Monitoring of Non-functional Requirements of Business Processes based on Quality of Service Attributes of Web Services

Evando S. Borges[1], Marcelo Fantinato[1], Ünal Aksu[2], Hajo A. Reijers[2] and Lucinéia H. Thom[3]
[1]*School of Arts, Sciences and Humanities, University of São Paulo, São Paulo, Brazil*
[2]*Department of Information and Computing Sciences, Utrecht University, Utrecht, The Netherlands*
[3]*Institute of Informatics, Federal University of Rio Grande do Sul, Porto Alegre, Brazil*

Keywords: Business Processes, Monitoring, Business Level Agreement, Service Level Agreement, Quality of Service.

Abstract: Business monitoring approaches usually address indicators associated with processes only at the service level; i.e., related to the services implementing the processes. Monitoring at the service level raises technical measures geared to Information Technology (IT) managers. Monitoring of Key Performance Indicators (KPIs) is usually carried out at a higher level, but transversely to the organization's processes, i.e., uncoupled from the processes. We present a component designed to aid in strategic alignment between business and IT by monitoring Non-Functional Requirements (NFR) of processes based on Quality of Service attributes. This component aims to allow business managers to monitor process executions by focusing on the indicators that truly respond to the execution of such processes. We evaluated the component via a proof of concept.

1 INTRODUCTION

In the global corporate landscape, with wide competition among organizations, real-time monitoring of Non-Functional Requirements (NFR) of business processes are a competitive edge, complementing the monitoring of functional requirements of processes. An organization that quickly realizes that a part of its process is not responding as expected, in terms of execution time, for example, can decide before a negative process outcome (Lubinski, 2008). Several types of NFRs can be typically associated with processes, such as those related to performance, security, availability, parallelism and cost (Presman and Maxim, 2014).

To monitor process NFRs, these requirements need to be specified systematically. The solution chosen to implement the processes can cause an impact on how monitoring of process NFRs can be carried out. Process automation and execution are commonly supported by Service-Oriented Architecture (SOA), using web services technology (Curbera et al., 2002; Sheng et al., 2014; Fahad et al., 2015). Service Level Agreement (SLA) are defined in terms of QoS attributes, which are the NFRs *per se*. QoS attributes and levels are commonly called Service Level Objectives (SLO). NFRs for process models can be defined as Business Level Agreements (BLAs) – like SLAs, but at the process level – as proposed as in the basis of this work (Salles et al., 2013; Barros et al., 2014; Salles et al., 2018). *Business Activity Monitoring (BAM)* refers to real-time access to critical business performance indicators to improve the viability of business operations (Lubinski, 2008).

Existent approaches aimed at the monitoring of process NFRs purely work from the technical point of view; i.e., they address the individual monitoring of each of the services that implement a process. This technical point of view is useful for the Information Technology (IT) team, which needs to follow the performance of the services that implement the processes. For business areas, this view is overly detailed and needs to be translated into a high level view for them to understand (Salles et al., 2018). A particular business area may be interested in a part of a critical process that will only present a problem if a full set of services presents any problem (Dumas et al., 2018) 1.

There are approaches to monitoring indicators at the highest organizational level – the Key Performance Indicators (KPIs) (Carmo et al., 2017). However, KPI monitoring is usually carried out crosscutting the organization's processes. Thus, this monitoring specifically targeted to business managers is often decoupled from processes. In addition, KPI monitoring is often not aligned with monitoring the services used to implement the processes.

Consider the loan process model in Figure 1.

Borges, E., Fantinato, M., Aksu, Ü., Reijers, H. and Thom, L.
Monitoring of Non-functional Requirements of Business Processes based on Quality of Service Attributes of Web Services.
DOI: 10.5220/0007700605880595
In *Proceedings of the 21st International Conference on Enterprise Information Systems (ICEIS 2019)*, pages 588-595
ISBN: 978-989-758-372-8

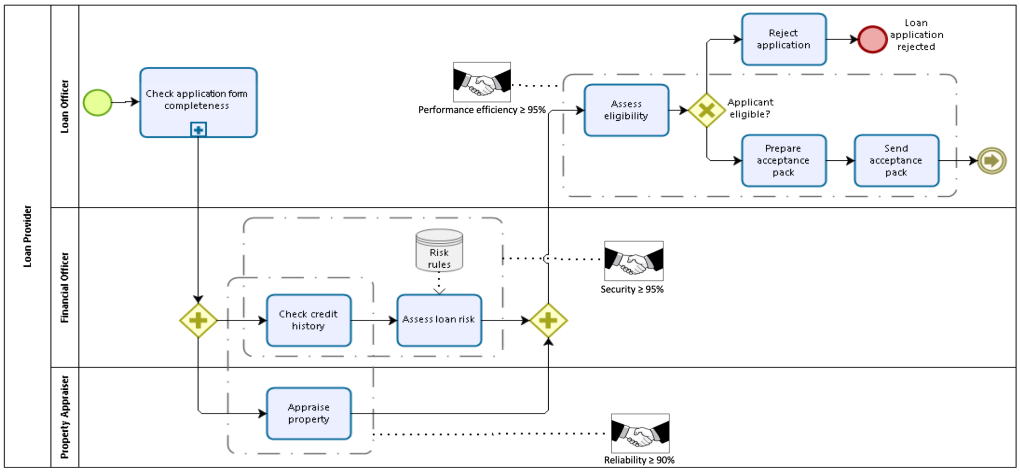


Figure 1: Loan business process model – scenario example of process-level NFRs [(Dumas et al., 2018), page 111, adapted].

