AARMS Vol. 13, No. 1 (2014) 31–46.

Component based IT interoperability solutions, a novel approach

MUNK Sándor[1]

In our days the range of information activities supported by IT equipment, and the volume of information stored in IT systems is continually growing. So cooperation among different organizations is practically impossible without extensive, meaning preserving information exchange between their IT systems. Practically all today's interoperability solutions are based on a previously agreed intermediary representation (formatted message standard, or standard data elements), but these solutions have a number of limitations. This paper outlines the foundations of a novel, component based realization of IT interoperability solutions. For this reason it summarizes the foundations of IT interoperability, analyses the goals, and possibilities of component based solutions outlines an architecture of component based interoperability solutions, and finally determines basics of its components.

Introduction

Due to the evolution of information technology it is continually expanding the range of information activities; processes supported by IT equipment, as well as the volume of information stored in IT systems. Consequently cooperation among different organizations is already practically impossible, or at least not sufficiently effective without extensive information exchange (specifically data exchange) between the IT systems of these organizations. This necessitates that IT systems be able to exchange information without human intervention, preserving the meaning, that is to say be interoperable. The significance of IT systems' interoperability is particularly great in such complex systems of organisations as the defence sector (military forces, law enforcement, disaster management, etc.), public administration, as well as alliances and regional integrations of such spheres.

Interoperability between IT systems can be assured by dissolving the differences between the systems, and by meaning preserving transformations between data with different formats, contents, and interpretations. Practically all interoperability solutions of our days are based on previously agreed intermediary representations, formatted message standards, or standard data elements. Traditional IT interoperability solutions typically devolve the tasks of transformation between different information representations to the relevant IT systems, so the interoperability capabilities, functions embodied in these solutions do not become "public property". Another significant problem is, that the adaptation to emerging, and changing information exchange needs, the improvement of interoperability capabilities are limited, and time–consuming.

Dominant components of IT interoperability solutions are software applications, application components that implement meaning preserving transformations. The effective develop-

1 National University of Public Service, Faculty of Military Sciences and Officer Training, Budapest, Hungary, munk.sandor@uni-nke.hu

ment and maintenance of capabilities flexibly adapting to dynamically changing interoperability needs requires application of state–of–the–art IT development methods, and solutions. These include, among others, component based, service oriented, or middleware solutions. From an interoperability point of view the most important properties are: the utilization of available capabilities, the dynamic expandability, so it seems appropriate to research the application of component–based solutions.

The purpose of this paper is to outline a novel, component based realization of IT interoperability solutions. For this reason it:

- summarizes the foundations of interoperability, and IT interoperability, the types, limitations, and problems of IT interoperability solutions;
- presents the essence of component based software development, analyses its goals, and possibilities in IT interoperability solutions;
- finally outlines an architecture of component based interoperability solutions, and determines concepts, types and main attributes of its basic components.

IT interoperability solutions

Interoperability between IT systems is not an end in itself; its objective is to ensure the conditions of operational interoperability, and as a part of this, information interoperability of organizations, necessary for their effective cooperation. In practice today's solutions based on standardization increasingly face implementation barriers and problems. In the following (based fundamentally on [1], [2], [3], [4], [5]) we summarize the conceptual basics of IT interoperability, then we introduce the concept, and types of IT interoperability solutions, finally we analyse the limitations, and problems of these solutions related to characteristics of interoperability environments of our days.

Basics of interoperability

Different interpretations of ***interoperability***, a concept first appearing in the military sphere, agree that it is a relation, a mutual capability between or among two or more objects to support cooperation and interoperation. A fundamental type of interoperability is operational interoperability between active actors (organisations, forces, etc.), a relation between/among actors cooperating to achieve a common goal and the overall mutual capability necessary to ensure successful and efficient cooperation. The preconditions of cooperation and operational interoperability are different part–capabilities, such as interoperabilities on functional areas (command, and control, logistics, etc.), as well as information interoperability, and technical interoperability, which are the basis of any kind of interoperabilities. (see details in [1])

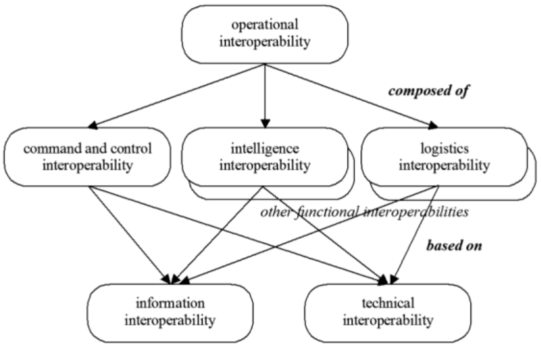
operational interoperability
composed of
command and control interoperability
intelligence interoperability
logistics interoperability
other functional interoperabilities
based on
information interoperability
technical interoperability

Figure 1. System of interoperability types [1: 129.]

