This is a pre-print draft of the following paper, please reference it correctly when citing it:
Oliveira, K., J. Castro, S. España and O. Pastor (2013). Multi-level autonomic business process management. 14th Working Conference on Business Process Modeling, Development, and Support (BPMDS 2013). S. Nurcan, H. Proper, P. Sofferet al. Valencia, Spain, Springer LNBIP. **147:** 184-198.

Multi-level Autonomic Business Process Management

Karolyne Oliveira[1], Jaelson Castro[1], Sergio España[2], Oscar Pastor[2]

[1]Universidade Federal de Pernambuco—UFPE, Recife, PE 50 740-560, Brazil
[2]Centro de Investigación en Métodos de Producción de Software, Universitat Politècnica de València, Valencia, Spain

{kmao, jbc}@cin.ufpe.br
{sergio.espana, opastor}@pros.upv.es

Abstract: Nowadays, business processes are becoming increasingly complex and heterogeneous. Autonomic Computing principles can reduce this complexity by autonomously managing the software systems and the running processes, their states and evolution. Business Processes that are able to be self-managed are referred to as Autonomic Business Processes (ABP). However, a key challenge is to keep the models of such ABP understandable and expressive in increasingly complex scenarios. This paper discusses the design aspects of an autonomic business process management system able to self-manage processes based on operational adaptation. The goal is to minimize human intervention during the process definition and execution phases. This novel approach, named MABUP, provides four well-defined levels of abstraction to express business and operational knowledge and to guide the management activity; namely, Organizational Level, Technological Level, Operational Level and Service Level. A real example is used to illustrate our proposal.

Keywords: Autonomic business process models, workflow management, self awareness, context awareness

1 INTRODUCTION

System components and software have been evolving to deal with the increasing complexity of stakeholder needs. Automating the management of computing resources is a major challenge.

Some recent works have proposed the use of Autonomic Computing (AC) concepts [1] to help to effectively manage enterprise systems and applications [3]. Indeed, a major application area for autonomic computing is aimed at freeing system administrators from the details of system operation and maintenance, improving robustness of systems and decreasing the total cost of ownership [2, 6].

Autonomic Computing Systems (ACS) are able to: (i) Self-configure; (ii) Self-heal; (iii) Self-optimize and, (iv) Self-protect. The secondary properties of autonomic systems are: (i) Self-awareness: An autonomic system requires to know itself; (ii) Context-awareness: An autonomic system should be aware of its execution environ-

ment by exposing itself and discovering other systems in the environment; (iii) Openness: An autonomic system should be able to function in a heterogeneous environment and be implemented on open standards and protocols; (iv) Anticipatory: One critical property from the perspective of the users is that an autonomic system should be able to anticipate its needs and behaviors to act accordingly, while keeping its complexity hidden [10].

However, the vision of autonomic computing should be not restricted to the area of system administration, management and maintenance. For example, it can also be applied to the area of process-aware information systems to effectively and efficiently deal with changes in several aspects of these applications.

Business Processes and Business Process Management (BPM) are essential in many modern enterprises. They constitute **organizational** and **operational knowledge** and often perpetuate independently of changes in the personnel or the infrastructure [4]. Autonomic computing principles can also be adapted to help organizations survive in dynamic business scenarios.

A Process that is compliant with AC principles is referred to as Autonomic Business Process (ABP) [9] [12]. ABPs (a.k.a. autonomic workflows) must have the capability to adjust to **environment variations (**context). If one **component service node** of an ABP becomes unavailable, a mechanism is needed to ensure a business process execution is not interrupted [8] [18]. ABP differs from traditional workflow in the fact that it relies on autonomic techniques to manage adjustments during its execution. Therefore, it enables dynamic and automatic configuration of its definition, activities and resources. It also allows self-optimization and self-healing. Furthermore, autonomic workflows must have intelligence to analyze situations and deduce adaptations at run-time.

This work investigates the application of autonomic computing principles can to business processes management. Our goal is to help organizations survive in dynamic environments. More specifically we want to face an open challenge in the area, which is to promote modularization and separation of concerns (SoC) in ABP Models [14].

This paper proposes a framework that exploits the high variability in service-oriented system environments by using models of the system context and by providing autonomic adaptations which rely on operationalizations of Non-Functional Requirements (NFR). This framework guides its adaptation according to a Multi-level Autonomic Business Process named MABUP, whose modeling process is presented in [9]. The benefits are manifold and include addressing scalability problems and improving the understandability of ABP in complex scenarios [11].

A MAPE cycle (Monitor-Analyze-Plan-Execute) is used, considering both the system (self) and the instrumented BPM (context).

This paper is structured as follows. Section 2 presents the background and related works. In Section 3, we introduce a running example. Section 4 proposes a multi-level autonomic business process management approach. Lastly, Section 5 discusses the proposal and outlines further research..

