Domain Specific Monitoring of Business Processes Using Concept Probes

Adrian Mos(✉)

Xerox Research, 6 chemin de Maupertuis, Meylan, France
adrian.mos@xerox.com

Abstract. This paper proposes a monitoring framework that has business concepts at its core. Rather than relying on generic mechanisms to provide monitoring data, it proposes the notion of concept probes that fully match the business concepts used in the definition of business processes. These concept probes combine monitoring information from business process execution as well as service execution into aggregate information that makes sense from a business concept point of view. The approach has far-reaching implications: firstly, it provides superior understanding of the various execution parameters of the business concepts used in processes (including performance, correctness and context), with potential to aid Business Process Management (BPM) and Service Oriented Architecture (SOA) governance. Secondly, it helps with setting application-wide alarms and constraints potentially corresponding to Service-Level-Agreements, on a concept-level. For a given concept, such constraints can be set-up with immediate effect in all the business processes that use it. Thirdly, this approach gives technical users a deep understanding of the contribution of each of multiple application layers (BPM, SOA, operating system, various other technical layers) to the combined performance of a particular business concept. This can lead to faster reaction time in fixing problems, changes in business partners (that provide better services) or improvements in the underlying infrastructure or application parameters.

Keywords: BPM · Monitoring · SOA

1 Introduction and Scope

Business Process Management (BPM) and Service Oriented Architecture (SOA) are two important paradigms in today's enterprise solutions. They each bring a level of agility to business applications. BPM addresses the methodology and tools that enable the management of business processes (BPs) including their evolutions throughout their lifecycle. BPM Suites (BPMS) are complex software stacks that execute business processes and connect them to various enterprise resources such as the personnel directory, the various legacy applications and potentially the organization's SOA. An enterprise SOA typically manages the reusable technical services used to execute tasks that occur in business processes and it is often hosted in a cloud environment. Their functionality, granularity and interfaces define their level of reuse across a multitude of business processes. In general, the closer the SOA services match the business requirements, the faster it is to implement new business processes. In practice, SOA has

F. Toumani et al. (Eds.): ICSOC 2014, LNCS 8954, pp. 213–224, 2015.
DOI: 10.1007/978-3-319-22885-3_19

often drifted away from its initial promise of matching IT with business and has evolved in the IT domain, enabling a certain kind of agility at the IT solution level, i.e. making it faster for IT departments to implement new applications, much like previous paradigms such as component-based software or object-oriented development.

The typical approach to develop business applications using BPM and SOA with today's state of the art tools involves the definition of business processes using a generic language such as the widely-used Business Process Model and Notation (BPMN) [1]. This language contains elements such as "activity", "gateway", "signal" or "flow". Users need to describe their BPs by assigning textual labels such as "payment" or "customer registration" to the generic BPMN elements. Users who design BPs are generally users with good business knowledge and understanding of the various roles in the organization.

In contrast, users who design and create SOA services are typically architects and developers with much less business know-how. There is a certain connection between these two classes of users (business and technical) as, naturally, the SOA must eventually fulfil some useful functionality for the business applications, yet they have fundamentally different concerns. This translates into a disconnect between BPM and SOA which is solved through manual "glue" in the form of SOA connectivity parameters in the BP descriptions. This connectivity typically translates into configuration forms that are associated to certain BPMN activities.

Once the BPs are designed and fully configured, they can be run by the BPMS. The BPMS will manage their execution and will also direct SOA calls to the appropriate SOA services, as required. It is very important to be able to extract information related to the execution of the BPs. Such information can be used to understand what really happens with the various activities, whether they execute correctly or not, how long it takes to execute them, what data is passed around and whether pre-established thresholds for various parameters are exceeded or not. Such information is extracted using monitoring infrastructure that connects to the BPMS and collects data as the BPs execute. The monitoring infrastructure is typically tightly woven into the BPMS. Similarly, for SOA data collection, the monitoring infrastructure can leverage the execution environment, such as a specific Enterprise Service Bus in order to collect metrics of interest.

