Abstract—Many forward-thinking organizations have adopted domain-specific languages (DSL) as the preferred method for describing business processes. Using DSL-based descriptions helps in removing uncertainty from the semantics of process models. DSLs can evolve in a managed way and with proper versioning of individual processes the original intentions of process designers can be preserved over time. However in collaborative projects, business processes written in different DSLs need to be converted to a common denominator format to facilitate exchange. Due to its widespread adoption, BPMN is ideally placed to serve as the exchange language for complex, cross-domain collaborations. This paper presents an approach for automatic two-way synchronization of domain-specific process models with BPMN diagrams. This approach can be valuable when collaboration between different stakeholders with different expertise is required, as well as when the company wants to leverage its investments in a BPM suite across its process portfolio. In addition, this approach ensures that changes to processes executed through the BPMS are valid with respect to their domain representations, minimizing the potential for runtime problems that are difficult to understand.

Index Terms—domain-specific language, BPMN, BPM, modeling, tooling

I. INTRODUCTION

Large organizations today face a growing challenge of managing heterogeneous process collections containing business processes that correspond to various practices and domains. When growing by acquisition or indeed by diversifying in new areas, new business processes need to be added to the company portfolio and it is often extremely difficult to match and preserve their semantics in the context of the existing practices.

Some companies adopt Domain Specific Languages (DSLs) for the description of their business practices in the hope that the explicit semantics inherent to DSLs can help alleviate some of the management challenges. Indeed, using DSLs can be highly beneficial because they allow business experts to focus on their domain of expertise while explicitly formalizing the domain so that ambiguities are reduced to a minimum [1][2]. This is in contrast to the approach often used today of a single generic language such as Business Process Model and Notation (BPMN) [3]. While BPMN is presented as a simple-to-use business process description language, its generality makes it very difficult for people outside the initial group of process designers to understand the precise semantics of the respective processes. This is because the full meaning of process elements and their interactions is only known to the group of initial designers. In the best cases, conventions may be used that are well described in company-wide repositories. However, if BPMN is enriched with annotations that are standardized across the organization, then it becomes very similar in intent and use to a DSL (albeit less powerful). Similarly, shared ontologies can be used to facilitate annotation of BPMN elements with the purpose of preserving meaning and comparing processes produced by different sources [4][5]. While using ontologies can prove cumbersome in some enterprise settings, simple annotation of process elements can be valuable. In these cases we consider the resulting ‘annotated BPMN’ a DSL for the purpose of the discussion in this paper.

In many organizations, even without an explicit effort to create DSLs for business processes, a large number of DSLs may exist due to previous application development choices. In particular, the use of Model Driven Architecture [6] approaches for various application-generation needs results in the creation of domain specific languages that although not specifically targeted at business process design, do solve business needs. Therefore there is value in understanding how and where they fit within the BPM space. It is important to understand that DSLs are not necessarily simple to use, or simpler to use than BPMN. They also demand an upfront investment, when creating the language. However, they are inherently more precise and they have the potential to facilitate sharing of know-how in the enterprise as well as managing ever-changing processes through their generational capabilities (transforming processes into executable artifacts).

When various DSLs coexist in the organization it may be difficult to gain oversight in the development of business processes that span several domains. Usually managers that need to oversee such developments are not experts in all of the respective domains and for this reason they require a simplified and unified view of the processes. Even domain experts responsible for parts of the overall business process design may need to be able to communicate with their counterparts from the other domains in a simplified way. This is valid when the domains represent different fields of expertise but also when
they represent various snapshots of the domain expertise in time, in cases where the business practices have evolved in a particular domain and new process designs need to leverage older existing processes.

Another reason for the coexistence of DSLs with a unifying generic approach is that many organizations today have significant investments in business process management suites (BPMS) that typically execute and manage processes written using a standard language such as BPMN (and / or BPEL [7]). Bringing the DSLs to a common BPMN denominator can help in leveraging the BPMS investments while preserving the specificity and power of DSLs. This is of particular relevance when processes evolve: having a strong connection between the generic execution language and the languages used by domain experts ensures that invalid designs are much less likely to occur during the process lifetime.

It is therefore essential to have the technical capability to automatically generate on-demand, generic and simplified representations of business processes written in various DSLs. This paper argues that the appropriate common language for this is BPMN, which has a graphical representation familiar to many business process designers. A good metaphor for illustrating the usefulness of a common exchange language is to compare the activity of business process design and development with the activity of document authoring. As such, DSL-based business process design can be seen similar to using a fully-featured word processor or CAD tool, while BPMN-based exchange can be compared to using PDF for printing or for sending documents to people that are not authors and that might not have the expertise in the same tools originally used to create the content.

