Complex Event Processing for Context-Adaptive Business Processes

Gabriel Hermosillo, Lionel Seinturier, Laurence Duchien

INRIA Lille - Nord Europe - University Lille 1 - Laboratoire LIFL - CNRS UMR 8022

Abstract

As the amount of data created by today's pervasive environments increases exponentially, there is a stronger need to decipher the important information that is hidden among it. By using complex event processing, we can obtain the information that really matters to our organization and use it to improve our processes. However, even when this information is discovered, the processes remain static and cannot be changed to adapt to the actual scenario, diminishing the advantages that can be achieved. In this paper we present CEVICHE, a framework that combines the strengths of complex event processing and process adaption. CEVICHE deals with the different languages used to detect complex events while presenting a single interface to the user.

Keywords: Complex Event Processing, BPEL, Context adaptation

1. Introduction

In today's pervasive and ubiquitous environments, the amount of data sent by the different devices in the environment is constantly growing and there is a strong need to transform this data into relevant information that can help to make the right decisions at the right moment. To take advantage of this information, it is required to discover time, cause and belonging relationships between the different data to give them a more useful meaning according to the context in which they are obtained, to generate complex events from simple and isolated events. Complex Event Processing (CEP) is an emerging technology for finding relationships, in real-time, between series of simple and independent events from different sources, using previously defined rules [1]. For example, if pressure decreases, the air is humid and the weather is warm, a CEP system could infer that a hurricane is being formed.

The CEP technology can be used, among a lot of other things, to enrich the enterprise's existing processes, by introducing rules that will allow the capture of relevant information from the different steps of the process [2]. For example, in the scenario of a retail store that keeps a record of its inventory in an existing ERP system and wants to keep a live monitoring of its stocks in order to prevent shortage. To achieve this, the store installs a CEP engine that will monitor the products movements through their life cycle in the store's process by receiving and analyzing all the events generated by every change in the state. Since the objective is to monitor inventory, the CEP engine will only keep the events related to changes in the inventory and forget about the rest. By creating the necessary CEP rules, the configuration is set to specify the lowest acceptable stock of product that the store can have to avoid a shortage, e.g. a 10% for normal products and a 5% for some low-demand products. Whenever a product reaches a minimum, the CEP engine alerts the managers so they can make a supply order.

In addition to that, CEP can also be used to predict unexpected situations. To complement the previous example, we can say that because of a global pandemic alert, hand sanitizers are very popular and are selling a lot more than usual. Given this demand, the store will run out of hand sanitizer before they can resupply it, even with the minimum stock alert. By adding some specialized CEP rules to analyze the frequency of sells of each product during the last 4 or 5 hours, the engine could polarize these values to know in advance (if the sells rates are kept) that it will need to resupply before the expected time, which will allow them to react in time even before it reaches the minimum level.

Email address: {firstName.lastName}@inria.fr (Gabriel Hermosillo, Lionel Seinturier, Laurence Duchien)

By doing the analysis of these kind of complex events, we can also prevent the theft of merchandise from stores, by creating relationships between the amount, kind and movement of the products inside the store [3]. However, there are some occasions in which it is not enough to be able to obtain relevant information from simple raw data, but there is also a need to adapt the process according to the new context in order to continue in an optimal way, and this is why we developed CEVICHE (*Complex EVent processIng for Context-adaptive processes in pervasive and Heterogeneous Environments*).

The purpose of CEVICHE is to create a system capable of generating, from a process definition, a context adaptation according to a set of previously defined rules, using the information obtained from each step of the process, and also from additional data sources that can be incorporated to cover different needs. For example, if the retail store needs to send a load of perishable products and the sensors detect that the temperature outside is too hot, e.g. 40°C, CEVICHE could adapt the process so that all the deliveries of perishable products are held until dusk (when the temperature should be lower).

With CEVICHE, we will improve the adoption of CEP by providing a solution to one of the problems it has, which is that until today there is no standard way to define the rules that relate the simple events to generate complex events, so we bring forward the *Standard Business Process Language* (SBPL) and a translation framework that will be able to adapt a single process definition to different CEP engines. The syntax of the SBPL will be based on the *Event Processing Glossary* [4] created by the Event Processing Technical Society and the *Core Business Vocabulary* created by EPCglobal [5] to make it as standard as possible. The translation is done by using a special plug-in for each CEP engine, so at the end the users only have to specify the process definition once even if they want to change to another engine, just by using the corresponding plug-in.

The rest of this paper is organized as follows. Section 2 explains the CEVICHE framework and its architecture. Section 3 presents some of the related work. Finally, section 4 concludes and discusses some future work.

