Abstract—Software applications need to be more flexible and communicative across their environment. Nevertheless, it is not always obvious as they face incalculable heterogeneity and constantly evolving technologies. Interoperability today imposes a great decoupling effort. This will promote the reusability of the application potential and above all a facility for interoperability management. In this study, a conceptual interoperability framework namely Software Framework Interoperability (SFI) is proposed. It is based on a standardized architecture while characterized by a model-driven approach. The aim of SFI is to decouple the software bricks while ensuring their interoperability.
Keywords: Interoperability, Standardization, Model-Driven Engineering, Software Engineering.

I. INTRODUCTION

In the course of recent advances in interoperability, the need for an interoperable information system has become inevitable due to the pervasive problems for the manufacturing software. Software vendors are left with no option but to maneuver their utmost to a mechanism in which they can highlight cooperation. Herein, we refer this cooperation as the mechanism of relating their application with other systems. Moreover, this cooperation is in line with the solution customer. Our software is required to be interoperable with the customer’s solutions. Definitions on interoperability have been reviewed in [1], [2]. Generally speaking, interoperability is the ability for two systems to understand each other, and to employ functionality of one another. Linguistically inter-operation means that one system performs an operation for another system. According to [3] and from the computer technology point of view, it is the faculty for two heterogeneous computer systems to function jointly and to give access to their resources in a reciprocal way. We can assess interoperability on several levels: data, services and processes, regarding a given business context. Furthermore we can not deal with interoperability issues without studying enterprise integration. Other works include [4] considering the enterprise integration as a process of ensuring the interaction between entities and necessity to achieve a specific domain objective. The integration stage may cover many levels, in our context we focus on application integration.

Today, interoperability is a major challenge for companies. Currently, the software must follow the technological evolution and organizational in order to maintain a constant performance whether functional or financial. In this context, several software vendors like our organization, Berger-Levrault, have begun to look for ways to extend the life of software. Whether new or old applications that were developed in virtually rigid and closed environments. In our case with Berger-Levrault, we have observed the utmost significance of decomposition after the analysis of our existing solutions. However, numerous problems are generated by this fact. It will not be enough to decompose the system, it must first be evaluated to identify the interoperability locks. These locks can be strongly linked to dependencies related to the hosting environment. Once we arrive at the evaluation of the application conceptually, we should minimize the gap between the conceptual aspect and the technical aspect. This step is crucial because it will determine the decomposition efficiently.

II. STANDARDIZED ENTERPRISE ARCHITECTURE

According to ISO 15704 [9], an architecture is a description of the basic arrangement and connectivity of parts of a system. It is either a physical system, a conceptual object or tangible entity. Researchers such as [10] consider the architecture as a description of the system and its components in order to guide the implementation dealing with the structure of the Information System (IS) components, their relationships while taking into considering its evolution over time. Enterprise architecture permits managing complexity and risks that is related to various factors such as: organizational (context, stakeholders, etc), technological (interfacing) etc.

We explained earlier that an architecture is an abstract description of the information system. Enterprise Architecture Interoperability (EAI) does not focus on how a specific company is actually structured, neither on how it operates. However, it tries to establish the links between the software bricks by concepts, modeling tools, ... which are necessary to the outcome of the solutions interconnection.

A. Manufacturing software capability profiling for interoperability (ISO 16100)

The ISO 16100 series enables manufacturing software integration by provision of a common interface specifications. This interface ensures the information exchange among software which are developed by different vendors. Furthermore
it offers a software capability profiling using a standardized method to enable users to select set of softwares. This set of softwares is chosen such that they embrace their functional requirements, the standard covering conformance test method ensuring the the software integration.

The capability profiling of manufacturing software units is the key step. The capabilities of an application are represented as an activity tree organized hierarchically [11]. Each activity is based on its manufacturing domain. Additionally, an activity is well defined by a unique name with semantic information. These activities are modeled by capability class structure (CCS).

A Manufacturing Software Units (MSU) is a set of CCS. The Figure 1 demonstrates what we have just stated.

B. Open systems application integration framework (ISO 15745)

This standard proposes an Application Integration Framework (AIF). It is based on a set of rules which describes an interoperability profile. This profile provides a common application integration environment for a given application domain.

Unlike the ISO 16100 standard, the AIF model encompass several models more precisely. These three models include: (1) Process Integration Model (PIM): It presents views of control processes and the information flow. (2) Resource Integration Model (RIM): It brings the resources necessary for the proper functioning of the PIM. (3) Information Exchange Integration Model (IEIM): It identifies the syntax, semantics and structure of information. And as a second role, it follows the sequencing of the information produced by the PIM.

The creation of profile takes place on five steps [12]:

1) Documenting the functional requirements as noted by a PIM.
2) Selecting the appropriate base specifications for the object interfaces denoted in the integration models.
3) Selecting (conforming) sets of options, or subsets, in the base specifications.
4) Combining references to a set of compatible base specifications in order to meet the identified application functional requirement.
5) Describing it in terms of an interface definition language.

