Designing internal control points in partially managed processes by using business vocabulary

Yurdaer N. Doganata,
IBM T. J. Watson Research Center
Hawthorne, NY 10532
yurdaer@us.ibm.com

Abstract— The challenges of automatically detecting compliance failures in partially managed processes are addressed and a solution is proposed. Detecting compliance failures by creating internal controls is a requirement of enterprise risk management framework. In case of unmanaged or partially managed processes, lack of full process visibility and the dependence on in depth knowledge of IT system and business application code are the main challenges of generating internal controls. Integration of business provenance management system with a business rule management system is proposed to overcome these challenges.

I. INTRODUCTION

Detecting compliance failures help organizations better control their operations and remain competitive. The quality of product and services can not be ensured in a business if the processes do not conform to design goals and comply with the rules and regulations. Moreover, organizations may be subject to serious financial penalty as well as civil and penal consequences if they failed to comply with established guidelines, rules and regulations. Hence, the impact of non-compliance may have devastating consequences [1], [2].

Compliance can be managed relatively easy when business processes are fully automated, well structured and documented [3]. In a fully automated structured business process real time information about the status of various activities can be collected by business activity monitoring software [4]. Hence, the trace of the business operations is completely visible when processes are managed by software applications. In modern enterprises, however, operations are not fully automated. Business activities span across systems and organizations integrating legacy and newly developed applications and activities often relay on human interactions in the absence of predefined control structures. Activities are not controlled by a single system or predefined rules, in other words, they are either partially managed or unmanaged.

Enterprise Risk Management framework such as COSO ERM requires creating internal control points to detect compliance failures for the purpose of preventing business risk from occurring [5],[6]. Internal controls can provide a reasonable assurance that the goal of an organization is met. Traditionally, auditors are used to check the status and the effectiveness of internal controls; however, this is a costly and time consuming approach. Automating the auditing tasks is possible provided that the internal controls are expressed in terms of business entities that are produced during the execution of a business process. This paper focuses on the challenges of creating internal control points to detect compliance failures automatically where the processes are either unmanaged or partially managed and proposes solutions for these challenges.

In a business environment where activities are not fully controlled, there are primarily two challenges in detecting compliance failures. The first challenge is to create internal controls when end-to-end operations are not fully visible and the business artifacts that are needed to define a control point are not always available. The second challenge is to enable creating internal controls without depending on in depth knowledge of IT system and business application code. Business people should be able to create new control points without requiring the application code to be modified every time there is a change. Hence, for a successful automation of detecting compliance failures, the visibility of operations needs to be increased while the gap between IT systems and business people is decreased. This requires tracking, capturing and correlating relevant aspects of the business operations and enabling internal control point creation by using the language of business people.

Traditionally internal control points are implemented by the IT organization based on the requirements prepared by business people [7] [8]. This is mainly because the internal controls are buried into the application code[9]. Implementing new internal controls by IT department every time there is a need is very costly and not flexible. In today’s enterprises, business people like to test different internal controls without requiring changes in the application code every time a new control is created.

In the next section, the problem of traceability in partially managed processes is tackled by employing business provenance technology [10]. The goal is to provide enough operational visibility to define internal controls from business process artifacts. In Section III, integrating the execution trace into a business rule management system [11] to business province management system is discussed in response to eliminating IT dependency in creating internal controls. The idea is to make the relevant business execution data available to a business rule management system to define internal controls by using business vocabulary.
II. INCREASING VISIBILITY OF UNMANAGED PROCESSES

The efficacy of internal controls depends on the visibility of the underlying process. Visibility of an unmanaged process is measured by the amount of relevant process artifacts that can be captured and distinguished. Hence, capturing and correlating the business events within the context of the business operations is an important part of increasing the visibility of a business process.

This section overviews the business provenance management system [10] that has been proposed and used to increase the visibility of unmanaged or partially managed processes in response to the first challenge. The business provenance subsystem monitors the various underlying systems across which the process executes, and is responsible for gathering relevant events (such as tasks being performed, data being accessed or modified and so on), and correlating these into coherent business process traces. Each relevant event produced by the IT system is stored in a provenance graph as a particular type of node or edge. Central to this process is the development of the provenance data model, based on the IT implementation of the process and the context of the business operations. During the development of the provenance data model, nodes and edge types that are expected to be produced at runtime are modelled. The data model is based on the known types of events that the IT systems produce. A relation between a resource record and a task record shows who was involved in executing that particular task. A relation between a data record and a task record reflects the effects of the task on the business artifact or the task dependency on its availability. A business goal and the associated internal controls often combines aspects by describing which tasks should be performed, when, how and who should be involved.