2 RELATED WORKS

Strohmaier and Yu in [13] have presented the first attempt to apply autonomic computing principles to workflow management. Their work shows that certain levels of autonomy can already be achieved with available techniques and introduces the concepts of the different degree of "autonomy" in workflow systems. However, no novel technique is proposed.

In order to deal with autonomic features, an interesting issue emerged when composing applications: the ability to bind and re-bind abstract activities to concrete services at run-time. Several researchers have addressed this issue. Lee et. al. [7] discussed managing run-time adaptations in long-running scientific workflows. In their work, adaptations have been described in terms of the functional decomposition of autonomic managers into monitoring, analysis, planning and execution components. Mosincat et al. [8] proposed fault tolerant execution of BPEL processes executing dynamic binding of services, performing a process transformation before the execution, for using a selection component at run-time. In Haupt et. al. [5], the workflow and the services comprising it are treated as managed resources controlled by hierarchically organized autonomic managers. Their work attempts to treat the complexity problem of autonomic workflows by using hierarchical workflow but, it does not explore any modularity technique neither how these different levels can guide different kinds of adaptations.

The modularization helps to treat the complexity of large-size models [11]. Once the management of business process models can be realized at different levels of granularity, there is a tradeoff between monitoring granularity and diagnostic precision. The finest level of monitoring granularity is at the functional level where all leaf level tasks are monitored. The disadvantage of fine-grained level monitoring is high monitoring and diagnostic overhead. Coarser levels of granularity only monitor higher-level business goals. In these cases, less complete and high-level monitoring data is generated, leading to multiple competing diagnoses. Both monitoring and diagnostic overheads are lower. The disadvantage is that if requirements denials are found, multiple diagnoses are returned, each pinpointing possible failures [17].

Generally, self-adaptive software is a closed-loop system with feedback from the self and the context. The Autonomic Computer System's building block, named autonomic manager, constantly interacts with managed element in order to maintain its equilibrium in face of incoming perturbations. The MAPE cycle (Monitor-Analyze-Plan-Execute), represented in Figure 1, is an implementation of the classical feedback control technique [6].

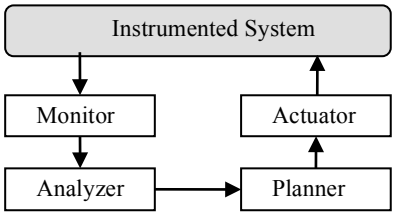

Fig. 1. Closed Loop Control Mechanism

Monitoring the execution context represents another important aspect to implement an autonomic behavior and for reacting to events. However, even if, they exploit workflows to design applications and to specify causal constraints among activities and pages, the major works do not use contextual information in the autonomic business process management system [15].

Furthermore, these works do not explore another crucial issue that is the expressiveness of the operational knowledge in the business process. This problem is related to how the metrics impact the in the management of business process and how to offer understandability of autonomic behavior.

The analysis of the related work shows that there is a lot of interest in the realization of systems and approaches able to deal, dynamically and automatically, with composition, binding, failures and other aspects of definition and execution of composed processes. However, these works do not explore the modularization of business process that helps to treat the management of autonomic business process in complex scenarios and expressiveness of autonomic features in business process models.

3 RUNNING EXAMPLE

A running example will be used to illustrate our approach. The example consists of a large system that requires to be managed in an autonomic manner. Specifically we treated the characteristic of Self-Optimization. We examine the CAGED (General Register of Employed and Unemployed), a project under the Ministry of Labour and Employment of Brazil (MTE) and governed by law 4923/65. It supports the submission of monthly declarations of change of company's employees due to dismissal or admittance (CAGED movements). The deadline for submission is the 7th day of every month. The data submitted are related to the previous month (i.e. its competence). The declarations are processed to generate operational and statistical data for the ministry of labor and employment.

Considering these characteristics, the process is started when the MTE opens the competence of CAGED reception. In this case, the declarants can send their CAGED declaration. The submit process a CAGED declaration is composed of a selection (including filling) of a CAGED file (detect the movements), sending this file and receiving a CAGED Receipt. During the reception, MTE can decide to start the processing of the files that provide operational data for MTE. Depending on timing constraints, such as holidays or some other guidance, MTE staff closes competences of reception. After all checks in CAGED reception and processing, operational staff closes the reception processing and starts statistical processing.

Having in mind that the CAGED submission is an activity that is critical to the success of CAGED process as a whole, we explain the main points that must be considered in its management. The capture operation of a CAGED file can be performed in different manner: (i) Generate CAGED File, that in general is executed through payroll systems of accounting firms which generate a CAGED declaration (in a predefined layout) without MTE analysis and signature; (ii) Generate Analyzed CAGED File, that is executed by declarants through a MTE offline system that generate a

CAGED File with analysis and signature; and (iii) Generate Short Analyzed CAGED File, that is performed by declarants to generate and send CAGED declarations with a maximum of 36 movements.

