Interoperative end-user process modelling for process collaborative manufacturing

Lai Xu*a, Paul de Vriezea, Keith Phalpb, Sheridan Jearya and Peng Liangb

aSoftware Systems Research Centre, Bournemouth University, Bournemouth, United Kingdom; bState Key Laboratory of Software Engineering, Wuhan University, Wuhan, China

(Received 30 May 2011; final version received 8 March 2012)

As business environments change rapidly, the ability to quickly set up a collaborative automated business processes is desirable. Collaborative business processes are increasingly driven by business agility, adaptability and flexibility, particularly in a modern manufacturing enterprise environment. Traditionally, collaborative business processes are used among big organisations. Many of collaborative business process systems are designed for a long-term use with central control. It is a great challenge for enterprises to adapt processes to the today’s business pace. Today’s enterprise users demand many situational collaborative business process applications which handle business needs within a short period and which normally do not need to support too many users. The traditional centrally controlled systems are not designed for such situational applications. In this paper, we explore how business process mashups can be used for manufacturing enterprise collaboration. We highlight the modelling of end-users process collaboration from both a control flow and a data flow perspective. Based on our analyses, an end-user process modelling approach is proposed for process enterprise mashup applications. Our approach, illustrated by reference to an example pharmaceutical collaborative manufacturing case, will support collaboration among users with different levels of modelling skills and expertise in a pharmaceutical manufacturing enterprise environment.

Keywords: collaborative manufacturing; process modelling; end-user process modelling; process collaboration, process-oriented mashups

1. Introduction

The Internet lies at the core of a connected world, acting as a conduit for the exchange of information, allowing tasks to be processed collaboratively and enabling the formation of communities amongst users with similar interests. This Internet interconnected world has increased both business and personal efficiency and performance. Internet provides an infrastructure for building collaborative business processes both flexibly and quickly.

Service orientation is an approach to information system design that consists of a way of thinking of business process management in terms of computational infrastructure, services, service-based development and outcomes of those services (Papazoglou and Georgakopoulos 2003). Service-oriented architecture (SOA) is a significant computing paradigm and is being embraced by organisations worldwide as the key to business agility. Web 2.0 technologies such as AJAX enable efficient user interactions in web applications. This can be leveraged for successful service discovery, selection, adaptation, invocation and service construction without end-user restrictions. SOA and Web 2.0 technologies also balance automatic integration of services and human interactions, separating content from presentation in the delivery of the service. Integrating SOA and Web 2.0 technologies into a service-oriented application connects business processes in a horizontal fashion.

Mashups are a new Web 2.0 technology that leverages the integration between web services, web API’s and web 2.0. Current mashups are used primarily for data aggregation, combining data from different sources to create valuable information. Online mashups have the capability to combine data with (possibly external) functionality to create and produce useful outputs. The term ‘mashup’ implies easy and fast integration, usually made possible by access to open APIs and data sources to produce results that the data owners did not originally envisage.

We proposed the concept of business process mashup (de Vrieze et al. 2009, 2011, Xie et al. 2009), which allows users to specify their needs, find related Web resources and eventually execute the resulting process for rapidly building collaborative business processes. The first step in enabling business process mashups is to allow business users who have no professional modelling skills to model business processes. Thus, end-user process modelling for business process mashups aims to be a step towards the
enablement of business process modelling to an end-user group in an enterprise. The finished business process models provide the basis for process automation.

In this paper, we first look at related work in Section 2 and identify the modelling requirements of end-user process applications and introduce the end-user process modelling principles for business process mashup applications in Section 3. Section 4 explains the end-user collaborative process model. This is followed by discussion of details of process modelling related with both a control flow and a data flow perspectives in Sections 5 and 6, respectively. A pharmaceutical manufacturer collaborative case is used to demonstrate our modelling method in Section 7. The paper ends with conclusions and future research.

2. Related work

We examine related work from two perspectives, namely collaborative business process and business process modelling. Further, we look at the relationships between OASIS reference architecture foundation for SOA (OASIS 2011) and our work.

