On Combining Business Process Integration and ETL Technologies

Albert Maier, IBM Deutschland Entwicklung GmbH
Bernhard Mitschang, Universität Stuttgart
Frank Leymann, Universität Stuttgart & IBM Software Group
Dan Wolfson, IBM Austin

Abstract: In this paper we contrast two important information technologies in the realm of Business Integration: Process Integration and Extract-Transform-Load (ETL) Technology. After a short characterization and description of either technology, we argue for a technology crossover that results in a synergy of the available technologies, and thus realizes a more complete and superior approach. A technological study as well as show cases will provide insight into the approach proposed.

1 Introduction

Business Integration is seen as a key technology to enhance the IT landscape and the productiveness of today's companies. Typically, Business Integration technology is separated into three levels (Portal Integration, Process Integration, Information Integration), whereof (Business) Process Integration [HW04] is commonly perceived as the driving force. ETL technology is core to Information Integration.

Process Integration comprises a set of capabilities that include among others the ability to integrate and manage legacy applications, enterprise application systems, and business partners as well as decision makers. At a more technical level the integration of information with people and (business) processes is addressed and realized by a service-oriented architecture.

For business process modeling and enactment a standardized language called BPEL4WS (Business Process Execution Language for Web Services, for short BPEL, [BPEL4WS]) has been devised. BPEL allows for the definition of both business processes that make use of Web services and business processes that externalize their functionality as Web services. Like other language approaches, BPEL supports a process-oriented notion to describe the relevant processes and activities of a business and among business partners.

BPEL fosters a two-level programming model: the lower level consists of executable software components in the form of Web services [Ley03] that realize the basic activities, and the upper, more abstract level consists of a process model that defines the potential order in which the activities making up the business process have to be carried out. In most cases, the process models are defined in a graph-like fashion and supported by corresponding graphical design tools [WSADIE]. From a modeling point of view, process design tools favor the description of control flow over data flow: For example, BPEL offers various language constructs to express control flow patterns such as loops, branch, and join, but data flow is defined implicitly by specifying (process) variables that basically represent input/output data of activities.

Since the activities are described as Web services, the actual implementation can be done in any language, based on any programming model, and for any execution platform. Furthermore, the concept of Web services allows postponing implementation selection decisions to runtime, thus exploiting the available execution environment to a maximum [Ley04].

Currently, there are already a number of products out on the market that support the BPEL4WS standard by means of graphical design tools and corresponding execution engines, e.g. Oracle's BPEL Process Manager [ORA04] and IBM's WebSphere Business Integration Server Foundation [WBISF].

ETL technology is used to provide a common, consistent representation of disparate data that is previously non-integrated and physically distributed across multiple systems. ETL processes mostly use a graph-like model to design sequences of data manipulation steps that are used to extract data from data sources, to remove data inconsistencies, to transform, restructure, correlate, consolidate, and finally to store the data for subsequent usage, e.g. to load it to a shared data warehouse. Typically, ETL processes are expressed in terms of control and data flow. However, unlike Business Integration processes, ETL process definitions and their corresponding tools favor data flow over control flow. In addition, ETL products come with a huge set of predefined functions, e.g. for transformation, correlation, restructuring. Since there is no standardization available, each product offers its proprietary graphical design language that is based on a proprietary data and control flow model, and implemented by a proprietary engine designed for a mostly platform-dependent runtime environment.

Today's ETL products like Ascential [ASC], Informatica [INF], Oracle Warehouse Builder [OWB], and IBM's DB2 Warehouse Manager [DWM] are able to design data provisioning processes that exhibit (simple) control flows that are capable of very complex "data flow" activities within a control flow.

As can be seen from the previous discussion, both technologies (Business Integration and ETL), although targeting different application areas, build on a process model that reflects data flow as well as control flow concepts. Hence, a comparison of the underlying techniques seems to be valuable from a technical as well as economical perspective. Exactly this is the focus of this paper.

