Abstract—Continuous need for collaborations of large number of systems has created a requirement to achieve interoperability of the systems involved in collaborations. The subject of the research work is to find a solution that ensures interoperability of those systems whose business is rigid and there is no possibility of introducing standards. The existing solutions dealing with a similar problem are described. However, those solutions are very limited when flexibility and easy expandability are needed. This paper proposed an approach based on the domain-specific knowledge base that contains the referent rules and models for each system which participates in collaboration.

I. INTRODUCTION

There is a need to find the flexible solution to achieve interoperability in systems whose business processes are rigid and in which there is no possibility for the introduction of standards due to business policy and rules on which the business is based. In order to facilitate the achievement of the interoperability of the large number of systems, we work on improving the existing solutions of interoperability of information systems. The goal is an adaptive solution to achieve interoperability of the large number of systems, which can be adapted to the business of each individual system.

II. THE MAIN PROBLEM

The main problem of our research is how to achieve interoperability in the case of a large number of systems that need to communicate with one system as a broadcaster or service provider. The most common case of this communication occurs when due to changes of the regulations in a country all companies of some type need to start communication with the system of some institution.

When the business processes and rules of these companies are not rigid or business processes are not yet defined there is possibility for introduction of standards in this communication. Two the most recently used global standards in business communication are EDI and XML. [6][7] The goal of this project assignment is to achieve the interoperability of the systems, taking into account the fact that the existing information systems of these companies are more or less developed as closed systems with established business processes and adopted standards.

The problem occurs in different data structures that these companies use in their systems and different sequence of activities in some of the business processes. In addition, there are some certain business rules that are specific only for one of the companies. Therefore, we need some flexible solution to achieve interoperability of these systems.

Fig. 1 represents the customary mode of communication of two software systems based on direct communication of their services. This is widely accepted and a good way of communication if one of the systems adapts to the needs of another system, taking into account the data structures, business processes and rules. If these systems want to use different types of data structures in their communication, described communication mode is not good solution.

This typical problem of interoperability in many of proposed solutions is solved by the mediator layer, which do necessary transformations for systems. The generally accepted solution, Enterprise Service Bus architecture implementation, is based on the mediator layer and it is an essential part of the implementations that require the layer of mediator. [1] One solution that introduces a dimension of knowledge, whose usage can reach the interoperability, is the approach suggested by the authors in [2].

The ESB architecture and the introduction of the knowledge dimension are the main components of the approach described in [3]. They define the distributed knowledge by using ontology and quite formal, which in our case was not an option, because due to the need for faster and easier maintenance of the solution, there is a necessity for the domain-specific knowledge base.

ESB is very good solution for described problem of interoperability if we have final and not very large number of companies. The solution is to configure adapter for every pair of systems in communication, represented in Fig. 2. But, if this number of systems increases too much, it would be very hard to maintain this architecture.

III. THE ADAPTIVE INTEROPERABILITY OF INFORMATION SYSTEMS

This paper proposes the approach that is based on the introduction of domain-specific knowledge base that contains the referent rules and models for each system that subscribes to the usage of services of the main system as broadcaster or service provider (the system that exposes services). Another one domain specific approach is described in [5]. The main difference is that this approach is based on domain-specific language, while our approach is based on domain-specific knowledge base. The proposed architecture is shown below in Fig. 3. Like you can see some stereotypes are defined, to simplify understanding of this architecture in any domain where it is used.
The adapter is defined, which performs the transformation of data from one format to another by consulting the knowledge base where the transformation rules are stored and defined for each user system in relation to the system that exposes services. In some points the adapter layer also must consult the system, the base system that introduces this architecture with earlier defined and exposed services to external user systems.

It is important to note that the knowledge base and the adapter layer belong to the system that provides the services. The internal database maintenance is also done by this system. Namely, when there is new system that initiates the collaborating with the existing system provider, in their mutual communications the formats of data that the new system wants to use are defined, and next to that, the way to carry out some business functionality or the execution order of activities, depends on what level there is a need to achieve interoperability – the level of data, the level of service, the level of the process or the level of the business systems. In the defined format, the rules are stored in the knowledge base like referent rules and models.

The knowledge base has its own services, depending on the type of physical implementation, for example stored procedures. All the knowledge about of rules, types of structures and flows is in the data not in structure of tables. So, it is very easy to make some new rules or flows or to change the type of structure some user system wants to use.

Just like in similar proposed solutions based on the mediator layer, the user system is not aware of the existence of the adapter layer. The main difference is that we define only one adapter layer no matter how many systems users participate in communication with system provider.

