Goal-oriented Autonomic Process Modeling and Execution for Next Generation Networks

Monique Calisti and Dominic Greenwood

Whitestein Technologies AG
Pestalozzistrasse 45, 8044, Switzerland
http://www.whitestein.com

Abstract. The effective modeling, execution and maintenance of business and operations processes, such as those described by the eTOM process framework, is of utmost importance to Telecom organizations, especially those transitioning toward NGN infrastructure. Many Business Process Management Systems, BPMS, available today however are significantly restricted in their support for intuitive and expressive process models, run-time process agility and rapid process adaptation to cope with changing business and operational conditions. This paper discusses a means of mitigating these limitations with a highly expressive goal-oriented process management language, GO-BPMN, and innovative autonomic BPMS called LS/ABPM. Together, GO-BPMN and LS/ABPM offer an intuitive, business-driven path to creating directly executable goal-oriented process models whose structure encodes multiple degrees of freedom through the potential for late decision-making. Executing models can be structurally modified in real-time in response to autonomic feedback from underlying systems they are managing.

Key words: business process management, next generation networks, autonomic goal-driven process navigation, service provisioning, eTOM

1 Introduction

The implicit objective of Next Generation Networks, NGN, is to facilitate the provisioning of rich media services to the telecoms customer with short lead times and at low cost to both provider and consumer. Indeed, the widespread proliferation of low-cost bandwidth and equipment is now readily facilitating the entry of many new service providers the majority of which need to provision their services through established operator infrastructure. This naturally presents huge business potential, but only if operations can be handled effectively, i.e., safely and flexibly, at the business process level [1].

In this perspective, business and operations processes, such as those specified by the eTOM framework [2], describe how telecommunications operators and service providers should perform their daily business to, amongst other things, ensure business continuity, consistency and business-level interoperability. These processes are typically put into practice using Business Process Management Systems, BPMS, which realize and administer a set of often diverse business processes involving people, organizations and technologies.

The major problem is that most contemporary BPMS solutions rely on well-understood, but inherently static and visually anaemic approaches to process modeling and execution. These approaches are often significantly inadequate when addressing the intrinsic flexibility and responsiveness to change required by many business processes like for instance service activation, service change management, new product assembly, procurement and supplier integration.

In particular, many of these processes face a significant level of run-time uncertainty and require process flexibility to cope effectively with changing business requirements and conditions. The primary reason for this is that most process models must be defined at design-time, rather than determined in real-time, i.e., at run-time. This implies that they can become *bloated* through the necessity of coding-in all possible options at design time due to the inability to change dynamically once in execution, *convoluted* through variation in the complexity and unpredictability of process structure, and *brittle* through an inability to adapt to real-time changes in business or operational deployment-specific conditions.

In response we propose an innovative *agile business process navigation approach* employing goal-oriented autonomic process management [3]. The approach targets the known limitations of contemporary BPMS by producing directly executable goal-oriented process models whose structures encode multiple degrees of freedom for late decision-making. The goal-oriented formalism creates a clean separation between statements of desired system behaviour, and the potential means to achieve that behaviour, encoded as plans. This approach is directly related to policy-based management of telecoms systems in that policies can both constrain decision points leading to goal satisfaction and/or be enacted through a process task taken toward achieving a goal [4]. In the former respect, predefined policies are used to control process execution by acting as conventions expressing constraints on goal directed decision points in the process model. In the latter respect, tasks can be linked to policy engines inducing a particular policy to become active in accordance with process expectations. Although not within the scope of this paper, we are currently investigating the integration of our approach to goal-oriented BPM with autonomic policy based network management [5].

Thus autonomic goal orientation offers a transition from design-time defined processes to real-time determined processes. The ***Goal Orientation*** aspect offers a powerful, visually intuitive method of modeling and executing processes accessible to business managers and process analysts alike. Processes are described as goal hierarchies, with every leaf goal linked to one or more plans describing that part of the overall process to be executed in order to achieve the goal. Because plans can be selected at run-time, flexibility is built-in to the process structures allowing workflows to be altered safely in real-time without any need for halting or re-starting the overall process. Moreover, ***Autonomic BPM*** builds on goal orientation to offer process responsiveness to change by creating feedback loops not only between the process engineer and the process model, but also between the underlying systems (human or computational) affected by the process tasks. This allows executing goal-oriented process structures to be structurally adapted in real-time in response to autonomic feedback from the underlying systems they are managing.