2.1. Solutions for collaborative business process

There are existing solutions to support collaborative business processes across multiple organisations. The complexity of the implementation of such collaborative business process automation depends on the flexibility of the process to be automated, the cardinality of participating business processes and the correlation of collaborating process instances. Setting e-contracts among involved partners’ workflow systems is one of the solutions for supporting collaborative process automation. Web service orchestration and choreography is another solution by exchange information and data in a loosely coupled environment.

Contracting among involved partners’ workflow systems collaborative processes is used to facilitate collaborations, while collaborations origin from contracting among organisations. Previous work on contracting (Chiu et al. 2002, Colombo et al. 2002, Xu and Jeusfeld 2003, Milosevic et al. 2006) has discussed the support of collaborative business processes on modelling processes, tracing processes and (pro-active) monitoring processes. The contracting approach is suitable for long-term business collaboration, and it is too expensive for solving situational application to satisfy flexibility requirements in virtual enterprise environment.

Web service orchestration and choreography provides a loosely coupled mode for a collaborative business process to be implemented by exchanging information and data. The approaches like WS-BPEL (Andrews et al. 2003), OASIS ebXML (http://ebxml.org), RosettalNet (http://rosettanet.org) and IBM WebSphere are able to provide collaborative business processes with messaging exchange mechanisms.

The OASIS ebXML business process specification schema (OASIS ebXML BPSS) (UN/CEFACT 2003) provides a standard framework by which business systems may be configured to support execution of business collaboration consisting of business transactions. However, OASIS ebXML BPSS only supports two partners involved business process collaborations. When multiple partners are involved in business process collaborations, the specification needs to be broken down into multiple bilateral relationships. This bilateral model easily results in increasing loads and complexity of tracking business processes. As a result of this and other limitations, OASIS ebXML BPSS-supported business collaborations lack flexibility and agility needed to respond to a changing environment.

Specifications, like WS-BPEL and WS-CDL, support service orchestration and choreography according to predefined processes or rules (W3C 2005). Such a static coordination mode cannot easily capture the dynamics of business processes in collaborations.

Traditional solutions for collaborative business process are designed and implemented by process architects and software engineers. Being agile in adapting business process of organisations to market dynamics, the new approach should look beyond the traditional solutions through collaborative interactions and dynamic virtual enterprise solution binding (Camarinha-Matos et al. 2003). Our business process mashup however aims to handle dynamic, flexible, end-user friendly creation, modification and automation of business processes.

2.2. Business process modelling

Business process modelling is the first step of business process management (van der Aalst 2004, Zur Muehlen and Rosemann 2004). Business process models are usually designed by domain process experts and professionals. The purposes for modelling organisational processes are various (Reijers 2003). One reason of modelling processes is to provide a way of documentation, communication and collaboration of business needs. Such business needs can be making strategical decisions, verifying compliance with regulations and so on. Another aim of process modelling is to provide the basis of process execution. Processes are executed according to process models, which contain application logic, under business process management systems like workflow systems.
In the business environment, MS Visio, MS PowerPoint, MS Word and Wikis are used by most business people to document processes. In the enterprise IT world, when we deal with process modelling for the process execution purpose, i.e. when building workflow management systems, the business process modelling languages are designed for IT expertise with strong domain knowledge.

Business process modelling languages like Petri Net, Event Process Chains (EPC), Business Process Modelling Notation (BPMN), UML activity diagram, UML collaborative diagram or Business Process Execution Language (BPEL) require major training efforts and thus are only accepted by certain groups of people. Our work aims to enable the end user and introduces an intuitive and guided way to model business processes.

van der Aalst et al. (2000, 2001) propose a new modelling approach based on proclets, performatives and channels. A proclet represents only one aspect or one element of a whole business process using Petri Net style notations. Our end-user process language can also be used to describe a proclets or used as a requirements acquisition tool to describe a part of a whole process.