The CAGED Declaration can be generated in two ways: CAGED File without signature of analysis or an Analyzed and Signed CAGED File. The first one can be analyzed through the activity of Generate Analyzed CAGED File or to be analyzed and sent by a service. An analyzed CAGED file, the second one, is sent without additional analysis; in this case, two manners are possible: CAGED File with less than 1500kb can be sent through a service that only verifies MTE signature. CAGED Files with more than 1500kb must be sent throw a desktop tool that optimizes its transmission.

The previews tasks are executed by the declarants that send their CAGED file to MTE. A MTE reception service is available to receive and store these files and triggers the generation of a CAGED Receipt to the declarants.

CAGED is complex system that is hard to be managed. Some qualities attributes must be assured according to its SLA such as availability, response time, processing time, etc; and all these with less human intervention. In this sense, business process models must be able to adequately represent: (i) Mission-critical Activities; (ii) The way the activities are executed and monitored; (iii) Which quality attributes and environmental context must be monitored and to what autonomic features; (iv) How systems needs to be adapted in case of some quality attributes deviation.

4 MULTI LEVEL AUTONOMIC BUSINESS PROCESS MANAGEMENT

To provide autonomic business process, we considered the use of modularization and separation of concerns to represent the different features. In Figure 2, we depict an overview of our approach to the management of a multi-level business process model that includes two main stages: (i) The modeling phase exploits the modeling of four abstraction levels: Organizational Level, Technological Level, Operational Level and Service Level; (ii) The management phase includes a MAPE module that uses the modeled autonomic business process and the metrics provided by the systems to guide the adaptation.

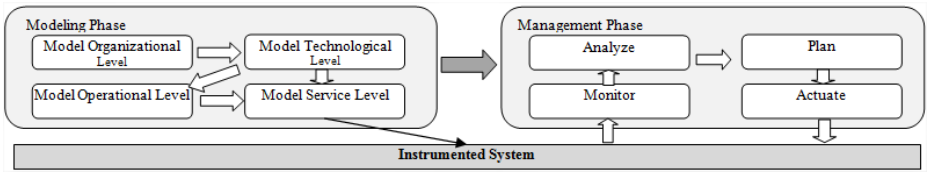

Fig. 2. Overview of the modeling and management phases of MABUP

In the sequel we present our Multi-Level Autonomic Business Process approach, named MABUP, which consists of well-defined abstraction level and a closed-loop mechanism to provide system adaptation (see Figure 2). As mentioned, our framework considers in the management of the processes both context information and quality attributes of the system.

4.1 Modeling Phase

In this section we present the four well-defined abstraction levels of MABUP. In the modeling phase, we used a combination of two languages; namely Communicative Event Diagram and BPMN (see Figure 3). Due to lack of space, we depict a simplified version of the models; for instance, we do not capture all context variations required for self-configuration, self-healing and self-protection. Hence, this paper focuses on the self-optimization feature of the "Declarant submits a declaration" critical communicative event.

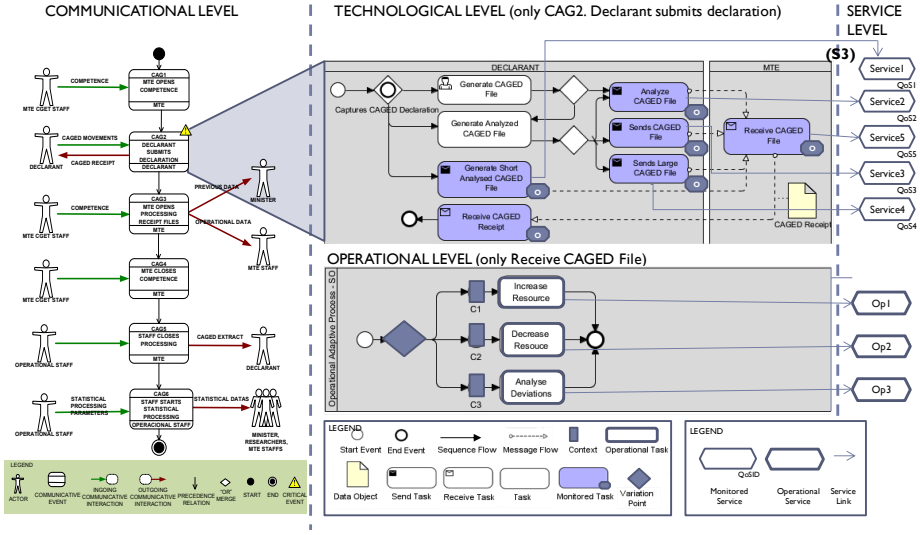

Fig. 3. Organizational, Technological, Operational and Service Levels

Model Organizational Level

The Organizational Level is an important abstraction level in our approach since it makes business process models more immune to changes in the support technologies by focusing on the essentials of the business behavior. It supports the modularization of Autonomic Business Process by providing a level which presents critical activities that must be refined and monitored.

