Emergency Services: Process, Rules and Events

Mauricio Salatino, Esteban Aliverti, and Demian Calcaprina

Plugtree
salaboy@gmail.com

Abstract. The Emergency Service Application was built as a blue print for architect applications using the Drools and jBPM5 platform. The main goal of this application is to show how we can model and implement a complete and complex business scenario using declarative approaches like: Business Rules, Business Process and Complex Event Processing. From the technical point of view, this application shows how components like a Rule Engine and Business Process Engine can be integrated with technologies that are widely used in most system architectures. Components - like Distributed Caches, Transactional Frameworks, Messaging Systems, Web Services Stacks and Web Frameworks - can be combined to work around the declarative power of rules, processes and events.

1 Application Scope, Domain and Requirements

The application was created to represent complex scenarios that are being executed by an Emergency Services company that deals with concurrent emergencies within a city. The company needs to solve different situations where different entities need to be coordinated to deal with an emergency situation.

The Emergency Services Application shows how we can provide a tool that helps the company to improve their services by giving them full visibility of their actions, traceability of their resources, suggestions and advice based on the context without sacrificing any degree of flexibility that they need to solve real life situations. One of the most important requirements of the application is related to the the fact that most of the emergencies can be classified under different categories based on their characteristics. We can model guidelines to solve each of these categories but in real life, each emergency will be treated differently, depending on the context, the state of the company and their resources. For each particular situation, different actions will be evaluated, suggested and executed.

Two generic situations will be described by this paper: Heart Attack Emergencies and Fire Related Emergencies. But the application is in no way limited to these two scenarios.

To design the application we gather, understand and formalize the knowledge that the company experts use to drive their activities. The formalization of this knowledge will be introduced in the following sections where the Business Processes, Business Rules and Events modeled for this application are described.

2 Declarative Knowledge Representation

The application uses the concept of procedure to define the set of business processes, business rules and domain specific services that will be used to deal with an emergency. We have defined a set of default procedures that describe the activities that can be executed during specific emergencies. The "Default Heart Attack Procedure" can be used to drive a standard set of activities that needs to be executed each time a person suffers a heart attack. The "Default Fire Emergency Procedure" describes the same but for emergencies that include fire situations where we need to coordinate the firefighters department in order to mitigate the dangerous situation as soon as possible.

Both procedures set up the basic activities that the company needs to execute in each specific situation, but in no way do they limit the company to add, remove or execute more activities in parallel.

Before executing these procedures, the company needs to identify the context of the emergency by executing a set of activities that were designed to pick up the initial information and find out what is happening. These activities are contained in a Generic Emergency Procedure that is executed each time the phone rings in the central offices.

3 Generic Emergency Procedure

This procedure will initiate the information about an emergency and is based on the initial information that is being gathered by the phone operators of the company. Using this information, a suggestion mechanism based on Business Rules will be in charge of suggesting the most appropriate, specific procedures out of all the available procedures.

This generic procedure is driven by the following business process:

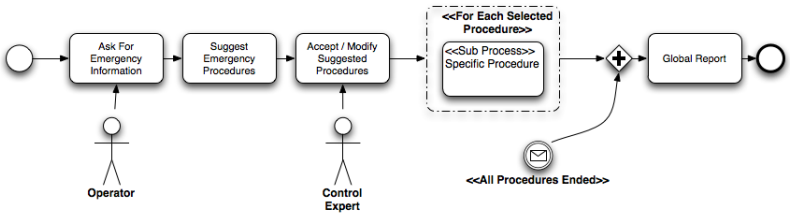

Fig. 1. Heart Attack Emergency Procedure

The Suggest Emergency Procedures activity inside this business process uses a set of rules to analyze the context of the emergency gathered to suggest a set of procedures that fits with that specific situation. These rules will evaluate the status of the company in order to suggest viable procedures.

Once the Suggestion rules are executed, the control expert in charge reviews the suggestions and modifies or approves the selected procedures. The business process automatically will start each selected procedure in parallel and it will wait until it receives a notification that all the specific procedures have ended.

The main purpose of this generic procedure is to speed up the activities that need to be executed for every emergency that the central offices handle. The suggestion rules help the experts by giving them a clear set of procedures that can be executed based on the current status of the company and also on the contextual and semantic information gathered for each specific situation. Once the procedures are started, a separate group of resources will be used to monitor all the activities executed for each emergency.