Our previous work has identified key issues in the implementation and use of a lightweight business process modelling environment (de Vrieze et al. 2009, 2011, Xie, et al. 2009). One of the key issues is in supporting end-user process modelling. Xie et al. (2010a,b) discuss lightweight business process modelling issues only from a control flow perspective.

After examining related work from the collaborative business process and business process modelling perspectives, we further look at the relationships between OASIS reference architecture foundation for SOA (SOA-RAF) and our work.

First, The SOA-RAF as a standard remains abstract in nature. It focuses on an approach to supporting business with the information technology needed to support it (OASIS 2011). In our work, we look at how to support real business processes (i.e. more traditional workflow systems supported business processes/collaborations), which are more complicated than current service orchestration or service choreography. There are differences between traditional workflow system-supported business processes/process collaborations and current service orchestration and choreography-supported business processes/process collaborations. The traditional workflow system-supported business processes/process collaborations contain more sophisticated control flow and data flow patterns. For example, they support multi-instances of activities and different data patterns. Current service composition and orchestration and choreograph solutions do not support multi-instances and only provide a global data pattern. In our work, full control flow and data flow patterns are not completely supported. But we do support multi-instances for the control flow and go beyond the global data pattern for the data flow. Our work does not provide a reference architecture but looks at new a process modelling method and implementing a process engine with support for a subset of process-related control flow and data flow patterns based on the principles of end-user friendly, lightweight, context awareness.

In general, the SOA-RAF provides a common framework and language for discussing SOA from many different views: enterprise architects, business IT architects or other senior executives. Our work provides more concrete architecture (de Vrieze et al. 2009, 2011), modelling methods (Xie, et al. 2010a,b) and an implemented execution engine (de Vrieze et al. 2011).

3. End-user process modelling requirements and principles

In this paper, end users refer to ordinary users who have a professional background, knowledge and understanding of the tasks and processes they are trying to support but who are not professional software developers (or modellers). Neither have they received significant training in programming, modelling or system design. To enable such end users to participate in modelling and execute the process models, there are five technical requirements.

- The language is designed for the end users without professional modelling skills. The complexities of process creation need to be hidden. The way of modelling needs to be integrated in the way people do their work, i.e. directly within the context of their work.
- The final process model should be an executable process model. The end-user process modelling language is designed for an end user to specify and automate business tasks and activities. Therefore, the modelling language must be executable. For being the executable process modelling language, the language needs to provide logic information, which is associated with precise semantics that can be used to automatically validate and simulate business processes.
- Another requirement is providing easy access to the modelling environment for end users. To ensure business user buy-in, there needs to be a focus on usability. This comprises both simplicity of the approach and a low footprint for the solution. Ideally, users can model processes using
simplified notations in a Web 2.0 environment and draw up on modelling best practices.

- Active participation is required for supporting end-user process modelling by combining both structured and unstructured information and publishing them as a single point of reference to the organisation, such as blogs or wikis. A process wiki allows the community to participate in discussions and to provide comments or ratings in relation to process proposals.
- There are also requirements for the end-user process editor. It should provide an intuitive user interface. It needs to ensure the user is given advanced guidance during the modelling activities. Furthermore, errors, misspelling and inconsistencies should be avoided from the beginning.

Following the summary of the requirements for end-user process modelling, it is possible to detail our four modelling principles for business process mashups (Xie et al. 2010a):

- The first principle is to minimise the need to design from scratch. For many end users, it is difficult to start to create a model from scratch and it is easier to design a process model based on an existing example. Learning by examples is a common learning style in a community environment where pre-design, highly reusable process models and templates can be selected. These existing models can also serve as a learning model or a base for end users to create a new process model.
- The second principle relates to context awareness. Context awareness helps users to select not only the right model elements but also with mapping labels to concepts. Duplication of concepts should be avoided even where users with different backgrounds or speaking different languages use different terms to refer to the concepts. The context-aware principle allows identification, storage and representation of business process models. A business process is instantiated depending on specific context information, e.g. time, location, user profile, device and usage conditions.
- The third principle involves the concept of reusability. In the process modelling, field workflow patterns are regarded as reusable elements. We adopt a subset of workflow patterns as a basis of our process patterns and process templates. Process patterns, process templates, process fragments, reference models, best-practice models and further example models are provided in association with the process modelling language; these are all reusable.
- The fourth principle is the community. A Web-based community environment provides recommendations and ranking, accompanied by commenting on process models and templates. These process templates will initially be related to usage in the early stages of development, but we expect them to be both significantly improved and expanded within the community environment.