In Chapter 2 a classical Business Integration scenario is used to highlight typical business process features and to characterize the BPEL technology. In a similar way, in Chapter 3, a classical ETL scenario is used to show the peculiarities of typical data provisioning processes and their corresponding language approaches. Chapter 4 provides a strict comparison of the underlying technologies, discusses the benefits of a technology crossover, and outlines a solution concept. An outlook to the industrial perspective of this enhanced process model and supporting technology in Chapter 5 closes the discussions of this paper.

2 Business Integration

In the following we present a typical business processes: Booking a trip (see

Figure 1). The process begins by receiving an itinerary via e-mail from a customer. Next, the soundness of the itinerary is checked by a staff member of the travel agency. After that, the corresponding flights are booked by sending a request to an airline. If the trip requires staying overnight, hotel rooms are booked by communicating with a hotel. After booking the flights and perhaps the hotel rooms the customer's credit card is charged by communication with the credit card company. Whenever an error occurs while checking the soundness of the itinerary, booking flights or hotel rooms, and charging the credit card, the customer is contacted by some staff of the travel agency.

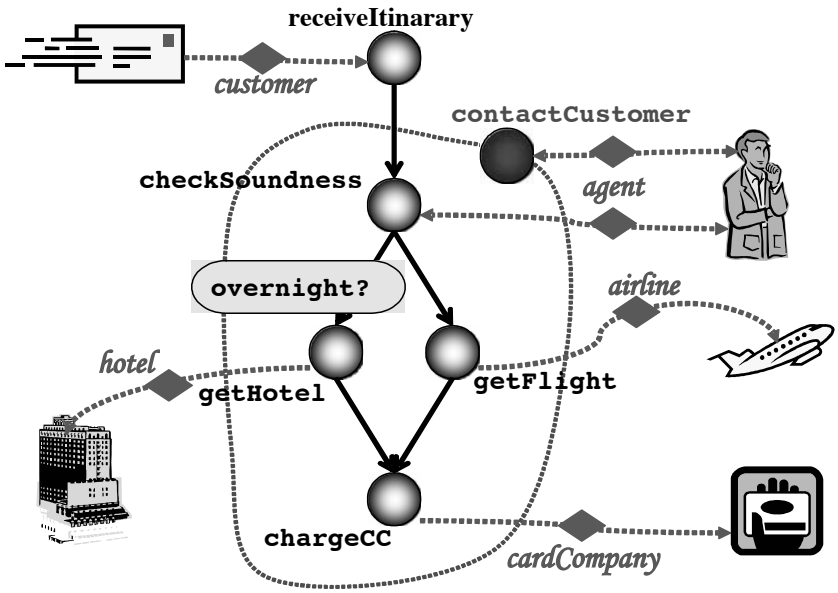

Figure 1: A Sample Business Process -TripBookingProcess

The following listing specifies this sample process in BPEL; note, that the listing is not complete because we only want to focus on the aspects key for our subject area. Line 01 starts the BPEL listing and provides the name for the process model. The kind of interactions a process has with its "partners" is defined by the **partnerLink** element which begins at line 02. Nested into this element are the individual partner links; each partner link specifies which functions a process offers to a partner and which functions it expects a partner to provide to the process. The partner link at line 03 describes that the interaction with a customer requires that the customer plays the **Recipient** role ("**partnerRole**") and the process ("**myRole**") plays the **TravelAgent** role; roles in turn are described in terms of port types elsewhere. A process receives and sends messages, and it might need some other data structures for holding and manipulating intermediate results. Each such data structure is referred to as **variable** in BPEL (e.g. line 05) and the **variables** element (line 04) collects all of these data structures into what is sometimes called the "process context".

The specification of the structure of the process model follows next. The **flow** element in line 06 specifies that process model is described as a graph. The set of edges used to connect the various activities (which are the nodes of the graph) are separately specified in the **links** section (line 07). An edge is referred to as **link** in BPEL (line 08). The graph begins with the **receiveItinerary** activity (line 09): It is a basic activity and it specifies that it **receive**s a message from the outside. The message is expected from a customer which is specified via the **name** attribute of the **partnerLink** element nested within the activity; and the customer is further expected to use the **orderTrip** operation of the **TravelService** port type; when the message is received it is stored in the **Order** variable (which has been defined within the **variables** sections at line 05). Line 10 indicates that the **receiveItinerary** activity is source of the link called **Itinerary**-to-Check; line 16 says that this link ends at the **checkSoundness** activity.