In Figure 1, business managers (i.e., the "process owners") should be able to verify whether specific parts of the process are operating according to their goals. It may be relevant for the "Financial Officer" to execute the activities "Check credit history" and "Assess loan risk" having security[1] as an NFR. This NFR is needed to guarantee a high level of data integrity and access control in this part of the process because of the sensitivity of the information involved. From the business manager's perspective, this is what matters in terms of NFRs involving these two tasks, and for which they would like to be informed about the process's ability to comply with. Technical details of services implementing these tasks, such as those involving scalability, testability, operability or stability are not their focus of interest but rather of the IT team. Similar cases in Figure 1 relate to requirements "Performance efficiency" and "Reliability" and the corresponding set of activities aggregated by each of them. Considering monitoring levels, business managers would not like to be awakened at dawn because some service is down. However, they would like so if this incident had a major impact on delivering a critical part of the process under their responsibility.

A comprehensive solution for monitoring NFRs that considers both business and IT perspectives should rely on the definition of NFRs at both levels. Based on this goal, the *StrAli-BPM (Strategic Alignment with Business Process Management) framework* was proposed before (Salles et al., 2013; Barros et al., 2014; Salles et al., 2018). The StrAli-BPM framework requires that NFRs are first specified in process models by business analysts. Later, NFRs at the process level are then used as the basis for defining the Quality of Service (QoS) attributes of the services by IT experts. These services are those that will implement the process activities. In this way, NFRs at process and service levels relate to each other.

This paper proposes a component for the monitoring of process NFRs as part of StrAli-BPM. The proposed component is called *StrAli-BAM – Strategic Alignment with Business Activity Monitoring*. Although StrAli-BAM is designed as a StrAli-BPM component, it is adaptable to similar contexts. The proposed component can monitor NFRs at the process level based on QoS attributes. This is possible given that StrAli-BPM first uses NFRs at the process level to derive the QoS attributes of the corresponding services. StrAli-BAM aims to enable different areas and organizational levels to share a common cockpit or dashboard for real-time monitoring, providing an additional contribution to strategic alignment.

This paper presents in the following sections the proposed component and its evaluation.

2 THE StrAli-BAM COMPONENT

In the broader context of the StrAli-BPM framework (cf. Figure 2), the new component was designed to enhance strategic alignment between business and IT by the monitoring of process NFRs. The monitoring of process NFRs is carried out via the monitoring of BLAs associated with business processes modeled in BPMN. This monitoring is also associated with monitoring SLAs and should be associated with KPI monitoring in the future.

Figure 2 shows the original framework (white elements) extended with the new component (gray ele-

[1]The NFR nomenclature adopted in this paper, for both process and service levels, is based on a dictionary proposed specifically for this purpose (Castro et al., 2019).

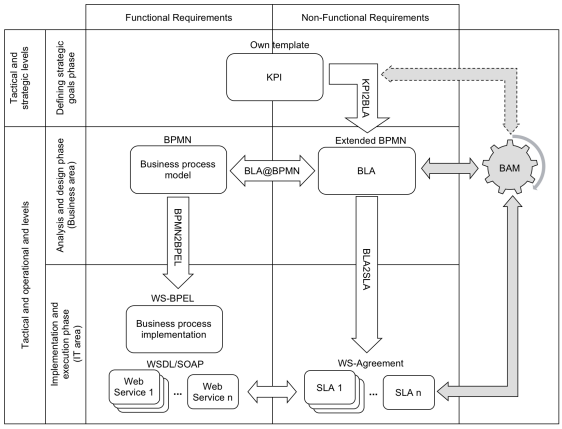


Figure 2: Extended StrAli-BPM framework.

ments). The gray double-dashed arrow shows the intention to add the monitoring of the KPIs used in the creation of the BLAs being monitored.

The primary intent of the monitoring offered by the proposed component is targeted to process NFRs, i.e., to BLAs. BLAs fill the gap between all levels of NFRs – from those extracted from KPIs to those derived to SLAs. As processes are run via their 'executable' versions in Web Service Business Process Execution Language (WS-BPEL), the SLAs of the corresponding web services are the directly monitorable elements in this infrastructure. Thus, BLAs are monitored only indirectly, since they are associated with BPMN process models in a non-executable version. As a result, the monitoring strategy of the proposed component considers the 'top-down' creation history of SLAs from BLAs to conduct a 'bottom-up' monitoring of BLAs based on SLAs.

Figure 3 shows the component architecture, comprising a process execution infrastructure (*Oracle SOA Suite 12c*) and a monitoring module (*BLA Monitor*). Via the execution infrastructure, the process is carried out using the *Oracle BPEL 12c* and *Oracle Mediator 12c* platforms integrated via *Oracle Instance*. The *BLA Monitor* must process SLOs (*SLO Processor*) and display BLA reports (*BLA Report Viewer*). It receives the SLOs from the execution infrastructure via the *SLO Collector* interface.