The four principles are not independent; rather they support and interact with each other. The minimising of learning effort is supported by reusability of the process template or models and by retrieving process templates or models using context information along with the community providing an active learning environment.

In the following sections, we focus on end-user collaborative process model and control flow and data flow design for end-user process modelling.

4. End-user collaborative process mode

Figure 1 illustrates the end-user process model of collaborative processes using UML class diagram notations. This model describes the relationships among all elements within the collaboration process model.

There are two parts of the collaboration process model. The side to the right of the dotted line in Figure 1 presents the structure of a collaboration model. The left part includes the label information of the collaboration process model. Both parts are sharing context information ‘ContextInfo’ and a common ontology ‘OntologyComponent’.

ProcessModel has different ‘States’ and different ‘States’ can be transferred by an ‘activity’. A single ‘ProcessModel’ comprises ‘activities’ which captures cohesive business functionalities. For example, an ‘invoice process model’ includes activities ‘requires invoice’, ‘create invoice’ and ‘post invoice’. The process which each process model represented is eventually invoking services to achieve the results.

An ‘Activity’ is either an ‘AtomicActivity’ or ‘CompositeActivity’. The AtomicActivity is the smallest unit of activity, e.g. a concert Web service or a computing resource. A ‘CompositeActivity’ consists of several other activities, either atomic or composite. An CompositeActivity entails a logical functionality that invokes an (atomic or composite) ‘ServiceOperation’ of another ‘Service’, or it can be a control flow construct like decision, merge, fork and join, or it can simply be a human task.

‘ProcessPattern’ refers to eight workflow patterns (van der Aalst et al. 2003) (see Figure 3).
‘ProcessTemplate’ is a combination of different process patterns which represents a process segment within a business domain. We separate ‘NonSequenceFlowObject’ from ‘SequenceFlow’ to ensure the alternating ordering of ‘SequenceFlow’, i.e. arrows and other objects, and thus avoid the situation where two ‘SequenceFlow’ elements link directly to each other. 

NonSequenceFlowObject includes ‘Start’, ‘End’, ‘Activity’, ‘SequenceFlow’, ‘Operator’, etc. which can be found from Figure 2. An abstract concept ‘Operator’ includes ‘OR’, ‘AND’ and ‘XOR’ gateways.

The left part of Figure 1 contains the label information of the collaboration process model. Every process model has a number of model elements, which

Figure 1. End-user collaborative process model.

Figure 2. Adopted symbols from BPMN.
can be process activities, sub processes or other process segments. These model elements are extended by labels that are human readable. There can be multiple labels per element, where the context is used to select the most appropriate.

Every business concept has a unique semantic meaning and thus is linked to a unique concept of ontology. In order to make these abstract business concepts meaningful to different end users, *business terms* are used to describe a business concept in a natural language. Both, the business concept and the business terms might be restricted to a specific business context. For example, a business activity with the concept name ‘SEND INVOICE’ may have the following business term representations ‘send invoice’, ‘send customer invoice’ or ‘send CI’. The business terms are based on a certain business term grammar. The grammar allows the system to be aware of grammatical variations of a term (e.g. singular or plural). The ontology provides a common understanding regardless of business process modelling notation, syntax or terminology used.