This activity (line 11) is an activity performed by a human being. But BPEL does not support the definition of such kind of activities, i.e. BPEL has to be extended accordingly. In [KKL04] corresponding extensions have been proposed: A **staff** element (line 12) is used to specify a predicate (line 13) that is used at runtime to determine which person can handle the task. In the specific example, all people playing the **Agent** role will be informed that the **checkSoundness** activity has to be done. Finally, one of the notified people will perform the task, and the data from the process context associated with the activity will be available to be passed to the service used to work on the activity. The **checkSoundness** activity is the start point of the **Check-to-Hotel** link (line 14); furthermore, a transition condition (line 15) is associated with that link that specifies that the link is only followed when overnight stay is required.

Fault handling in BPEL is associated with the concept of a scope. A scope embraces a part of process model and associates common fault handling with that part of the process model; note, that it also furnishes joint compensation handling with that part, but we are not dealing with compensation based recovery in what follows. Line 17 begins a **scope** and line 19 specifies that each fault occurring in that scope (and which is not handled locally) is dealt with by the **contactCustomer** activity. Within this scope, an airline company is contacted as well as a hotel chain (if needed), and payments are initiated by getting in touch with the credit card company of the customer: Whenever one of these activities does not succeed, the **contactCustomer** activity is run.

The **assign** activity at line 21 is of interest to understand the next sections: Assignments are used within BPEL to manipulate the process context, i.e. variables. In our example (line 22), the **customerInfo** part of the **Order** variable is copied into the corresponding part of the **Payment** variable. This variable in turn is used as **inputVariable** (line 26) of the **chargeCC** activity, i.e. the corresponding variable as constructed by the **assign** activity is send as a message to the service implementing **chargeCC**.

Example: BPEL Listing for TripBookingProcess

```
01 <process name="TripBookingProcess" … >
02   <partnerLinks>
03    <partnerLink name="Customer"...
                  partnerRole="Recipient"
                  myRole="TravelAgent"/>
        ...
     </partnerLinks>
04   <variables>
05       <variable name="Order" messageType="trip"/>
      ...
     </variables>
06   <flow>
07    <links>
08       <link name="Itinerary-to-Check"/>
        ...
      </links>
09     <receive name="receiveItinerary"
               partnerLink="Customer"
               portType="TravelService"
               operation="orderTrip"
               variable="Order">
10       <source linkName="Itinerary-to-Check"/>
       </receive>
11     <invoke name="checkSoundness" ...>
12       <bpelp:staff>
           <bpelp:potentialOwner>
13           <staff:membersOfRole role="Agent"/>
          </bpelp:potentialOwner>
         </bpelp:staff>
14       <source linkName="Check-to-Hotel"
```

```
15                transitionCondition="overnight='true'"/>
          <source linkName="Check-to-Flight"/>
16        <target linkName="Itinerary-to-Check"/>
       </invoke>
17     <scope>
18      <faultHandlers>
19        <catchall>
20          <invoke name="contactCustomer" .../>
          </catchall>
        </faultHandlers>
        <invoke name="getHotel"...>
        <invoke  name="getFlight"../>
21      <assign>
22        <copy>
23          <from variable="Order"
                    part="customerInfo"/>
24          <to   variable="Payment"
                    part="customerInfo"/>
          </copy>
        </assign>
25      <invoke name="chargeCC"
26              inputVariable="Payment"...
        </invoke>
      </flow>
    </scope>
  </process>
```

It should be obvious from the example, how the control flow of a process is specified explicitly in BPEL. Data flow is implicitly specified based on variables and assignments: An activity that receives a message specifies the variable to which the message received must be copied. Once received the message is stored persistently in the specified variable. An activity that sends a message specifies the variable from which the message should be taken from. Such underlying variable might be constructed from other variables beforehand. For example, activity A in Figure 2 receives a message that is copied into variable v^2. The assignment activity B accesses both, variable v^2 as well as v^3 to construct variable v^4. Variable v^4 is used as message to be sent out by activity D. Note, that the dashed arrows are only indirectly represented in BPEL syntax: For example the arrow from v^4 to D is represented by the **`inputVariable`** attribute of D's **`invoke`**.

