Deriving biological applications from domain specific process models

Stefan Jablonski*, Matthias Faerber*, Bernhard Volz*, Stefanie Genthner**

*) Lehrstuhl für Angewandte Informatik IV, Universität Bayreuth
{stefan.jablonski, matthias.faerber, bernhard.volz}@uni-bayreuth.de,
**) stefanie.genthner@gmx.de

Abstract

In this paper we present how the process modeling and execution tools iPM and iPE can be used to model and execute biological processes. The main focus of this paper is on the flexibility of iPM and iPE with respect to the customization to the biological application domain. We will demonstrate the flexibility of our modeling methodology by giving two examples: Modeling the invocation semantics of web services used in the biological application domain and the processing of streamed data.

1 Introduction

Data integration is a central aspect in many biological applications. This ranges from storing large quantities of experimental data to retrieving information from external databases through the invocation of external applications (databases, data warehouses etc.). Another facet of these applications is the manipulation of data in a series of work steps which have a predefined order of execution. Our goal is to provide a software system that supports researchers in the biological domain. This software allows specifying customized processes that support the diverse data integration tasks in biological applications. Due to the heterogeneous structure of existing biological databases, the crucial requirements of such a system are flexibility and customizability.

We present a software solution for supporting the development of biological applications which are based on process models. Following the model driven software design approach our system consists of three components:

- a modeling component to define customized processes
- a generic version of a biological application from that customized biological applications can be derived, and
- a compiler that transforms the generic biological application into a tailored process execution system which is based on the process model.

In this paper we mainly focus on the data integration task of such kind of applications. We do not dwell on the biological problems nor do we provide new solutions for them. The main purpose of this paper is to propose a general concept that is coping with the challenging tasks of data integration in biological applications. We will first show an example for a biological process which uses several external databases in order to fulfill its goal. Querying these databases is done using web services. After the discussion of this example process, we will introduce our process modeling approach called Perspective Oriented Process Modeling. Here we show and explain its extensibility which makes it a primary choice for modeling biological processes as it can be easily adjusted to the biological application domain. Two examples for

such adjustments are discussed which arise from the exemplary biological process: The integration of web services and the integration of data streams into the semantics of the modeling language. The need describing the use of data streams arises from the fact that data streams are typical for scientific workflows and especially the search for similar sequences of proteins – as shown in the exemplary biological process – results not in one specific data item but a list of similar sequences. Here also an extension has to be made to our modeling approach as the classical workflow idea does not deal with that kind of data. In the fourth chapter we will then demonstrate the transformation of the model of a biological process into an executable application by using a special compiler for process specifications: The output can again be easily adjusted to the needs of the application domain. We end this publication by given a short summary in the fifth section.

2 Biological Processes

To illustrate the execution of a biological process we use the prediction of the three dimensional structure of proteins that is proposed by [14]. Although this process can be replaced by (almost) any other biological process without loss of generality, we have chosen this process for three reasons. First, the prediction of the three dimensional structure of proteins is one of the main goals of life sciences as the protein function depends on its structure. The sequence analysis is found in many biological examinations like the protein structure prediction, the building of phylogenetic trees and the analysis of metabolic and regulatory pathways [2]. Second, and for us even more important, this process makes use of distributed databases. These databases are used to compare the sequence with many others which are already characterized. The third reason why we chose this process is the creation of a data stream inside the process; this kind of data is typical for scientific workflows and can also be often found in the biological application domain. So this example provides a very practical application scenario where the core part of our research, the data integration task, can be investigated in a very realistic manner.