A. The knowledge base

Our solution introduces the domain-specific knowledge base that contains the referent rules and models for each organization (company) that subscribes
to the usage of services of the main system. When there is a new system that needs to start the collaboration, after the data format, some business functionality and the execution order of activities are defined, that is inserted in the knowledge base. The structure of the knowledge base shown below in Fig. 4 is generic and covers a variety of business domains. What makes it domain-specific is domain-specific knowledge in her warehouse. It contains the rules, not the transformation mappings to get the rules. Therefore, it is general and easy to maintain. Every system in communication with the main system is the organization in this base. For every organization we keep track of data structures that are in use. One structure has its own type and its own classification. The type of structure is defined like input or output, depending on whether it is a format in which we receive or we send data. The classification of structure is some global defined set of data like DataSet, JSON, XML or some similar structure. It the classification has defined schema for validation, we also keep it like the attribute of the structure for organization. Every structure has its own list of parameters, with their names and types. We define all business flows in all organizations we communicate with. Every organization could have some specific business rules for each or some of the flows. These rules are required to be evaluated before doing activates of exposed services. The rules can be of two different types: the value rule and the algorithm rule. In the next subsection it will be more about that. We keep history of all changes in the organization.

B. The adapter layer

The adapter layer component uses the services of other components of the system to make decisions and perform different actions based on them. This layer is based on a generic algorithm that includes flows for all exposed services of system. The generic algorithm is illustrated in Fig. 5 using the UML activity diagram.

In our generic algorithm we use mentioned stereotypes to show components of the system to which the used service belongs. The algorithm uses services of the knowledge base to inform of structures some system have in use – input and output structure type. When adapter has corresponding input structure to data received, algorithm can go on. Then the adapter gets all parameters of that structure. One of them is code of company that gives us information about the company that called the external service. When is well-known from which company system received request, by consulting the knowledge base the adapter gets all business flows defined for that company. If received request corresponds to some defined business flow in the next step is possible to get all the business rules for that flow. All rules must be met before the operations on data are done. If the rule is the value rule, it is necessarily to get that value from the knowledge base because algorithm needs it to continue. If the rule is the algorithm type the adapter layer gets its name and that algorithm must be satisfied before business process execution continues. These rules typed as algorithms need to be execute at internal system. This communication is also done by the adapter layer. When all rules are met, the algorithm gets output structure for company which is in progress, formats data in appropriate structure type and sends it back to the company.

The adapter layer is not technologically dependent. It can be implemented like Java or C# class, by using BPEL orchestration or some other type of implementation. There are different possibilities.

IV. THE APPLICATION OF THE PROPOSED APPROACH IN THE DOMAIN OF INSURANCE BUSINESS

In the domain of insurance business there is a need that all insurance companies start communication with The Association of insurers of a certain country. These are existing companies with rigid business processes and rules and there is no possibility to introduce standards. Due to changes in legal regulations in this business domain there is a need for all business processes of insurance companies to be centralized. At the country level would be keep track of all valid insurance and keep the registers of all persons, vehicles, land and so on. Here we can easily recognize the described problem where one system exposes services to a large number of other systems. The main system is the system of The Association of insurers of a certain country and systems users are all insurance companies of that country. The problem occurs in different data structures that insurance companies use in their systems, different sequence of activities in some of the business processes, such as policy issuance process. Also, there are some certain rules of business that are specific only for one of the insurance companies. So, as proposed by Athena approach, there was a commitment to allow interoperability at all levels - business systems, business processes, services, and data. [4]
In this domain are defined three different flows of business to work with policy:
1. policy issuance,
2. information on premium level and
3. policy cancellation.

The Association of insurers provides three types of services, for these three types of flow. Some insurance companies in their business have defined only some of these business flows, while some insurance companies use all three business flows. So there is a need to keep track of flows at the level of each insurance company.

All this knowledge would be defined in the knowledge base. Also if something changes in business of some company and the new flow is defined or some old is deactivated, we need to change that in the knowledge base. The history is needed because of the policies that are issued by old business rules and regulative.

To simplify the problem and facilitate the understanding of the same consider the case of two insurance companies, two systems as users with their different business rules and requirements – insurance company A and insurance company B. The A company in its business has defined all three types of flows in the domain of insurance, so it uses all three business flows. From the other side, the company B uses only one flow - the policy issuance.

Consider some of pre-defined business rules in communication with each of the two insurance companies:

1. The policy number in both systems users are defined like the insurance company code + the next free number of the main system register. So, for both insurance companies this company code is defined like one referent rule in the knowledge base for the flow of the policy issuance.

2. The A company like input format can send any of formats that are defined by the main system and the company B wants to use Dataset of parameters in format key value. This is defined like the structure of data of input type and Dataset classification.