According of the principles of Communication Analysis, Communicative Event Diagram provides a notation to specify the Organizational Level of a business process [1]. In this notation, there is a special primitive for process modelling, named communicative event. It represents the triggering of an activity that receives an incoming message, processes it and provides an output. Hence, it represents the organizational behavior resulting from a given change in world (subject system), intended to account for that change by gathering information about it. A communicative event is a set of actions related to information (acquisition, storage, processing, retrieval, and/or distribution), that are carried out in a complete and uninterrupted way, on the occasion of an external stimulus.

Communication Analysis offers guidelines to allow the identification of communicative events. The following modularity criteria are used to guide modeling [3]:

- Trigger unity, that presumes that the event occurs as a response to an external interaction by an actor that triggers organizational reaction;
- Communication unity, that describes that each and every event involves providing new meaningful information;
- Reaction unity, that outlines that an event triggers an Information System (IS) reaction, which is a composition of synchronous activities. Events are asynchronous among each other. This criterion defines a temporal encapsulation.

MABUP introduces the concept of Critical Event, a special kind of communicative event which is a mission-critical activity. Critical events must be refined to provide information about its behavior and indicate the sub-activities that are monitored according to autonomic features.

In this sense, Figure 3 presents the Organizational Level of Business Process of CAGED, where we highlight the critical event "CAG2 - Declarant submits declaration". It is an activity that has as input all CAGED movements and, as an output, the receipt. Stakeholders highlight that it is critical to the success of the CAGED process.

Model Technological Level

Technological Level represents the refinement of a critical event processing to model important aspects that can impact software adaptation, such as:

- *Present different alternatives to perform an activity*: Some activities can be executed in different ways in a company. It is important to map these differences to analyze if they require special ways to be managed.

In our example, the generation of file can be performed in three different manners: (i) Generate CAGED File; (ii) Generate Analyzed CAGED File; and (iii) Generate Short Analyzed CAGED File. If one of these activities became unavailable, another alternative can be executed to guarantee the system operability until all processes return to an optimum state.

- *Indicate External dependences*: External dependences are important to be expressed as they can lead to interoperability problems. In that sense, interoperability demands human intervention coordination related to the efforts to ensure performance, scalability, correctness, or reliability of applications in the presence of concurrency and failures [2]. We consider that this calls for autonomic characteristics such as optimization, healing and protection. Furthermore, when a system relies on an external service, some kind of service level agreement (SLA) is required.

Figure 3 indicates that the "Receive CAGED Receipt" task is impacted if the "Receive CAGED File" task becomes unavailable or has a poor performance.

- *Highlight monitorable tasks*: Some events are too complex and have to be processed by different kind of components. Hence, some are monitorable and others not.

For example the “Generate analyzed CAGED File” task is executed outside the MTE domain by an offline desktop tool that is not expressed in SLA as a monitorable module. In our example the monitored tasks, such as “Receive CAGED File” are depicted in gray.

- *Define the autonomic characteristics and symptoms of monitorable tasks*: Indicate the autonomic principles that will be considered to monitor the tasks. The SLA document is a good source of information as it presents the quality attributes and their respectively values that must be assured.

In the running example, we only consider the self-optimization (O) principle to all monitorable tasks.

Model Operational Level

Operational level indicates the operational knowledge required to manage the process. Business analysts define the contextualization of the monitored task using their knowledge about the business domain to identify information that can affect the process and express the operational knowledge to manage it. The information is codified in terms of contexts, context variability and operational tasks.

The variability analysis is focused on context variability. Both variants and variation points are related with the contextual information. They indicate how the contexts variation can affect the autonomic actions. Operational Task expresses autonomic actions in the systems to assure the optimal state of the system and express a refinement of the autonomic part of a Monitored Task.

In our running example, we defined Self-Optimization (O) as the desired autonomic principle. As previously noted, it is related to the efficiency NFR, which in turn can be decomposed into others characteristics such as Response Time and Space Utilization. In this example, we deal only with Response Time attribute.

For instance, a variation point of the Receive CAGED File (see Table 1) task is associated with the response time. This variation point is a contextualization for the alternative ways how the adaptation actions are selected to respond to environment changes. The alternatives are represented as variants that are associated with the variation point. In the example, the variants are actions to control or regulate the response time deviation of the CAGED file reception. The tasks are associated with contexts describe in expressions that activate the presence of a variant in the variation point.