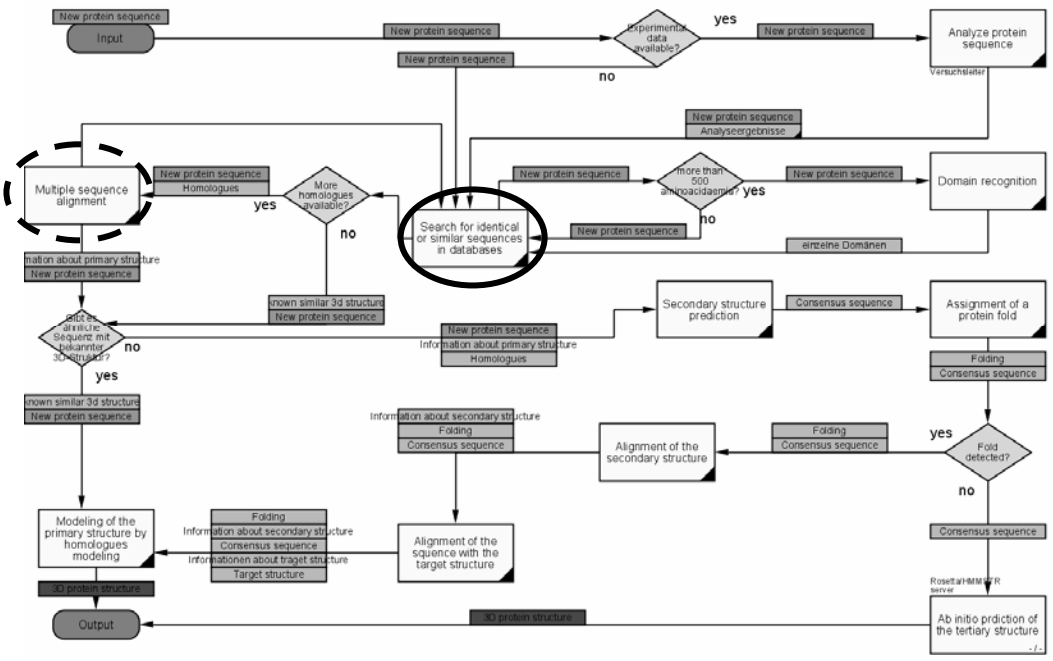

Figure 1. The prediction of a three dimensional structure of proteins as proposed by [14] modeled with the iPM process modeling tool.

We will not go into any detail of the process proposed by [14] which is shown in Figure 1. However, two of those steps that are concerned with the integration of external databases will be discussed shortly.

The starting point of this process is an unknown protein sequence. This sequence has to be compared with others from sequence databases in order to find homologues. The corresponding step in the process is marked with a circle. This search can be performed on several freely available databases that can be queried via the internet. For instance, some databases offered by the European Bioinformatics Institute (EBI, [4]) can be used for this purpose. Queries can be submitted by using either a plain web interface or a more programmatic approach, namely a web service. Examples for web services which can be used for a homology search at the EBI are: WSFasta, WSWU-Blast2, WSMPsrch or WSScanPS. As a result e.g. the WSFasta web service issues a "list" of similar sequences which can be seen as a stream of single sequences: This becomes even more obvious when a search is performed in an asynchronous manner where the results are not collected and sent back at once but whenever they are found. Processes following this special step will then not work on one single sequence but on a stream of them. As a consequence the processes after this search operation must be running as long as there are sequences present in the stream. Only after the stream has ended, the processes can be stopped.

A second process in which external services are used is marked with a dashed circle: Here a multiple sequence alignment has to be performed. Again the EBI offers a web service called ClustalW which achieves this. Also other tools can be used in order to fulfill this task but they are not discussed here as we want to focus on the data integration itself.

3 Model

The goal of our research is to support biological processes as shortly discussed in the previous sections. Especially, we want to be able to specify such processes that perform the diverse tasks of data integration as are encountered also in the example of Figure 1. The typical challenges from a data integration perspective are (from [3] and [11]):

- The structure of the data must not be the same in every database; i.e. it can be necessary to adjust the format of the data when it is used.
- Different databases or more general – data sources – offer a variety of languages and methods of accessing them.

To cope with the requirements compiled here, we pursue the following approach. We start with the assumption that such a biological application can well be modeled as a (biological) process. To execute such a special process, besides the functionality of a conventional process execution system (e.g. a workflow management system, [6]), some special functionality is needed. We choose the process modeling system iPM as a basis for modeling the biological process; we then choose the process execution system iPE [7] to execute that process. Both systems are characterized by their special ability of customization, i.e. they can be tailored to specific requirements of a special application area. The result of tailoring is called in the context of iPM and iPE a domain specific extension. In our case, we need a domain specific extension for the special issues of data integration in biological applications.

In order to discuss domain specific extensions of the two systems iPM and iPE we firstly have to introduce the basis of the iPM process modeling system, the so called perspective oriented process modeling approach as proposed in [6]. After that, two domain specific extensions for the biological data integration domain are discussed: They cope with the integration of external data sources using web services and the processing of data streams. As a result we will see that only by adjusting some perspectives of our modeling tool, these two examples

can be very quickly integrated into the modeling tool. Also iPE which is responsible for the generation of executable applications can be easily adjusted: We will show here that new templates for the code generation are enough for tailoring the output of iPE to the domain extensions mentioned above.

3.1 Perspective Oriented Process Modeling

The rationale behind the perspective oriented process modeling concept is to have a modularized process model that can easily be tailored to specific needs of particular application domains. One of our experiences was that mostly standard process models cannot be used in those application areas. We also wanted to avoid having to implement a completely new process model for those specific domains; rather we aimed at a kind of flexibly customizable toolkit that might be tailored for specific applications.