The remainder of the paper presents the approach and places it in a concrete technological landscape. Section 2 presents an overview of the approach without technical details while Section 3 illustrates an implementation choice by presenting a possible instantiation using open-source modeling technologies. Section 4 gives a walk-through scenario of a possible implementation from a user perspective. Related work is briefly discussed in Section 5 while Section 6 concludes with a summary and a discussion of the challenges and future work.

II. OVERVIEW OF THE APPROACH

This section gives a technology-independent overview of the approach which can be implemented in a variety of design and development environments, as illustrated in the following section with a possible instantiation in an open-source context.

The diagram in Fig. 1. presents the main elements involved as well as their interconnections.

For illustrating purposes, the diagram shows two domain-specific editors, one for Domain 1 and one for Domain 2 each supported by a respective DSL. Naturally, any number of DSLs can be supported by the approach. The editors are based on meta-models that contain the various DSL elements relevant to and corresponding to graphical editing constructs of business processes. The assumption (derived from the reality of many graphical editors today) is that changes in the process diagrams made by the users are constantly synchronized with the corresponding instance of the domain meta-model. Conversely, if the meta-model instance is changed by outside actions, these changes are reflected visually in the process diagrams. This synchronization allows changes to be propagated between the editor meta-models and the Common Meta-Model.

The Common Meta-model is a central, simplified representation of the main generic concepts common to business process descriptions, such as activity, flow, gateway and so on. It is significantly simpler than fully-fledged BPMN because its objective is simply to extract the essence of the structure of various business processes, and not to offer a complete modeling language. Converting between the Common Meta-model and the domain-specific meta-models is done using model-based transformations. The transformations are written at meta-model level and their execution is performed at model instance level. The transformations need to be triggered by modules that are connected to the editors, ideally plug-ins that integrate into the editors transparently.

They communicate with the Model Transformation Engine which is the entity responsible for executing the transformations.

Once transformed into an instance of the Common Meta-Model, the processes can be converted to the generic language of choice. In the diagram, the Generic Language Meta-model box illustrates the fact that the conversions between the generic language and the simplified common language are performed using meta-model operations, similar to the cases described above. The Generic Meta-model is assumed to be at the foundation of any generic business process editor and indeed of general-purpose business process management suites.
The reason why DSL-based business processes are not directly transformed into the generic language is to increase flexibility in the framework. As the generic language (usually a standard language) evolves, only the transformation between the Common Meta-model and the Generic Meta-model needs to be updated (in contrast to having to update all transformations between individual DSLs and the generic language). In addition, several generic languages (or indeed targets of various natures such as deployment artifacts for different platforms) can be supported incrementally over time. When new targets are added, new transformations need of course be added between the Common Meta-model and the new meta-models corresponding to the new targets, on a 1-to-1 basis. This approach allows scalability and flexibility in supporting a variety of environments.

III. INSTANTIATION USING ECLIPSE TECHNOLOGIES

The approach presented in Section 2 can be implemented in a variety of environments and editors. This section presents a possible implementation using widely available open-source Eclipse [8] technologies and editors. There is a wide range of editors and tools related to business process modeling and execution in the Eclipse ecosystem with a significant number grouped under the Eclipse SOA [9] top level project. These include BPMN [10] and BPEL [11] editors, the Stardust BPMS [12], the Java Workflow Tooling (JWT) project [13] and the Mangrove modeling framework [14] as well as a variety of runtime management projects. In addition to these projects that are related to business process modeling and execution, there are a number of generic Eclipse projects that are focused on providing modeling support, most notably the Eclipse Modeling Framework (EMF) [15][16] and the Eclipse Graphical Modeling Project [17]. These projects contain a variety of technologies that can be used to support the model transformation approach described above as well as the integration with the graphical editors. A potential instantiation of the approach in this context is illustrated in Fig. 2.

The domain-specific languages in this example are Xeproc [18] and extended BPMN respectively, however any other Eclipse-supported domain-specific language and editor could be used instead. Xeproc is a DSL for modeling document processing tasks as XML pipelines and it has an associated EMF-based meta-model as well as a graphical editor. The second DSL in this example is considered to be represented by an extended BPMN that corresponds to a particular domain. As mentioned in the introduction section, BPMN enriched with standardized conventions can be considered a domain-specific language in some situations. Naturally, the BPMN meta-model is the generic version needed to support standard BPMN editing but transformation plug-ins geared towards the extended BPMN can make use of the standard semantics of the extensions to the basic elements.