The fundamental problem with the state-of-the-art approaches is that they collect and present data at the level of process definition, which is generic. The fact that BPs have been designed and deployed using a generic language such as BPMN inherently means that monitoring data is collected in a generic way, with respect to the business domain. Therefore, reusing the example above, monitoring data will be collected for elements such as "activity", "gateway", "signal" or "flow" with no correlation to the business concepts of "payment" or "customer registration" apart from the simple matching of the textual labels to the monitored generic elements. This causes a number of problems:

- It is hard to make use of the monitoring data in order to present meaningful metrics to business users, without significant configuration efforts for each BP.
- It is difficult to correlate the execution of business concepts to the execution of services in the SOA layer as well as to the parameters of the runtime infrastructure.

- It is difficult to set wide-ranging Service-Level-Agreements (SLA) that affect all BPs equally. For instance it may be necessary to specify that all "payment" operations, regardless of the BP in which they occur, must execute in less than 20 s. In today's BPM solutions this would typically imply manually changing each of the BPs in which the payment activity occurs in order to establish this SLA, or refactoring all BPs to use the "payment" activity as a sub-process and then setting the SLA to the sub-process.

In summary, the above-mentioned problems with today's monitoring approaches are due to the fact that most existing solutions are domain-independent and technology-dependent. In contrast, the approach presented in this paper is domain-dependent and technology-independent with a focus on BPM/SOA environments.

2 Overview of the Approach

This proposal aims to address the shortcomings of today's monitoring capabilities for BPMS/SOA applications. It does so by adding a layer of abstraction on top of existing capabilities, rather than replacing them. This ensures compatibility with a wide range of existing systems and platforms. In fact the proposed approach is technology-independent. The approach entails the creation of Concept Probes (CPs) that are monitoring entities corresponding directly to business concepts. These CPs provide an aggregated view of the various execution layers involved in the execution of a particular business concept present in BPs. Once the CPs are created, they need to be bound to the monitoring capabilities of the existing infrastructure, effectively acting as an extra monitoring layer on top of BPMS and SOA monitoring capabilities. In addition to the CPs, this approach introduces the notion of Business Process Probe (BPP) which corresponds to each deployed business process and which is essentially a composition of the CPs that correspond to the concepts used in the BP.

Figure 1 below uses an example to illustrate the conceptual placement of the probes in the context of a typical BPM/SOA environment. The figure illustrates a simple business process deployed into the BPMS and connected to the SOA services by links from activities **Aa** and **Ac**, with **Aa** requiring services **S1** and **S3**, and **Ac** requiring **S6** respectively. These links represent regular web service calls such as SOAP or RESTful invocations. The rectangle represented above the BPMS represents the monitoring capabilities that are available for the respective BPMS. Usually these capabilities include the generation of events when activities execute, the computation of execution times for activities and generally the reporting of various states in which the BPs operate. The rectangle below the SOA Runtime represents the monitoring capabilities of the SOA environment (which is typically an Enterprise Service Bus or other forms of SOA-based middleware such as an SCA runtime [2]). These capabilities typically include monitoring the service invocations, computing execution times for various service operations and general reporting of the states in which the services operate.

To reduce clutter, the image above does not present other monitoring layers that may be available in an enterprise environment, such as cloud monitoring, operating