A. Capturing and storing business events

The trace of a business process is obtained by using recording clients which process application events and transform them into provenance events. The captured data must be relevant and specific to the business operation under consideration. Capturing application event data may require access to underlying system logs, email repositories, or user specified directories where documents are stored. The recorder client processes application events, transforms them into provenance events and records them in the provenance store. A provenance event contains a subset of application data that needs to be stored as business provenance. The captured data must be relevant and specific to business control points. To avoid redundancy and possible exposure of sensitive data, recorder clients do not copy all application data.

The captured data is then typed according to the proposed data model by using the specifications of the business and stored. Once the provenance data is stored, relations among the entities are established by running analytics. The data correlation and enrichment component links and enriches the collected data to produce the provenance graph. To do so, the analytics components have access to the content of the provenance store. The enriched business data is accessed through a query interface and analysed to verify business control points in two different styles. Firstly, a query can be deployed into the provenance store to emit results in real-time, feeding existing dashboard systems to display key performance indicators for example. Secondly, a query front-end enables visualization and navigation through the provenance graph from the outside.

B. Business Provenance Graph Data Model

The types of graph nodes should be general enough to support various meeting data without lack of generality. We propose to extend the following data types for meeting visualization which are proven sufficient to represent any business process:

Data Type: The business artifacts that were produced or exchanged during the business process such as documents, e-mails, database records, etc.

Task Type: A task record is the representation of a particular task activity that utilizes or manipulates data and is executed by the process resources.

Resource Type: A resource record represents a person, a runtime or a different kind of resource relevant to the selected scope of business provenance; an actor of a particular task is an example of such a record.

Custom Records: Custom records provide an extension point to capture domain specific, mostly virtual artifacts like compliance goals, alerts and checkpoints.

Nodes of the provenance graph represent these four classes of records. Edges between nodes are made based on correlation between two records, Relation Records represent the edges. These are the records generally produced as a result of relation analysis among the collected records. Some relations are rather basic on the IT level, like the read and write between tasks and data. Other relations are derived from the context.

C. Example: ‘New position open process’

Fig. 1 shows a sample business process to open a new position. The example is taken from Lombardi user guide [12]. In the example depicted by Fig. 1, the hiring manager submits a job requisition for a new position. If this is for a new job position, the requisition is routed to the general manager for approval. If this is for an existing position, the requisition is routed directly to human resources. The general manager evaluates the submitted requisition and either approves it or rejects it. This is represented by “Approve/reject requisition” task. If approved, the requisition is routed to human resources. Otherwise, it is terminated and the hiring manager is notified. Based on the data model described in the previous section, the records that are created for this process are as follows:

Data records: Job Requisition, GM’s approval, GM’s rejection, List of job candidates.
Task records: submit job requisition, approve/reject requisition, find job candidates, and notify hiring manager.

Resource records: hiring manager, general manager, human resources, system

Relation records: actor, generates, manager, next task, submitterOf, approvalOf

Figure 1. New position open process

Figure 2. A trace of “New Position Open Process”

Table I. Storing the Provenance Entities of an Execution Trace

<table>
<thead>
<tr>
<th>ID</th>
<th>CLASS</th>
<th>APPID</th>
<th>XML</th>
</tr>
</thead>
<tbody>
<tr>
<td>PE1</td>
<td>Resource</td>
<td>App01</td>
<td>&lt;ps:person ps:id=&quot;&quot; ps:class=&quot;resource&quot; &gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><a href="">ps:appId</a>app01&lt;/ps:appId&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><a href="">ps:name</a>Joe Doe&lt;/ps:name&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&lt;ps:timestamp value=&quot;&quot;/&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&lt;email&gt;<a href="mailto:jdoe@acme.com">jdoe@acme.com</a>&lt;/email&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&lt;manager&gt;Jane Smith&lt;/manager&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&lt;role&gt;Sales Manager&lt;/role&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&lt;/ps:person&gt;</td>
</tr>
<tr>
<td>PE2</td>
<td>Task</td>
<td>App01</td>
<td>&lt;ps:submission ps:id=&quot;&quot; ps:class=&quot;resource&quot; &gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><a href="">ps:appId</a>app01&lt;/ps:appId&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><a href="">ps:start</a>&quot;...&quot;&lt;/ps:start&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><a href="">ps:end</a>&quot;...&quot;&lt;/ps:end&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&lt;email&gt;<a href="mailto:jdoe@acme.com">jdoe@acme.com</a>&lt;/email&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&lt;manager&gt;Jane Smith&lt;/manager&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&lt;/ps:submission&gt;</td>
</tr>
<tr>
<td>PE3</td>
<td>Data</td>
<td>App01</td>
<td>&lt;ps:jobRequisition ps:id=&quot;&quot; ps:class=&quot;data&quot; &gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><a href="">ps:appId</a>app01&lt;/ps:appId&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&lt;ps:reqId=req001&lt;/ps:reqId&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&lt;ps:timestamp value=&quot;&quot;/&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><a href="">ps:type</a>new&lt;/ps:type&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&lt;dept&gt;dept501&lt;/dept&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><a href="">ps:position</a>Sales&lt;/ps:position&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&lt;/ps:jobRequisition&gt;</td>
</tr>
</tbody>
</table>