III. MODEL-DRIVEN ENGINEERING

For decades, software researchers and developers put their efforts to create abstractions that can help them to improve the design of their software. These abstractions may facilitate the ability of deploying applications around a complex environments. From the beginnings of computer science, these abstractions included both “platform” technologies and languages. These abstractions in the literature were approached as models.

A model is an abstraction of a system, modeled as a set of facts constructed with a particular intention. A model should be able to answer to any questions about the modeled system.

To make the software manufacturing process more flexible and fluid, Model-driven engineering has shown significant improvements in the development of complex systems by focusing on a more generalized abstraction than classical programming. It is a form of generative engineering in which all or part of an application is generated from the models. The crux of this idea is to use as many different modeling languages (Domain Specific Modeling Languages -DSML) as the chronological or technological aspects of the development of the system require. The definition of these DSMLs is called meta-modeling.

A meta-model is a model that defines the expression language of a model [5], which is the modeling language. The notion of meta-model leads to the identification of a second relation, linking the model and the meta-model used to construct it, called conforming to.

A. Model-driven Architecture approach

Following the emergence of the key concept of meta-model as a model description language, several works concerning meta-models have emerged in order to bring their advantages in areas such as software development which interests us.

This has also led to the emergence of a major problematic at this time. It is related to the incompatibility of variety of meta-models. The most relevant and logical answer was to propose a metamodel like MOF (Meta-Object Facility) [13]. As a model it aims to limit the number of levels of abstraction. The difference between a meta-model and a meta-meta-model is that the latter has the ability to describe itself.

The Object Management Group (OMG) bases its hierarchy on the Model-Driven Architecture (MDA) approach, the organization of the modeling is described by a pyramid (see Figure 2 for explanation).

The OMG adopted the MDA in 2000 [6] in order to fully exploit the advantages of the approach. In follow up, the consortium defines the details of the architecture while including several standards, notably UML, MOF and XMI.

The basic principle of the MDA approach is to rely on the UML standard to describe separate models for the
different phases of the application development cycle. More specifically, the MDA advocates the development of models:

- Requirement Computation Independent Model (CIM) in which no computer considerations appear.
- Analysis and Design Platform Independent Model (PSM).
- Code Platform Specific Model (PSM).

The prime objective of the MDA approach is the development of perennial models (PIMs), independent of the technical details of the execution platforms (JEE, .NET, ...) in order to allow the automatic generation of all the code models (PSM) and also targeted to obtain a significant gain in productivity.

The transition from the PIM level to the PSM level involves model transformation mechanisms. We will provide its detail in the section III-B.

B. Model-Driven Interoperability

Model-driven interoperability is a model-driven approach addressing interoperability issues. This approach introduces different abstractions of interoperability to reduce the gap between business models and code level. The MDI definition is based on the three levels of MDA (see Figure 3) transformation automation approach. An approach which enables analysis across different abstraction for the user until the execution phase. These models must be conceptualized in such a way that it can be interoperable with other models developed by other enterprises but using the same MDI architecture.

IV. RELATED WORK

Several works were proposed in Model-driven interoperability over the past 15 years. These efforts were introduced either by research committees or by software editors. We shall cast a view upon some of the notable ones in forthcoming sections IV-A and IV-B.

A. ATHENA Project

ATHENA is an integrated project sponsored by the European Commission in support of the Strategic Objective Networked businesses and government set out in the IST 2003-2004 Workprogramme of FP6. The main contribution of ATHENA was manifested in an AIF interoperability framework, which is the result of the improvement of a number of existing frameworks. Specially several concepts were taken from the IDEAS project [7]. It uses concepts, languages and a meta-model in order to achieve conceptual integration. This reflects the fact that the model-driven approach is the fundamental aspect of the AIF. It offers developers precisely with software integration guidance enabling them to achieve the interoperability state. The ATHENA interoperability framework (AIF) is structured into three layers shown in Figure 4.

- Conceptual layer: Focuses on conceptual aspects such as languages, meta-models, and model relationships. It provides all the basic tools to organize the interoperability system.
- Applicative layer: Focuses on methodology and standards. It supplies guidelines, principles and patterns that can be used in order to puzzle out the interoperability issues.
- Technical layer: Focuses on technical development problems in first step, then focus on solving problems related to communication and information environments.
B. European Interoperability Framework (EIF)

The European Interoperability Framework [3] aims to support the European Union’s strategy of providing European citizens e-Government (public services exposed via information systems and the Internet) while ensuring interoperability between the different services addressed to the public (citizens, companies).

The EIF is defined as an abstraction which regroups all the policies, standards and guidelines that were developed outside the EIF project by the different countries of the European Union. These rules aim to describe how a collaboration between two entities can persist while ensuring interoperability.