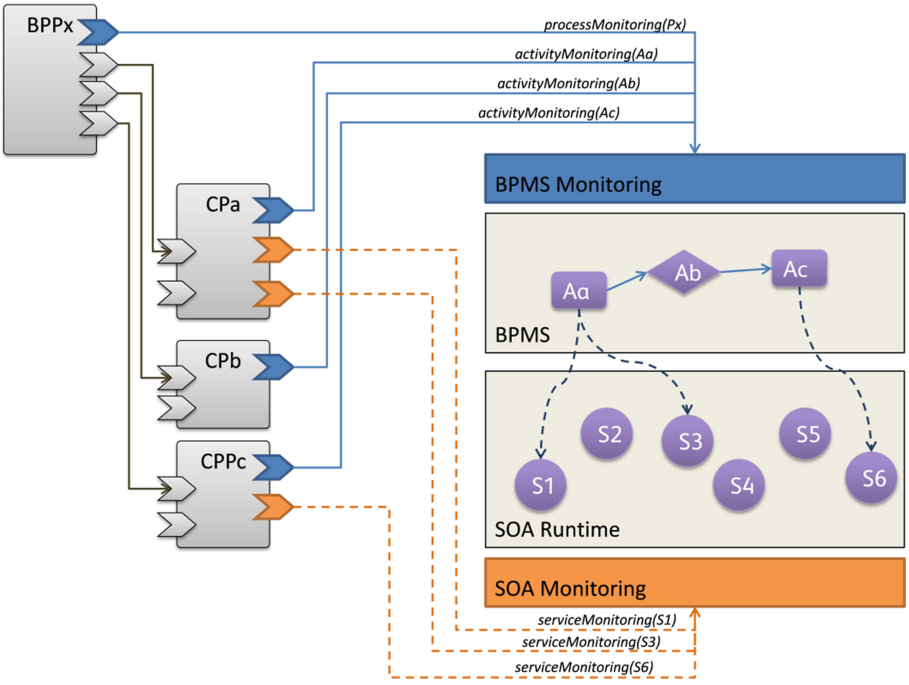

Fig. 1. Approach overview

system monitoring or network monitoring. These are however taken into account and mentioned in the description of the probe structure below.

Figure 1 presents three concept probes **CPa**, **CPb** and **CPc** that correspond to the business concepts **a**, **b** and **c**, used in the illustrated BP through the activities **Aa**, **Ab** and **Ac**. In addition to the CPs, the image also shows a business process probe, the **BPPx**. This corresponds to the example BP and it uses the three CPs to aggregate BP-level information. The outgoing lines from the CPs represent their connections to the BPMS and SOA monitoring systems. For example, since **CPa** is a probe specifically generated for concept **a**, it will interrogate the BPMS monitoring system with regard to the activity **Aa** and the SOA monitoring system with regard to services **S1** and **S3**. These connections are generated based on the knowledge that concept **a** is used in activity **Aa** and it requires services **S1** and **S3**. This knowledge comes from concept mappings described in the section below. The lines are labeled with abstract functions that simply illustrate what kind of data they collect from the monitoring systems.

The CPs therefore leverage existing monitoring capabilities using specific bindings related to the concepts they need to match. They aggregate the required data from the various BPMS and SOA monitoring systems into meaningful information that matches the business concepts used throughout a business application. Similarly, the BPPs aggregate the monitoring data from CPs corresponding to the monitored BP with additional BP-specific monitoring information that is generated by the BPMS monitoring system (such as timestamps and duration for the process execution, user roles

and other process-specific data). Note that the information provided by a BPP is significantly richer than that provided by BPMS monitoring systems for a given BP because it includes the breakdown of monitoring information for each of the concepts used in the BP as well as the aggregated BP-level data. Naturally, modern BPMS monitoring systems can make the correlation between a BP and it composing activities but this approach consolidates monitoring information in a conceptual layer that adds the semantics of the contained concepts.

3 Concept Mappings

In their simplest form, concept mappings are connections between business concepts and the SOA services that are used by them. The concepts are then used in all of the BPs in various combinations, resulting in a variety of BPs.

The starting point is the following sets that are known:

1. Set of services $\mathbf{S = \{s_1, s_2, \ldots s_q\}}$
2. Set of processes $\mathbf{P = \{p_1, p_2, \ldots p_m\}}$
3. For each process $\mathbf{p_k}$, a set of activities $\mathbf{Ak = \{a_{k1}, a_{k2}, \ldots a_{ky}}$ where **y** depends on the complexity of $\mathbf{p_k\}}$.
4. The set of all activities in all the processes $\mathbf{A = A_1 \cup A_2 \cup \ldots \cup A_{|P|}}$