The remainder of this paper is organized as follows. Section 2 describes the essential principles of our goal-oriented autonomic process modeling and execution approach. The Living Systems Autonomic Business Process Management Suite, LS/ABPM, which realizes this approach, is then presented and an overview of its core components is given in Section 3. By focusing on a typical NGN composite service provisioning use case, a concrete case model is then proposed in Section 4. This aims to show how GO-BPMN can be adopted in the NGN context as a powerful and intuitive notation for business processes modeling and execution to empower business decision makers and IT administrators at different levels. On the other hand, this hopes to stimulate discussion, as summarized in Section 5, about which specific business process modeling and execution aspects might need to be further developed/refined to properly address the specific needs of increasingly dynamic NGN.

2 Goal Oriented Autonomic Process Management

In day-to-day business operations, it is quite natural to set goals, decompose a goal into sub-goals, define or reuse plans, and routinely track and check the execution of chosen plans in order to detect problems as they occur (or even better before they do), and to take appropriate actions [6].

On the other hand, todays dominant process management approaches focus almost exclusively on procedures. The concept of what the procedure is meant to achieve, and why, typically remains implicit in the mind of the humans who designed it. Because of this, the increase in process management automation that occurred with the increasing availability of BPM systems has also shifted the focus away from goals and plans in favour of procedures.

The consequence is that processes have become more efficient in execution but more rigid in structure. To support efficiency without sacrificing agility we propose that goal-orientation be placed at center stage.

2.1 Goals and Plans

Using a goal-oriented approach separates the statement of desired system behaviour, from the possible ways to perform that behaviour. A desired result is described by achievement conditions to make true and as maintenance invariants whose violation must be avoided: achieve goals and maintain goals, respectively.

The possible ways to obtain a result are represented by plans. These are essentially process graphs decorated with the conditions under which they become applicable and the results they obtain when successful.

Plans consist of a structured aggregation of tasks connected with standard Business Process Modeling Notation (BPMN) flow logic [8]. The tasks are contained within a plan body, which has a context condition describing when that body can be executed.

We developed a specific extension to the widely used BPMN visual modeling language, called GO-BPMN, or Goal Oriented BPMN [3]. The language introduces new model elements for goals, plans, relationships between them, and context variables for process-wide state preservation.

In accordance with GO-BPMN, a **goal** is an objective function that becomes active whenever its specified preconditions are met. Once any plan associated with a goal complete, the goal is considered to be satisfied. GO-BPMN specifies the following two goal types:

– ***Achievement Goals*** represent a condition or a state of a process that is to be achieved. These goals can thus be used to represent explicit objectives, needs, desires, etc. achieved during process execution. An Achievement Goal can be characterized by one or more pre-conditions that must hold before the goal can complete, i.e., its sub-goals or plans are committed to. Moreover, such goals can have skip conditions where, if true, the goal is considered as achieved. Figure 1 illustrates the GO-BPMN model notation for an achievement goal using in its decoration graphical form.
– ***Maintenance Goals*** represent a process state that is to be maintained true. These goals can thus be used to represent explicit invariants in process execution. A Maintenance Goal is characterized by a maintain-condition that must hold during the life-time of the goal. If the maintain-condition is false, any sub-goals or plans are committed to. Figure 2 illustrates the GO-BPMN model notation for a maintenance goal.

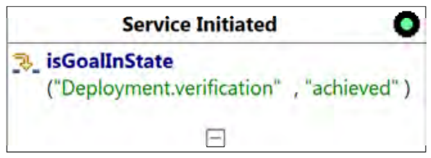


Fig. 1. Achieve Goal with precondition referring to a previous goal.

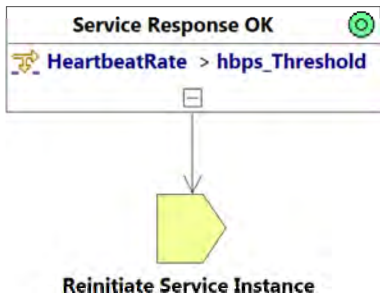


Fig. 2. Maintain Goal with plan to be executed if guard conditions are false.