Given that in our running CAGED there is a deadline for declaration submission, it is important to measure the daily reception. As observed in Figure 4, the reception rate peaks during the first seven days of the month. This trend must be considered to allow the selection of a correct adaptation. Trend analysis methods [16] can help business analysts to predict future outcomes tracking variances in historical results.

In order to treat Response Time deviations that may be related to the performance of the “Receive CAGED File” activity, we defined in Figure 3 (Operational Level) three different tasks (operationalizations): (i) Increase resource; (ii) Decrease Resource; and (iii) Analyze deviation. The 3 contexts (C1, C2 and C3) present in Figure 3 (Operational Level) are defined in Table 1.

Table 1. Contextualization of Autonomic Business Process

Monitored Task	Contextualization			
	Variation Point	***Variants***	***Context***	***Quality***
Receive CAGED File	Response Time Deviation	Increase Reception Rate	C1: ReceptionTrendIsOK= true and LastThreeCycleIncreasing= true	Response Time >= 220ms
		Decrease Reception Rate	C2:ReceptionTrendIsOK= true and LastThreeCycleDecreasing= true	Response Time <= 110ms
		Deviation Reception Rate	C3:ReceptionTrendIsOK= false and LastThreeCycleIncreasing= true	Response Time >= 220ms

Considering the above, the operational level presents the new interconnected concepts in ABP models: variation point, variants, operational task and can be defined as follow:

$$OL = \{VP, Var, OT, Rel\{VPxVarxOT\}\}$$

where: OL is the Operational Level; VP is Variation Point; Var is Variant, OT is Operational Task and Rel expresses the relationship between then. OL is a conjunct of VP, Var and OT where VP is linked with Var and Var is related with OT.

Model Service Level

Both monitorable and operational tasks should be linked to system services. In a service-based mission-critical system, adaptation is an activity with the objective of delivering an acceptable/guaranteed service based on SLA (Service Level Agreement).

One of the key components in SLA is SLO (Service Level Objective) which specifies QoS (Quality of Service) metrics governing service delivery. Each SLA may include several SLOs, each corresponding to a single QoS parameter related to quality factors.

In the Service Level, the services linked with monitorable tasks are called monitorable services that should be checked according to the parameters presented in the SLO. Whereas the services linked with operational tasks are named as operational services that have the objective of returning the system to an optimal state in an autonomic manner. For example, the "Receive CAGED File" activity is linked to "Service5" that should assure a response time of 190 ms. The concepts used in this level are related as follow:

$$MonitoredService \subseteq Service \text{ and } OperationalService \subseteq Service \; (MonitoredService \cap AutonomicService = \{\} \wedge MonitoredService \cup AutonomicService \subseteq Service)$$

That is, *MonitoredService* and *OperationalService* are a sub-conjunt of Service and are mutually exclusive because other services may exist (ie. a disjoint and incomplete inheritance of *Service*).

4.2 Management Phase

Monitor

The monitoring component collects indicators provided by the system from time to time (cycle). The monitor checks both context information and NFRs related to the

system. In our approach we assume the use of context sensors, which are computational entities that provide raw data about the operational environment presented in the instrumented BPM; and service quality attributes that must be assured. The monitor module verifies the processes at a well-defined level of modularity (Technological Level) and affects services-oriented systems with coarse grained adaptation at run-time.

As shown in Figure 5, the monitor module is divided in Context Monitor, QoS Monitor and Monitor Engine. The context monitor reads and processes the contextual data provided by the instrumented system. The QoS Monitor reads and processes the log database of all monitored services that support all monitored tasks and their respective metrics according to the SLO. The Monitor Engine processes the information provided by Context Monitor and QoS Monitor, collects the relevant data according MABUP model and passes it to the diagnostic component for analysis.

Algorithm 1 defines how the monitor process works, ie. it obtains the MABUP model, treats the monitored tasks, collects the metrics of each one and returns the data to be used by analyzer module.

Algorithm 1. Monitor algorithm

```
monitor(MABUPModel model)
1. MonitoredTaskValue[] mtvs
2. VariationPointValue vpv
3. VariationPointValue[] vpvs
4. for each tl ∈ getTechnologicalLevels(model)
5.   do for each mt_i ∈ getMonitoredTasks(tl)
6.     MonitoredService ms ← getMonitoredService(mt)
7.     OperationalLevel op ∈ getOperationalLevel(mt)
8.     do for each vp_j ∈ getVariationPoints(op)
9.        vpv ← getVariationPointValues(mt,ms,vp)
10.       vpvs[j] ← vpv
11.    end for
12.    mtvs[i] ← getMonitoredTaskValues(mt, vpvs)
13.  end for
14. end for
15. return mtvs
```

The *getMonitoredTaskValues* method (line 12) localizes, though QoS Monitor, the metrics of the monitored service expressed in the Service Level based on the identification of the monitored task. It calculates the average of the metrics obtained in a cycle by all requested services that support the monitored task as follow:

$$f(s) = \frac{1}{n}\sum_{i=1}^{n} QoS(s_i)$$

where: f(s) the function to verify the QoS of a service s; n is the number of requested services in a cycle; QoS(s_i) is the metric obtained for a requested service.