The goal of concept mapping operations is to determine the following sets

1. Set of concepts $\mathbf{C = \{c_1, c_2, \ldots c_n\}}$
2. $\mathbf{CM = \{(c_j, S_j): \forall\ c_j \in C;\ S_j \subseteq S\}}$ which contains for each concept its list of services, e.g. $\mathbf{(c_4, (s_1, s_3, s_8))}$
3. $\mathbf{AM_k = \{(a_{ki}, c_j): \forall\ a_{ki} \in A_k;\ c_j \in C\}}$ which contains for process $\mathbf{p_k}$ its activities and the concepts they map to
4. $\mathbf{AM = AM_1 \cup AM_2 \cup \ldots \cup AM_{|P|}}$ which contains for each activity in all processes the concept it maps to

This paper does not make a claim about a particular method of obtaining concept mappings. However, this section briefly discusses this aspect in order to demonstrate feasibility of the entire approach. The existence of the sets described above is a requirement for the proposed method to function. There are two important aspects to be discussed about concept mappings, namely concept identification and concept use, which can be seen as two required stages in the application of the method.

3.1 Concept Identification and Use

Obtaining the sets **C** and **CM**, described above, requires that the business concepts that are used over and over again in the business processes be clearly identified together with their required usage of the SOA. There are three main potential approaches for concept identification:

- Automatic Top-Down: this approach corresponds to the desirable approach for modern organizations that will create new business processes in the future.

It assumes that the concepts are defined by business experts and eventually bound to SOA services in a deployment stage where their service requirements are mapped to available SOA assets. An in-depth discussion of this approach is presented in [3].

- Automatic Bottom-Up: this approach assumes existing legacy BPs are already deployed and functional in an organization, so it is best suited for existing BPs deployed in BPMS/SOA environments. It leverages extraction capabilities from execution logs to cluster and identify commonly used concepts and their correlations to SOA services. Many such approaches are possible, for instance [4, 5].
- Manual Top-Down: this approach is a downgraded version of the first approach (Automatic Top-Down) and it can be applied in organizations that do not use a deployment stage for BPs that connects them automatically with the SOA. It has the same characteristics as the first approach but it requires the manual annotation of concepts with SOA services, rather than using the service binding information that the first approach has.

The three approaches can be used in combination in some cases, for instance the Automatic Bottom-Up may need help from Manual Top-Down to increase quality of the results. In all cases the result is a list of concepts with their related SOA services.

After concepts are identified, it is necessary to obtain the **AM** set by mapping the BPs (existing or future) to the concepts. This involves matching BP activities to concepts. Such matching is closely related to the method for identifying concepts. When using Automatic Top-Down, the BPs are in fact composed of activities directly matching concepts, so there is no ambiguity, each activity corresponds to a clearly identified concept. When using Automatic Bottom-Up identification, the concepts are extracted from BP activities so matching activities to concepts requires simply storing the correlations between the activities and the extracted concepts. When the Manual Top-Down approach is used, BPs need to be annotated with the concepts manually, this requires the manual creation of the connection between activities and concepts (i.e. **Aa** activity to concept **a** in the example above).

When all the required mappings are available, the probes need to be created, instantiated and deployed. The specific technical means for generating and running monitoring probes are out of the scope of this proposal. The generation can be done using a variety of existing methods such as code generation or template instantiation for instance. Once they are generated they need to be executed as they need to be running entities managed by a monitoring framework. There exist a variety of monitoring frameworks that can be used for managing these probes and this proposal does not aim to propose a new monitoring system. Rather it proposes a new layer of monitoring probes that can be executed in any extensible monitoring system. A very common technology that can be employed for merging the proposed approach with existing monitoring frameworks is Java Management Extensions,[1] supported by a large variety of infrastructures, both commercial and open-source.

[1] http://en.wikipedia.org/wiki/Java_Management_Extensions.