Information interoperability is a key prerequisite for cooperation, for operational interoperability. A fundamental condition of successful and efficient operation of complex organisations, organisational systems, and groupings is the sufficient level of information exchange between the components, the sharing and coordinated exploitation of information necessary for cooperation. According to our interpretation, information interoperability is a mutual capability of different actors necessary to ensure exchange and common understanding of information needed for their successful cooperation. [1: 128.] This is made up of two basic components: the ability to exchange information, and the ability of the common understanding of information.

During *information exchange* one party (the sender) converts a portion of its knowledge into a form suitable for transfer; this representation will be transferred to the other party (the receiver), who interprets it, and builds into its knowledge. So in fact not the information itself is transmitted, but a representation, and an "other" information is created based on this representation. Common interpretation means that the intended meaning of the representation owned by the source — to the extent necessary for cooperation — equals the interpretation of the receiver.

Information interoperability requires different components, and capabilities. According to the *levels of abstraction of information representations* these components can be classified into three (technical, syntactic, semantic) levels. The technical level is related to the handling — creation, transmission, delivery — of material (physical) representations carrying information. The syntactic level is the level of the intermediary — perceptual or symbolic; traditional, or technical — representation, handling what is related to the languages, and formats used during information exchange. Finally the semantic level is the level of meaning carried by syntactic representations, the systems of concepts, and the knowledge representations used. To implement information interoperability it is necessary to ensure the conditions of meaning preserving information exchange on all the three levels.

IT interoperability has become an increasingly important, essential condition of information interoperability. Information exchange between cooperating actors increasingly happens without human assistance (machine to machine = M2M), by direct data exchange between the actors' IT systems. During exchange, and if necessary, transformation of data stored, handled in IT systems, it is necessary to ensure, that source and target data carry the same meaning, or to be more precise similar enough meaning for cooperation, for all the concerned actors. So IT interoperability is a mutual capability of IT systems, devices, applications to send, receive, exchange data — with possible transformations — preserving the meaning assigned by the primary user community. [2: 105.]

Basics of IT interoperability solutions

In case of IT systems (devices, applications, application components, hereinafter briefly IT systems) exchanging information with each other, an interoperability problem occurs, when at some level, there are differences, disagreement in the representations used, or in their interpretations. If the representations and their interpretations are the same, then nothing is to be done. However if there are differences, either the capability to use a shared, uniformly interpreted intermediary representation should be created, or the meaning preserving transformation between different representations should be realized during the process of information exchange (communications).

An ***IT interoperability solution*** is an IT system, device, application, or application component that's purpose is to resolve heterogeneity between disparate, heterogeneous IT systems, to ensure conditions of meaning preserving information exchange. Information exchange between two IT systems is always implemented with the help of a certain intermediary representation, which is carried by the communication network used. Meaning preserving transformation, the essential component of the IT interoperability solution, or parts of it can be implemented in relation to, as a part of the related systems, or independently from them.

The purpose if interoperability solutions, forming part of the systems, *interoperability interfaces, connectors, wrappers* is to implement the meaning preserving transformation between the system's internal, native representation, and the intermediary representation used. A given system may be connected to a more collaborative environment, so it can have more interoperability interfaces to different intermediary representations.

Interoperability solutions that are independent from the related systems — analogously to appropriate network devices — can be *interoperability gateways* performing transformations between intermediary representations used by the different collaborative environments. In addition to the centralized solution, the necessary transformations can be implemented in a distributed manner, in the form of an *interoperability infrastructure* based on the services of multiple components. Interoperability infrastructure can be a value–added service layer of the IT network, or an independent middleware layer. The advantage of the system independent solutions is that each system may use their own intermediary representation, and do not have to conform to other systems, to the changes in their range.

Today's IT interoperability solutions in practice are primarily standardization solutions. They are based on standards widely accepted, or agreed on by a community of interest for the different forms, components, and levels of information exchange that in the following we summarize based on [3]. The essential characteristic of interoperability solutions is based on global, or community of interest specific standards, established by formal agreements, or developed from practical experience, and the ability to exchange information by standardized solutions is the responsibility and task of the cooperating IT systems. Standardization of IT interoperability affects the three levels of interoperability to varying degrees. The technical level is characterized by general, widely used solutions, and this gradually covers the syntactic level. At the same time a semantic level dealing with content oriented questions is basically characterized by community of interest specific solutions.

Today the ***standardization solutions*** can be classified into three major groups, including document format standards, message format standards, and data element standards. *Interoperability solutions based on document format standards* standardize the formats of different (textual, spreadsheet, drawing, image, audio, and other multimedia) documents exchanged between IT systems. They do not address the issues of production, processing, transmission, and reception of these documents.