2. The CEVICHE framework

CEVICHE is a framework that intends to facilitate the integration of CEP into existing business processes and to allow these processes to be dynamically adapted to different circumstances. With this framework we want to address mainly four challenges: adaptation, dynamicity, integration to business process, and non-dependency to a specific CEP engine. To face the first challenge, adaptation, we use an aspect-oriented approach, which allows the system to add or change functionality and facilitates the task of separating concerns. By analyzing the current events with CEP and using context information, we can decide when and how to adapt the system at run-time, thus giving a solution to the dynamicity challenge. For the third challenge, we use these solutions in a business process environment by integrating them with the *Business Process Execution Language* (BPEL) . Finally, CEVICHE aims to be able to work with any CEP engine available. For that, as part of this framework, we defined the SBPL, which will gather all the information about the processes, contextual environment, business rules, and adaptation conditions. Using the specifications in the SBPL file, CEVICHE will translate them to the chosen CEP engine using a translation plug-in, making the system independent and allowing the user to choose the engine she prefers.

2.1. CEVICHE architecture

CEVICHE is composed of three main parts: a user interface to create the SBPL file, a translation framework to manage the plug-ins for each CEP engine, and an aspect manager to deal with the process adaption. The aspect-oriented approach allows us to separate cross-cutting concerns from the core business processes and to integrate new functionalities or change the existing ones at run-time. CEVICHE also relies on different technologies to achieve the process adaptation, as shown in Fig. 1. First of all, CEVICHE needs a CEP engine that will be in charge of capturing and filtering all the events surrounding the process to create complex events that are relevant to the business. In order to manage the business processes, we need a *Business Process Management* (BPM) engine or a CEP engine capable of doing this. For the process definition, CEVICHE uses BPEL, since it is a standard language used in the industry for this purpose. BPEL is an orchestration (focused on the execution of the process) XML-based language that allows the definition of a process and its interaction with different entities using Web Services.

To deal with context-awareness we use COSMOS [6], a component-based development framework that manages the context information in ubiquitous environments. COSMOS allows the building of a hierarchical network of context

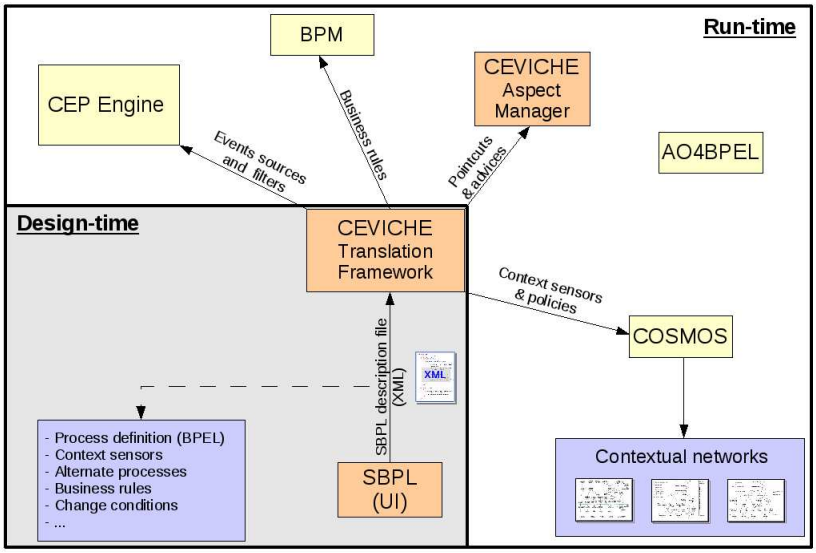

Figure 1: The architecture of CEVICHE

nodes that will gather the information obtained by each component and send it to the upper nodes to get a general response. Finally, we use an aspect-oriented extension to BPEL4WS called AO4BPEL [7], to adapt the existing processes. The advantage of using AO4BPEL is that we can change the business process specifications at run-time without the need to redeploy them, avoiding to lose all the ongoing transactions by doing that.

To configure the system, the user is provided with an interface where she can capture the business rules, process definitions and all the context data sources needed to adapt the processes. This information is sent in an SBPL file to the translation framework, which uses a specialized translation plug-in to adapt the information in the file to the specific CEP engine's format. This way, whenever the user wants to use another CEP engine, the only thing that needs to be done is to change the plug-in, without rewriting all the specifications of the business processes. The translation framework will then use that information to provide the CEP engine with the business rules in the appropriate format and to send the corresponding part of the information to each component of the architecture, leaving all the parts ready to interact during the process execution.