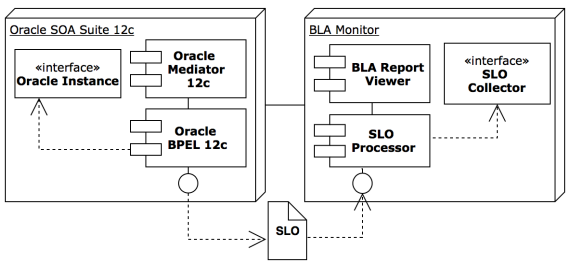


Figure 3: Architecture of the StrAli-BAM component.

2.1 BLA Monitor

Aiming at strategic alignment, the monitoring of process NFRs should be offered to different organizational areas and levels via an integrated dashboard, with distinct levels, types and groupings of information. Figure 4 shows, via an ArchiMate model (TOG, 2018), the architecture of the designed BLA monitor, based on SLAs and their respective SLOs.

Per Figure 4, the main elements of the *BLA Monitor* component are *SLO Processor* and *BLA Report Viewer*. Although this component works directly with StrAli-BPM, it has been designed in a decoupled way, assuming it can be used in other similar contexts.

The first step of SLO processing is to *Register BLA*. In this step, the BLA and SLA artifacts – i.e., the monitoring targets, generated by the other components of the StrAli-BPM framework – should be consumed as the initial input. The registration of a BLA to be monitored with its respective SLAs is then carried out. This action is carried out via the *Import BLA* interface, for the *Register BLA* service, generating the *SLO* data object in the database. For each SLO within an SLA, a unique identifier is generated and its QoS attribute and level are identified and also stored in the database. The BLA register must consider all the content (of a BLA and its respective SLAs) needed to enable the correct and complete BLA monitoring, according to the meta-models proposed in earlier works (Salles et al., 2018). For example, different penalty and bonus thresholds for each BLA (when applicable) and the number of runs to be taken for averages calculations must be brought in properly.

The second step of SLO processing is to *Collect SLO Result*. This processing is triggered whenever a service with at least one associated SLO runs for the process being monitored. The associated service *Collect SLO Result* stores the measured value of the QoS attribute for the corresponding SLO; this value is stored in the database as the *SLO Monitoring Result* data object. The *SLO Record* interface receives requests for SLO result collection.

The third step of SLO processing is to *Process SLO Result* by comparing the measured value (stored in the *SLO Monitoring Result*) with the expected level of *SLO*. This comparison results in the *BLA indicator* data object, which serves as input to the service *BLA Report Viewer*. As each SLO is processed, its SLAs and BLAs must have their execution results set to – *passed*, *failed* or *warning*. Depending on the number of penalty or bonus thresholds associated with a corresponding BLA, distinct levels of results should be used for the *passed* and *failed* options, since monitoring may show *passed* results with different bonus

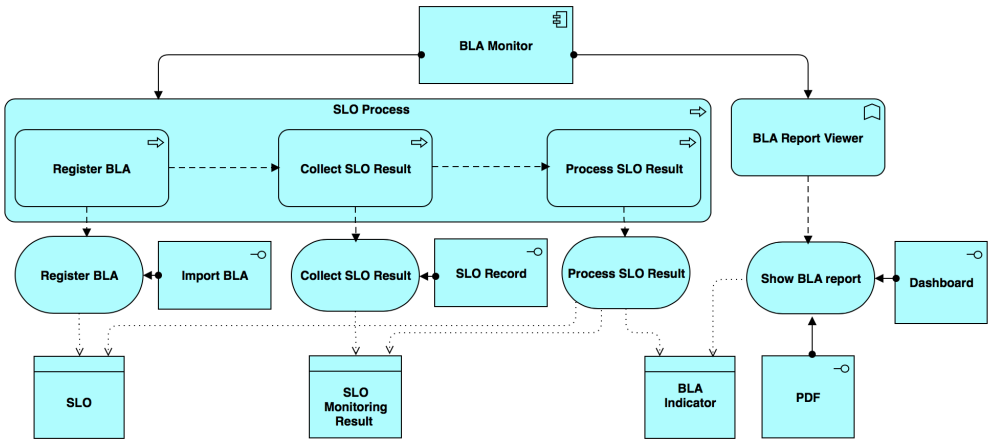


Figure 4: BLA monitor architecture.

levels (*passed level 1*, *passed level 2* etc.) or *failed* results with different penalty levels (*failed level 1*, *failed level 2* etc.). The numbers of runs to be taken for averages calculations, as registered during the *Register BLA* for each BLA and SLA, must be considered.