4 Probe Structure and Functionality

The concept probes are capable of collecting an arbitrary number of metrics, such as execution time or execution status. The approach described here does not focus on a particular metric, as the same approach can be used to measure several metrics. For illustration purposes, execution time is used when an example is required. In the description of the structure of the probe, metrics are identified by Greek symbols such as Metric α or Metric β…

The connections of the probe to the various existing monitoring infrastructures are explained in the paragraphs below. All CPs contain the same three main components, illustrated in Fig. 2. The first, **Raw Data Collection** is charged with collecting data for any given metric from any of the collection points, represented in the image to the right edge of the CP composite structure. For a given concept c_j, its corresponding data collection points are:

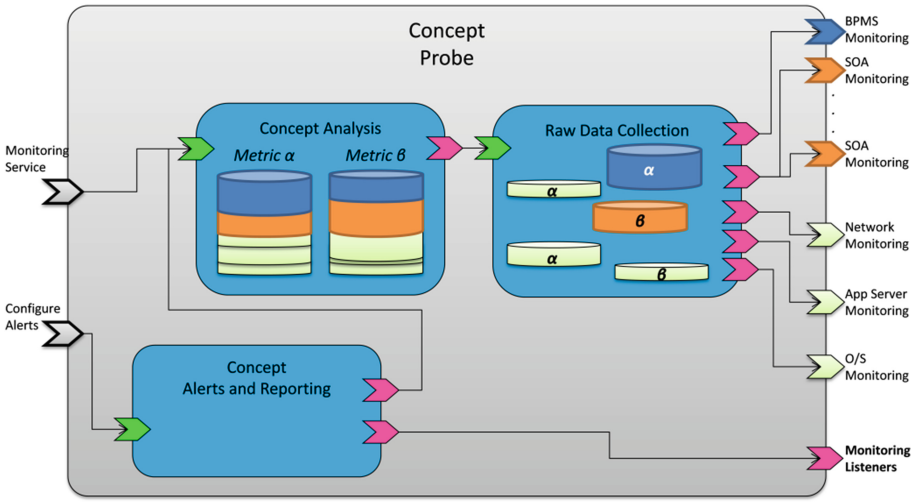

Fig. 2. Concept probe structure

- One BPMS monitoring point that collects data from the BPMS with regard to activities that map to this concept, i.e. $CA_j = \{a_{xy}: (a_{xy}, c_j) \in AM\}$. This effectively means that there is one probe per concept regardless of the number of activities that correspond to this concept. The reason for this is that the approach emphasizes the value of monitoring each individual concept regarding of where it is used in the business processes. So each time an activity is executed, the concept probe corresponding to the concept associated to the activity is notified.
- Several SOA collection points that each map to the SOA Runtime monitoring capabilities for each of the SOA services that this concept maps to, i.e. S_j. These points extract execution information for the services that are related to the concept.
- Several other collection points that can collect information potentially to be correlated with the above-mentioned collection points. This includes Network

Monitoring, App Server Monitoring and Operating System Monitoring. These extra collection points can give useful information regarding the context of the metric values. For instance, a service execution can be perceived as slow if network latency is very high. Similarly, if the OS processes are not scheduled properly by the OS or if the Application Server is not scalable, these can impact the execution of the BPMS and the SOA layers. Therefore, these extra collection points can potentially be very useful, although they are not essential. They are given here as an example of extra aggregation capabilities of the probes.

The second component, **Concept Analysis**, is tasked with aggregating raw data obtained from the collection component into composite metrics. These composite metrics are data structures that present the aggregate monitoring information combining the individual metric data for the BPMS, SOA and other collection points, for the concept.

The data structures give an aggregate value if appropriate (such as *total execution time*), as well as a breakdown of this value or contextual information pertaining to this value for the individual collection points. This can include the individual execution time of services and of the process activity in the BPMS, as well as values for network latency, resource utilization in the application server or process scheduling in the operating system. Similarly, cloud-related data can be obtained such as the virtual machine utilization for the server executing the SOA services or BPMS elements. Individual methods for obtaining these values are out of the scope of the presented approach, as the approach is concerned with the architectural entities that the monitoring framework has, not their detailed implementation which is often straightforward. This concept is also queried by outside entities for metric values (represented as the Monitoring Service port of the CP).