Tailored, domain specific process models have two main advantages:

- They are very appropriate to provide a communication means between domain experts and IT people who have to implement those applications.
- They can effectively and efficiently implement special features of an application domain without having to introduce cumbersome detours which lead to "spaghetti" code.

In the perspective oriented process model a process is composed of several different perspectives (aspects). Each perspective focuses on a different part of a process. The five basic perspectives are:

- **Functional perspective**

  The functional aspect is the core of the process description. It is used to describe the structural composition of a business process. It defines the work steps that have to be performed in order to achieve the goals of the process. The work steps normally build up a structured hierarchy and can be regarded as processes themselves.

- **Data and dataflow perspective**

  Through this perspective the input and output data of a process are defined. This yields into the description of the data flow between process steps.

- **Control flow perspective**

  The control flow describes the order in which process steps have to be executed. Also this perspective is used to describe parallel and alternative paths of execution in a process model.

- **Operational perspective**

  The operational perspective identifies tools that are used while executing a process step. These tools are applications like statistics programs or specific tools that perform searches in a database. Another kind of tools would be mapping programs which map the data coming from an external data source with a different format into the format used inside the process. This can solve one of the main issues of data integration as mentioned above.

- **Organizational perspective**

  The organizational perspective describes responsibilities in the process model, i.e. agents that are qualified to control the execution of work step. Typically, roles are associated with process models that describe eligible people or systems that are responsible to perform a work step.

Although most perspectives can – in principle – be modeled independent of each other, only the combination of several perspectives form a complete structure of a process. Figure 2 shows how a modeling construct is constructed following the idea of process perspectives. The one and only fixed part of a modeling construct is a so called process skeleton. This process skeleton knows about the perspectives currently relevant to form a modeling construct. The collection of all relevant perspectives thus forms a modeling construct. The collection of all modeling constructs constitutes a process modeling language. By introducing domain specific variants of the perspectives, domain specific modeling constructs can be defined. For each modeling construct, syntax, semantics and how it is presented in a modeling tool like iPM must be specified.

More details about the iPM modeling tool can be found in [13] and especially more details about the meta modeling approach standing behind the perspective oriented approach can be found in [1].

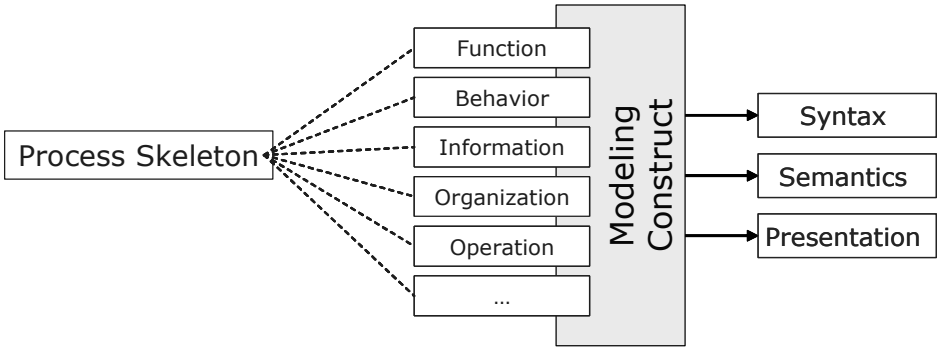

Figure 2. A modeling construct is built out of several perspectives.

3.2 Extension 1: Wrapper generation for web service integration

The five aspects that were described in the previous section set up the core part of every process model. Yet they are not complete. They can be extended with other aspects to include other views into the process model. This was successfully demonstrated in several other domains like the medical [12, 9] or quality management domain [8].

In this paper the focus is on the biological domain and the integration of external databases into a process. So far no information about accessing applications via web services has been added to the perspective oriented process modeling. In order to be able to include those external databases, some extensions have to be made.

The first approach of using only the operational aspect to model the access to external databases with the help of databases leads to one restriction: web services, defined through their WSDL document, lack semantic information, i.e. how the single methods of a service need to be invoked in order to get the desired result. This would mean to provide not only the WSDL of a web service but also a wrapper which encapsulates the semantics of the whole service in one single method. The execution system will then later on call this wrapper, invoke the method and by that use the web service. The main downside is that such a wrapper has to be provided for every web service which should be used during the execution of a process. Moreover the order of the invocations of single methods offered by the web service might be different depending on the concrete application. Here two or even more variants of the same wrapper are necessary to reach the goal of invoking a web service. Furthermore these wrappers cannot be generated but must be written by hand as there is still no semantic information on the method invocation available.

A second approach uses the operational aspect slightly different: If the operational aspect of a process has been set to "WSDL", the interpretation of this process inside the execution

environment changes – now the processes inside this work step are interpreted as a state chart which defines the semantics of the method invocations and gives a detailed specification on how a web service can be called and which parameters are being passed on in between the single calls. Only this change in the interpretation allows the specification of the semantics we need at the execution time. A positive side effect is the use of the already existing modeling language: No other language is required to describe the invocation order; the modeler uses the same tool.