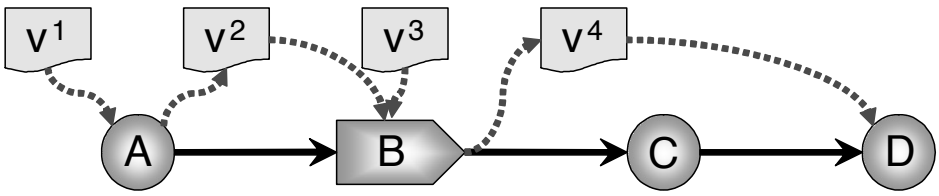

Figure 2: Dataflow in BPEL

Once a process model has been defined it can be deployed into a runtime environment [KKL04]. Deployment especially requires to determine which services (in the sense of an executable available from within the environment) to use as implementations of the port types making up all partner links of the process model. Typically, instantiating a process model then consists of the workflow engine navigating through the graph structure of the process model and interacting with the service bound at deployment time to each of the activities reached.

BPEL is geared towards supporting the logic of business processes ("programming in the large"), but not to be a general-purpose programming language. From a user's and experience point of view, it is beneficial to combine BPEL with a standard programming language to directly express business functions ("programming in the small'). Such an approach has been proposed as BPELJ [BGK04]. BPELJ consistently extends the BPEL process model to provide transition conditions and assignments specified in Java. In Chapter 4 we suggest a similar extension for process models that deals with data manipulations only.

3 ETL

In the following we present a typical ETL application. In the scenario described we are looking at a company that manufactures consumer goods for sale to other businesses. The financial department wants to track, analyze, and forecast the sales revenue across geographies on a periodic basis for all products sold. The company has decided to create a data warehouse for the sales data. The source data is stored in different formats in different operational systems. It has to be cleansed and transformed before being moved into the warehouse.

A star schema design is used for the warehouse. A star schema [In02] is a specialized design that consists of multiple dimension tables, and one fact table. Dimension tables describe aspects of a business. The fact table contains the facts or measurement about the business. Here, the star schema includes three dimensions: Products, Markets, and Time. The facts in the fact table include orders of the products over a period of time.

The processing logic for building the small star schema is shown in figure 3. The three dimension tables can be build in parallel, the creation of the fact table is performed subsequently. Finally there's a check whether the fact table could be successfully populated. If not, an administrator is alerted by email, else this data warehouse is ready for queries.

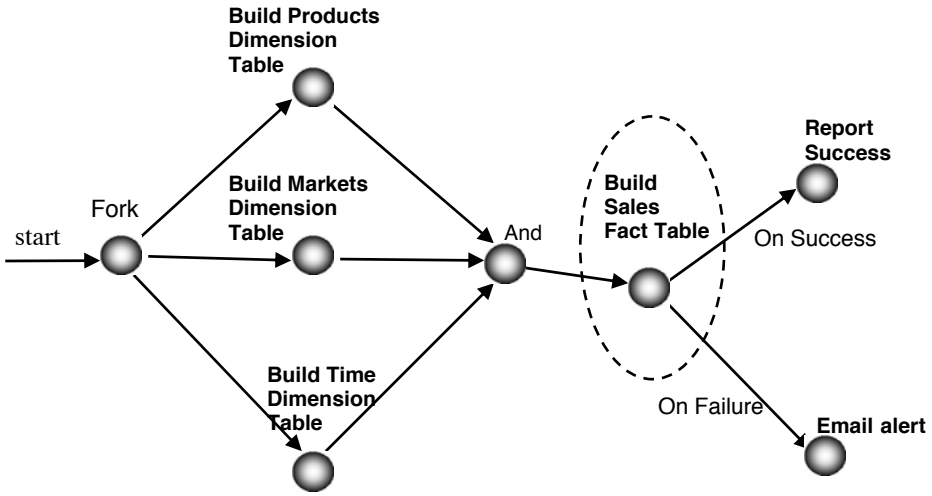

Figure 3: A Sample ETL Scenario (Control Flow Level)

Today's ETL products (Extract, Transform, Load) like Ascential [ASC], Informatica [INF], Oracle Warehouse Builder [OWB], and DB2 Warehouse Manager [DWM], are able to support such scenarios and to design the sometimes very complex "data flow activities" triggered by such a control flow. The **`Build Sales Fact Table`** activity for example might include steps like extracting the data from different sources, normalizing values (e.g. currencies), performing checks for correct formats and value ranges, and aggregating the data before finally inserting it into the target **`Sales`** table.