4 Default Heart Attack Procedure

As soon as we identify a heart attack situation, the system will automatically suggest to the expert which procedures best fit based on contextual information. If the Default Heart Attack Procedure is selected, the activities described by the following business process will be executed:

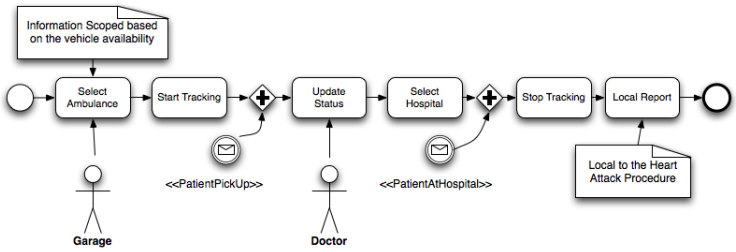

Fig. 2. Fire Emergency Procedure

This sequence of activities and events defines exactly how the company must deal with a Heart Attack situation. Briefly, an ambulance will be selected according to the company status and the patient information and it will be dispatched to the emergency location. Once that ambulance arrives, the Doctor will send an updated report to the central about the situation. This information will be correlated with the emergency location, the location of different hospitals based on distance, and the availability of the health-care services in each hospital to select the most appropriate facility. Once the patient is at the hospital, a report about this procedure will be created.

This business process is enriched by different sets of rules that are being executed in different activities to take automated actions to speed up the service and the human involvement times. For this particular use case, a set of rules is being defined to automatically select the best hospital based on the available

information. Taking advantage of the Complex Event Process features provided by the platform, another group of rules are defined to take care of more dynamic and reactive aspects that need to be covered.

The following rule is one of the set of rules created for the hospital selection mechanism:

Listing 1.1. Rule 2

```
rule "Select Closest Hospital"
  ruleflow-group "hospital-selection"
  when
    $pI: WorkflowProcessInstance( $pid : id )
    $emergency: Emergency( $type: type.name )
    $selectedHospital: Hospital() from accumulate (
       $hospital: Hospital() from externalEntities.getAllHospitals(),
         hospitalDistanceCalculator(
           new HospitalDistanceCalculationData(
             $hospital, $emergency )
           )
         )
  then
    String callId = ( (Call) $pI.getVariable("call") ).getId();
    //Send Hospital Selected Message
    MessageFactory.sendMessage(
      new HospitalSelectedMessage( callId, $selectedHospital ) );
end
```

This basic rule calculates the closest hospital to the emergency location iterating the location of all the available hospitals. The available hospitals are being dynamically calculated based on periodical updates reports received in the central.

The following business rule uses the temporal operators to analyze and react based on the patients vital signs. Once the patient is inside the ambulance, all his/her vital signs are sent to the central offices and monitored by a set of rules that are designed to analyze anomalous situations and generate warnings. These warnings are automatic reactions executed by the system when a specific pattern is found in multiple sources of real time events. These warnings can be used to influence the hospital selection, the route to reach the selected hospital, or even trigger new on-demand procedures.

Listing 1.2. Rule 3

```
rule "Patient heart attack pattern"
  when
    ArrayList( $num : size > 7 ) from collect (
      PulseEvent( processed == false, $pulse: value )
        over window:time(1s)
        from entry-point "patientHeartbeats" )
  then
    MessageFactory.sendMessage(
      new PatientMonitorAlertMessage(
        callId, vehicleId,
        "Warning, patient suffering a heart attack ",
        new java.util.Date() ) );
end
```

This simple rule evaluates in real time the events that are coming from a stream called “patientHeartBeats”. If we find more than 7 events per second filtering the values of those particular events, we can say that it is very likely

that the patient is having a heart attack. The application provides a configurable module that allows us to set up different devices to be used as input for these events. We have designed a set of bindings for the Wii Remote Control Accelerometer, the IPhone Accelerometer and Android Devices Accelerometers that can be plugged as event sources. Based on the values that are being sent by these devices, the rules will react if a pattern is found.

5 Default Fire Emergency Procedure

This procedure will be executed each time that the company needs to deal with a fire situation. Once again, a set of business processes and business rules will compose this procedure. For this procedure, we will analyze a business process that describes a more dynamic set of activities that needs to be executed.