Figure 4 shows an example of this modeling technique: It describes how the "WSFasta" web service of the EBI can be integrated.

The execution of the service starts by calling the method "runFasta" which needs two parameters of complex nature (i.e. they are composed out of other values). "inputParams" summarizes mainly four fields, leaving space for providing a valid email address, the name of the database the searching algorithm should use, the name of the search algorithm and a flag which indicates if the search should be done in an asynchronous or synchronous manner. Figure 3 shows the definition of the type as given by the WSDL document: Here more parameters exist but many of them could be set to null and are not necessary for the exemplary call described later.

The complex nature of the parameter in the process modeling tool is indicated in Figure 4 by the little black triangle in the lower right corner. The second input parameter – data – then holds the type of input data, e.g. "sequence" and the actual data that should be passed to the service. Sometimes not all of these parameters are variable – some may be fixed; especially the flag for specifying an asynchronous method call can be set already during modeling time as the semantics of using the web service strongly depends on this value: Modeling an asynchronous call is also possible but needs different constructs. In order to avoid a possible error this flag should be set to "false" (i.e. the call will be of synchronous nature). A value for an input parameter is set by adding a so-called data class to one specific data container, called "constant". The data itself then carries an attribute called "value" which gives the static value that cannot be changed. An execution system is later on not allowed to generate code which allows any other system to change the value of this parameter.

```
- <complexType name="inputParams">
  - <all>
      <element name="program" type="xsd:string" />
      <element name="database" type="xsd:string" />
      <element name="moltype" nillable="true" type="xsd:string" />
      <element name="histogram" nillable="true" type="xsd:boolean" />
      <element name="nucleotide" nillable="true" type="xsd:boolean" />
      <element name="topstrand" nillable="true" type="xsd:boolean" />
      <element name="bottomstrand" nillable="true" type="xsd:boolean" />
      <element name="gapopen" nillable="true" type="xsd:int" />
      <element name="gapext" nillable="true" type="xsd:int" />
      <element name="scores" nillable="true" type="xsd:int" />
      <element name="alignments" nillable="true" type="xsd:int" />
      <element name="ktup" nillable="true" type="xsd:int" />
      <element name="matrix" nillable="true" type="xsd:string" />
      <element name="eupper" nillable="true" type="xsd:float" />
      <element name="elower" nillable="true" type="xsd:float" />
      <element name="dbrange" nillable="true" type="xsd:string" />
      <element name="seqrange" nillable="true" type="xsd:string" />
      <element name="outformat" nillable="true" type="xsd:string" />
      <element name="async" nillable="true" type="xsd:boolean" />
      <element name="email" type="xsd:string" />
    </all>
  </complexType>
```

Figure 3. The type inputParams is defined in the WSDL document of the web service [4].

The "runFasta" method invocation returns an id for the job which has been started. As this scenario is using the synchronous search mode, the "poll" method is only called once: It will wait until the search result is available. This result is then passed to the output of the process.

For successfully calling the "poll" method and retrieving the output, a second parameter "type" is required. The value for this parameter – in this special case – is fixed: "tooloutput" has been set to model the invocation. Although the parameter is a result of the invocation of "runFasta", it has been put onto the flow in between the two methods: Which parameters are really an output of an invocation and which are defined through the model can be resolved by parsing and interpreting the WSDL document. Please note that Figure 4 shows the type name of each parameter as name for the parameter for better visualization purposes. In general parameters can be given arbitrary names: The dependency in between parameters and their type can be again modeled using data classes of iPM.

Although we have put the main focus on web services, the approach described here can also be used to integrate databases using other technologies. This is possible as the interpretation of the modeling constructs and their composition has not been fixed in iPM but is left open to any program which is getting the process specification as input. Only those applications have to have an interpretation for each construct found inside a process model. One example for such an application is presented in section four of this paper; iPE transforms a process model into an executable application and can also generate a web service specific wrapper.

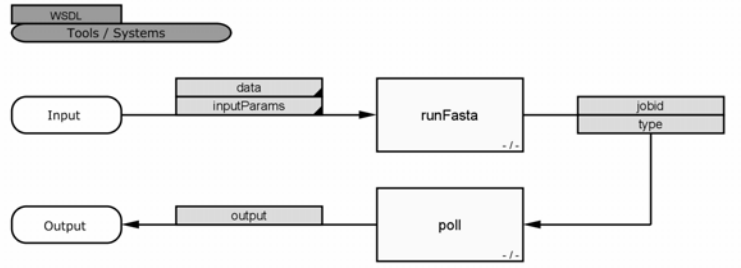

Figure 4. Example for modeling a very simple invocation of the WSFasta web service of the EBI.