Figure 4 shows part of the data flow needed to build that **`Sales`** table. Typically graphical tools for designing data flows show both "operation nodes" and "data set nodes" at the canvas (in contrast to graphical tools for designing business processes and control flows that show "activity nodes" only). The two nodes in the top left corner of figure 4 represent a file respectively an operational table, both containing information about orders. The orders stored in the file contain currency information in $ amount, the orders in the table in Euro amount. The Euro amounts will be converted into $ amounts before the order data is merged into a single set. In a subsequent step the orders are checked for validity, e.g. each order needs to be associated with a valid customer and each date value needs to have the format for a correct date. In this example the resulting table **`ValidOrders`** contains columns with the order number, and then a pair of columns for each product, one column holds the amount charged for that product, and the other column contains the quantity sold. This is a typical spreadsheet format, with lots of columns, but very far from a normalized table. Therefore, a transformation to pivot the data is applied; the result is stored in the **`Orders`** table that is used as input for building the final **`Sales`** fact table.

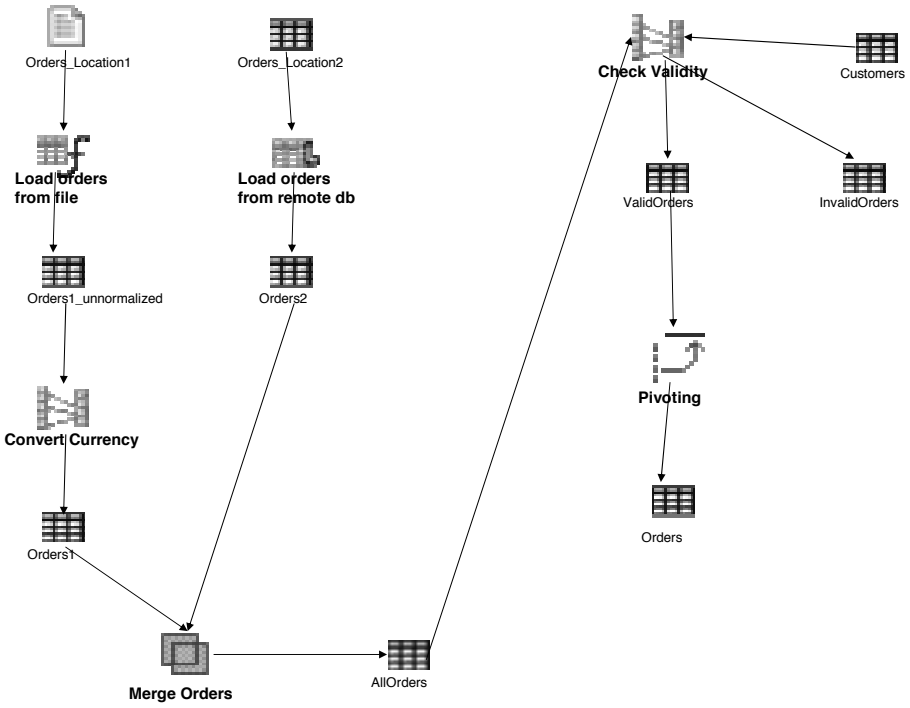

Figure 4: A Sample ETL Scenario (Data Flow Level)

The focus of ETL products is on the description and execution of data flows. Typically a very rich set of data operations is offered, the number of transformation functions might easily go into the hundreds (typically all SQL functions can be leveraged) and the load and extract operations support many different systems and formats. On the execution side even data in the terabyte range can be moved and transformed. In contrast, the control flow capabilities are usually weak. Often it's even not possible to design a control flow separately with a design tool, but control flow aspects are mixed with data flow aspects and the functionality might be restricted to specify sequencing conditions for data flows.