The basis of *interoperability solutions based on message format standards* is formatted messages belonging to semi–structured information. To meet the specific information exchange needs of a community of interest, standard messages are defined, which are based on standardized data elements, message fields, and standardized message structure processable by IT equipment. Message fields are standardized for each message, but can be re–used in more messages. Bit–oriented message formats are intended to support time–critical (real–time and near–real time) information exchange, while character–oriented message formats support the less or non–time critical information exchange.

Interoperability solutions are based on information exchange data models, which are based on the set of standardized data elements satisfying all the relevant information exchange needs of a community of interest. Data elements describing the characteristics and relationships of objects, which are subjects of information, are arranged into a single data model, where each data item can have only one version, specifying the content, format and possible values. These standardized data elements then can be used both in formatted messages and during database replication. Meaning preserving transformations between data models of the different IT systems and the common information exchange data model are the responsibility of the respective systems. Nowadays different communities of interest have designed and develop continuously a number of information exchange data models.[2]

Limitations and problems of IT interoperability solutions

During the examination of specialities of IT interoperability solutions, including their limitations and problems, one cannot abstract from the scope of interoperability to be ensured. Interoperability problems and the range of applicable solutions of a given IT system are determined by the characteristics of its interoperability environment. IT interoperability environment of a given IT system can be interpreted as all those IT systems (devices), that are in direct information exchange relation with the given system (without human interaction). IT interoperability environment includes information (or rather data carrying this information), handled, or exchanged by the given systems (see details in [4] and [5]).

Today's existing, as well as planned IT interoperability solutions mostly related to the so–called *elementary interoperability environment*, that is characterized by a well defined, permanent, close, functional area cooperation, and functional similarity of cooperating partners. These solutions are based on the same conceptual and methodological basis, the application of a *single common intermediary representation* (standardized messages, standardized data elements). Their implementation requires the following tasks:

- definition of information exchange needs of the given cooperation group;

2 Joint C3 Information Exchange Data Model (JC3IEDM), National Information Exchange Model (NIEM), Justice Information Exchange Model (JIEM), European Information Exchange Model (EIEM), Aeronautical Information Exchange Model (AIEM), etc.

- comparison, and reconciliation of the content (interpretation) and format of information by different actors, involved in information exchange;
- creation of the intermediary representation used in information exchange, and its agreed interpretation;
- finally the implementation of the necessary transformations between the internal representations of the different actors and the intermediary representation.

Functional similarity implies the identity or similarity of the scope, and content of information handled, which facilitates the establishment of the common intermediary representation, and the implementation of meaning preserving transformation between the internal and intermediary representations.

In order to maintain interoperability satisfying the requirements, in case of new, emerging information exchange needs, or changes in existing needs the above tasks should be cyclically repeated. The main characteristic of interoperability solutions based on common intermediary representations is the significant *turnaround time* (measured in half years, or years), which follows from the time required to reconcile changes, and on the other hand the implementation of appropriate modifications.

With the expansion of the scope of cooperating partners and the content of cooperation (the range of information exchanged) the implementation opportunities of elementary interoperability solutions are gradually narrowing. In case of a so–called *complex interoperability environment,* that is an extensive cooperation group and widespread information exchange with differentiated content, on most communities of interest the possibility to coordinate application domain specific versions of information decreases. Due to specific needs of different functional, application areas, standardization, and realization of uniform solutions is limited, particularly on a semantic level. This is because each community of interest has its own terminologies, which have concepts identical, or partly different to those used by others, as well as concepts unique to them. Consequently instead of a single intermediary representation *more,* complementary, existing parallel to each other *intermediary representations* are needed.

In a complex interoperability environment a given IT system is in connection with more IT systems that are members of more previously known communities of interest, and these communities develop their interoperability solutions independently of each other, or only a partially coordinated way. The number of these communities is usually few, rarely greater than 2–5. IT systems know, and use intermediary representations of more communities of interest ("they speak several languages"). Conditions of interoperability in this case can also be created in advance; the appropriate interoperability solution for the given system can be previously developed, and continuously maintained.

In a complex interoperability environment different intermediary representations in many cases form a multilevel, hierarchical system. The core of the system is a representation, which is shared by the whole cooperation (application) area, and ensures the exchange of information relevant for all, or the majority of cooperating actors. Some parts of intermediate representations related to specific communities of interest are common with the central representation (overlapping it, or are mapped to it), other parts are area–specific.

In case of *dynamic interoperability environment* a given IT system is also in connection with IT systems of more communities of interest, but their number is far higher than in the complex environment of interoperability, their range is dynamically changing, some of them appear only in relation to a specific task, operation. Consequently, unlike in the case of the two types mentioned before, conditions of interoperability previously can be ensured only partially, or only in the previously known areas of cooperation. In case of dynamically occurring cooperation areas of interoperability solutions should only be (fully) implemented and adapted to the given situation during the phase of the preparation, and/or the execution of the operation.