BLA Report Viewer shows the results to users via *Dashboard* and *PDF* interfaces. The *BLA Report Viewer* service uses the indicators generated by the SLO processing to display, in real-time, the information of the BLA monitored and its respective SLAs. The dashboard should allow the visualization of the generated data via distinct perspectives and detail levels. General indicators can be merged on the main screen, useful for both business and IT managers. From the overview, managers should be able to view details via distinct perspectives, such as by process, by activity, by service, by BLA, by SLA, by NFR type. Managers should also be able to choose, for example, criticality levels, i.e., processes with a high *warning* or *failed* degree. Given a process and its BLAs, responsible managers should be able to do a top-down reading to understand which SLAs are causing their BLAs to break. Similarly, a bottom-up view can also be used; for example: from a service whose SLA is not being met, it should be possible to view which processes and BLAs are affected. Finally, managers should be able to access data related to penalties and bonuses resulting from executions, also via distinct perspectives and levels of detail.

2.2 Event-based Monitorable Execution Infrastructure

For the monitorable execution of processes, an event-based processing infrastructure is proposed. The technical solution adopted as a proof of concept is based on the *Oracle SOA Suite 12c*[2] platform, which offers comprehensive integration. The following platform components were used: *Oracle BPEL 12c* and *Oracle Mediator 12c*. Integration between the components was done via *Oracle Instance* as presented via an ArchiMate diagram in Figure 5.

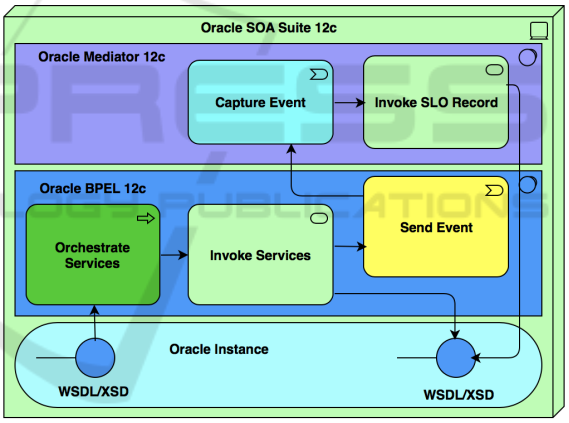


Figure 5: Event-based monitorable execution infrastructure.

Oracle BPEL 12c orchestrates the services composing the process and thus receives as input the WS-BPEL file to be executed. To enable the monitoring of the process execution, the services to be monitored need to be first identified for monitoring. The WSDL interface of each service with an associated SLA is extended with a canonical scheme fragment for monitoring, for each SLO in the SLA associated with it.

A *canonical scheme*[3] was used to ease interaction between services, enabling reuse. The canonical scheme is defined via XSD, cf. Figure 6. It specifies the general structure of an SLA goal – i.e., an

[2]License for development, testing and prototyping.

[3]Canonical scheme is a design pattern applied in SOA to ease the exchange of data between services (Erl, 2005).

SLO – to be reused in the WSDL interface of the services to be monitored. For each SLO to be measured in the same service, a canonical scheme must be instantiated and embedded as an extension of its WSDL interface. An SLO is composed of: *Id* (identifier of the SLO to be monitored); *QoSAttribute* (type of the NFR to be monitored, such as response time, availability, throughput); and *QoSMeasuredValue* (result of the attribute measured when the service runs).

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="SLO" type="SLOType" />
    <xs:complexType name="SLOType">
        <xs:sequence>
            <xs:element name="Id" type="xs:string" minOccurs="0" />
            <xs:element name="QoSAttribute" type="xs:string" minOccurs="0" />
            <xs:element name="QoSMeasuredValue" type="xs:string" minOccurs="0" />
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

Figure 6: Canonical scheme for monitoring.

per Figure 5, the execution and monitoring of a process begin with *Orchestrate Services* receiving the set of WSDL/XSD interfaces. The service orchestration needs to *Invoke Services*, which makes a request to the services presented by the platform instance via WSDL/XSD interfaces. Services that are identified for monitoring, via one or more instances of the canonical scheme, trigger events. This is done via Event Definition Language (EDL) – an XML dialect of the Oracle SOA Suite 12c platform. As a result, *Send Event* is triggered via defined rules for sending events, which are captured in the Oracle Mediator 12c layer by *Capture Event*. Finally, the *Capture Event* module gets the event request and triggers the *Invoke SLO Record* module. Then, the *Invoke SLO Record* module forwards the corresponding SLO so that monitoring is continued by the proposed component, as in Figure 4 (via the *SLO Record* interface).

2.3 Prototype Tool

To show the component feasibility, functional tests were carried out via a prototype to perform a proof of concept of the component. The prototype allowed to simulate the execution and monitoring of a process according to the proposed component.

The prototype was developed with Oracle Java 8 (for back-end development), Spring Boot[4] (for configuration management), Spring Tool Suite[5] (as IDE) and Angular JS[6] (for the front-end development).

Figures 7–9 show examples of the developed dashboard prototype. They were designed following the specification presented in Section 2.1 for the *Dashboard* interface of the *BLA Report Viewer*.

[4]http://spring.io/projects/spring-boot
[5]http://spring.io/tools
[6]http://angular.io

Figure 7 shows the screen proposed to consolidate the results. The two charts at the top show a summary of cumulative results for all the organization's processes for BLAs and SLAs, including the percentages of *passed*, *failed* and *warning* results. Considering the BLA and SLA meta-models on which the proposed component is primarily based (cf. Section 2.1), there may be different levels of *passed* and *failed* for some BLAs and SLAs. The two bottom charts show a summary of the cumulative results but separated by BLA and SLA, including all those being used by the organization's processes at that time. This same consolidation structure can be used to show, for example, the data of a single process of the organization.