If all this information is present in a process model, a wrapper for one specific web service and the modeled calling scenario can be automatically generated.

3.3 Extension 2: Data stream processing

Another extension for iPM for the biological application domain is the integration of data streams. Biological processes can produce data streams as shown in the second section of this paper and thus the modeling language must be able to cope with them. Data streams can be integrated by adjusting two perspectives:

- Data and dataflow perspective

  Data items inside a process can now identify two things: a single data item storing only one value or a stream of data of the same type. In order to distinguish these two types, one additional attribute has to be introduced which specifies whether the given data is a stream or not. Also the semantics of the dataflow changes: Without streams the data flow indicated one single exchange of data in between two processes. With streams a data flow stands for a continuous communication in between two processes up to a certain point – which is the end of the stream. As the processing of one single data item can last an arbitrary amount of time, the data flow must be buffered such that incoming data is not lost while a work step is still dealing with data received earlier.

- Operational perspective

  An application which executes a single work step must now also cope with data streams. Instead of stopping after working on one single item, the application needs to run in a loop: As long as there are data items inside the stream, the application is not allowed to end in order to avoid unnecessary waiting time during startup and

shutdown of an application. The application must only be stopped after the last data item has been received. As it can be rather hard to identify the end of a data stream, a special marker sequence can be send through the stream which causes all applications to shut down after they have forwarded the marker.

4 Model Transformation

In this chapter we want to describe in detail the transformation of a process specification into an application. We have developed a tool named iPE which uses a templates based approach for generating an application from a set of templates specially tailored to one application domain. We will also show how the first extension for the biological application domain can be integrated into this transformation process by providing a new service template for the generator.

4.1 Transformation of process models in general

The iPM process modeling tool does not allow executing a process model right-away. Instead, it is transformed into an executable form using a compiler for process specifications called iPE. This compiler uses the process model and a set of templates as an input and generates a domain and user specific application – or in general an output of arbitrary type. As the compiler is a general approach, one template it gets as input is a special module which defines the modeling language used for the process model. Other templates define the architecture (e.g. three tier architecture), the type (stand-alone application, web application, plugin, etc.) and the features (support for interfacing with external applications, including information delivered by a knowledge management system etc.) of the output.

The transformation of a process specification into an executable application is again described as a process (compare Figure 6). First, the input model is transformed into representation which is neutral in respect of the modeling language used to describe the process specification; one can see this form of representation as a superset of the domain specific modeling languages iPM supports in principle. In the next step, the single components and layers of an iPE application are being generated out of this neutral representation and the set of templates. By providing domain specific templates as input parameters for the transformation process, the output of the compiler is again domain specific; as the user can also choose which templates fits his needs best (concerning the application type), the iPE application is also user specific. The amount and type of templates provided for the compiler also defines the features the generated application will support later – e.g. it is possible to exchange the template for the Runtime Persistence Service which is implemented on a database system with one which is using the normal file system of the machine.

The execution semantics of a process is as follows: For each step, an operational aspect is defined which specifies an application responsible for carrying out that particular step. In case no operational aspect is found, a default application will be started which simply displays the input and output data of a process step. Furthermore this default application can be used to alter the data currently displayed to the user.

So far, we have constructed one set of templates which generates a web application running inside a servlet container like Apache Tomcat. The templates use the Java Server Faces technology for building up the default application. Other client-side applications like Microsoft Word or Excel can be used for the execution of one particular step: Here the corresponding application is started on the client side; when the user finished the working step carried out by such an application the data ("document") is transmitted back to the server and stored inside a database or a file system.

The iPE application is following the classical three tier architecture; an overview of the architecture is shown in Figure **5**. Each layer in this architecture can be easily exchanged. We reach this by defining a set of interfaces which must be fulfilled by each part of an iPE application and which give clear responsibilities to each layer.

The main part of an iPE application following this architecture is the Process Control Layer (PCL). The PCL manages several services and calls them when necessary. This layer is also used as an interface in between a frontend (here the default application) and the single services; in this case it delegates incoming calls from the frontend to the services which store the data of the process. Each service knows the exact position and all process data at the moment it is called by the PCL. Thus it possible for every service to execute functionality based on the actual position in the process.