5. Control flow modelling

In order to simplify business process modelling, models must be highly reusable, giving process flexibility and minimising designs made from scratch. There is a wide agreement that patterns can both accelerate the process of designing a solution and reduce modelling time (Medicke and McDavid 2004). Patterns enable participants of a community to communicate more effectively, with greater conciseness and less ambiguity (Buschmann *et al.* 2007, Tran *et al.* 2007). A pattern is an abstraction from a concrete form which keeps recurring in specific non-arbitrary contexts (Riehle and Zullighoven 1996). The use of patterns is a proven practice in the context of programming, as shown by the impact made by the design patterns of (Gamma *et al.* 1995) as well as various other patterns.

We have adopted eight symbols from BPMN. The symbols adopted are presented in Figure 2. This notation was chosen because BPMN has been well accepted by business users (Zur Muehlen 2008) and our research is focused on the design of an executable process modelling language and not on the design of

![Figure 3. Business process patterns.](image-url)

![Figure 4. Pharmaceutical manufacturing collaborative scenario.](image-url)
notation. This minimum number of notation means a user will have minimised learning efforts. Research shows that the average subset of BPMN used in process models consists of just nine different symbols (Zur Muehlen 2008). The notations for pool and lanes are dropped because they are used for expressing process collaborations. End-user process modelling process collaboration can be expressed from an individual business partner’s perspective and an integrated view will be generated by the process editor. We have further adapted the notation for multiple instance business processes.

We have also used the eight most frequently used workflow patterns (see Figure 3) from van der Aalst et al. (2003) as our process patterns. The patterns can be captured within most common business process models and have a well-defined formal foundation.

These workflow patterns from van der Aalst et al. (2003) are however too fine grained and have insufficient information on the context and results to represent a reusable solution. Therefore, we introduce process templates that are different combinations of process patterns. The processes represented by a process template are sound. Certain process templates can be enriched with the information that they are valid for different domains, i.e. business context. The examples of the process templates are shown in Figure 5 with an example case context in Section 7.

Using patterns, the soundness of process models can be guaranteed in certain ways (Gschwind et al. 2008). The patterns can further be used to develop domain-specific process templates. Following context-aware modelling principles, it is possible to adapt the best-fitting process artefacts during selections, e.g. a user can fill in context information such as particular industry, location and process name; a specific process template will appear to the users accordingly.

6. Data flow modelling

Data flow patterns aim to capture the various ways in which data are represented and utilised within processes. Russell et al. (2004) present 40 data-related patterns for process-aware information systems which are divided into four distinct groups: data visibility, data interaction, data transfer and data-based routing.

- Data visibility relates to the extent and manner in which data elements can be viewed by the various activities of a process.
- Data interaction focuses on the manner in which data are communicated between active elements within a workflow.
- Data transfer considers the means by which the actual transfer of data elements occurs between activities or subprocesses and describes the various mechanisms by which data elements can be passed across the interface of an activity or subprocess.
- Data-based routing characterises the manner in which data elements can influence the operation of other aspects of the workflow, particularly from a control flow perspective.

See Di Lorenzo et al. (2009) for a discussion on data integration-related issues of the current mashup tools. Data flow operators have to be performed on either the structure of the data or on the data itself. Data are generated and updated using different data refresh plans, such as pull and push strategies. The pull strategy is based on frequent and repetitive requests from the client; there are two ways to handle pull interval. In a global strategy, this will be set for the whole application. But in the local strategy, each data source is given its own refresh interval. In the push strategy, the client does not send requests but registers to receive update notifications.

According to the definition of the data flow patterns, the data operators and data refresh strategies only cover data transfer, data interaction and data routing. Data visibility is not covered by current data-oriented mashups. Most popular data operators in the current data-oriented mashups are ‘Union’, ‘Join’, ‘Sort’ and ‘Filter’. A comparison of IBM mashup tool, Yahoo Pipes and Microsoft popfly is given in Table 1. It shows a rough mapping different coverage of a data flow perspective between a process-oriented mashup and data-oriented mashup tools.

In short, a significant difference between data-oriented mashups and process-oriented mashups is whether ‘data visibility’ is covered. Without different ‘data visibilities’ at different levels of business processes and different instances of business processes, certain business processes, such as multiple business processes, cannot be supported which is not a problem for data-oriented applications, but it is a problem for process-oriented applications. Therefore, data visibility is certainly a topic of interest in our research on the topic of our business process mashup.