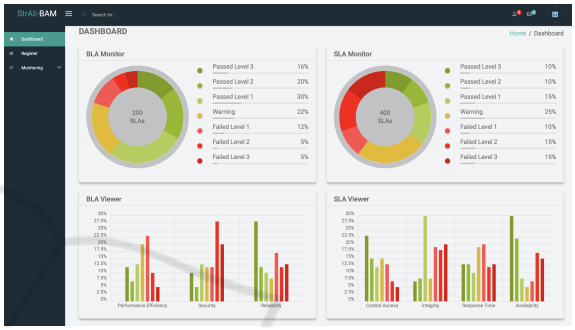


Figure 7: Dashboard prototype – indicators overall.

Figure 8 shows data detailed by BLA. For each BLA of a process, its basic information such as name and description (i.e., NFR type, operator and main target) is shown. It is also possible to check its current status in terms of *passed*, *failed* or *warning* consolidated result, based on the last average calculation. It is also possible to consult the historical basis for each BLA. For each BLA, one can also view the data of the SLAs derived from it and hence associated with it, as in the figure for BLA *Security* and process *Loan Provider*. SLAs are shown followed by their basic information. Thus, if a BLA has *failed* or is on *warning*, it is possible to verify the root cause for this.

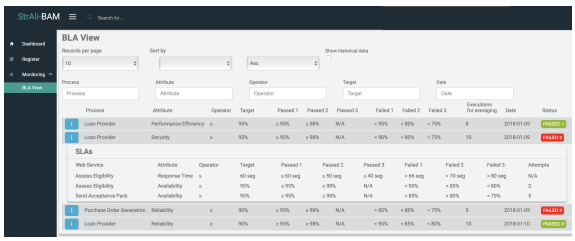


Figure 8: Dashboard prototype – BLA view.

Figure 8 shows the option to fully detail one BLA. The BLA *Performance Efficiency* is illustrated here. The first two data frames show the same data as the previous screen. The following frames detail the

monitoring results for each process instance execution considering the data shown in the first two frames.

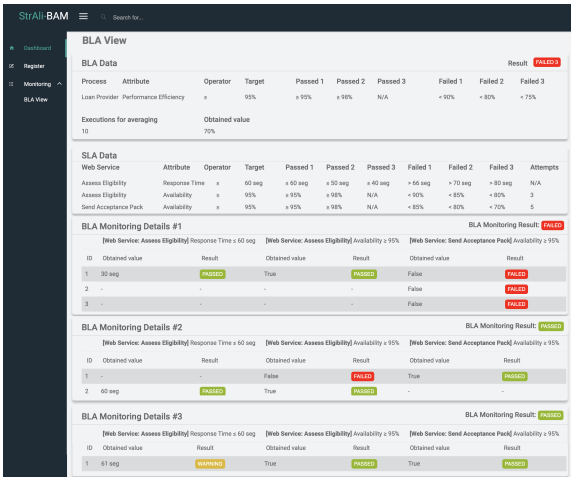


Figure 9: Dashboard prototype – BLA monitoring details.

3 EVALUATION

The loan process model in Figure 1 was explored to evaluate the proposed component. The BLA "Performance efficiency[7] $\geq 95\%$" (associated with the activities *Assess elegibility*, *Prepare acceptance pack* and *Send acceptance pack*) was chosen for this evaluation. This BLA was defined considering the rules defined by the BLA meta-model on which the proposed component is primarily based (cf. Section 2.1). The details of the BLA are not shown graphically in the process model, but only in terms of the element properties in the tool. Some of these details can be seen in the prototype presented herein (cf. Figure 8). In Figure 8, it is possible to consult, for example, the different thresholds for the application of penalties in case of non-compliance with the target and different thresholds to be awarded bonuses in case the goal is reached more satisfactorily than expected. For example, for this BLA, the target is 95%, with a tolerance of 5% and three penalty levels ($< 90\%$, $< 80\%$ and $< 75\%$). The lower the level of performance obtained, the higher the penalty applied. On the other hand, there are two levels of bonus ($\geq 95\%$ and $\geq 98\%$); the higher the level, the higher the bonus.

Using the proposed infrastructure (cf. Figure 5) and component (cf. Figure 4), the process was executed and monitored. The process model in Figure 1 was implemented in WS-BPEL as in Figure 10. The dotted red line highlights the code fragment corresponding to the implementation of the three process activities associated with the BLA under analysis. Four services were used to perform these three activities (two services for activity *Assess elegibility*, one service for *Prepare acceptance pack* and one service for *Send acceptance pack*). To achieve the target of this BLA, three SLAs were defined: 'response time' and 'availability' for the first service used to execute *Assess eligibility* and 'availability' for the service used to execute the activity *Send acceptance pack*[8]. Some details of each SLA are in Figures 8 and 9, including their targets, different penalty thresholds and different thresholds for granting bonuses, and maximum number of attempts per process instance. The Fixture Factory[9] tool was used to generate random data (i.e., the mocks) to simulate the SLA processing.