Goals can be decomposed into *hierarchies* as shown in Figure 4. In this example the top-level goal can only complete once all three sub-goals have completed. Note that sub-goals may not need to complete successfully if the precondition on the top-level goal allows for this. All three sub-goals may execute concurrently.

A **plan** is that part of a process which specifies the functional tasks to be performed to achieve goals. Typically plans can be expressed in the form of a BPMN model and characterized by a guard-condition that must hold before the plan can be executed. Guard conditions can incorporate the expression of context associated with the systems affected by the plan structure resulting in context-aware plan selection and execution.

As illustrated in Figure 3, multiple plans can be available to satisfy any given leaf goal at run-time. Dynamic plan selection is performed by evaluating plan guard conditions against the current process context described by the collection of goals that are presently in an active state. Selected plans are immediately executed. Additional policies can be specified to select between plans when more than one has true guard conditions.

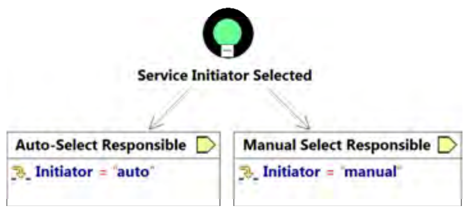


Fig. 3. Situation where two plans are available to a goal condition.

2.2 Autonomic Process Control

Once a process model has been created it can be directly executed as a process instance by associating it with an autonomous process controller that then becomes responsible for the instances entire execution. This consists of two major related functions:

1. Triggering the process goal hierarchy and controlling its run-time execution.
2. Initiating an autonomic feedback loop between the system being affected by the process instance, and the executing process instance model itself.

The latter of these two functions is responsible for initiating real-time adaptation of a process instance by using event triggers and, if desired, logic reasoning over system behaviour. This brings about both process flexibility and resilience. The system may include software, hardware, human and physical resources including the constraints and policies defining their use. Two of the possible effects of this adaptation are dynamic goal decomposition and dynamic optimization.

A process controller can self-optimise a process by assessing whether a goal hierarchy can be partially fragmented into sub-goals to parallelize execution or achieve partial results. This is particularly useful when a goal cannot be achieved due to non-satisfiable precondition and where segmenting the goal into sub-goals will allow at least some proportion of the goal condition to be met. Also, temporal pre-conditions can be adapted to alter the order of goal succession when possible and appropriate. Thus, the autonomic feedback loop is used to sense when goal hierarchies can be reformulated according to strategic beliefs held by the autonomous controller, and then acting to restructure the process as appropriate. An example of how this might restructure a process instance model is shown in Figure 4, where the ‘Service Deployment Verified goal has been fragmented into three sub-goals that are performed concurrently.

A process controller can also self-optimize a process by assessing whether an existing plan can, or should, be structurally altered (re-constituting the task steps), removed entirely or have its preconditions modified. Equally, a new plan may be introduced in real-time. New plans may be discovered from plan repositories or constructed on-the-fly. In this case the autonomic feedback loop is used to sense the basis for a decision whereby alterations in the process plan structures are made. An example of one case in which this might restructure a process instance model is shown in Figure 5, where the ’Auto-Select Responsible’ plan from Figure 3 has been replaced entirely with the alternative plan - same name but different precondition.

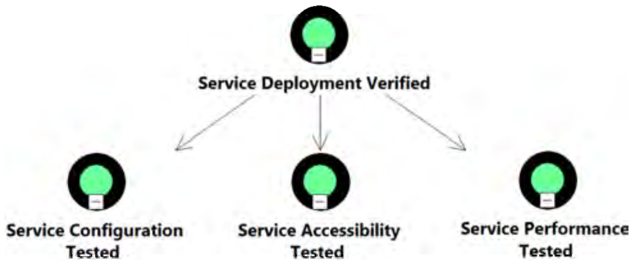

Service Deployment Verified
Service Configuration Tested
Service Accessibility Tested
Service Performance Tested

Fig. 4. Fragmentation of a goal into three sub-goals.