4 Technology Crossover

In this chapter we want to analyze and compare the two technologies that have been described before. At first sight it seems that business process technology (exemplified by BPEL) and ETL do not have common characteristics:

- Business process technology

  The goal of business process technology clearly is to model, manage, and process business process descriptions focusing on control flow issues.

  A service-oriented and message-based model is adopted that provides for a 2-level programming model: the basic level refers to the activities, and the higher level orchestrates these to constitute the final business process. A tuple-oriented model is applied, i.e. input/output to/from activities is done via BPEL variables that address a single object. Similarly, BPEL variables are used at the process level to express conditions that are for example part of BPEL loop constructs.

  There is a well-accepted standardization, with the BPEL language standard for expressing process choreography, and the Web service standard for the activity service level. By means of the Web service approach, an independence from the implementation level of activities and the execution platform is achieved. Basically, a flexible scripting of opaque activities is provided.

  The standard does also address extended transaction support for short and log-lived transactions based on a compensation approach. Error handling is another strong point. Though not standardized yet, user interaction is a further very valuable concept provided by current products.

  In BPEL, all data manipulation issues are hidden within an activity. The Java-snippet support introduced by the BPELJ proposal [BGK04] extends the BPEL process model in various aspects: data manipulation and transformation tasks are expressible via Java coding and Java code can be used in conditions.

- ETL technology

  The goal of ETL technology is data provisioning based on a rich and extensible set of data management functions.

  Data management functions are primarily set-oriented concentrating on movement and transformation of potentially large data sets. A set of available and for the current project necessary transformations are scripted to build the required data management and data flow aspects of the ETL process.

  For ETL, there are no standards available. Hence each product expresses some form of proprietary language for function definition and function scripting and a proprietary execution engine that, in turn, is mostly platform dependent. Furthermore, there is no general error handling defined and no user interaction provided.

However, at a more abstract level the two technologies do have commonalities. Among the most important ones are:

- To both sides, there is a clear joint notion of a process model that deals with both control flow issues and data flow issues.

- On the business process side control flow is the primary one and data flow is subsidiary, since the classical notion of business processes that are biased on control issues is predominant.

- On the ETL side it is just the opposite: data flow is the core issue and control flow is added on the side.

As can be conceived quite clearly, an enhancement to business process technology and especially to its standard BPEL4WS could be done with respect to its data provisioning capabilities. Vice versa, an enhancement to ETL technology could be focusing on extending its limited control flow capabilities.

Before further discussing these obvious extensions to either technology it is important to reflect the needs of the applications in some more detail: clearly, there are true ETL application scenarios as well as true business process application scenarios. However, there is an increasing amount of so-called mixed scenarios on both sides that require a combined technology that reflects control flow and data flow as equal modeling concepts.

In the conventional application areas for business processing, processes that reflect some amount of data management and data processing are well conceived. This is exemplified by the following:

- Information gathering
  Flows orchestrating the gathering of various portions of information (e.g. from a number of backend systems) to comprehensively describe the object of concern (e.g., a customer, user, client).

- Information casting
  Flows orchestrating updates against a number of (relational) target systems (e.g. backend data management systems).

- Batch processing
  Offline processing of for example accumulated orders turns out to produce so-called batch flows that show a high volume of data processing, on one side because the volume of orders to be processed could reach MByte to GByte range and on the other side because most processing activities in this flow refer to some kind of data processing on backend systems.

The peculiarity of this list of application areas is that although processes are being addressed, significant emphasis has to be spent on data provisioning tasks. Hence the problem is to somehow attach complex data provisioning tasks to defined business processes.

Similarly, one can find conventional ETL application areas, where control flow issues are of primary concern, as for example:

- Building a data warehouse
  After having designed a set of distinct ETL steps, each one constructing a single table of the target data warehouse, it is necessary to build the overall warehouse process, i.e. to orchestrate the whole ETL application.

- Content management
  Document management is steadily evolving from pure storage and management of documents into so-called enterprise content management, where the focus is among others (like rich media support, effective search capabilities) on document workflow. Document workflow basically defines the whole lifecycle (e.g. ingestion, search, storage) of a document that is managed by a content management system.