In dynamic interoperability environments implementation of interoperability requires other then pre–planned and prebuilt ways, and methods. To describe an adequate capability of an IT system in such an environment, a new concept should be introduced. *Adaptive interoperability* is a capability of an IT system to ensure the necessary conditions of the meaning preserving information exchange in a dynamically changing cooperation (interoperability) environment with other — previously known or unknown — IT systems, without IT development efforts, within user defined time limits. [4: 71.]

The *overall conclusion* is that currently used IT interoperability solutions based on intermediary representation provide an efficient solution only in case of well–defined, long–term and close cooperation. In addition their development and maintenance requires considerable coordination and time, they reaction to new information exchange requirements is difficult and slow. Finally the implementation of the necessary transformations between the internal and intermediary representations is fully passed on to affected systems.

Component–based approaches and interoperability solutions

In case of some level of heterogeneity of cooperating systems the conditions of interoperable (meaning preserving) information exchange between IT systems is implemented by IT applications realizing the transformation of the information flow. These applications, apart from simple cases, are complex software systems which realize a number of different — syntactic, semantic, and procedural — transformation functions. For effective and efficient software development different development approaches, methodologies have emerged, one of which is component–based software development. In the following (based essentially on [6], [7], [8] and [9]) we summarize the fundamentals of component–based software development, and then examine how this approach can be utilized, realized in case of interoperability solutions.

Basics of component–based software development

Component–Based Software Development (CBSD), or Component–Based Software Engineering (CBSE) is a software development methodology, based on the idea of an engineering approach, which states:

- do not invent, and do not fabricate everything again;
- build on reusable components;
- take, and adapt existing parts, components;
- standardize. [8]

This approach is based on the separation of software system development, and software components development. Accordingly, software systems should be built up (as far as possible) from prefabricated, existing components, and software components should be developed so that they can be used in different systems. This approach occupies an intermediate position between a completely individual development and the use of ready–made solutions.

The basic objective of component–based software development, fitting the line of former modular programming, structured programming, object–oriented programming, distributed software development, and other similar methodologies, is faster, more scalable product manufacturing, matching user needs; ease of modification and expandability; as well as reusability of partial results of development. Component–based software development difficulties include those not easy to find, learn or adapt to the appropriate components, and those where it is more difficult to make reusable components, then unique ones.

According to a widely accepted definition a *software component* "is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties." [7: 41.] Software components are cohesive units (from some, usually functional, point of view), implementing specified functionality, loosely coupled with their environment.

The concept of components are strongly connected to such general design, and development principles as decomposition, modularity, abstraction, and encapsulation. Effectively usable components can be developed by decomposition — that is division into more manageable parts — of complex systems (functions). During decomposition the intent is to increase the strength of internal relationships, and reduce external dependencies. For their users the components are abstractions of a particular function, or group of functions, whose quality significantly determines the usability of the component. Finally the components, by encapsulation, include data and procedures necessary to accomplish the functions provided, at the same time hiding the details from their users.

Software component interfaces are those means between the users and the component that enable the interconnection. Technically an interface is a set of operations that can be invoked by clients. Each operation's semantics is specified and this "contract" is normative for the users and for the developer of the component too. Specification of an interface is an abstract description of functionality; services provided by the component can be informal, formal, and mixed. A specification always contains functional aspects, usually includes pre– and postconditions of operations and possibly extra–functional elements (performance, capacity, availability, security information, and environmental requirements) too. [7: 50–57.]

Software component models and infrastructures (frameworks) are fundamental conditions of component–based software development. Component–based systems rely upon well–defined standards and conventions (component model) and a support infrastructure. A component model is a set of rules that determines the framework of a component's development (forms of communication, data representation, conditions of deployment, etc.), while a component infrastructure is a platform, on which the components 'work' and which provides different (e.g. communication, or resource management) services to the components. [9: 23–25.] In practice many different software component models and related infrastructures have evolved.[3]

In addition to dissimilarities, there are many similarities among *component–based approaches and service–oriented approaches* so popular nowadays. Both approaches are based on loosely coupled, interoperable, reusable components. However, compared with existing component–based solutions, service–oriented approaches took a step forward, because in fact they regulate only communication between components and do not contain provisions

3 Microsoft Component Object Model (COM), Microsoft .NET Framework, Enterprise Java Beans (EJB) and Java Platform Enterprise Edition (J2EE), Common Object Request Broker Architecture (CORBA).

for their language, and platforms. There are also differences in performance and security between components deployed in their environment and web services available through Internet protocols. Of course, in changing the interfaces, it is possible to implement the functionality of a software component as a web service, or provide a web service in the form of software component.

Basics of component based interoperability solutions

For the analysis of the application of the component based approach in interoperability solutions:

- first we overview the arguments for and against this approach and what opportunities exist;
- then we outline a basic framework of a component based interoperability solution.

The use of component based software solutions are justified, if on the given application area there exists *widely used functions, in many IT systems, products,* implementation of which, in the form of software components can be economical. This is obvious in the case of interoperability solutions, since most of the systems involved in information exchange use the same or similar information representations[4], and several types of data elements (temporal characteristics, spatial characteristics, names, etc.) used in many systems.