From the business process application perspective, all data flow operators can be implemented using data transfer Web services. In a business process enterprise mashup, the mashup providers do not limit the data flow operators. Collecting a library of Web services which can support different data operators is useful and powerful for end users.

We have chosen 13 out of 40 dataflow patterns, i.e. three dataflow patterns for data visibility, five patterns for data interaction, three patterns for data transfer mechanisms and two patterns for data-based routing.
Figure 5. Example of process templates.

(a) Never wait, execute every time

(b) Wait for all to finish

(c) Wait for the first to finish and ignore the others
Multi-instance patterns are selected, which BPEL does not support. In the context of data visibility, three data flow patterns are supported.

- **Scope data** pattern introduces a pattern where data elements can be defined which are freely accessible by a subset of activities but are not available to other activities in the process instance.
- **Process instance** data pattern presents a pattern where data elements are supported which are specific to a process instance. They can be accessed by all components of the process model during the execution of the process instance.
- **Environment data** pattern is a pattern where data elements which exist in the external operating environment are able to be accessed by components of the process during execution. Direct access to environmentally managed data by process activities or instances during execution can significantly simplify processes and improve their ability to respond to changes in the broader operational context. External data may be sourced from a variety of distinct locations including external databases, applications that are currently executing or can be initiated in the operating environment and services that mediate access to various data repositories and distribution mechanisms.

Concerning the data interaction, five data flow patterns are adopted.

- **Global shared data** pattern describes an ability to communicate data elements between one activity instance and another within the same instance. The same data elements are shared between activities typically via access to globally shared data. No explicit data passing is required. Variables can be bound to scope within a process definition which may encompass a number of activities, but there is also the ability for messages to be passed between activities when the control passes from one activity to another, as in BPEL4WS.
- **Activity to environment – push-oriented** data pattern is where an activity initiates the passing of data elements to a resource or service in the operating environment. The passing of data from an activity to an external resource or service is most likely to be initiated by the activity itself during its execution. Depending on the specific requirements of the data passing interaction, it may connect to an existing API or service interface in the operating environment or it may actually invoke the application or service to which it is forwarding the data.
- **Environment to activity – pull-oriented** data pattern allows a process activity to request data elements from resources or services in the operating environment. Activities require the means to proactively seek the latest information from known data sources in the operating environment during their execution. This may involve accessing the data from a known repository or invoking an external service in order to gain access to the required data elements.
- **Environment to activity – push-oriented** data pattern provides ability for a process activity to receive and utilise data elements passed to it from services and resources in the operating environment on an unscheduled basis. This pattern is particularly important in areas of volatile information where updates to existing data elements may occur frequently but not on a scheduled basis. This pattern relates to the ability...
of tasks to receive new items of data as they become available with without needing to proactively request them from external sources or suspend execution until updates arrive.

- **Activity to environment – pull-oriented** data pattern allows an activity to receive and respond to requests for data elements from services and resources in the operational environment. In some cases, the requests for access activity instance data are initiated by processes outside the process environment. These requests need to be handled as soon as they are received but should be processed in a manner that minimises any potential impact on the activity instance from which they are sourcing data.

Three data transfer mechanisms are supported in our lightweight process modelling language:

- **Data transfer by value – incoming** pattern allows an activity to receive incoming data elements by value, removing the need to have shared names or a common address space with the component(s) from which it receives them. In BPEL4WS, data elements are passed between activities using messages.
- **Data transfer by value – outgoing** pattern allows a process component to pass data elements to subsequent components as values, removing the need to have shared names or a common address space with the component(s) to which it is passing them. BPEL4WS provides the option to pass data elements between activities using messages, an approach which relies on the data between process components by value.
- **Data transfer by reference – unlocked** pattern allows communicating data elements between process components by utilising a reference to the location of the data element in some mutually accessible location. No concurrency restrictions apply to the shared data element.