Analyze

In our approach, this component has the objective to evaluate the metrics obtained by the monitor. Considering the measures obtained by the monitor, the diagnostic component checks the contexts expressed in the Business Process Model. In case of some deviation, the planner (section 4.2.3) selects the appropriate operational task that represents autonomic interventions in the system.

Considering the management of business processes, it is important to assure the service level agreement related to these processes. In this sense, the analyzer module verifies the monitored metrics in the business process model and in case of deviation above a predefined threshold the module verifies the operational contexts that affect the process and the variants related to these contexts. The analyzer module is divided in two components: Domain Assumption Verifier and Contextual Business Process Model Diagnosis.

The "Domain Assumption Verifier" component identifies all quality attributes related to the monitored task and in case of metric deviations the "Contextual Business Process Model Diagnosis" component analyzes the contexts values. Algorithm 2 is a simplified version of this process. According to lines 2-4 the "Domain Assumption Verifier" component reads all monitored task values obtained by the monitor module and verifies the QoS deviation. The isQoSDeviated method (line 3) reads the Service Level, gets the expected metrics and compare them against the obtained values. In the "Contextual Business Process Model Diagnosis" component, the getDeviatedVariant method (line 5) checks all Variation Points and evaluates the context and quality expected in each Variant. The Analyzer module returns all the deviated variants.

Algorithm 2. Diagnosis algorithm

```
analyze(MonitoredTaskValue[] mtvs)
1. Variant[] deviatedVars
2. for each mtv  mtvs
3.   do boolean isQoSOut ← isQoSDeviated(mtv)
4.   if isQoSOut then
5.      deviatedVars += getDeviatedVariant(mtv)
6. end for
7. return deviatedVars
```

In our running example we had explored the monitored task "Receive CAGED File" that has the Self-Optimization (O) as the desired autonomic principle and in its operational decomposition presents the Variation Point related to the Response Time attribute. In this sense, we explored three scenarios according the data provided in Figure 4 that are related to the tree variants presented in this Variation Point:

- Scenario 1: In the beginning of the month the reception rate is increasing time-to-time and this is an expected change in the environment according to the CAGED characteristics. The system provided the following information: ReceptionTrendIsOK=true; LastThreeCycleIncreasing=true (Contextual); and ResponsteTime=382ms (QoS);

- Scenario 2: After the deadline to send the CAGED File, the competence is closed to receive files of previous month and opens to receive the next competence. As result, reception rate decreases time-to-time. Considering this, the system provided the following information: ReceptionTrendIsOK=true; LastThreeCycleDecreasing=true (Contextual); and ResponsteTime=80ms (QoS);
- Scenario 3: The reception is out of the reception peak and the reception rate is increasing, as shown in Figure 4 (10/06/2011). In this case, the system must evaluate the deviation but not increase the resource. Considering this, the system provided the following information: ReceptionTrendIsOK=false; LastThreeCycleIncreasing=true (Contextual); and ResponsteTime=457ms (QoS);

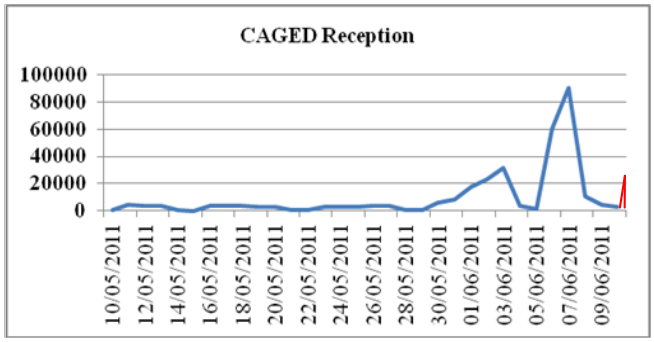

Fig. 4. CAGED Reception Rate

Considering scenario 1, 2 and 3; in different moments the analyzer module would return respectively the variants: Increase Reception Rate, Decrease Reception Rate and Deviation Reception Rate. As can be observed, it is important to emphasize that even if a quality attribute is out of the defined value, the deviated variants will only be returned only if there is contexts that enables it. For example if the system returns a response time of 350ms in the execution of the task "Receive CAGED File" and there is no context that influences in this value, the autonomic engine will not return deviations.