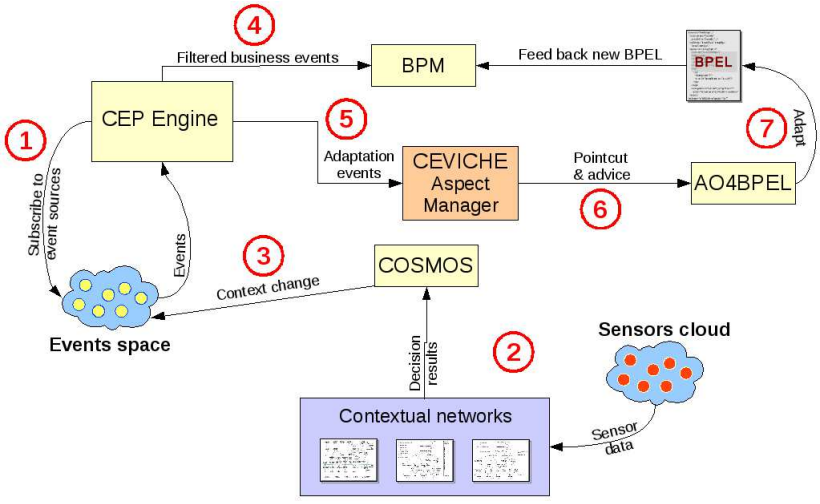

Figure 2: Information flow for adaptation

Once the initial setup is ready and all the components have been properly configured, the process starts and the information begins to flow from one component to the other, as seen in Fig. 2. In the first step, the CEP engine subscribes to the different sources of events, here called the *events space*, which will provide the engine with the information it needs to take decisions and create complex events. At the same time, in step 2, COSMOS will use the context policies provided by CEVICHE during the setup process to generate the context hierarchies needed to monitor the context information. Whenever there is a change in this information, it will be reported as an event and sent to the CEP engine (step 3). The CEP engine will gather all the events, filter the interesting ones according to the business rules and find relations that can generate complex events, which are then sent to the BPM to be considered in the process (step 4). The CEP engine is provided with some rules, which specify the conditions under which the process would need to be adapted. Whenever these conditions are met, the engine sends the information about the process to the CEVICHE Aspect Manager (step 5). Depending on the information received, the Aspect Manager will select the steps of the process that need to be changed and the corresponding changes to be done. Then, it uses AO4BPEL (step 6) to modify the process at run-time to be considered by the BPM (step 7).

3. Related work

CEP and BPEL. As we mentioned earlier in this paper, CEP is an emerging technology and the use of it in the business processes is a recent topic of interest and research [8, 9]. An analysis of scenarios of composite event patterns comparing BPEL and BPMN is done in [8]. The authors analyze patterns of events that go from conjunction and cardinality to time relations and event consumption possibilities. The conclusion of their study is that neither BPEL nor BPMN are capable of supporting complex event scenarios in their specifications, so there is a need to integrate event pattern descriptions into the process definitions language, but they do not mention the need to adapt the process according to those complex events.

In [9], we present a service to add traceability to the RFID tagged products by using Complex Event Processing. When the RFID events are captured, they are transformed into business events that correspond to the business rules defined in the process which allows the users to have a better understanding of the status of their products, i.e. the product's location and the environment it has been exposed to.

Business process adaptation. The need to adapt a process has been a topic of interest in the recent years and there have been different approaches that offer solutions to it [10, 11, 12, 13]. In [10], the authors propose to deal with process adaptation by adding a web service repository that will handle the web services to invoke in each case. Whenever an invocation of a web service is done, the call is intercepted and the repository is checked for changes in the process definition, before the invocation of a web service. If there have been some changes, then it examines the available web services in the repository and chooses the one that best suits the criteria, otherwise the invocation is executed as usual.

The authors in [11] use an aspect-oriented approach introducing `executable models`, which are used to represent the cross-cutting concerns. They use `open objects`, which are representations of the state of the elements in the model, to monitor the invocation of services and adapt the process by weaving the interaction with other models before (activation) and after (deactivation) the call to the service.

Another aspect oriented implementation, using the Spring .NET framework, is presented in [12]. They use a contract-based approach to assign a web service to each instance of an execution call. To achieve adaptation of the process they can change the contract at run-time to assign a new web service for the call. They are also capable of adapting an existing implementation of a web service by using aspects to weave the new behavior.

An adaptation of the BPEL language called VxBPEL is presented in [13]. Here, the authors insist on the need of flexibility and variability in the service-based systems and the lack of them when deploying BPEL processes. They extend the BPEL language to add new elements like `Variation Points`, which are the places where the process can be adapted and `Variants`, which define the alternative steps of the process that can be used. VxBPEL also accepts new `Variants` to be added at run-time, allowing the systems to be adapted without redeploying the process.

BPEL context adaptation. The work that is closer to our proposal is the one presented in [14]. Here the authors present a plug-in based architecture for self-adaptive processes that uses AO4BPEL. Their proposal is to have different plug-ins with a well-defined objective. Each plug-in will have two types of aspects: the `monitoring aspects` that

will check the system to observe when an adaptation is needed and the `adaptation aspects` that will handle the situations detected by the monitoring aspects. Whenever the conditions of a `monitoring aspect` are met, it uses AO4BPEL to weave the `adaptation aspects` into the process at run-time. In our approach we deal with the monitoring part using the rules deployed in the CEP engine, which will detect special situations (by relating simple events) and select the aspects to be used to adapt the process.