The editors corresponding to these DSLs, i.e. the Xeproc editor and the BPMN editor with added extensions are both extended with transformation plug-ins that trigger the transformations to the common meta-model.

The common meta-model is the Mangrove Core, which is part of the Eclipse Mangrove SOA Modeling Framework. It is a relatively simple EMF-based meta-model that aims to unify business process and SOA elements. Mangrove does not aim to provide a fully-featured business process repository to hold large collections of processes, such as Apromore [19]. Instead it focuses on preserving the common elements of business processes and architectural constructs from the various related diagrams, effectively keeping them in sync.

In addition to Mangrove Core, the Mangrove project contains a variety of plugins for model transformations as well as some editor extensions. The model transformation plug-ins contain code that convert the supported EMF meta-models to Mangrove Core and vice-versa. They are invoked from editor plugins that are connected to supported editors through standard extension points.

Typical targets for generic BPMN models are the BPMN editor itself or indeed runtime engines that would take either native BPMN 2.0 models or BPEL scripts generated from BPMN. A variety of BPEL engines are supported through Eclipse.

This example architecture illustrates how existing technologies and frameworks can be used to support the presented approach. The Eclipse environment allows non-intrusive, incremental support to be added to an existing business process modeling environment. Through the addition of sets of plugins corresponding to various functionalities described in this section, the environment can be upgraded to support generic model exchange between various domain experts and
organizational roles. This has the potential to significantly benefit communication and collaborative design.

IV. SIMPLE SCENARIO

To illustrate the functionality of the sample architecture described in Section 3, this section presents a simple scenario from a user perspective. The context is that of a travel agency that needs to design a business process for the creation of trip package proposals for its clients. A fragment of the process, responsible for the selection of the appropriate options to be presented to the client, is presented in Fig. 3. The image shows a simplified BPMN process with extensions as they could appear in an extended editor for a travel-industry-specific language. The icons associated to BPMN activities indicate the specification of precise semantics from an extension library available to the designer.

Fig. 3. Travel Package Analysis

The first activity ‘Select Potential Packages’ involves collecting the appropriate travel packages based on the decision by the sales department regarding the client budget. The outcome of this task is a list of potential travel offers. The next activity, “Check Availability” involves checking the availability of each of the selected packages, based on the appropriate travel periods. The following activity, “Decide Priorities”, orders the available packages based on criteria internal to the travel agency and corresponding as much as possible to the preferences expressed by the client. Lastly, for the top offers (according to the ranking), the travel agent must generate the documents containing all the necessary information about the packages so that they can be sent to the customer. This last activity involves DSL-based descriptions of the document-processing operations, to be performed on a system that is Xeproc-compatible (i.e. that is parameterized and controlled using the Xeproc DSL).

The document-processing expert uses Xeproc to design the appropriate process for the generation of the actual document containing the package to be sent to the customer. This process is described in the screenshot in Fig. 4. that shows the diagram created with the Xeproc editor.

In the Xeproc process, the Document Processing System is instructed to split the input (the top offers identified by the travel agent) into individual pages, extract and highlight destination titles, identify the hotel and destination descriptions, highlight the prices, put everything together and generate a summary and table of contents. The result, which is a multi-page document resembling a custom travel brochure will be packaged together with a customer response form.

After the two domain-specific processes have been designed in their respective editors, the designers may want to inspect the combined results in a generic BPMN diagram that hides away the specificities of the domains. This could be useful when holding progress reviews where process managers need to assess the advancement of the design activities. By using the transformations depicted in Section 3 a standard BPMN diagram can be generated. The connection between the “Create Client Proposal” activity in Fig. 3. and the Xeproc process shown in Fig. 4. can be made by simple naming convention (name of DSL injected into the activity name) so that the generation plug-ins know where to generate BPMN messages. The result of the generation is illustrated in Fig. 5. which shows a two-lane simplified BPMN process containing the combined activities extracted from the two domain-specific processes.