- Grid processing
  Processing and especially data provisioning tasks in a Grid environment can be described in a process-oriented fashion. This is especially supported by the OGSA standardization approach that subsumes specific data access an integration services (DAIS)).

The peculiarity of this list of application areas is that mostly processes are addressed that knit together complex data provisioning tasks.

All the scenarios and discussions from above clearly advocate for a combined technology that treats data provisioning issues at the same level as (business) process choreographing tasks. For this we propose to extend the given and standardized BPEL4WS process choreographing language by well-known SQL functionality and additional data provisioning functionality like transformation, correlation, and restructuring. A technical solution to this is to treat these BPEL extensions similar to the Java extension that was proposed to BPEL as part of the BPELJ approach [BGK04]. We name this proposal BPEL4SQL. BPEL4SQL will support “SQL snippets” as BPEL activities and in BPEL conditions. SQL snippets will provide read and write access to BPEL variables. Another explanation model for the BPEL4SQL approach is that it can be seen as an embedded SQL approach for BPEL.

5 Summary and Industry Trends

In this paper we have argued that a technology crossover that merges the strong technologies underlying process integration and ETL results into a truly enhanced technology that in turn provides to both discussed application areas an extended functionality.

Traditionally, the worlds of ETL and process integration have served different needs. As discussed in Chapter 4, there are an increasing number of application scenarios that would benefit from an integrated environment. There are various activities on the product market that indicate an evolvement towards such an integrated environment. Some companies like Microsoft and IBM own strong products in both areas, and are currently driving the standardization of BPEL. Others are completing their portfolio by buying the missing pieces, e.g. Oracle recently bought Collaxa, the vendor of a BPEL workflow engine. Traditional vendors of ETL technology, e.g. market leaders like Ascential and Informatica are currently evolving their proprietary infrastructure to integrate with a service oriented architecture, for example by allowing their processes to be called as services and by being able to consume external services. Underpinned by these trends we could see that the basic technologies are already in place with the market leaders.

Bibliography

[ASC] Ascential DataStage, http://www.ascential.com/litlib/index.html

[BGK04] M. Blow, Y. Goland, M. Kloppmann, F. Leymann, G. Pfau, D. Roller, M. Rowley, BPELJ: BPEL for Java, BEA Systems & IBM Corporation, 2004 http://www-106.ibm.com/developerworks/webservices/library/ws-bpelj

[BPEL4WS] OASIS Web Services Business Process Execution Language TC http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel

[DWM] IBM DB2 Universal Database Data Warehouse Editions - DB2 Universal Database Data Warehouse Enterprise Edition, http://www-306.ibm.com/software/data/db2/udb/dwe/edition-ee.html

[HW04] G. Hohpe, B. Woolf: Enterprise Integration Patterns, Addison-Wesley 2004

[In02] W. H. Inmon: Building the Data Warehouse, Wiley; 3 edition, 2002

[INF] Informatica Power Center, http://www.informatica.com/

[KKL04] M. Kloppmann, D. König, F. Leymann, G. Pfau, D. Roller, Business process choreography in WebSphere: Combining the power of BPEL and J2EE, IBM Systems Journal 43(2) (2004).

[Ley03] F. Leymann: Web Services: Distributed applications without limits, Proc. BTW'03 (Leipzig, Germany, February 2003), Springer 2003.

[Ley04] F. Leymann: The Influence of Web Services on Software: Potentials and Tasks, Proc. 34th Annual Meeting of the German Computer Society (Ulm, Germany, September 20 – 24, 2004), Springer, 2004.

[ORA04] Orchestrating Web Services: The Case for a BPEL Server, An Oracle White Paper, June 2004.

[OWB] Oracle Warehouse Builder, http://www.oracle.com/technology/products/warehouse/index.html

[WBISF] IBM WebSphere Business Integration Server Foundation, http://www-306.ibm.com/software/integration/wbisf/

[WSADIE] IBM WebSphere Studio Application Developer Integration Edition, http://www-306.ibm.com/software/integration/wsadie/