Advantages of CEVICHE. The advantage of CEVICHE's architecture is that it is not necessarily linked to the third party technologies that it uses. So, even though it is not its main purpose, just as it can change from one CEP engine to another, the way to adapt the processes could also be changed from AO4BPEL to some other approaches, by creating the proper plug-in for it.

For the work presented in [14], an advantage of their work is that the monitoring aspects can be hot-deployed to their BPEL engine while with our approach the changes in the rules might not be considered at run-time, depending on the CEP engine. On the other hand, this difference also shows an advantage for our proposal, since we are not tied to a single engine and we can use any CEP rules already defined for the monitoring process, while in their case we would need to create a new plug-in for each new situation we want to monitor.

4. Conclusions and future work

Process adaptation and Complex Event Processing are two topics that are creating a lot of interest in the research community, however there is still no integration of both domains. In this paper we presented CEVICHE, a framework that integrates CEP and process adaption according to the context and that deals with adaptation, dynamicity, integration to business process, and non-dependency to a specific CEP engine. As part of the CEVICHE framework, we proposed the SBPL to deal with the different languages of the CEP engines, allowing the users to write their process specifications only once and deploy it in the engine they want. As future work, we plan to work on the definition of a RESTful architecture to leverage on the deployment of CEVICHE components and facilitate the evolution of the architecture by adding or changing components.

5. References

[1] Guangming Wang and Gonglian Jin. Research and Design of RFID Data Processing Model Based on Complex Event Processing. In *CSSE '08: Proceedings of the 2008 International Conference on Computer Science and Software Engineering*, pages 1396–1399, Washington, DC, USA, 2008. IEEE Computer Society.

[2] Tao Ku, YunLong Zhu, and KunYuan Hu. A Novel Complex Event Mining Network for Monitoring RFID-Enable Application. In *PACIIA '08: Proceedings of the 2008 IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application*, pages 925–929, Washington, DC, USA, 2008. IEEE Computer Society.

[3] Nicholas Huber and Katina Michael. Minimizing Product Shrinkage across the Supply Chain using Radio Frequency Identification: a Case Study on a Major Australian Retailer. In *ICMB '07: Proceedings of the International Conference on the Management of Mobile Business*, page 45, Washington, DC, USA, 2007. IEEE Computer Society.

[4] EPTS Event Glossary v1.1. `http://www.ep-ts.com/component/option,com_docman/task,doc_download/gid,66/Itemid,84`, 2008.

[5] EPCglobal's Core Business Vocabulary. `http://www.epcglobalinc.org/standards/cbv`, 2009.

[6] Romain Rouvoy, Denis Conan, and Lionel Seinturier. Software architecture patterns for a context-processing middleware framework. *IEEE Distributed Systems Online*, 9(6):1, 2008.

[7] Anis Charfi and Mira Mezini. Ao4bpel: An aspect-oriented extension to bpel. *World Wide Web*, 10(3):309–344, 2007.

[8] Alistair P. Barros, Gero Decker, and Alexander Großkopf. Complex events in business processes. In *BIS*, pages 29–40, 2007.

[9] Gabriel Hermosillo, Julien Ellart, Lionel Seinturier, and Laurence Duchien. A Traceability Service to Facilitate RFID Adoption in the Retail Supply Chain. *International Workshop on RFID Technology (IWRT)*, 2009.

[10] Fernando Antônio Aires Lins, José Carlos dos Santos Júnior, and Nelson Souto Rosa. Adaptive web service composition. *SIGSOFT Softw. Eng. Notes*, 32(4):6, 2007.

[11] Mario Sánchez and Jorge Villalobos. A flexible architecture to build workflows using aspect-oriented concepts. In *AOM '08: Proceedings of the 2008 AOSD workshop on Aspect-oriented modeling*, pages 25–30, New York, NY, USA, 2008. ACM.

[12] Syed Saif ur Rahman, Nasreddine Aoumeur, and Gunter Saake. An adaptive eca-centric architecture for agile service-based business processes with compliant aspectual .net environment. In *iiWAS '08: Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services*, pages 240–247, New York, NY, USA, 2008. ACM.

[13] Michiel Koning, Chang-ai Sun, Marco Sinnema, and Paris Avgeriou. Vxbpel: Supporting variability for web services in bpel. *Inf. Softw. Technol.*, 51(2):258–269, 2009.

[14] Anis Charfi, Tom Dinkelaker, and Mira Mezini. A plug-in architecture for self-adaptive web service compositions. In *ICWS '09: Proceedings of the 2009 IEEE International Conference on Web Services*, pages 35–42, Washington, DC, USA, 2009. IEEE Computer Society.