However this 'simple' condition in practice often does not appear to be sufficient. According to experience, the application of the theoretically suitable methodology is mostly bogged down in the business interests of software product manufacturers and in the diversity of component architectures. Software solutions implementing the same and similar functions are typically used only in various products of one manufacturer (and not necessarily in the form of software components), and not in products of different manufacturers (see e.g. different image and video format converters). The same is the case of existing interoperability solutions.

Another aspect of the use of component based solutions can be the *contribution to specific capabilities, properties of the software product.* The literature formulates a number of system properties (so called "ilities"), that can be easier, and reliably achieved with component–based solutions. These include among others extensibility, tailorability, and through these adaptability (to changes, changing needs).

In case of an appropriate support architecture, by replacing software components, or adding new components, capabilities, services of a component–based system — in a plug–and–play manner, similar to technical solutions used as examples — can be extended, enhanced without software development. In the world of interoperability solutions, due to dynamic changes in cooperating partners and their information exchange requirements, the new and changing message formats, and data elements, the significance of this option is invaluable. Limitations of the implementation are first of all the variety of component models and infrastructures, and the lack of a unified interoperability architecture.

Finally the reason for the component–based solutions can be the *special expertise needed to implement some functions.* In case of interoperability, in a lot of application areas there are (domain specific) knowledge intensive functions, where implementation of software requires

4 EDI, military message formats (Message Text Format, MTF), XML, etc.

not primarily IT, but domain specific expertise. Technical and syntactic level transformations usually require minimal knowledge of speciality. However semantic level transformations are basically based on speciality based knowledge, after that their IT implementation — a bit of exaggeration — is simple, almost mechanical. For example it is the domain experts' responsibility and capability to determine the order, and rules of transformations between different, but similar classification attributes, or between dates in different calendar systems. These domain area expertise functions should be implemented in the form of widely usable components.

The conclusion is that in case of interoperability solutions there exist a number of sub functions, which play an important role in many, or almost every system, so their implementation in form of reusable components could provide significant benefits. Without a detailed analysis these include for example the decomposition of structured information representations (e.g. formatted messages, tabular data, etc.) into elements and their relations, or the meaning preserving transformation between different formats of data elements. In case of interoperability solutions, due to dynamic changes in information exchange requirements and limitations of solutions based on preliminary coordination, standardization is of the utmost importance for the role of component–based solutions supporting extensibility, adaptability. Finally the independent implementation of interoperability components can be justified by the need of specialized domain knowledge.

The *basics of a framework for component–based interoperability solutions* can be built on functional decomposition, partition into functional components of a meaning preserving information exchange between IT systems. In our opinion two basic dimensions of decomposition are the abstraction levels, and the structural architecture of information representations.

It is easy to see how *transformations between technical (physical), syntactic and semantic representations* can lead to different functions, independent in nature, and may form individual components. In the following — although their role in specific interoperability solutions (e.g. tactical data links) is significant — we do not deal in detail with components handling physical representations. This is common in the interoperability literature: in case of analysis of interoperability solutions the lower levels of information exchange are usually considered solved, and a communication infrastructure is assumed, that provides an existing bit– or character–oriented channel for information exchange between cooperating systems.

It is also fairly clear that a meaning preserving transformation of a complete information exchange unit (document, message, query, response, data stream, etc.) can be decomposed into *transformations of composite and elementary information units*. Software structure determined by data structure, object–structure has long been used and is a still prevailing approach of software development. From our point of view it is important to emphasize that individual information exchange units typically are made up of parts used more than once, and these parts are built from often used elementary units. Thus software components implementing transformations of intermediate and elementary level representational units may be ideal for multi–use (reusable) building blocks. Hereinafter — due to reasons of brevity and differences in characteristics — we do not deal in detail with the issues of interoperable transformations of natural language texts and multimedia representations, our discussions will be narrowed to structured and semi–structured data, information representations.

In the following we outline *a comprehensive model of interoperability solutions* that could form the framework for the analysis of the structure and components of component–based solutions.

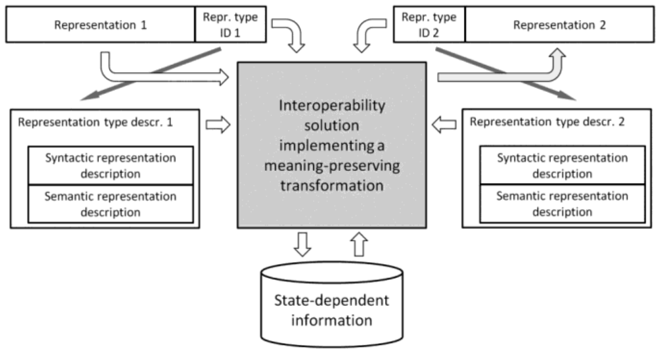
Representation 1
Repr. type ID 1
Repr. type ID 2
Representation 2
Representation type descr. 1
Syntactic representation description
Semantic representation description
Interoperability solution implementing a meaning-preserving transformation
Representation type descr. 2
Syntactic representation description
Semantic representation description
State-dependent information