Business control points associated with the business rules can be created using this description in terms of provenance graph entries. A business control point is satisfied if certain vertices and edges exist in the provenance graph. Hence, it is possible to claim that a business control point is a sub graph of the provenance graph.

While the relevant details of the process trace is stored in Error! Reference source not found., as exemplified above directly from recorder clients, creating internal controls is not
a trivial task. Extensive IT skills are required to manage the data stored in a database. Another problem is that the Error! Reference source not found. does not contain the semantic information related to the business process. The table provides for a structured representation of business process traces without using the business vocabulary. A semantic mapping between the IT level details stored in Error! Reference source not found. and the business language is needed to be able to provide a business level view of the trace.

Often implementation of internal control points depends on IT departments in creating, testing and deployment of internal controls by business people. There exist tools that help creating internal controls as an expression of business policy in a form that is understandable by business users. The next section describes how one of these tools, namely ILOG JRules BMRS [11], can be utilized to create internal controls in the language of business people. Other tools can also be used without losing the generality of the concept.

D. Semantic Mapping: Creating Business Vocabulary

Business users need to create internal controls using familiar business terms. ILOG JRules BMRS [11] is one of the tools that provide for a rule development environment to create business vocabulary which is called ‘verbalization’. The data model used to create provenance graph as explained above is the model against which the internal controls are run. The first step of the solution is to generate an executable java object model (XOM) from the provenance data model. This way the nodes and the edges of the graph and their attributes are directly linked to XOM java objects through getters and setters methods. In order to enable editing internal controls by using business vocabulary, the next step is to map XOM to business vocabulary by using so-called Business Object Model (BOM). With this vocabulary, internal control points can be described by using the language of business people.

A BOM in a rule management system contains the classes and methods that the artifacts of internal controls act on. As an object model, the BOM is very similar to a Java object model. It consists of classes grouped into packages. Each class has a set of attributes, methods, and possibly other nested classes.

BOM-to-XOM mapping defines the correspondence between the BOM and the XOM used at runtime. As an example, consider the data node PE3, jobRequisition, as shown in Fig. 2. The java class that represents this data object is given below. The associated XOM data object for jobRequisition is constructed by the attributes of PE3.

```java
package myCompany;
public class jobRequisition
{
    public String class = "data";
    // Construct for jobRequisition
    public jobRequisition (String managerGen, String applId,
    String reqId, Boolean type, String position) {
        this.managerGen = managerGen;
        this.appId = applId;
        this.reqId = reqId;
        this.timestamp = timestamp;
        this.type = type;
        this.position = position;
        this.dept = dept;
    }
}
```

In the execution model (XOM) jobRequisition is a data class with attributes or data members. When the BOM is created from the execution model, class attributes are verbalized as navigation phrases and the methods are verbalized as action phrases. Some of the sample navigation phrases generated for jobRequisition data class in are listed below. The association between java class members and the navigation/action phrases constitute the BOM. The content of BOM includes statements like:

```java
myCompany.jobRequisition#concept.label = Job Requisition
myCompany.jobRequisition.managerGen#phrase.navigation = 
    {general manager} of {this}
myCompany.jobRequisition.appId#phrase.navigation = 
    {application ID} of {this}
myCompany.jobRequisition.reqId#phrase.navigation = 
    {requisition ID} of {this}
myCompany.jobRequisition.position#phrase.navigation = 
    {offered position} of {this}
myCompany.jobRequisition.type#phrase.navigation = 
    {position type} of {this}
```

Optionally, managerGen can be accessed from a hashtable by using a get method such as getmanagerGen where dept and managerGen are the <key, value> pairs:

```java
// Method to get the name of the hiring manager’s name
public String getmanagerGen(String dept) {
    String managerGen = (String) hashMap.get(dept);
    return managerGen;
}
```

Hence depending on the underlying data model used in capturing and correlating the process artifacts, either navigation or action phrases are used for verbalization. In this case an action phrase is generated for getting the name of the general manager as

```java
myCompany.jobRequisition.getmanagerGen#phrase.action =
    {general manager} of {this}
```

Hence, when the phrase "general manager" appears in an expression, BOM indicates that ‘general manager” is obtained by using the getmanager method of the jobRequisition class.