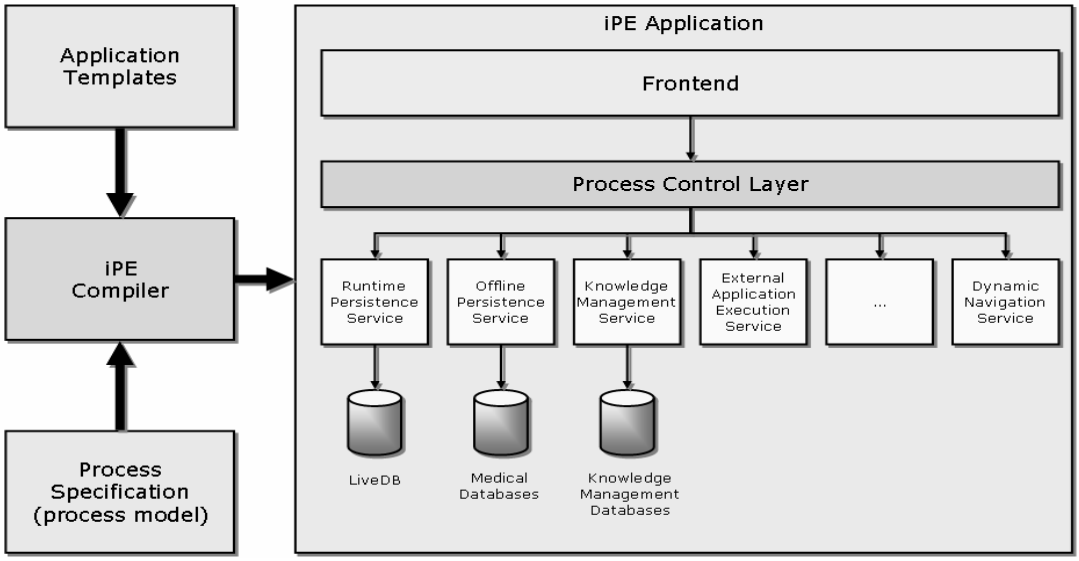

Figure 5. From the process model to an iPE application.

The many services listed in Figure **5** encapsulate the following functionality:

- Runtime Persistence Service: The RPS is responsible for supporting the execution of the biological process. As stated earlier, an iPE application consists of input and output masks. During the execution of a process, data is gathered and/or computed through the execution of single steps. The RPS keeps track of the execution, stores the data and also allows going back in the execution history. Data of the RPS is to be stored per process instance for a short period of time as the amount of data significantly grows during the execution.
- Offline Persistence Service: In contrast to the RPS, the OPS is responsible for storing the data won during the execution at given positions in the process model to a special, external data storage. This storage only contains the data produced by the process but no information on the flow of execution. Summarizing, the OPS is responsible for synchronizing the database of the RPS with external databases.
- Knowledge Management Service: The KMS can gather information of any kind from a knowledge management system and display it in the user interface. This is meant to support the user of a biological process e.g. by displaying additional information about protein structures and/or sequences. One can also think about using this service for providing information about the execution of former processes.
- External Application Execution Service: Invoking programs which provide a web service is one part of integrating external applications. Another service which is able to interface with other applications is the EAES. The difference is that this service

bridges an iPE application with a set of predefined applications. This is currently used for connecting office applications like Microsoft Word or Microsoft Excel with an iPE application, allowing the user to fill out a preformatted document which is later sent back to a server machine and stored inside a database or the file system.

- Dynamic Navigation Service: The execution of a process does not necessarily need to follow a predefined path through the model – instead, the actual path strongly depends on the data gathered through the process and the execution history. The DNS decides which step is to be executed next. The decision can be made either directly by the service or with a user interaction over the frontend.