The third component**, Concept Alerts and Reporting** relates to the ability of the probe to give specific reports about the execution of the concept and most importantly to register alerting rules and react accordingly. This component allows the registration of SLA requests through the Configure Alerts port and uses the Concept Analysis component to constantly compare the aggregated metric values with the required thresholds. If SLA thresholds are exceeded it can notify registered Monitoring Listeners. These listeners are external entities (out of scope of the presented approach), which can be connected to the monitoring probe and notified of important alerts and events.

There is one **Business Process Probe (BPP)** per business process deployed. Similarly to the concept probe, the business process probe contains a set of three components with distinct responsibilities, as illustrated in Fig. 3.

The **Raw Data Collection** component collects the monitoring data from the CPs that correspond to the activities of the business process monitored by the BPP.

For the BPP_k corresponding to a process p_k, the data collection points are:

- One BPMS collection point that collects monitoring data for the execution of p_k in the BPMS. This can include contextual information (e.g. user name) for the required metric as well as metric values for the business process (e.g. execution time of p_k as seen from the BPMS).
- Several connections to each of the CPs required by the BPP_k. These are the CPs that correspond to the set of process concepts $PC_k = \{c_i \in C: \forall (a_{kx}, c_i) \in AM_k\}$.

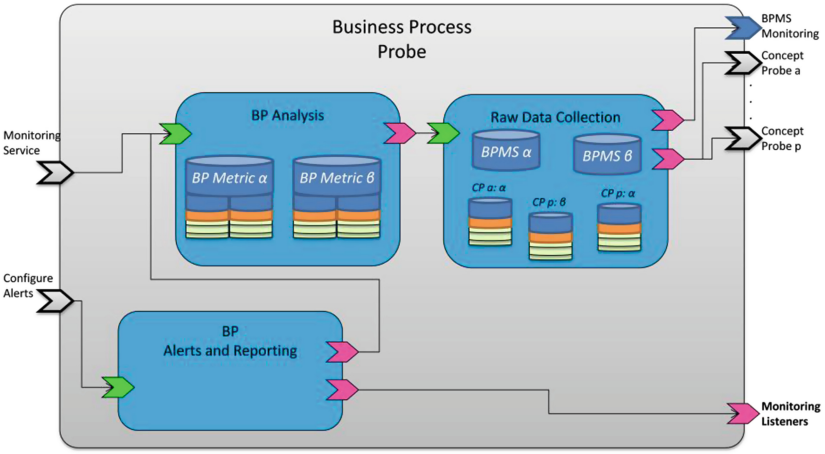

Fig. 3. Business process probe

These are used in the aggregation of monitoring data corresponding to each concept used by the P_k. If a concept appears several times in the process (due to several process activities mapping to the same concept), this concept will count several times in the aggregation. This is part of the logic of the **BP Analysis** component.

The **BP Analysis** component is very similar in functionality to the Concept Analysis component of the CP, except that it aggregates data from the BPMS and the various CPs, rather than from the BPMS, SOA and the extra monitoring collection points. To this end it aggregates the BPMS-collected data corresponding to the execution of the process together with the already aggregated data of each of the CPs it connects to. The CPs correspond to monitoring data for the individual activities that compose the process so a simple way to visualize their composition is putting them side by side, under the complete process data. An example is the total execution time of a process composed by the sum of the individual execution times of its activities. It may be useful to understand why a process executes in a given amount of time, and the composed metric would be able to show its individual components, highlighting the concepts that take the most amount of time. This can be decomposed further by showing why the individual concepts take so long to execute, by drilling down into individual services that are used for the concepts as well as the other monitoring data collected. The last component, **BP Alerts and Reporting** has identical functionality to the CP-level component, Concept Alert and Reporting, but refers to the entire BP.

5 Related Work

There are many commercial tools and academic approaches to monitoring business processes and services in a BPM/SOA setting. Some of them tackle only BPM while other only SOA, with a number of them tackling both, however without providing the same level of monitoring as the proposal presented here. This is because the vast

majority of the approaches stay generic with respect to the business domain, even though they may do some monitoring aggregation.