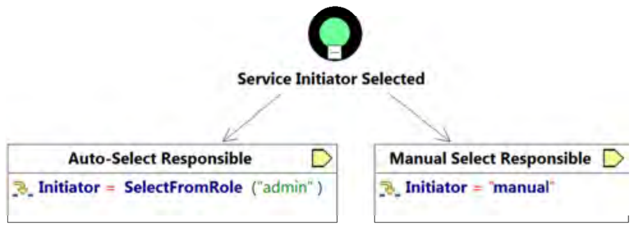

Service Initiator Selected
Auto-Select Responsible
Initiator = SelectFromRole ("admin")
Manual Select Responsible
Initiator = "manual"

Fig. 5. Replacement of a plan - with reference to Figure 3.

A further feature of this method of process control is that process models can be updated using a modeling tool and re-loaded into the process controller in real-time. The controller will manage dependencies and state preservation, ensuring that model execution remains uninterrupted as the adjustments are blended in non-disruptively.

Interactions between process instances are managed via communication between the process controllers responsible for those instances. This is local if the instance is managed by the same controller and remote if not. Interactions can be simple bindings between the goals and plans of different processes or more complex (potentially semantic) relationships coordinating the activities of more that one controller. When multiple process instances are interacting, the influence from autonomic feedback loops is carefully monitored and controlled to ensure all effects are traceably causal and without unexpected side-effects.

3 The LS/ABPM Suite

The GO-BPMN approach is at the core of the Living Systems Autonomic Business Process Management Suite, LS/ABPM. The business goals to be achieved are defined in the process model, providing business-oriented modeling. This goal-based process model is directly executed in the run-time environment, striving to achieve the goals by navigating the goal hierarchy. LS/ABPM is particularly well suited to businesses where processes are required to swiftly adapt to change and where the processes do not, or cannot, simply follow a strict, predefined sequence. LS/ABPM is built using the Living Systems Technology Suite [7], LS/TS, middleware and it includes the following main components:

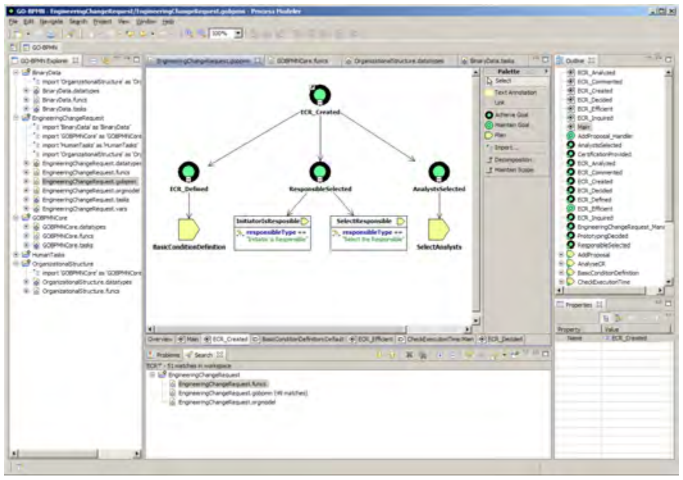


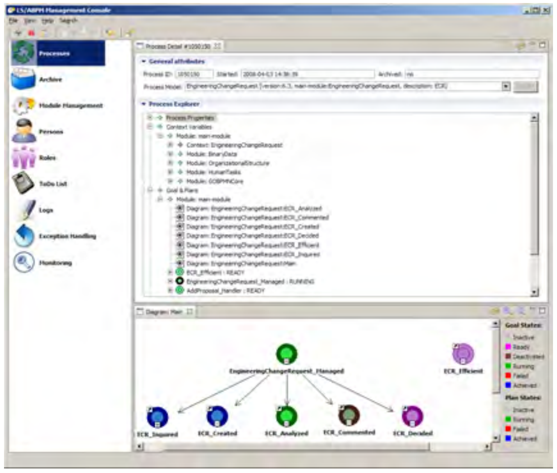
Fig. 6. The LS/ABPM Process Modeler supports GO-BPMN based design, test and validation of business processes.