The main advantage of this framework is that it offers an open environment that can be adapted to the evolution of public administration. This adaptation may cover changes in needs and even the evolution of technology. The EIF addresses four levels of interoperability:

- **Legal interoperability**: Aligned legislation so that exchanged data is accorded with proper legal weight.
- **Organizational interoperability**: Coordinates processes in which different organizations achieve a previously agreed and mutually beneficial goal.
- **Semantic interoperability**: Precise meaning of exchanged information which preserved and understood by all parties.
- **Technical interoperability**: planning of technical issues involved in linking computer systems and services.

V. CONTRIBUTION

Our contribution is comprised of realization of a conceptual interoperability framework that could allow us to decouple the software bricks with two objectives:

1) It Promotes the reusing the potential of existing solutions in order to reduce the cost of developing new products while enhancing the competitiveness of the company. In fact, this reuse will save out time as well as efforts in terms of implementation;

2) Preparation of an interoperable solution with other targeted systems: this is ensured when the application is decomposed. We did not take back the standard as it is, we maneuvered it. In this way, it can effectively answer to our context as well as to optimized solution to our expectations.

In order to arrive at this solution, we have taken over the fundamental aspects of ISO 16100 for the construction of the capability profile. We have incremented to this model in such a way that it describes the functional environment of the application in more persuasive way. Indeed, at the moment, applications in principal are dependent on several entities (ie: ORM, Server container, etc.). This requires us to take these entities into consideration as soon as we evaluate the solution in order to avoid incompatibility problems.

A. Overview of the framework’s functional principle

In the presence of an opportunity for collaboration, the developer tries to create a virtual profile for the solutions used by the customers. Once the target solution is virtualized, the developer put an effort to retrieve the template profile that can be integrated into the final solution of the client and that exist in the database profile. We will specify that a template represents a set of profile.

At this stage, we can face two scenarios (see the Figure 5):

- If no template matches then the developer will receive an acknowledgment.
- If the developer request succeeds then we proceed to the next step.

Thereafter the framework merges the selected profiles with the virtual client profile, we will demonstrate the detail later on. The objective of this step is to unify the profiles by creating a single interoperable model.

Once the model is generated, we perform a model transformation in order to obtain a shape of an interoperable implementation.

B. The framework in detail

As we described in section V-A, our framework aims to restrict the coupling of solutions while ensuring interoperability. To achieve our goals, we investigated the related research on the profile in two aspects. The first point is expressed in a capability profile which serves to decompose a software application into a set of functionalities. The second aspect revolves around modeling of the software context.

1) Capability Profile Model: Inspired from the existing state of art work related to ISO 16100, We shall extend our observation that the capability profile needs to be extended. The extension must define the context of the environment where the application is running.

The objective being as we said before, dividing the software into features, this will allow us to modularize even better our applications on the one hand. But above all, we will be able to promote the re-usability of the potential of existing solutions. For example, if we have a solution that manages employees in a hospital, we can inevitably use part of this solution to create another solution for managing employees in another sector. However, in our context of Berger-Levrault company, we preferred assembling our solutions by domains (ie. healthcare, E-Gov).

As the shrewed readers can notice from the Figure 6 that we have decomposed each solution into a set of methods which operates with resources. The latter will allow us to analyze the process. A process is composed of a sequence of activities, each activity will correspond to a capacity.

A capability template represents a set of capability class. It can be exposed as a profile. However, applications tend to rely more and more on external execution frameworks. This is why we thought of modeling the environment of execution in the following.

2) Runtime Profile Model: New applications often use rather heterogeneous environments and subsequently represent dependencies. We considered and eventually worked around the same fact. That’s why we designed a runtime profile that will complement the capacity profile. The environments that can be attached to applications can be
an Object-Relational Mapping (ORM), containers that use resources as config files or an environment variable empowering synchronization with the operating system. The structure of the model is shown in the Figure 7.

3) Model Mapper: The process of merging sub models is not only non trivial but also a substantial phase in the course of conceiving, realizing and developing the parameters of a model in software engineering. We will begin by merging the runtime model profile with the capability profile. The next phase is to bind the profile of the solutions that are available with the virtual profile of the client.

The mapper is based on a model-oriented architecture. We took this choice in order to fluidify the passages from the conceptual gap to the technological gap.

VI. CONCLUSIONS

In this study, we have proposed a conceptual framework for interoperability that evaluates the capabilities of each functionality included in a solution. The evaluation aims subsequently to divide the solution into a set of functionalities while having the vision to export the dependency elements of the environment.

Throughout realizing this approach, we have noticed that the profile as presented in the standards is static. This will pose plenty of problems. Indeed, the profile need to be up to date and on track in the current state of the solution. Eventually along these course of actions, it can provide us with stable and exact interoperability models.

To overcome the problems linked to the maintenance of profile, we intend to continue this work through research that consists essentially into an automated update process. We have planned to introduce updates through source code annotations.

REFERENCES