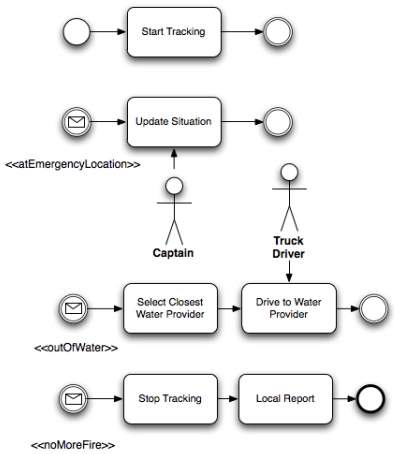

Fig. 3. High Level Architecture

This more unstructured process allows us to represent a situation where we send one or more fire trucks to a fire emergency. Each truck will have a limited amount of water that can be recharged in the fire departments. This process is being driven by the events that are being received in the central offices, which allow us to coordinate if we need more trucks; or if the situation is under control, we can reduce the number of trucks that we are using.

For this procedure a set of rules is defined to control the water tanks and select the closest water provider; a planning algorithm is also used to calculate the initial amount of trucks required to deal with the fire situation.

6 Inferences, Correlation, Aggregation and Dynamic Knowledge Composition

In order to provide the flexibility required to handling these complex situations we need to provide generic mechanisms to gather the knowledge required to deal with each emergency on demand. This application was designed to allow company experts to pick different business knowledge assets to solve specific situations. Composing different pieces knowledge into a single runtime will allow us to build very flexible and reactive services that can solve high and low level situations. These knowledge runtimes will be populated with the information that belongs to the specific situation. Once the runtime is created and populated with information, the rules, processes and events will be analyzed and the correspondent actions will be triggered.

Each knowledge runtime will be an isolated entity that can be distributed in different nodes of a computer grid allowing the application to scale. Each of these knowledge runtimes will provide a context smart enough to solve the specific situation that causes its creation.

7 Architectural Overview

This application was created to take advantage of different technologies to solve very specific problems. All these technologies are being used to demonstrate how we can solve all the technical problems that arise when we try to provide a solution that needs to drive a company. The following technical components are being used to solve infrastructural problems such as scalability and robustness, delegating the business logic and business definitions to the Business Rule Engine and Business Process Engine.

Current technical components that are being integrated to the application:

- Distributed Cache (Infinispan)
- NOSQL Graph Based Database (Neo4J)
- Query and Graph Transversal frameworks (Gremlin and Chyper)
- Reliable Messaging System (HornetQ)
- Web Frameworks for Presentation Layer (FreeMarker, Spring MVC)
- Interaction Component for dynamic form builder (Smart Tasks)

The architecture of the application was created with the concept of distribution in mind. Usually the problems that we want to solve using this approach are extremely complex and can involve huge amounts of data, therefore each knowledge runtime can be instantiated in different physical or virtual machines. Using different techniques, we can coordinate and monitor these knowledge runtimes so decoupling them in order to improve performance and scalability. Most of the interactions are being handled by messaging queues which allow us to configure the reliability of the channels completely decoupled from the problems that we are trying to solve.

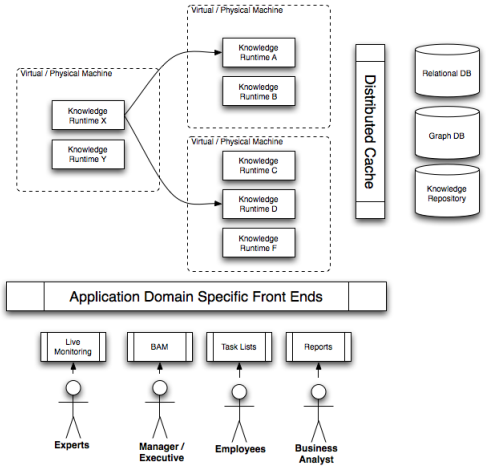

Fig. 4. Caption here

8 Conclusions

The application right now has as the main goal to show how can we mix the declarative power of rules, business processes and complex event processing to create application that can be understood by business people. The future of the application will be driven by the Drools and jBPM5 projects that are continuously evolving. Ontologies, smart and distributed agents, more powerful domain specific languages, predictive models and planning algorithms will be included as part of the design of the application architecture to test and demonstrate how all these features can be complemented to provide a more flexible platform to build applications.

Older versions of this application were presented in six international events during 2010 and 2011. For more information about the application, you can browse the main developers blogs [3, 1, 2] . All the application features are open to the community and we encourage people to participate from the project to learn about these technologies. The application source code is fully available to download [4] and is licensed under the Apache Software License 2.0.

References

1. Aliverti, E.: Blog. `http://ilesteban.wordpress.com/` (2011)
2. Calcaprina, D.: Blog. `http://dcalca.wordpress.com/` (2011)
3. Salatino, M.: Blog. `http://salaboy.wordpress.com/about/` (2011)
4. Salatino, M.: Emergency Services Application. `https://github.com/Salaboy/emergency-service-drools-app` (2011)