– ***Process Modeler***. The core idea behind this component is to offer an intuitive, easy to-understand and use set of tools and methodologies that both

business and IT users can deploy to design, test and validate GO-BPMN business processes.
With the LS/ABPM Modeler, see Figure 6, the end user can define goals' and plans' hierarchies, application-specific business data, functions, tasks, context conditions, and implement plans with standard BPMN elements.
Moreover, this component allows the definition of organizational structures enabling the mapping of human activities to specific users, roles and organizational units.
In addition, process models are modular, allowing collaboration on large models, domain-specific modules, or libraries. In this way, different people can work on each reusable module and later consolidate their results into a whole model.

- ***Process Navigation Engine***. The core idea is that GO-BPMN process models created with the Process Modeler (see above) are directly executable by the Process Navigation Engine. This component is responsible for assembling at run-time the actual business process, by creating a path that takes into account model changes and plans alternative.
  According to given business goals/rules and other relevant context conditions, the Process Navigation Engine selects and orchestrate the appropriate plans in real-time. In particular, sanity conditions can be defined and the system ensures them through continuous monitoring and triggers corrective action as appropriate.
  The LS/ABPM Engine implements a data-type driven Web renderer used for human-centered activities. Such a render can be customized to implement application specific front-ends.
  Moreover, active coordination and cooperation between multiple process models can be achieved through message-driven synchronization between process controllers. This is an essential feature that enables to autonomically resolve competing goals and plans.
- ***Management Console***. This component provides powerful tools for the deployment, management and control of processes and other system administration tasks.
  First of all, continuous visibility of process execution and events is realized through detailed monitoring of running process instances, as displayed in Figure 7. At any point, the achieved, running, and waiting goals of a process can be inspected, as well as the corresponding pending tasks. This can be used by supervisors to fine-tune a process during execution.
  The LS/ABPM Management Console also supports crucial control tasks such as activation/de-activation of business goals and fine-tuning of context variable governing running process instances.
  Moreover, this component supports typical administrative activities such as data persistency, user management, role-based assignment of personnel, and security features.
- ***Application Frameworks & Process Component Libraries***. LS/ABPM has been built as a domain independent system. However, domain-specific BPM solutions can be built by making use of the LS/ABPM SDK.

This package enables the development of system specific sets of task libraries (either human or system executable), functions, and user interface front-ends. These latter ones can also be easily enhanced or substituted by other technologies (e.g., Web frameworks) according to the specific needs.



Process ID: 1050150 Started: 2008-04-13 14:38:39 Archived: no

Process Model: EngineeringChangeRequest [version 6.3, main-module EngineeringChangeRequest, description: ECR]

- Process Properties
- Context Variables
  - Module: main-module
    - Context: EngineeringChangeRequest
    - Module: BinaryData
    - Module: OrganizationalStructure
    - Module: HumanTasks
    - Module: GOBPMNCore
- Goal & Plans
  - Module: main-module
    - Diagram: EngineeringChangeRequest:ECR_Analyzed
    - Diagram: EngineeringChangeRequest:ECR_Commented
    - Diagram: EngineeringChangeRequest:ECR_Created
    - Diagram: EngineeringChangeRequest:ECR_Decided
    - Diagram: EngineeringChangeRequest:ECR_Efficient
    - Diagram: EngineeringChangeRequest:ECR_Inquired
    - Diagram: EngineeringChangeRequest:Main
    - ECR_Efficient : READY
    - EngineeringChangeRequest_Managed : RUNNING
    - AddProposal_Handler : READY

Fig. 7. The LS/ABPM Management Console showing a running process.

4 Case Model: Composite Audio-Video Feed Service

This case study describes the process for creating a notional NGN composite service which provisions an audio-video feed created in real-time by combining media streams from several service sources. This Composite Audio-Video Feed, CAVF, service process is considered to be a type of telecommunications Business Service Provisioning (BSP) which typifies the envisioned form of NGN services that are statically or dynamically assembled from multiple services potentially served by multiple providers. BSP is a critical process for telecommunications operators, service providers and systems vendors as it is central to the vision of NGN. BSP, in one form or another, is addressed by both the TMF eTOM and ITIL v3 specifications.

A concrete example of the CAVF service in use is when a service provider wishes to create an audio-video (AV) feed which dynamically splices feeds from several sources to create a interleaved AV stream. The primary feed will be, for example, a television show or movie, with advertising feeds spliced into the primary feed in accordance with the target audience.

The CAVF service process has been selected as an example case due to it lending itself particularly well to process expression in terms of business goals.