Plan

The main objective of planner module is planning the interventions that will be executed in the system according the information provided by the analyzer module. As shown in Figure 5, the planner module has two components: The Planner Execution Selector and the Process Status Manager. The component Planner Execution Selector reads all deviated variants obtained through analyze module and then obtain, from the Operational Level, the Operational Tasks related to these variants. The status of each selected Operational Task is managed through the Process Status Manager component to assess if this task can be executed. The status can indicate if there are other executions of the same Operational Task in the moment (in_progress) or if there are some previous plans that have the same operation and have not stated yet (uncommitted). In this case, the Operational Task does not enter in the plan that will be executed.

Regarding our running example, considering Scenario 1 (defined in section 4.2.2), the planner module returns the “Increase Resource” Operational Task. While in Scenario 2 the “Decrease Resource” Operational Task is returned. Finally, in the Scenario 3, the module returns “Analyse Deviations”.

Actuate

The actuator module has the objective of actuating in the instrumented system. For this reason, the module has two components as presented in Figure 5: The Actuator Manager and the Operational Task Assigner.

According the Operational Task returned by the planner module, the Actuator Manager accesses the Service Level to obtain the Operational Service with the parameters that it needs to be executed. The Operational Task Assigner access the instrumented system to finally perform the adaptation.

Considering our running example, if there is a plan to perform the operational task “Increase Resource”, the Actuator Manager component will access the specification of the service named “OP1”. This specification has the name of the service that is: *expandReceptionMem(ResourceType rt, ResourceValue rv, SystemModule sm)* where *ResourceType* is the type of resource that the analyst wants to increase, *ResourceValue* is the percentage of growth and the *SystemModule* is the identification of the subsystem. These values are predefined as in this case as: rt=” memory”; rv=0.1; and sm=”cag.Reception”. It corresponds to 10% of increase of memory resource.

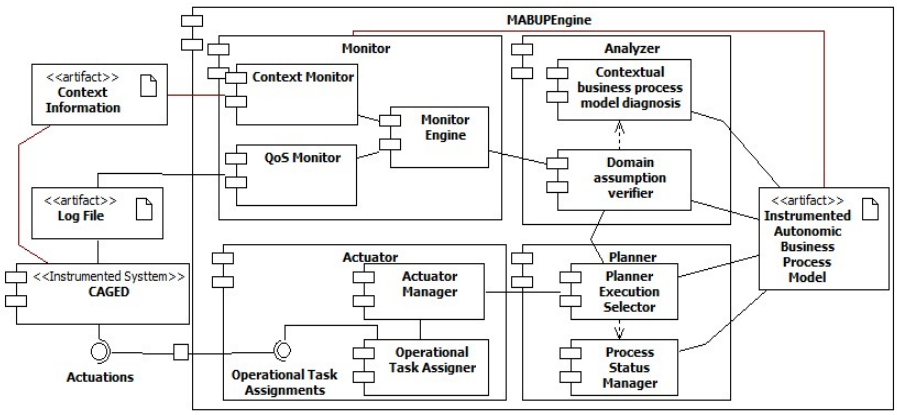

Fig. 5. MABUP Engine

5 DISCUSSION AND CONCLUSION

In this paper we outlined an approach for managing autonomic business processes following a multi-level strategy. The architecture an architecture of autonomic business process models that intends to represent the correct behavior of the systems from the business, technological, operational and service points of view. This proposal has

been divided into two parts: the modeling and the management phases. The modeling phase express how to obtain four abstract levels that instrument the business process model with autonomic features; namely, Organizational Level, Technological Level, Operational Level and Service Level. The management phase proposes a MAPE (Monitor-Analyze-Plan-Execute) engine that considers both instrumented ABPM (Autonomic Business Process Model) and the instrumented system obtaining information about the environment context and quality attributes to guide the adaptation.

The paper (1) defines the MABUP (Multi-level Autonomic Business Process) model used at runtime to represent the correct behavior; (2) presents the logical view on the architecture to manage the multi-level autonomic business processes; (3) introduces algorithms to perform the MAPE modules of our approach; and (4) shows how an existing system can be modeled to exploit our architecture.