3. The A company wants to receive data in the XML format, valid according to schema that is sent earlier. From the other hand, the company B wants to receive data in Dataset structure. These are also records in the knowledge base, defined like structure of data of output type and XML or Dataset classification.

4. The company B has one specific rule to issue policy only to clients who are known from earlier, i.e. it is the closed circle of clients. This would be the rule of flow in the knowledge base.

Now that we have all the facts on the business rules of these companies and their relationship with The Association of insurers, leading to the proposed approach we come to the following architecture shown below in Fig. 6. Here is the real architecture of n insurance companies where we use earlier defined stereotypes of our architecture and in Table I are defined all services shown in this architecture. Services that are exposed are three mentioned services. Every insurance company uses one or more of them. The adapter layer uses defined generic algorithm without any modifications and here are shown all services of the knowledge base used by it. Also, in this described example, the adapter needs service of internal system to execute algorithm of rule defined for flow policy issuance of the company B. Because there is no any changes the adapter layer generic algorithm shown above in Fig. 5 is appropriate in the domain of insurance.
business. Considering the knowledge base this structure is also unchanged, but all mentioned knowledge about the business of these systems is stored in it. So, all business rules are inserted like records in the below defined structure. It is very important to figure out that all knowledge is in the data not in the structure of model. In the next two tables Table II and Table III we are representing these four described business rules, in one table the rules about data formats and in another one the rules about the business flow policy insurance. In this representation we use flatted lists of data to avoid dependence on any type of physical implementation of a knowledge base.

V. THE SUMMARY

From the aspect of insurance companies and The Association of insurers, this approach is considered to be easy to implement and convenient, since adaptations are possible for each insurance company that is added as new participant in collaboration.

As mentioned at the beginning this approach is applicable in any domain of business, with minimal modifications of the solution in the domain where it is needed. The flexibility and easy expandability as the basic requirements are met. It is easy to define new rules for some system. Also there is possibility to eliminate some existing rules or to change type of structure some insurance company want to use.

That it is easy to introduce another of the company in business communication we have already shown, but it should be noted that if some new flow in work that is not covered by our generic algorithm appeared, it is possible to modify or expand the algorithm or to introduce some independent branch in the adapter layer. A further research will take into account this problem as well as the safety aspects of the proposed solution.

<table>
<thead>
<tr>
<th>Company</th>
<th>Name of work</th>
<th>Name of the rule of flow</th>
<th>Type of the rule of flow</th>
<th>Value of the rule of flow</th>
<th>Algorithm of the rule of flow</th>
<th>Date valid from</th>
<th>Date valid to</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Policy issuance</td>
<td>Company code</td>
<td>Value</td>
<td>0A</td>
<td>NULL</td>
<td>01/01/2017</td>
<td>NULL</td>
<td>TRUE</td>
</tr>
<tr>
<td>B</td>
<td>Policy issuance</td>
<td>Company code</td>
<td>Value</td>
<td>02B</td>
<td>NULL</td>
<td>01/01/2017</td>
<td>NULL</td>
<td>TRUE</td>
</tr>
<tr>
<td>B</td>
<td>Policy issuance</td>
<td>Only close circle of</td>
<td>Algorithm</td>
<td>NULL</td>
<td>Closed_Circle_Clients_algorithm</td>
<td>01/01/2017</td>
<td>NULL</td>
<td>TRUE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Company</th>
<th>Name of structure of data</th>
<th>Schema of structure of data</th>
<th>Structure of data type</th>
<th>Structure of data classification</th>
<th>Date valid from</th>
<th>Date valid to</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Dataset input</td>
<td>NULL</td>
<td>Input</td>
<td>Dataset</td>
<td>01/01/2017</td>
<td>NULL</td>
<td>TRUE</td>
</tr>
<tr>
<td>A</td>
<td>XML output 1</td>
<td>&lt;?xml version=&quot;1.0&quot; encoding=&quot;UTF-8&quot; ?&gt;</td>
<td>XML</td>
<td>&lt;xs:schema xmlns:xs=&quot;<a href="http://www.w3.org/2001/XMLSchema%22%3E">http://www.w3.org/2001/XMLSchema&quot;&gt;</a></td>
<td>01/01/2017</td>
<td>NULL</td>
<td>TRUE</td>
</tr>
<tr>
<td>B</td>
<td>Dataset input</td>
<td>NULL</td>
<td>Input</td>
<td>Dataset</td>
<td>01/01/2017</td>
<td>NULL</td>
<td>TRUE</td>
</tr>
<tr>
<td>B</td>
<td>Dataset output</td>
<td>NULL</td>
<td>Output</td>
<td>Dataset</td>
<td>01/01/2017</td>
<td>NULL</td>
<td>TRUE</td>
</tr>
</tbody>
</table>
REFERENCES