However, any other process defined by eTOM, ITIL, or other ad-hoc processes can be mapped into GO-BPMN and executed using the LS/ABPM navigation engine. In all cases investigated to date our approach offers clear advantages whether in terms of modeling clarity, model flexibility, sensitivity to business conditions, integration of multiple vendors, or a combination thereof.

The GO-BPMN model for provisioning the CAVF service is shown in Figure 8. The illustration shows only the goal-hierarchy, omitting all plans but three for the purposes of visual clarity. In complete models, each leaf goal must have at least one plan available for execution to be a valid model. The overall CAVF service provisioning goal, *Composite AV-Feed Service (CFS) Provisioned* is satisfied only when the five goals specified at the next, sub-layer are achieved (or de-activated, if allowed with the context of the model), and so on. The default condition is that all active goals may be achieved concurrently unless restricted by dependencies imposed by context variables specified in goal pre-conditions.

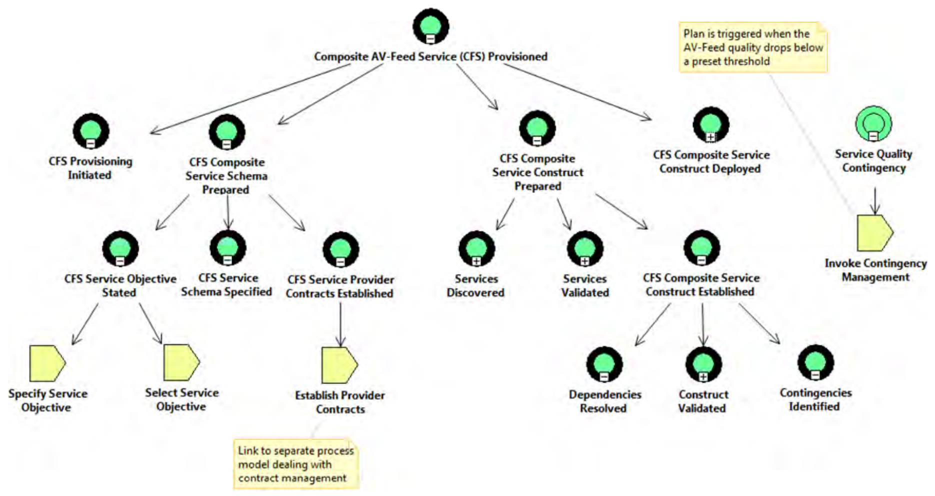

Composite AV-Feed Service (CFS) Provisioned
Plan is triggered when the AV-Feed quality drops below a preset threshold
CFS Provisioning Initiated
CFS Composite Service Schema Prepared
CFS Composite Service Construct Prepared
CFS Composite Service Construct Deployed
Service Quality Contingency
CFS Service Objective Stated
CFS Service Schema Specified
CFS Service Provider Contracts Established
Services Discovered
Services Validated
CFS Composite Service Construct Established
Invoke Contingency Management
Specify Service Objective
Select Service Objective
Establish Provider Contracts
Dependencies Resolved
Construct Validated
Contingencies Identified
Link to separate process model dealing with contract management

Fig. 8. GO-BPMN model for the provisioning of Composite Audio-Video Feed Service.

In this particular example the second sub-layer goals are:

CFS Provisioning Initiated: Achieved when process is setup.

CFS Composite Service Schema Prepared: Achieved when the schema detailing the objective and structure of the service construct is established, and contracts established with the various providers of specified services. Although not visible in this simplified iconic notation there are logical dependencies between the three sub-goals such that *CFS Service Provider Contracts Established* can only become active once *CFS Service Schema Specified* has been achieved, which in turn can only become active once *CFS Service Objective Identified* is achieved.

CFS Composite Service Construct Prepared: Achieved when required services have been discovered and validated, and the assembled service construct is established according to the three indicated third layer sub-goals which specify that service inter-dependencies must be resolved, the entire construct must be validated and contingencies identified for cases when, for example, a service fails or a service provider violates an agreed contract.

CFS Composite Service Construct Deployed: Achieved when the service construct is successfully deployed.