This BPMN representation is the result of applying the transformation from DSL editors to Mangrove Core and then to BPMN. However in some situations it might be desirable to follow the inverse path. For instance, the travel-domain expert that designs the process in Fig. 3. might want to do a high-level draft of what is expected from the Xeproc process. This draft could be sketched in standard BPMN, resulting in a simple chain of activities showing what needs to be done. Through reverse transformation from BPMN to Mangrove Core and further to the Xeproc editor, a basic initial diagram is created that contains generic elements. The Xeproc expert would then change the elements’ types to correspond to specific document processing tasks and would potentially even add, delete or change some of the activities. Using both transformation directions alternatively over time can result in better collaboration and reduced ambiguity of the overall process collection.
V. RELATED WORK

Most of the published research work in the area of DSLs is not connected particularly to the business process management area. Domain specific languages are seen primarily as great accelerators for the development of business solutions through efficient generation of code that is ready to be deployed. Typically solutions based on DSLs advocate the use of a clearly specified domain language that allows the definition of business logic that is then transformed into code which is eventually executed with minimal need for change by humans. While this is a valid and useful approach for writing applications, we believe that in today’s enterprise settings, it is important to leverage existing investments while incrementally adding advanced functionality that can accelerate development and evolution of business processes. This is one of the reasons we argue that it is important for DSL-based approaches to be integrated in standard BPM environments. This is in contrast to most existing approaches which advocate the use of DSLs as an alternative to BPM tools and suites. Sometimes a minimal level of integration is presented as desirable but it tends to focus on relatively isolated aspects such as support for database access or document flows [20].

On the other side of the related research space, the work in the BPM area focuses on improvements to generic business process languages and their corresponding suites. The generic languages such as BPMN and to an even higher extent its executable counterpart BPEL, require the addition of semantics through conventions rather than language constructs. For instance, the majority of commercial BPM offerings today that offer domain-specific capabilities do so using templates for business processes that correspond to individual industry verticals (e.g. healthcare, transportation or insurance). The problem of exchanging models between different domains does not really exist since all the templates are based on the same generic language, so no conversion is required. However this comes with the high cost of using static templates that offer little business flexibility.

Similarly, the approach taken by Business Process Interoperability (BPI) [21] relies on generic languages and technologies that use standardized conventions such as common industry-wide ontologies [22]. One of the main goals of BPI is to facilitate exchange and interoperability of business processes defined in various organizations. It encourages the use of Service Oriented Architectures that mutualize commonly used service interfaces to facilitate the automation of inter-organizational processes. While this is a worthy goal which can yield promising results in certain situations, it fully depends on the ability and will of organizations to agree on common standards. This is far from current practice for many high-value business processes in enterprises today. In addition, BPI does not focus on better collaboration within an enterprise, on sharing knowhow between stakeholders, and on automating constantly changing business processes.

The approach presented in this paper aims to go beyond existing solutions by leveraging the advantages of both worlds. It supports the use of DSLs in traditional BPM environments with transparent mechanisms for connecting various domain specific languages with the generic definition and runtime capabilities of existing business process management suites.

VI. CONCLUSIONS AND FUTURE WORK

This paper presents an approach for integrating domain-specific process descriptions into the typical BPM environments used in many organizations today. This has the potential to significantly simplify management of complex business processes that span multiple domains of expertise and to ensure that process evolutions do not incur unwanted behavior. As companies continuously grow their process collections, such an approach may become increasingly useful and it is essential that it be supported by tools seamlessly integrated into the existing investments in business process management suites. We have started identifying and implementing the required components for an instantiation of the approach using a mature set of open-source tools in the Eclipse ecosystem. To illustrate the approach, a set of prototype plug-ins today can deliver two-way transformations between the Xeproc DSL and Mangrove Core. Through the existing connectivity between Mangrove core and BPMN, these transformations can effectively connect the Xeproc DSL editor with the BPMN editor in the manner illustrated in Sections 3and 4. Through the extensible architecture promoted by Eclipse in general and Mangrove in particular, other existing connections can be leveraged such as BPEL, SCA [23] and JWT. Any other DSL can benefit from such connectivity provided the appropriate Mangrove plug-ins are added. We plan to further develop our Xeproc support as well as to improve the Mangrove Core and its various extensions in order to further facilitate the addition of new DSLs. In addition, we have experimented with adding Xtext [24] support to Mangrove in order to provide efficient means for integrating textual DSLs into BPM environments. Another area of exploration relates to the extension capabilities of BPMN 2.0.
and related editors in order to improve graphical display of DSL processes when merged into BPMN.

Although these experiments are limited to the Eclipse set of technologies, we believe they are highly relevant for any industrial-strength BPM suites many of which are actually built using Eclipse.

REFERENCES