Figure 11 shows a consolidated view of the results of the monitoring simulation for the loan process. The BLA is considered *failed* for an instance execution if any of the associated SLAs are finalized *failed* for that execution. The third SLA, referring to the second service, is only evaluated if the first service could be executed, i.e., if it was available. Per Figure 11, one can observe that the scenarios relevant for the business areas are different from IT. For example, an SLA that fails may not necessarily reflect a failure in the corresponding BLA. Thus, the business area needs not to be involved in minor technical issues, which can be handled in isolation by IT. The calculations of each BLA and each SLA is performed independently although the BLA calculation considers the result of SLAs that are associated with it. Each SLA presents an isolated view specific to IT management and the calculation to determine whether its target is being reached is performed in a parametrized way for each SLA. For example, for the SLA Response Time, the calculation is being done every five executions of the associated service. The same is occurs for BLAs, which in this case is being done every 10 process instance executions. In Figure 11, results are shown for 20 instance executions, which enabled to calculate the BLA average value twice, whose results were 90% (Warning) and 70% (Failed Level 3). Also, depending on the nature of the service, it is possible to parametrize the maximum number of attempts it will retry, in case its SLA is failed, before the corresponding BLA is considered having failed.

An example of a situation where a technical prob-

[7]Per the adopted NFR nomenclature, "performance efficiency" means: degree to which a process can efficiently use an amount of resources (such as software, products and hardware) under stated conditions (Castro et al., 2019).

[8]The definition of which services should have SLAs associated and the QoS attributes and levels best suited to achieve a BLA goal are outside the scope of this paper.

[9]http://github.com/six2six/fixture-factory

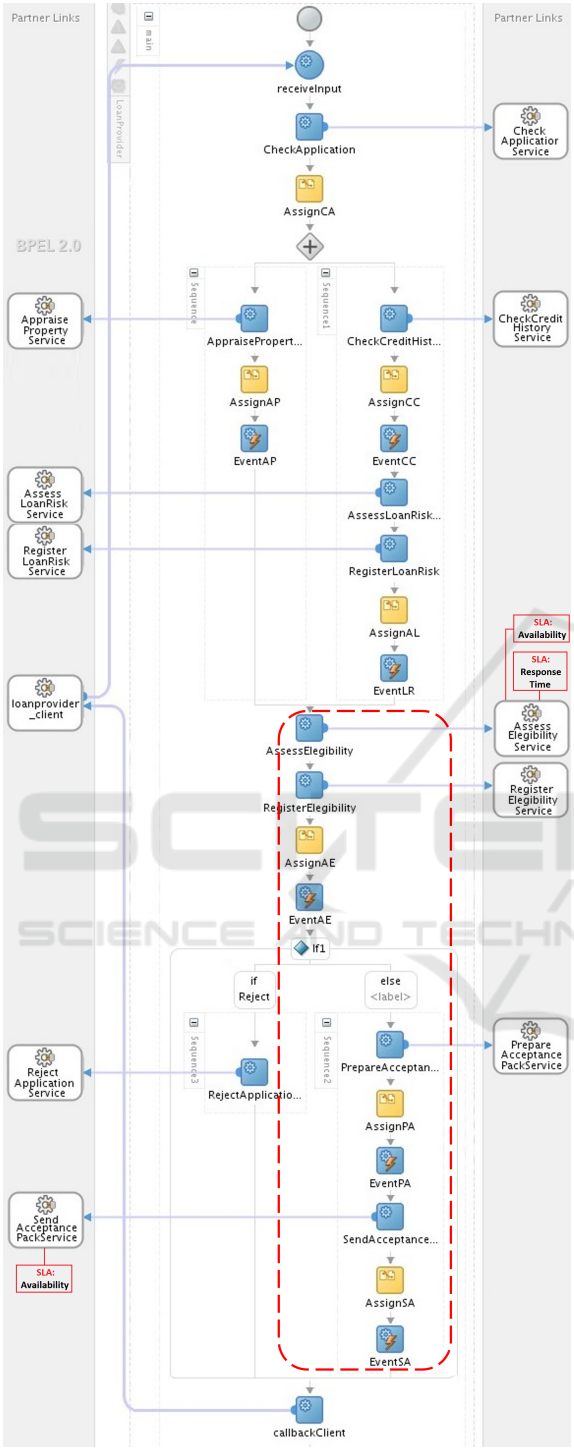


Figure 10: Executable version in WS-BPEL of the loan business process (for validation of the proposed component).

lem is not reflecting in the process is represented in the instance #4, where there was an availability problem for the service *Assess eligibility*, but the second attempt was successful. In addition, considering also the second and successful attempt, the reply was above the 60-second goal, albeit within the 6-second tolerance. In despite of this, the activity ended up being executed in 64 seconds, i.e., below the expected of 66 seconds including the tolerance, even in the face of an availability issue. Business managers may not be interested if a service is experiencing availability issues, but they may be interested if they find that this problem is affecting their business, which was not the situation in this instance execution. In addition, even the two services showing poor availability rates, the BLA is only on warning at the end of the first cycle of 10 instance executions, since the BLA has failed for only one of these executions according to its settings.