Fig. 3 depicts the verbalization of jobRequisition with attribute managerGen as the “job requisition” and “the general manager of the job requisition”. Similarly, the relations are also verbalized by using the same concept. There is a java class representation of each relation type. A relation
is an edge in the graph where the source and the target nodes are the members of the java class that represents the edge.

In short, the vocabulary is the set of terms and phrases attached to the elements of the BOM. Readers are referred to the user manual of ILOG JRules tool [11] for the details of verbalization of executable objects such as graph nodes. Vocabulary elements associated with the graph nodes are then used by internal control editor.

The internal control authoring tool (ILOG JRules) provides for editing capability in natural language. The business vocabulary generated in BOM is provided by using drop down menus in the rule editing tool. Hence, the drop down menus contain the associated vocabulary for every graph node and its attributes.

All internal controls have the same structure: a list of conditions to meet before performing a list of actions. It has four parts; definitions, if, then and else. The variables for the internal control point descriptions are defined in the definitions section. The conditions for the internal controls are listed in the if part, and the actions to be performed are listed in the then and else parts. Internal controls can be created by using Business Action Language (BAL) [13] and the vocabulary created for the provenance graph items. BAL consists of predefined constructs to build business rules and the operators that can be used in rule statements to perform arithmetic operations, associate or negate conditions, and compare expressions.

In the example below, the control point of the ‘New Position Open Process’ is expressed by using the definition and if-then-else structure. In the definition section, ‘Current job opening request’ is defined as the Job requisition with a string variable which is used to define the ID of the job requisition as follows:

\[
\text{set “the current request” to a “job requisition” where “the id of Job Requisition” is <string ID>}
\]

This definition links “the current request” to a data graph node labelled as “job requisition” with a specific “ID”. In addition, ‘hiring manager of the request’ is defined as the submitter of the ‘current request’; ‘general manager of the request’ is defined as the manager of the ‘hiring manager of the request’ as follows:

\[
\text{set “hiring manager of the request” to a “person” where “the relation of this person” to “the current job request” is “the submitter of job requisition”;}
\]

This definition links the “hiring manager of the request” to a graph node labelled as “person” which is connected to “the current request” data node as defined above with an edge labelled “the submitter of job requisition”. Finally, the “general manager of the request” is defined as the manager of the “hiring manager of the request” and linked to the manager attribute of the resource node “hiring manager of the request” as:

\[
\text{set “general manager of the request” to “hiring manager of the request”;}
\]
In the second part of the structure, the conditions are listed. The first condition is to check if the job requisition is for a new job opening. Otherwise, approval is not needed. The second condition is to ensure that the approval of general manager exists.

\textbf{If} all the following conditions are true:
- The "type" of "current job opening" is "new"
- "Approval" from the "general manager of the request" is not null

\textbf{Then}
Internal control is satisfied

\textbf{Else}
Internal control in not satisfied

This way the internal control expression is formed by using business vocabulary and with proper verbalization of the provenance data objects. Linking this expression to a sub-graph of the provenance graph that represents the control point is possible since the vocabulary used in the expression is directly linked to the graph nodes and edges. The first constraint of the expression is transformed into a graph query which retrieves a job requisition data node with type ‘new’ for a given job requisition id. The second constraint is transformed into a query that retrieves a data of type ‘Approval Status’ with a boolean status value ‘Approved’ for the same requisition id. Finally, last constraint corresponds to a query that returns the list of candidates for the same requisition id. The internal control point is generated as a custom node connected to the three data nodes defined by the constraints. The internal control is satisfied if all the specified edges exist.

IV. CONCLUSION

Integration of a business rule management system to the business provenance management helps closing the gap between the IT and business people. Proper verbalization of the provenance data model is crucial in making the connection between business vocabulary and the data types used to capture the trace. This article shows how to create internal control points by using a rule management system and linking these internal control points to the process execution trace.

The bridge between IT and the business vocabulary is built by connecting provenance data model to execution object model first, then to business object model, and finally to rule editing in business vocabulary. Once the proper verbalization is completed, the business people can create internal controls without a need to know data model and association to underlying provenance graph. Different verbalization for different business vocabulary is possible. This work suggests that the task of verbalization is a role that is executed after the provenance graph data is created.

This study shows that verbalization can be done over the execution trace without changing the application code. Hence, the approach is applicable to partially managed or unmanaged processes where activities are not fully controlled or monitored.

Future research includes improving verbalization process by adding business semantic into the provenance data model and automatic deployment of control points to provenance graph and continuous compliance checking.

REFERENCES