Service Failure Contingency: This is a Maintain Goal which persistently monitors the quality of the AV-Feed. If the quality, measured as a function of uptime and image quality, drops below a preset threshold a contingency management mechanism is invoked to correct the problem. The particular correction method used is not of concern, but may typically be to switch to alternative feed services. This is an example of autonomic feedback from the system under process management directly affecting process control flow.

The plans in Figure 8 are included as illustration of their use. The two plans *Specify Service Objective* and *Select Service Objective* both satisfy the *CFS Service Objective Identified* goal, with the plan pre-conditions (not shown) specifying that only one of these plans will be selected at run-time according to the value of a context condition set by a process user. In the case of these two plans the context criteria indicates whether the service objective should be specified ad-hoc or selected from a prescribed list; context conditions can consist of any relevant criteria such as cost, availability, performance, etc. Recall that context conditions can also be used to skip parts of the process, or to postpone the activation of a goal until after the achievement of others.

A third plan *Establish Provider Contracts* satisfies the *CFS Service Provider Contracts Established* achieve goal. The attached note indicates this plan contains a BPMN signal task which connects to a separate process which deals with contract management.

This model is intended as an example of how the CAVF service provisioning process may be modeled, at predominantly goal-level, using GO-BPMN. As with any other GO-BPMN model executed by the LS/ABPM process navigation engine, the goal-plan-task structure can be altered, re-loaded and executed at run-time, perhaps through the addition of a new plan or segmentation of a goal into two new sub-goals. In such a case, if an instance of the model is already executing its transaction-preserved state is transferred to the new instance; a set of violation rules ensure that model alterations cannot create inconsistencies with any critical execution aspects of the original instance.

5 Conclusions

The GO-BPMN approach offers a simple and intuitive method of modeling business processes by making use of concepts and artifacts that are easy-to-understand and to express directly by business managers and process analysts. Moreover, the intrinsic flexibility built-in to the process structures implies that workflows can be safely modified in real-time without any need for stopping or re-starting the overall process. Changes to any goal or plan in a GO-BPMN process model can indeed be made indipendently and do not have a ripple effect of consequences as they would have in a sequential process model. Hence, changes can be made at any time - even during execution.

While in this paper we focused on a specific case model, we argue that the combined adoption of the GO-BPMN approach and the LS/ABPM Suite has a much broader applicability and a great potential to enable the necessary BSS/OSS evolution towards the realization of the NGN vision. In particular we recognise that while valuable as standard guidelines, the prescribed process descriptions offered by frameworks such as eTOM and ITIL are frequently a poor reflection of in-use processes that rarely follow ideal cases. This strengthens the argument favouring goal orientation, which explicitly embeds sufficient process flexibility into model descriptions while ensuring that when executing they will always achieve required and expected goals.

Acknowledgments. Thanks go to the many colleagues at Whitestein Technologies involved with this work, especially Giovanni Rimassa and Roberto Ghizzioli.

References

1. Schwartz, S.: As Telco's Become Service-Centric, They Need to Think BPM. Billing and OSS World, (2005)
2. Enhanced Telecom Operations Map (eTOM). ITU-T M.3050 (2004)
3. Greenwood, D., Rimassa, G.: Autonomic Goal-Oriented Business Process Management. Proc. 3rd International Conference on Autonomic and Autonomous Systems, Greece (2007)
4. Raymer, D., Strassner, J., Lehtihet, E., and van der Meer, S.: End-to-End Model Driven Policy Based Network Management. Proc. 7th IEEE International Workshop on Policies for Distributed Systems and Networks, USA (2006)
5. Bahati, R.M., Bauer, M.A., Vieira, E.M., Baek, O.K. and Chang-Won Ahn: Using policies to drive autonomic management. World of Wireless, Mobile and Multimedia Networks (2006)
6. Tibco: Goal-driven business process management: Creating agile business processes for an unpredictable environment. Tibco Whitepaper (2006)
7. Rimassa, G., Greenwood, D., Kernland, M.: The Living Systems Technology Suite: An Autonomous Middleware for Autonomic Computing. International Conference on Autonomic and Autonomous Systems, ICAS, Silicon Valley, USA (2006)
8. OMG, Business Process Modeling Notation, `http://www.bpmn.org`
9. Information Technology Infrastructure Library v3 (ITIL), `http://www.itil-officialsite.com`