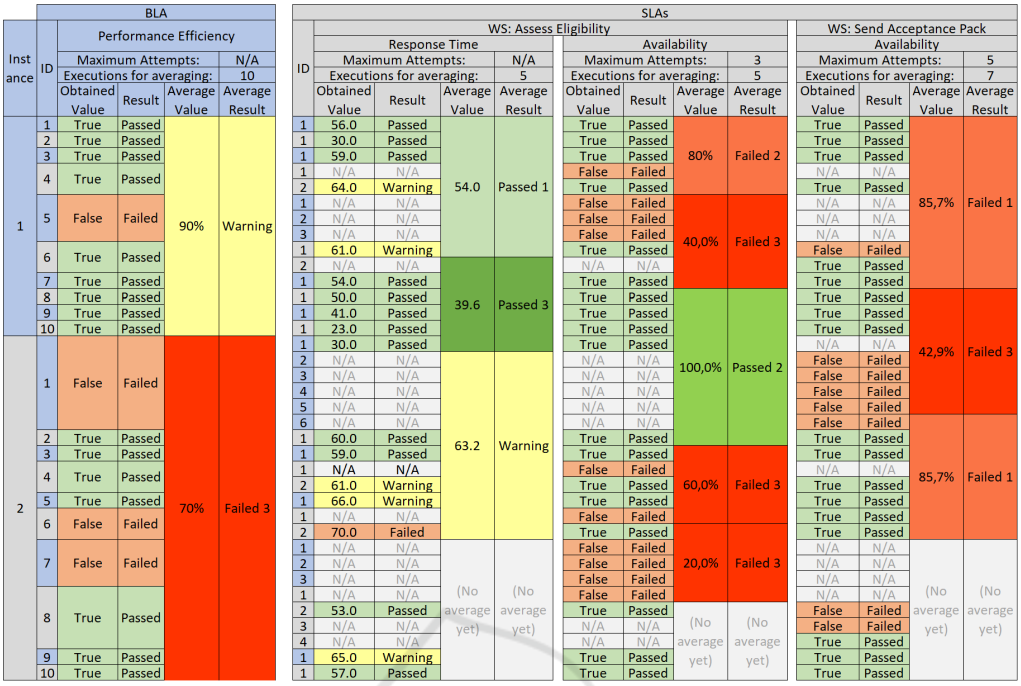
For the second cycle of 10 instance executions, the BLA evaluation deteriorates and is assessed below the lower threshold. One of the reasons is that the SLA response time behaved worse. However, this simulation presents only a proof of concept of how NFR monitoring can be treated at two different and complementary levels, with insights that may interest managers and teams of different areas and organizational levels. The way the data is presented in this table sought to present an overview of all the results of the simulation in a grouped form, which makes it difficult to understand. The emphasis is for a graphical tool to provide specific views for each profile, following guidelines for appropriate user experience.

The data in Figure 11 are not considering penalty and bonus values (individual or accumulated). The prototype still does not carry out them.

4 CONCLUSION

The objective of this paper was to present the StrAli-BAM component, designed to allow the StrAli-BPM framework monitoring process NFRs at both process and service levels. The component proposed is split into a monitoring architecture and an event-based infrastructure for execution and monitoring. While most approaches focus exclusively on web service monitoring and few others address business process monitoring only in terms of KPIs, the component presented herein aims at integrated monitoring.

For future work, we plan to: to finish the development of the indicator dashboard to view the data being monitored in real-time; include support for monitoring KPIs besides BLAs and SLAs; and conduct evaluations with scenarios closer to the actual settings in organizations dealing with this type of scenario.



BLA					
Instance	ID	Performance Efficiency			
		Maximum Attempts:			N/A
		Executions for averaging:			10
		Obtained Value	Result	Average Value	Average Result
1	1	True	Passed	90%	Warning
	2	True	Passed		
	3	True	Passed		
	4	True	Passed		
	5	False	Failed		
	6	True	Passed		
	7	True	Passed		
	8	True	Passed		
	9	True	Passed		
	10	True	Passed		
2	1	False	Failed	70%	Failed 3
	2	True	Passed		
	3	True	Passed		
	4	True	Passed		
	5	True	Passed		
	6	False	Failed		
	7	False	Failed		
	8	True	Passed		
	9	True	Passed		
	10	True	Passed		