Two ways of data-based routing are introduced in our lightweight process modelling language:

- **Event-based task trigger** data pattern indicates an external event to initiate an activity. There are three distinct scenarios for the pattern. First, the activity instance to be initiated is the first task in the process; this is equivalent in control terms to starting a new process instance which is the first ask instance. Second, the external event is triggering the resumption of an activity instance in the process that is in the middle of a workflow process; the activity instance has already had control passed to it but its execution is suspended pending occurrence of the external event trigger. Finally, the activity instance is isolated from the main control flow in the workflow and the only way in which it can be initiated is by receiving an external event stimulus.
- **Data-based routing** data pattern alters the control flow within a process instance as a consequence of the value of data-based expressions. Without data-based routing constructs, it would not be possible for a process to alter its operation in response to the data it was processing. Data-based routing provides the ability for the completion of one activity instance to trigger one or several subsequent activity instances depending on the outcome of an expression based on the values of various process data elements. It allows for example to treat low value purchases different from high value ones, even when initiated through the same process.

These 13 dataflow patterns provide an expressive power for the end-user language. We do not expect that end users will be able to understand them completely when they first start to complete a mashup. However, we expect that they will be easily learned from the existing process templates, reference models and other process fragments.

### 6.1. Relation among control flow, data flow and process validation

There are links between control flow and data flow in many ways. Data visibility in the data flow pattern can express where data elements are freely accessible by a subset of activities in the control flow. In the control flow, the conditions of the splits or joins can also indicate which data will be involved. For example, **Activity to environment – push-oriented** data pattern is where an activity initiates the passing of data elements to a resource or service in the operating environment.

Each control flow pattern can relate with any data flow pattern, vice versa. In a modelling environment, data type of parameters using the different data patterns is specified as an attribute of each activity within a control pattern.

On the one hand there is a common belief in the importance of patterns. On the other hand there is limited support for using patterns in today’s business process modelling tools. The basic control flow patterns (van der Aalst et al. 2003) are available in most business process modelling tools, and the YAWL system (van der Aalst and Hofstede 2005) supports all control flow patterns. However, applying even a basic pattern is under the full responsibility of the user.
during the process modelling stage, i.e. most modelling errors are a result of incorrect combinations of the exclusive choice, parallel split, simple merge and synchronisation patterns (Koehler and Vanhatalo 2007). Keeping this in mind, we are designing a process modeller which will provide assistance to end users applying patterns to maintain the soundness of the process model.

Sections 4–6 explained the end-user process model and how do model control flow and data flow in the end-user process modelling. We provide a pharmaceutical manufacturing collaborative scenario to demonstrate how the process can be modelled using our modelling language and method.

7. Model pharmaceutical manufacturing collaborative activity

At the pharmaceutical industry, all manufacturers are facing skyrocketing costs due to significant data accuracy issues. Multiple, incompatible data storage systems and a fragmented organisation are created by a series of acquisitions. Some departments use databases, while others keep a track of critical product information on spreadsheets (TIBCO 2006). The cost of manually reconciling and transferring data between systems, spreadsheets and departments is spiralling out of control.

In this scenario, we are looking at a collaborative process within a pharmaceutical company. There are seven departments involved in the process: marketing and sales, R&D, quality assurance (QA), production control, manufacturing, global packaging and delivery. We first demonstrate the collaborative process model. Further, we explain how the end user can get to this process model.

A user name Lorraine, who works for marketing and sales department, uses a business process mashup to handle a new customer requirement. This new requirement may only relate to packaging issues within certain region. The requirements can be declined or accepted.

Lorraine forwards a customers’ requirement to both R&D and QA departments. R&D and QA will each evaluate the requirements. The result of R&D evaluation can be a resolved product process or a suggestion to decline. After results from both R&D and QA come out, the requirement will be forwarded to the product control department. The production control department will check out production scheduling and product cost accounting, and a final decision of acceptance or declination will be sent back to Lorraine. An approved decision will be forwarded to the manufacturing department. Further activities such as global packaging and delivery follow. Figure 4 depicts a model of this collaborative process. This figure uses the symbols mentioned in Figure 2.