Figure 2. Overall model of interoperability solutions
(Created by the author)

The fundamental input of the interoperability solution — whether it is centralized or distributed, or part of the system(s) involved, or independent from them — is a given type of information representation (belonging to source side), and its fundamental output is another type of information representation (belonging to received side). For processing, and the creation of particular representations, syntactic and semantic descriptions are needed that can be provided, published by the involved parties, or may be generated by the interoperability solution providers.

In many cases permanent specifications are not sufficient for meaning preserving transformation; information from previous process of information exchange may also be needed. For the period of the information exchange, this information should be temporarily kept by the interoperability solution.

Architecture of component based interoperability solutions, interoperability functional units

As outlined in the previous section the basic function of an interoperability solution is to transform a given type of information representation to another type of information representation. In case of a component–based implementation to achieve this goal a comprehensive architecture and its constituent component types must be defined, which form the basis of the development of components fitting into this framework, and usable in several solutions. In the following we first outline an interoperability solution architecture, and then determine the essential features of its functional units.

Architecture of component based interoperability solutions

The functional architecture of interoperability solutions can be determined according to the three phases of transfer–based solution of computer (machine) translation. Its essence is that:

- with morphological, syntactic and semantic analysis the source text is transformed into a language–specific intermediate representation;
- source language intermediate representation is transformed into a target language intermediate representation;

- finally target language text is generated based on the target language intermediate representation.

Since our study is focused on the meaning preserving transformation of structured and semi–structured data (information representation) and the goal (requirement) is a 'perfect' solution, without the loss of information due to application circumstances, statistical translation techniques – considered nowadays the most popular – based on substantial preliminary data collection are not feasible. It can be easily stated that analysis (decomposition into component parts) and generation (building from component parts) of structured and semi–structured information representations, in relation to the processing of natural language texts, are practically not a problem. Based on the foregoing a proposed overall architecture of component–based interoperability solutions is shown in the following figure:

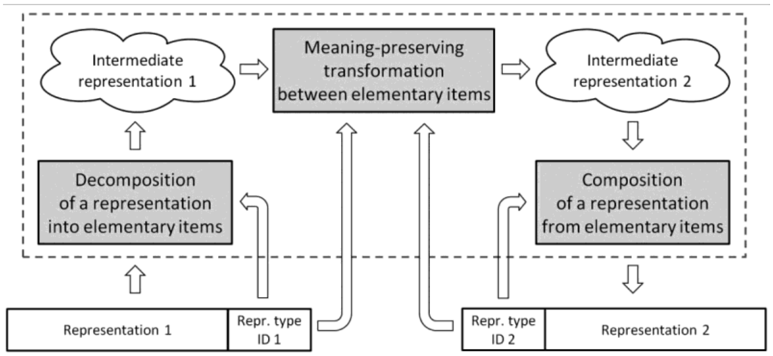
Intermediate representation 1
Meaning-preserving transformation between elementary items
Intermediate representation 2
Decomposition of a representation into elementary items
Composition of a representation from elementary items
Representation 1
Repr. type ID 1
Repr. type ID 2
Representation 2

Figure 3. Overall architecture of an interoperability solution (Created by the author)

Key components of the architecture are the *two intermediate representations* that form a connecting link between three functional units. The required degree standardization of these representations is the basic prerequisite of the component–based solution. To facilitate further transformations intermediate representations carry the information content of source and target representations divided into elementary components, in form of elementary statements related to individuals (objects), their characteristics (properties), and relationships. In case of structured and semi–structured data any knowledge representation language[5] suitable for description of propositional (statement oriented) information is appropriate for intermediate representation. According to the literature these knowledge representation types can be mapped to each other [e.g. 10, 11], however in this paper we are not dealing with this in detail, conversions between different types we consider solved by general purpose components.

Subtasks of an interoperability solution are implemented by three broad functional units. Two of these, *decomposition of the source representation into elementary components and composition of target representation from elementary components* are related to the system of interpretation (conceptual system) and format rules of a given side; they are independent of the features, characteristics of the other side. In case of structured (tabular) and semi–structured data (e.g. formatted messages) realization of the two transformations, based on

5 Resource Description Format ~ W3C Recommendation, Topic Maps ~ ISO 13250, Conceptual Graphs ~ ISO 24707.

database schemas, and message formats, is not a significant problem, it largely encompasses commonly used sub functions.

The third functional unit, *transformation between intermediate representations of the source and target contexts* is the essential component of the meaning preserving transformation. This functional unit resolves the differences in concepts, interpretations, and formats of the two contexts. Based on the needs of the application area, the transformation can be achieved in two ways:

- complete (as possible) transformation of source information into target information;
- production of required target information from the source information available.

