investments have to be made		Presumably Points Of Sale devices have to be replaced.	1) Treasury/liquidity management has to be adapted to the new strategic situation; 2) Front/middle/back offices have to be adapted; 3) Payments processes have to be adapted for compliance.
More competition			1) Small banks have to

			outsource the processing of payments to bigger banks or have to quit; 2) EU12 banks could end up with 18 – 29 billion (38%-62%) less in the revenue of direct payments for the period after 2010.

Table 33: Overview of disadvantages

→ They will not succeed to assemble the best of all countries against a cost price of the most efficient party.

What do you mean with 'best'?

→ Quality, efficiency, performance, security, reliability, low costs. An example is the direct debit of the Netherlands. In the end the SEPA Direct Debit will be used.

6) What are the biggest advantages of SEPA and for who is this an advantage?
   → Cost savings on the corporate site
   → Harmonization
   → Financial institutions are able to lay in at large extent, which could mean an increase of quality (e.g. better performance and less costs)
   → Less tailored work has to be done.