Industrial approaches such as IBM Tivoli [6] or Tibco Hawk [7] as well as many others do provide a wide array of monitoring capabilities. However they are tightly bound to the generic capabilities of the BPMS they are targeting, namely Websphere and Tibco BPM respectively for these two examples. They provide detailed monitoring data from a variety of sources but they do not offer domain concept probes or monitoring data at the level of abstraction that business designers need. However such approaches are typically suited for integration in the approach presented in this proposal, through the BPMS and SOA collection points. The domain probes would use JMX to extract and aggregate runtime monitoring data from such monitoring infrastructure. Therefore such approaches are fully complementary to this proposal.

Among academic approaches, there are approaches that recur to aggregation mainly to compose events from a low-level monitoring source (using Complex Event Processing queries) in order to extract more meaningful data out of the raw events. For instance they may aggregate events such as "process starts" with "process ends" in order to extract the aggregated metric "process execution time". Such approaches [8–12] use a variety of techniques to derive better understanding of raw events but they fundamentally still stay at a generic level with regard to the business domain. As with the commercial approaches outlined above, the presented proposal is complementary and could interact with such approaches using them as data collection points. There are also approaches that try to correlate execution events to the originating processes using some forms of traceability between model elements and execution events. For instance in [13] the authors argue for the existence of domain-specific patterns for interpreting events, without giving a complete solution. Their suggestion is in line with the proposal here in that they promote the idea of presenting information corresponding to domain elements, but they focus mostly of interpreting CEP events, while our proposal targets structured probes that connect directly with monitoring APIs. In addition, this proposal presents in detail the structure of the probes while the authors of [13] simply state that it would be useful to have domain understanding of events.

In summary all of these approaches ultimately recur to generic event analysis and do not provide a "native" monitoring probe layer that directly corresponds to the business concepts.

Lastly there are approaches (such as [14]) that try to trace the execution events back to modeling entities using unique correlation IDs. Similarly to the above-mentioned approaches, this remains generic with respect to the business domains and does not benefit from concept-level aggregation presented in our approach. Therefore it does not allow the creation of concept-based SLAs, nor does it offer concept-based metrics that span across the business processes. Like the other approaches above, it can correlate data back to business processes without further aggregation into business concepts.

The approach presented in this proposal provides the same level of functionality that a generic approach offers, but at the domain-element level using probes that correspond on a 1-to-1 basis with business concepts. These probes hook into the existing monitoring systems regardless of the technology they use to extract and represent events. These elements ensure that the approach offers the advantages explained in the previous sections and constitute important differentiating factors.

6 Summary and Conclusion

Existing monitoring solutions are typically technology-specific and generic with respect to the business domains. In contrast, the approach presented here is generic with respect to technology and domain-specific with respect to the business. This brings several interesting advantages. Having concept monitoring probes gives unprecedented insight into the execution of applications. Business users can understand how the processes execute in terms that are ideally suited to them. In addition, they can specify constraints and alerts for particular concepts that have immediate effect across the entire spectrum of the deployed business processes. Setting a Service-Level-Agreement for a concept would instantly translate into the constraint being applied to all the activities of all the processes using it. Similarly, specifying alerts or simply observing the concept behavior would apply to any execution of the activities related to it. In addition, each execution monitoring of such activities would be complemented by monitoring of the SOA services that are associated to the concept. Beside performance metrics, this approach brings important benefits in understanding whether a concept executes successfully.

For technical users or system administrators, this would give a breakdown of responsibilities for performance problems showing the individual contribution of each of the layers involved and each of the entities (e.g. services) that compose the concept execution. Similarly, when monitoring process execution, this approach promotes the use of process-level probes composed of concept probes. Each BPP would correspond to a particular process and it would provide the same benefits as described above, but at process-level. Therefore business users could understand how a process performs in terms of the business concepts that it comprises, while technical users could understand the impact of the various layers and entities involved in fulfilling the end-to-end process. This can help pinpoint SOA services that cause bottlenecks for individual processes, or explain why certain processes do not execute correctly, by showing the concepts whose execution fails.