Of the two, the second approach is more difficult, because its implementation — in case of source information is not sufficient — may require additional information. These may be available in the source environment (but outside of the source representation), or can be acquired from a general knowledge base, or a knowledge base established specifically for interoperable transformations.

Transformation between intermediate representations can be divided at *syntactic and semantic levels* and corresponding functional units. In practice syntactic transformations are conversions between data values appearing in elementary information, representing the value of a given characteristic of a given object. In this paper we consider a transformation as syntactic level transformation when it takes place between two different representations of the same data element concept[6]. They also include conversions between composite representations (e.g. dates, quantities). However we do not consider as syntactic the transformations between classification characteristics represented by different codes, since actually they are transformations between class concepts.

Semantic transformations are necessary when there are conceptual differences between the two parties, such as case transformations between the concepts – including object, characteristic, and relationship concepts – of the two interpretation environment required. Transformations between concepts and between data values sometimes are connected: to determine object concept data values (classification, or other characteristics) may be necessary, and data values (e.g. classification characteristics) may be determined based on object concepts.

Component based interoperability functional units

Basic components of interoperability solutions' functional architecture outlined on the previous section are interoperability functional units. A functional unit is a basic concept of IT: an entity of hardware or software, or both, capable of accomplishing a specified purpose. [13: 11.] Accordingly an *interoperability functional unit* is a hardware and/or software entity whose purpose is to accomplish meaning preserving transformations between different information representations.

Interoperability functional units may be IT application/device components, standalone IT applications/devices, and complex IT systems built on each other and other type functional units. For extensive usability, the interface of the interoperability functional units can be generalized in such a way that their input and output consist of bit sequence format information

6 Concept that can be represented in the form of a data element, described independently of any particular representation. A data element concept is composed of an object class [concept] and a property [concept]. [12: 5., 10.]

representation(s), and globally unique identifiers defining their type. Specifications, knowledge components of this representations (defined by the unique identifiers) may appear, may be available 'built in' to functional units, or independently, in some form of knowledge representation.

The following provides a broad, one by one overview of the basic characteristics, features of functional units implementing decomposition, composition, and transformation, and possibilities of their implementation in the form of reusable components.

The basic purpose of *decomposition and composition functional units* is to create elementary information representations from a composite information representation, and to build a desired (composite) information representation from elementary representations. During decomposition data values appearing in representations are extended with data type identifier, and object–, property–, and relationship–type concepts (more accurately with their type identifiers). In case of structured and semi–structured information representations in practice we encounter only a few structure types. These include tabular (matrix), and graph (basically tree) data structures.

In case of decomposition and composition functional units, the most appropriate sub functions to implement as widely reusable components are sub functions that implement syntactic analysis of the given representation[7] (analysing a sequence of symbols according to the rules of a formal grammar, and building a language data structure), and sub functions of syntactic generation (building sequences of symbols from a language data structure).[8] In case of general purpose components, for implementation, it is necessary to provide a standardized formal specification (database schema, or message format specification) of the representation structure.

In case of formatted messages another obvious possibility for component–based implementation is the decomposition of messages into message parts, and composition of messages from message parts. One of the basic objectives of major message format standards is the individual standardization of reusable message parts, and data elements. So in case of a new message, to create an interoperable transformation it is just enough to implement the decomposition, and composition functional units for the new message parts.

The purpose of *syntactic transformation units* accomplishing transformations between source and target intermediate representations is the meaning preserving conversion between different representations of the same data element concepts. This includes transformations between different formats of numbers, dates, times, and spatial characteristics, or character set conversions between textual characteristics. These are already available in the form of general purpose programs and subroutines. In fact their component–based implementation is only a question of implementing a suitable interface; it is not a major technical problem.

Transformations between textual (character string) data having internal structure, e.g. person names[9], and organization, organizational unit names, including their different language versions form a more interesting, and more significant task. To solve these tasks by component–based functional units, deeper level knowledge components are needed. Due to the need of the identification of objects that are of interest, meaning preserving exchange of name features (appropriate for identification) between different interpretation contexts is crucial. (in detail see [14]).

7 Parsing.
8 We can meet such solutions (XML parser, MTF parser, etc.) in the practice.
9 Surname (family name), given name, titles, other name parts, etc.

Purpose of *semantic transformation units* accomplishing transformations between source and target intermediate representations is the meaning preserving transformation between the conceptual systems (object, property, and relationship concepts). The background of this task is provided by the alignment, matching, and mapping of ontologies explicating these conceptual systems. This is obviously easier in case of interpretational contexts closer to each other, and much more difficult in case of more different ones.

Semantic transformation units, by concept categories, may be classified into object concept, property concept, and relationship concept transformation units. These include: organization concept, activity concept, part–whole concept, etc. transformation units. For transformation of a given concept, in addition to source and target ontologies, and the concept itself, additional information is available in the intermediate representation may be needed.