SLAs																	
ID	WS: Assess Eligibility								WS: Send Acceptance Pack								
	Response Time				Availability				Availability								
	Maximum Attempts:			N/A	Maximum Attempts:			3	Maximum Attempts:			5					
	Executions for averaging:			5	Executions for averaging:			5	Executions for averaging:			7					
	Obtained Value	Result	Average Value	Average Result	Obtained Value	Result	Average Value	Average Result	Obtained Value	Result	Average Value	Average Result					
1	56.0	Passed	54.0	Passed 1	True	Passed	80%	Failed 2	True	Passed	85,7%	Failed 1					
1	30.0	Passed			True	Passed			True	Passed							
1	59.0	Passed			True	Passed			True	Passed							
1	N/A	N/A			False	Failed			N/A	N/A							
2	64.0	Warning			True	Passed			True	Passed							
1	N/A	N/A			False	Failed	40,0%	Failed 3	N/A	N/A							
2	N/A	N/A			False	Failed			N/A	N/A							
3	N/A	N/A			False	Failed			N/A	N/A							
1	61.0	Warning			True	Passed			False	Failed							
2	N/A	N/A	39.6	Passed 3	N/A	N/A			True	Passed							
1	54.0	Passed			True	Passed			True	Passed	42,9%	Failed 3					
1	50.0	Passed			True	Passed	100,0%	Passed 2	True	Passed							
1	41.0	Passed			True	Passed			True	Passed							
1	23.0	Passed			True	Passed			True	Passed							
1	30.0	Passed			True	Passed			N/A	N/A							
2	N/A	N/A	63.2	Warning	N/A	N/A			False	Failed							
3	N/A	N/A			N/A	N/A			False	Failed							
4	N/A	N/A			N/A	N/A			False	Failed							
5	N/A	N/A			N/A	N/A			False	Failed							
6	N/A	N/A			N/A	N/A			False	Failed							
1	60.0	Passed			True	Passed			True	Passed							
1	59.0	Passed			True	Passed	60,0%	Failed 3	True	Passed	85,7%	Failed 1					
1	N/A	N/A			False	Failed			N/A	N/A							
2	61.0	Warning			True	Passed			True	Passed							
1	66.0	Warning			True	Passed			True	Passed							
1	N/A	N/A			False	Failed			True	Passed							
2	70.0	Failed			True	Passed	20,0%	Failed 3	True	Passed							
1	N/A	N/A	(No average yet)	(No average yet)	False	Failed			N/A	N/A	(No average yet)	(No average yet)					
2	N/A	N/A			False	Failed			N/A	N/A							
3	N/A	N/A			False	Failed			N/A	N/A							
1	N/A	N/A			False	Failed			N/A	N/A							
2	53.0	Passed			True	Passed			False	Failed							
3	N/A	N/A			N/A	N/A	(No average yet)	(No average yet)	False	Failed							
4	N/A	N/A			N/A	N/A			True	Passed							
1	65.0	Warning			True	Passed			True	Passed							
1	57.0	Passed			True	Passed			True	Passed							

Figure 11: Monitoring simulation results.

ACKNOWLEDGEMENTS

This work was funded by Fapesp, Brazil (grants 2017/26491-1 and 2017/26487-4, holder Marcelo Fantinato) and Capes, Brazil (grants 88881.172071/2018-01, holder Lucinéia H. Thom).

REFERENCES

Barros, V. A., Fantinato, M., Salles, G. M. B., and de Albuquerque, J. P. (2014). Deriving service level agreements from business level agreements: An approach towards strategic alignment in organizations. In *16th Int. Conf. on Enter. Inf. Syst.*, pages 214–225.

Carmo, A., Fantinato, M., Thom, L., Prado, E. P. V., Spínola, M., and Hung, P. C. K. (2017). An analysis of strategic goals and non-functional requirements in business process management. In *19th Int. Conf. on Enter. Inf. Syst.*, pages 262–273.

Castro, C. F., Fantinato, M., Aksu, U., Reijers, H. A., and Thom, L. H. (2019). Towards a conceptual framework for decomposing non-functional requirements of business process into quality of service attributes. In *21st Int. Conf. on Enter. Inf. Syst.*

Curbera, F., Duftler, M., Khalaf, R., Nagy, W., Mukhi, N., and Weerawarana, S. (2002). Unraveling the web services web: An introduction to SOAP, WSDL, and UDDI. *Int. Comp.*, 6(2):86–93.

Dumas, M., La Rosa, M., Mendling, J., and Reijers, H. A. (2018). *Fundamentals of Business Process Management*. Springer, 2nd edition.

Erl, T. (2005). *Service-Oriented Architecture – Concepts, Technology, and Design*. Prentice Hall.

Fahad, M., Moalla, N., and Ourzout, Y. (2015). Dynamic execution of a business process via web service selection and orchestration. In *Int. Conf. on Comp. Sci.*, pages 1655–1664.

Lubinski, T. (2008). Business activity monitoring: Process control for the enterprise. SL Corporation, Corte Madera, CA.

Presman, R. S. and Maxim, B. (2014). *Software Engineering: A Practitioner's Approach*. McGraw-Hill Education, 8th edition.

Salles, G. M. B., Fantinato, M., Barros, V. A., and de Albuquerque, J. P. (2018). Evaluation of the strali-bpm approach: Strategic alignment with bpm using agreements in different levels. *Int. J. of Bus. Inf. Syst.*, 27(4):433–465.

Salles, G. M. B., Fantinato, M., de Albuquerque, J. P., and Nishijima, M. (2013). A contribution to organizational and operational strategic alignment: Incorporating business level agreements into business process modeling. In *Int. Conf. on Serv. Comp.*, pages 17–24.

Sheng, Q. Z., Qiao, X., Vasilakos, A. V., Szabo, C., Bourne, S., and Xu, X. (2014). Web services composition: A decade's overview. *Inf. Sci.*, 280:218–238.

TOG (2018). The archimate enterprise architecture modeling language. http://www.opengroup.org/ subjectareas/ enterprise/archimate-overview. The Open Group.