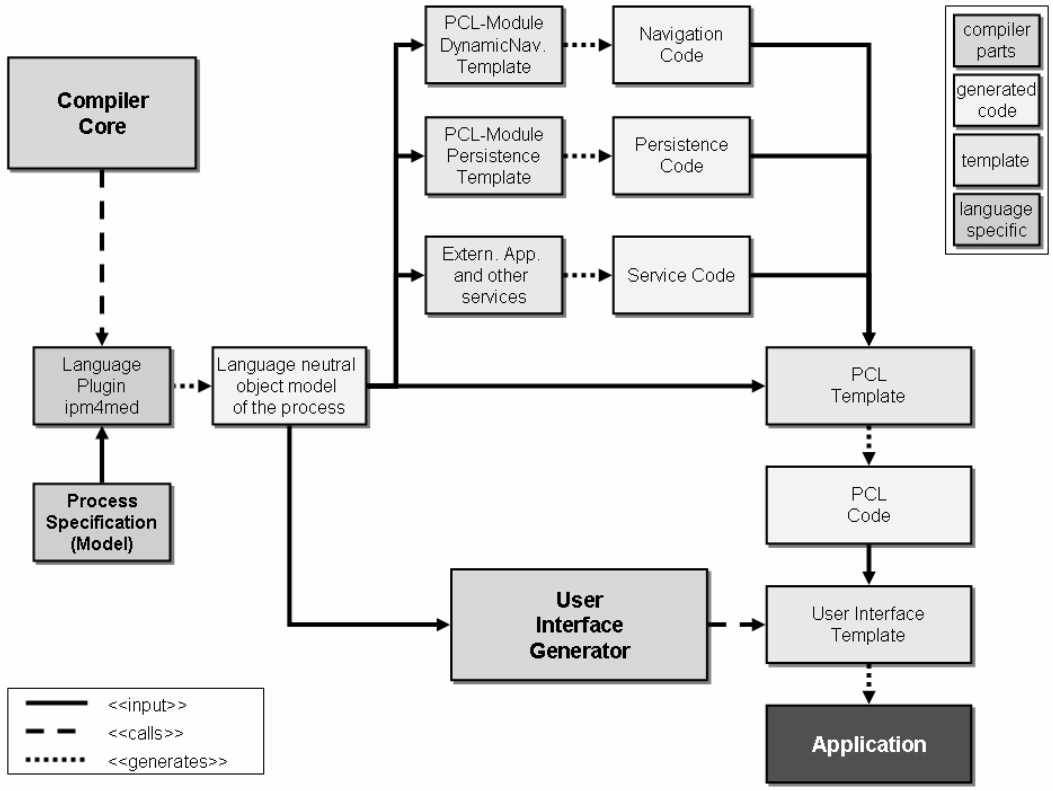

Figure 6. The schematics of the iPE compiler for process specifications

4.2 Automatic generation of wrappers for the invocation of web services

It is crucial to understand that the templates of iPE are not already finished blocks of application logic but generalized versions of a concrete solution which are tailored by the compiler to the needs of the application domain up to a certain point (compare Figure 6): The structure and the function of a service is already predefined by its template and cannot be adjusted through the generator. Furthermore every piece of information a template needs has to be put into the model if it cannot be derived from other information.

Only at the end of the generation process, all tailored services will be compiled into one executable application. Following this approach we do not need to provide a service for each web service which is accessed by the process but one generalized version: As an input for the generation of the specialized version, this template extracts the semantic information from the integrated meta model of the compiler, reads in the WSDL file and generates appropriate code. The WSDL file is on one hand used to generate proxy/stub components needed for the access of a web service; on the other hand, the WSDL file contains information about the parameter types and methods which can be cross checked with the model. Beside this, the WSDL also clearly specifies in which order parameters must be passed in a method call for

the service. All this makes the WSDL an essential part for the generation and it cannot be left out. In general this approach is applicable for arbitrary web services and allows us to generate specialized wrappers sufficing the needs of the application domain automatically from the model.

In order to integrate the domain specific extension explained in detail in section 3.2, two extensions have to be made: The user interface generator needs to be extended in order to handle these new kind of processes and a template for calling web services has to be written. The language neutral representation of the process model of the compiler does not need any changes since the modeling language is not changed – also the language plugin which transforms a given model into this language neutral representation does not need to be touched.

The user interface generator is responsible for generating an appropriate interface for the application. It creates a copy of the language neutral representation of the process model and modifies it. The modifications do not concern the semantics of the process but help to ease the development of user interface templates: The user interface generator can then tell clearly which process (beside the start and the end of a process) are executed through the user interface ("default application"). We have implemented one user interface based on the Java Server Faces (JSF) technology. By default, this generator creates one JSF page per process step. In order to avoid the creation of such pages for those processes which are describing the semantics of a web service invocation, it needs to be altered: If there is an operational aspect whose name is "WSDL" no interface needs to be created – instead the web service will be called later in the process.

5 Discussion

The modeling methodology presented in this paper is also applicable in applications the conventional perception of scientific workflow. According to this, a scientific workflow consists of automatically executable steps only. Our approach also allows to model work steps that are not executed by calling a certain application (e.g. a web service) but that describe interactive activities, i.e. activities that involve human interaction. This ability nicely broadens the applicability and the expressiveness of our approach. For example, non-computable decisions can be incorporated into a workflow like a human controlled scientific analysis.

In [5] the Taverna workflow execution engine, another approach for the execution of bioinformatics workflows based on graphical process descriptions, is presented. Although the basic principle is comparable to the method introduced in this paper, the implementation and the scope of the workflow itself differ.

Taverna follows a bottom-up approach in respect to the orchestration and enactment of workflows. Bottom-up means, that starting with external web services a user has to build and compose a workflow starting with the smallest possible unit. This way a workflow application is constructed using concrete elements (e.g. web service port types) to build an abstract workflow. The applied workflow modeling language supports this approach (SCUFL [1]).