Transformations between source and target concepts are not always feasible clearly, with complete accuracy, e.g. in case when there are differences between levels of detail of the two conceptual systems. Moreover conceptual differences between application requirements or different points of view may also make it difficult, or prevent the ‘transmission’ of meaning. As a consequence semantic transformations are the most difficult parts of meaning preserving transformations.

Conclusion

As a conclusion it can be stated that information interoperability is a key prerequisite for cooperation, for operational interoperability. Moreover IT interoperability is an increasingly important, essential condition of information interoperability, a mutual capability of meaning preserving information exchange between IT systems of cooperating actors without human assistance.

Existing IT interoperability solutions are primarily standardization solutions that can be classified into three major groups, including document format standards, message format standards, and data element standards. Currently used IT interoperability solutions based on previously agreed intermediary representations provide an efficient solution only in case of well–defined, long–term and close cooperation. In addition their development and maintenance requires considerable coordination and time, their reaction to new information exchange requirements is difficult and slow.

Component–based software development is an up–to–date approach of software development that is based on the separation of software system development, and software components development. Its principle is that software systems should be built up (as far as possible) from prefabricated, existing components, and software components should be developed so that they can be used in different systems.

The application of component–based software solutions is justified first of all when on the given application area there exists widely used functions, in many IT systems and products. Another aspect of the use of component–based solutions can be the contribution to specific capabilities, properties of the software product (e.g. extensibility, tailorability, and through these adaptability). Finally the reason for the component–based solutions can be the special expertise needed to implement some functions.

The basics of a framework for component–based interoperability solutions can be built on functional decomposition, partition into functional components of a meaning preserving information exchange between IT systems. In our opinion two basic dimensions of decomposi- tion are the abstraction levels, and the structural architecture of information representations. The functional architecture of interoperability solutions can be determined according to the three phases of transfer–based solution of computer (machine) translation.

Basic components of interoperability solutions' functional architecture are interopera- bility functional units, hardware and/or software entities whose purpose is to accomplish meaning preserving transformations between different information representations. These have more widely used sub functions which may be effectively and efficiently implemented in form of reusable components.

References

[1] MUNK S.: An analysis of basic interoperability related terms, system of interoperability types. *Academic and Applied Research in Military Science*, I 1 (2002), 117–131.

[2] MUNK S.: Necessity of a military IT interoperability infrastructure. *Revista Academiei Fortelor Aeriene*, V 1 (2007), 104–109.

[3] MUNK S.: *Katonai informatikai rendszerek interoperabilitásának aktuális hadtudományi kérdései.* MTA doktori értekezés, Budapest: MTA, 2007.

[4] MUNK S.: Concept and necessity of adaptive interoperability in case of military IT systems. *Science & Military*, I 1 (2006), 68–73.

[5] MUNK S.: Changes in the military information interoperability environment. *Revista Academiei Forţelor Terestre*, X 4 (2005), 39–51.

[6] HEINEMAN, G. T., COUNCILL, W. T. (eds.): *Component Based Software Engineering: Putting the Pieces Together.* Boston: Addison–Wesley, 2001.

[7] SZYPERSKI, C.: *Component Software: Beyond Object Oriented Programming.* 2nd Ed. Pearson Education, 2002.

[8] CHARAF, H. KONDOROSI K., LÁSZLÓ Z.: A komponens-technológia néhány aktuális kérdése. In. *Szoftvertechnológiai Fórum, Komponens alapú szoftverfejlesztés, NJSZT*. Budapest, 2003.12.04. www.inf.u-szeged.hu/stf/slides/e12.ppt (downloaded: 20 05 2013)

[9] BACHMANN, F., BASS, L., BUHMAN, C., COMELLA-DORDA, S., LONG, F., ROBERT, J., SEACORD, R., WALLNAU, K.: *Technical Concepts of Component-Based Software Engineering. Technical Report.* Pittsburgh: Carnegie Mellon University, 2000.

[10] LACHER, M. S., DECKER, S.: On the Integration of Topic Maps and RDF Data. In. *Proceedings of Semantic Web Working Symposium*. Palo Alto. August 2001, 331–344. [11] DELUGACH, H. S.: Towards Conceptual Structures Interoperability Using Common Logic. In. *Proceedings of the Third Conceptual Structures Tool Interoperability Workshop*. Toulouse, July 2008, 13–21.

[12] *ISO/IEC 11179-1 Information Technology — Metadata Registries (MDR) – Part 1: Framework. Second Edition.* 2004.

[13] *ISO/IEC 2382-1 Information Technology — Vocabulary – Part 1: Fundamental Terms. Third Edition*. 1993.

[14] MUNK S.: Névjellemzők interoperabilitása katonai informatikai rendszerekben. *Hadmérnök*, I 3 (2006), 121–136.