In short, by using patterns, the soundness of process models can be guaranteed in certain ways (Gschwind et al. 2008). These patterns can further be used to develop domain-specific process templates, and by following context-aware modelling principles, it is possible to adapt the best fitting process artefacts with appropriate (context dependent) selections.

Consider a simple example in the context of Lorraine which is an aggregating decision process of the kind we would want to support. Specifically, Lorraine wants to receive suggestions based on the reviews of R&D, QA and Production Control departments. After the activity ‘Setting Customer Requirements’ by Lorraine, two possible activities ‘Check Possible Production Process’ by R&D and ‘QA Customer Requirements’ by QA are triggered. The results of above two activities will be forward to Production Control at the same time. Activity ‘Check Production Scheduling and Cost Accounting’ will be performed by Production Control. Based on the results from R&D and QA and the result of checking scheduling and cost, the final decision on whether approve or decline the requirements from the customer will be sent to Lorraine. If the decision is to approve the customer requirements, further activities ‘Manufacturing’, ‘Global Packaging’ and ‘Delivery’ will continue.

Now we explain how Lorraine could select different process templates for establishing her final model. In our modelling environment, the modeller can select different process templates from recommendations. The recommendation can be based on the user profile and context information. When the limited recommendations are provided to the end users, the differential of the similar templates is emphasised.

Figure 5 includes three very similar process templates. In Figure 5(a), we see that the activity ‘forward decision’ is executed each time one of the three preceding tasks completed. It is a kind of ‘never wait, execute every time’. Figure 5(b) is similar but combines the individual decision into one piece of information. Therefore, it waits until each of the activities enabled by ‘Set Customer Requirements’ completes. If one of activities cannot be performed, activity ‘forward decision’ is delayed until all have completed. This template is finally adopted for the final process model. Figure 5(c) enables all three activities ‘Check Possible Production Process’, ‘QA Customer Requirements’ and ‘Check Production Scheduling and Cost Accounting’. The Activity ‘forward decision’ is executed after the first activity has completed. After ‘forward decision’, all other running activities are cancelled.
8. Conclusions and future research

Web-based technologies not only impact our communication patterns but also provide opportunities to bring information, knowledge and automation support to our daily activities. Process composition in a business process mashup provides an agile approach to adapting to changing business environments. The requirements of end-user process collaboration modelling have been analysed. Following requirements, principles of the end-user process collaboration modelling were identified. Further, the end-user process collaboration model was presented. This model describes relations among processes, services and context information. The current version of the end-user business process modelling language has eight graphical symbols and eight control flow patterns. A range of executable process templates are in use and development is continuing. Thirteen dataflow patterns have been chosen to support the current needs of perimeter enterprise applications. The primary evaluation of the end-user process model method and language can be found from Xie et al. (2010a,b).

We are starting new work in several areas. To fully exploit the potential of Web-centric compositions, we are exploring a lightweight business process modeller and context-aware process modelling. We are aiming to allow users to pick up process models and run them with only minor modifications. Further, we are also interested in annotating the resources using WSMO-Lite and Micro-WSMO.

We will also look at constraints linking service inputs and outputs that are currently not expressed in the process model and enrich the process so that the modelled processes can be composed and orchestrated conveniently in the ‘runtime space’.

Finally, we would like to emphasise that business process mashups will not completely replace core business process management systems. Business process mashup applications address different needs and are built for just a handful of users, applications that are used for only a few weeks or months or situational applications that address a small piece of functionality. Business process mashups focus on agility over stability and may use human involved exception handling instead of extensive manual correctness validation. Business processes can serve users with different modelling skills for collaboration within a virtual enterprise environment.

Acknowledgements

The authors would like to thank the EU SOA4All and FAST projects. Part of the work by Dr. Lai Xu and Dr. Paul de Vrieze was performed at the SAP Research, Switzerland.

References