Our approach is a top-down approach. It starts with an abstract process model that is detailed to concrete work steps. The abstract process model is far from being executable. However it can be used for communication between scientists which is a necessary phase when new scientific applications are developed. In order to alleviate such communication, a process is not described by executable services but by a more textual representation which can be easily understood by domain experts. After such an abstract (general) process has been modeled, it has to be refined and supplemented with execution specific attributes (i.e. web service

locators or port types). So, the abstract process model is transformed into an executable workflow.

Approaches like Taverna and our approach could easily be integrated into a single method. While the strength of our approach is in providing a structured modeling methodology (clear, extensible and abstract process language), Taverna provides a comprehensive model for the orchestration of web services. I would therefore be a valuable approach to start with an abstract model in iPM and later map that model to a Taverna workflow. This integrated method would leverage on the advantages of both basic approaches.

The current implementation of iPE does not yet support advanced auditing. Therefore, we want to integrate a log component which tracks execution information about workflows. This process log should than be used for the reproduction of data manipulations and should support the traceability of the process.

We are also planning to integrate quality-of-service figures which allow us to automatically respond on changing conditions during the execution of a process. For instance, when the data volume exceeds certain limitations it can be handy to compress the transmitted data.

6 Conclusion

In this paper the application of the perspective oriented process modeling approach for modeling biological processes has been shown. First a biological process has been introduced and used for identifying two main topics in the biological application domain. The integration of data and the processing of data streams. We then gave a description of how the modeling language iPM needs to be extended in order to cope with these two requirements. Even web services have been used for the integration of external data sources it also has been shown that the presented concept can also be applied to other techniques.

7 References

1 Addis, M., Ferris, J., Greenwood, M., Li, P., Marvin, D., Oinn, T. and Wipat, A.: Experiences with e-Science workflow specification and enactment in bioinformatics. In Proceedings of e-Science All Hands Meeting 2003, pp. 459-466, East Midlands Conference Centre, Nottingham. Cox, S. J., Eds., 2003

2. Backofen, R.; Bry, F.; Clote, P.; Kriegel, H.-P.; Seidl, T.; Schulz, K.: Aktuelles Schlagwort Bioinformatik. Informatik Spektrum, October 1999, (http://www.pms.ifi.lmu.de/publikationen/PMS-FB/PMS-FB-1999-11.html); retrieved: 2006-11-14

3. Bry, F.; Kröger, P.: A Molecular Biology Database Digest. International Journal On Distributed and Parallel Databases 13(1): 7-42, 2003

4. European Bioninformatics Institue, Website: http://www.ebi.ac.uk, WSDL for WSFasta web service: http://www.ebi.ac.uk/Tools/webservices/wsdl/WSFasta.wsdl, retrieved: 2006-11-14,

5 Hull, D., Wolstencroft, K., Stevens, R, Goble. C., Pocock, M., Li, P. and Oinn, T.: Taverna: a tool for building and running workflows of services. In Nucleic Acids Research 2006 34(Web Server issue):W729-W732; doi:10.1093/nar/gkl320, 2006

6. Jablonski S., Bußler C., 1996. Workflow management - modeling concepts, architecture and implementation. London. International Thomson Computer Press, 1996.

7. Jablonski, S.; Faerber, M.; Götz, M.; Volz, B.; Dornstauder, S.; Müller, S.: Configurable Execution Environments for Medical Processes. 4th. International Conference on Business Process Management (BPM), Vienna Austria, 2006, Vienna, Austria
8. Jablonski, S.; Faerber, M.; Schlundt, J.; Bridging the gap between SPICE Reference Processes and OU Processes: An iterative business process modeling approach. SPICE 2006 (Proceedings of the 6th International SPICE Conference on Process Assessment and Improvement (ISO/IEC 15504 standard)), Luxembourg, 2006.
9. Jablonski, S.; Lay, R.; Meiler, C.; Müller, S.; Process Based Data Logistics: A Solution for Clinical Integration Problems, International Workshop on Data Integration in the Life Sciences (DILS), March 25-26, 2004, Leipzig, Germany, 2004
10. Jablonski, S.; Volz, B.: Datenbankunterstützung für die Prozess-Meta-Modellierung, Technical Report, University of Bayreuth. Submitted for publication, 2006
11. Leser, U.; Rieger, P.: Integration molekularbiologischer Daten. Datenbankspektrum Ausgabe 6, 2003, pp. 56-66
12. Meiler, C.: Modelling, Planning, and Execution of Clinical Paths. PhD Thesis, University of Erlangen-Nuremberg, 2005
13. ProDatO Integration Technology GmbH: Handbuch zu iPM, http://www.prodato.de, 2007
14. Russel, R.; Structure prediction flowchart. (http://speedy.embl-heidelberg.de/gtsp/flowchart2.html); retrieved: 2006-11-14