A full prototype of the presented framework is in advanced stages of implementation, using Stardust BPMS [15] and Fuse ESB [16] as the target BPM and SOA layers, respectively. The implemented probes correspond to the concepts of a sample domain chosen for validation. They correlate data for BP activity execution with data for service execution and give a breakdown of each layer's contribution to various performance metrics. The data is then sent for display to Eclipse graphical editors where it is presented in the appropriate context alongside process design elements [17]. Such usage helps validate the added value of the approach and its two main components: domain-level concept mapping of monitoring data; and technology independence where existing BPM/SOA environments can be augmented to benefit from the collection and graphical display of relevant monitoring information.

References

1. Object Management Group, Business Process Model and Notation. http://www.bpmn.org/
2. OASIS Service Component Architecture. http://www.oasis-opencsa.org/sca/
3. Mos, A., Jacquin, T.: A platform-independent mechanism for deployment of business processes using abstract services. In: 6th International Workshop on Evolutionary Business Processes, EDOC, Vancouver, Canada (2013)
4. Pérez-Castillo, R., García-Rodriguez de Guzmán, I., Piattini, M., Weber, B., Places, A.S.: An empirical comparison of static and dynamic business process mining. In: ACM Symposium on Applied Computing, ACM, New York (2011)
5. Wang, J., Tan, S., Wen, L., Wong, R.K., Guo, Q.: An empirical evaluation of process mining algorithms based on structural and behavioral similarities. In: 27th Annual ACM Symposium on Applied Computing (2012)
6. IBM Tivoli. http://www-01.ibm.com/software/tivoli/
7. Tibco Hawk. https://docs.tibco.com/products/tibco-hawk-4-9
8. Pedrinaci, C., Lambert, D., Wetzstein, B., van Lessen, T., Cekov, L., Dimitrov, M.: SENTINEL: a semantic business process monitoring tool. In: First International Workshop on Ontology-Supported Business Intelligence (OBI 2008). ACM, New York, NY, USA (2008)
9. Pedrinaci, C., Domingue, J., Alves de Medeiros, A.K.: A core ontology for business process analysis. In: 5th European Semantic Web Conference on the Semantic Web (2008)
10. Mos, A., Pedrinaci, C., Rey, G.A., Gomez, J.M., Liu, D., Vaudaux-Ruth, G., Quaireau, S.: Multi-level monitoring and analysis of web-scale service based applications. In: Dan, A., Gittler, F., Toumani, F. (eds.) ICSOC/ServiceWave 2009. LNCS, vol. 6275, pp. 269–282. Springer, Heidelberg (2010)
11. Hummer, W., Inzinger, C., Leitner, P., Satzger, B., Dustdar, S.: Deriving a unified fault taxonomy for event-based systems. In: 6th ACM International Conference on Distributed Event-Based Systems (2012)
12. Mulo, E., Zdun, U., Dustdar, S.: An event view model and DSL for engineering an event-based SOA monitoring infrastructure. In: 4th ACM International Conference on Distributed Event-Based Systems (2010)
13. Ammon, R.V., Silberbauer, C., Wolff, C.: Domain specific reference models for event patterns for faster developing of business activity monitoring applications. In: VIPSI (2007)
14. Mulo, E., Zdun, U., Dustdar, S.: Monitoring web service event trails for business compliance. In: Service-Oriented Computing and Applications (SOCA) (2009)
15. Stardust BPMS. http://www.eclipse.org/stardust/
16. Jboss Fuse ESB. http://www.jboss.org/products/fuse
17. EclipseCON Talk: Modeling and Monitoring Business Processes with Mangrove, BPMN2 Editor and Stardust. https://www.eclipsecon.org/na2014/session/modeling-and-monitoring-business-processes-mangrove-bpmn2-editor-and-stardust