The real example demonstrates the feasibility of the presented conceptual architecture and highlights its applicability to a real-life scenario. Furthermore, the benefits of our approach are manifold and include:

- *Modularity of ABPM:* We have relied on Communication Analysis principles to deal with the overhead in the monitoring phase. The Organizational Level helps to indicate mission-critical activities that must be treated in an autonomic manner. The Technological Level, a refinement of the mission-critical activities, specifies the autonomic tasks that must be monitored.
- *Scalability*: Modularity helps abstracting processes, thus reducing the complexity of ABPM, and increases the scalability of our approach;
- *Separation of concerns:* In our MABUP model the relationship between the business, technological and operational knowledge are explicitly expressed in different and interconnected abstraction levels of the model.
- *Understandability*: The different well-defined levels helps to provide understandability of ABPM;
- *Expressiveness of autonomic features in BPM*: In contrast to other approaches that need a knowledge database to express the metrics that affects the adaptation, our approach provides the concepts of critical event, monitored task, context variability and quality attributes expressed in a BPM that guide the self-management at runtime. All these concepts are interconnected to indicate how the metrics impact each autonomic business process.
- *Guide the adaptation based on metrics, as context and quality attributes, contained in the BPM*: Considering the knowledge provided in our MABUP model, the management module guides the adaptation according the context and quality attributes that affect the operational tasks.

Our approach is part of an ongoing research endeavor. As such, much remains to be done. As future work, we plan to perform a controlled experiment to empirically evaluate our proposal.

ACKNOWLEDGMENT

Research supported by CAPES and Spanish Ministry of Science and Innovation project PROS-Req (TIN2010-19130-C02-02), the Generalitat Valenciana project ORCA (PROMETEO/2009/015), and co-financed by ERDF structural funds.

REFERENCES

1. Espana, S. et al. 2009. Communication Analysis: A Requirements Engineering Method for Information Systems. *Proceedings of the 21st International Conference on Advanced Information Systems Engineering* (Berlin, Heidelberg, 2009), 530–545.
2. Ganek, A.G. and Corbi, T.A. 2003. The dawning of the autonomic computing era. *IBM Systems Journal*. 42, 1 (2003), 5–18.
3. Gonzalez, A. et al. 2009. Unity criteria for Business Process Modelling. *Research Challenges in Information Science, 2009. RCIS 2009. Third International Conference on* (2009), 155–164.
4. Greenwood, D. and Rimassa, G. 2007. Autonomic Goal-Oriented Business Process Management. *Management*. (2007), 43.
5. Haupt, T. et al. 2011. Autonomic execution of computational workflows. *Computer Science and Information Systems (FedCSIS), 2011 Federated Conference on* (2011), 965–972.
6. Kephart, J.O. and Chess, D.M. 2003. *The vision of autonomic computing*. IEEE.
7. Lee, K. et al. 2007. Workflow adaptation as an autonomic computing problem. *Proceedings of the 2nd workshop on Workflows in support of large-scale science* (New York, NY, USA, 2007), 29–34.
8. Mosincat, A. and Binder, W. 2008. Transparent Runtime Adaptability for BPEL Processes. *Proceedings of the 6th International Conference on Service-Oriented Computing* (Berlin, Heidelberg, 2008), 241–255.
9. Oliveira, K. et al. 2012. Towards Autonomic Business Process Models. *International Conference on Software Engineering and Knowledge (SEKE 2012)* (San Francisco, California, USA, 2012).
10. Rahman, M. et al. 2011. A taxonomy and survey on autonomic management of applications in grid computing environments. *Concurr. Comput. : Pract. Exper.* 23, 16 (Nov. 2011), 1990–2019.
11. Reijers, H. 2008. Modularity in process models: Review and effects. *Business Process Management*. (2008), 20–35.
12. Rodrigues Nt, J.A. et al. 2007. Autonomic Business Processes Scalable Architecture. *Proceedings of the BPM 2007 International Workshops BPI BPD CBP ProHealth RefMod semantics4ws* (2007), 1–20.
13. Strohmaier, M. and Yu, E. 2006. *Towards autonomic workflow management systems*. ACM Press.
14. Terres, L.D. et al. 2010. Selection of Business Process for Autonomic Automation. *2010 14th IEEE International Enterprise Distributed Object Computing Conference*. (Oct. 2010), 237–246.
15. Tretola, G. and Zimeo, E. 2010. Autonomic internet-scale workflows. *Proceedings of the 3rd International Workshop on Monitoring, Adaptation and Beyond* (New York, NY, USA, 2010), 48–56.
16. Vedam, H. and Venkatasubramanian, V. 1997. A wavelet theory-based adaptive trend analysis system for process monitoring and diagnosis. *American Control Conference, 1997. Proceedings of the 1997* (Jun. 1997), 309 –313 vol.1.
17. Wang, Y. and Mylopoulos, J. 2009. Self-Repair through Reconfiguration: A Requirements Engineering Approach. *2009 IEEE/ACM International Conference on Automated Software Engineering*. (Nov. 2009), 257–268.
18. Yu, T. and Lin, K. 2005. Adaptive algorithms for finding replacement services in autonomic distributed business processes. *Proceedings Autonomous Decentralized Systems 2005 ISADS 2005* (2005), 427–434.