Adaptive Business Rules Framework for Workflow Management

Kanana Ezekiel, Dr. Vassil Vassilev,
Prof. Karim Ouazzane and Yogesh Patel
School of Computing & Digital Media, London Metropolitan University, UK

Abstract

Purpose – Changing scattered and dynamic business rules in Business Workflow Systems has become a growing problem that hinders the use and configuration of workflow-based applications. There is a gap in the existing research studies which currently focus on solutions that are application specific, without accounting for the universal logical dependencies between the business rules and, as a result, do not support adaptation of the business rules in real time. **Design/methodology/approach –** To tackle the above problems, this paper adopts a bottom-up approach, which puts forward a component model of the business process workflows and business rules based on purely logical specification which allows incremental development of the workflows and indexing of the rules which govern them during the initial acquisition and real-time execution. **Results** – The paper introduces a component-based event-driven model for development of business workflows which is purely logic-based and can be easily implemented using an object-oriented technology together with a formal model for accounting the business rules dependencies together with a new method for incremental indexing of the business rules controlling the workflows. It proposes a two-level inference mechanism as a vehicle for controlling the business process execution and adaptation of the business rules at real time based on propagating the dependencies between the rules. **Originality/value –**The major achievement of this research is the universal, strictly logic-based event-driven framework for business process modelling and control which allows automatic adaptation of the business rules governing the business workflows based on accounting for their structural dependencies. An additional advantage of the framework is its support for object-oriented technology which can be implemented with enterprise-level quality and efficiency. Although developed primarily for application in construction industry the framework is entirely domain-independent and can be used in other industries, too.

Keywords Business Process Modelling, Process Workflows, Business Rules, Dependencies, Rules Adaptation, Dependency Tree

Paper type Research paper

Introduction

Almost all workflow applications are based on some sort of rule-based systems. Business rules control the behaviour of the business processes according to the domain logic and the best practices in the domain (Rowe, Stephens & Guo, 2004). For example, in a data centre workflow application, a rule may exist to ensure rack utilization is less than rack capacity before the equipment installation process is executed. The biggest strength behind the use of business rules come from having multiple and changing business rules interacting with each other. However, as more and more rules are added and rule inter-relations are established business rules require extensive work in order to maintain their consistency. In business environment, an essential element for success is the degree to which the rules can quickly change and propagate these changes in real time. This paper presents a novel approach to automate workflow processes using a framework of business rules, meta-rules and business rules relationships. The approach introduces a component-based and event-driven model for development of the business workflows which is based purely on logic and is implemented using an object-oriented technology. A formal model for accounting business rules and dependencies together with a new method for incremental indexing of business rules controlling workflows is described. The paper is structured as follows: First, a review of related literature and applications is examined. Second, a framework for business rules model specifically introducing our approach is presented. Third, an ontology of two-levels rule-based approach introducing building blocks of business rules model is provided. Fourth, we describe formal definitions and model concepts transformation into dependency trees for business rules relationships. Fifth, presents a preliminary implementation work using **DROOLS** and future work. Finally, some implications of the new approach have been identified and discussed briefly before the final conclusion.

Related Literature and Applications Review

The trend in research studies of rule-governed business workflows is focused primarily on theories and practices of custom-tailored workflows and much less on exploring business rules dependencies and the necessity of adapting the rules to the changing conditions. There are several popular Business Rules Management Systems (BRMS) on the market today, but it is still very difficult to configure and automate workflow applications as the study by Cognizant revealed (Cognizant, 2015). To name the few, BRMSs have been explore by various authors such as Al Hilwa & Hendrick (2012), Macdonald (2010), Sainte Marie (2011), Haley (2013), Feldman (2011), Boyer and Mili (2011), Browne (2009) and others. In the typical case, the BRMSs use a rule engine for business rules management, providing APIs for modelling business rules and algorithmic inference. However, there are notable limitations of the possibility to manage the changes, which require updating the formulation of business rules. Although BRMSs in most cases allow for rules to be specified separately from the business processes, which supports two-step procedure of business process modelling and business rules specification, it remain impossible to specify the dependencies between the rules based on the relationships between workflow objects. This causes multiple changes to be necessary to adjust already configured workflows and to update existing business rules even in the case of simple change. The main reason for this situation is the lack of consistent model of the components of the business rules themselves. Typically, rules are composed out of events, conditions and actions, which are specified separately and are not related through the objects used to formulate them. This means that change made on the "condition" part of a rule will require invoking the whole rule rather than only the condition component. Externalizing different part of the rules would bring flexibility and increase the performance as only that part which needs changing would be processed explicitly, while the adjustment can be automated.

Some research studies suggest more flexible approaches towards workflow systems automation. Casati et al. (2000), for example, consider a workflow design based on rule-based approach to handle exceptions based on a separate description of workflow activities. Their approach provides a higher degree of flexibility during workflow design since it allows to model exceptional situations. Still, it remains difficult to describe and account the dependencies between the rules. It becomes even more complicated to deal with multiple changing rules as the rule management remains a tedious manual task. In fact, this is one of the main reasons why rule-based approaches have not been popular choice for managing workflows. Goh et al. (2001) investigate the use of Event Condition Action rules (ECA) to support workflows in product development. In their approach workflow activities are associated with ECA rules to govern how the activities which are executed. But the emphasis of their work is on high-level integration platform for building flexible workflows, rather than business rules, process structures and dependencies. Other authors in the literature such as Boyer & Mali (2011) and Anantaram (2007) suggested modelling of business rules as components themselves, separate from the business objects and the application logic. While the business user is free to define and modify the rules, the rules and their components are not defined in the same ontology. This approach does not allow rule classifications and the rule dependencies cannot be defined. Also, this separation hinders usability and adaptability of business rules. Geppert (1998) describes the implementation of an event-driven engine for distributed workflows, called EVE. To control the distributed workflows, they maintain an explicit list of events. Their approach addresses the problem of distributed events in

workflow execution by focusing on reactive event-based coordination and integration but because the inter-relations are not defined explicitly they still don't offer much flexibility in controlling the workflows in real time.

Framework for the Business Rules Model

To address issues discussed previously, we adopt an approach which relies on object-oriented modelling paradigm. The object orientation allows to define objects, classes and relationships between objects in a bottom-up manner, suitable for representing business rules complexities in a more structured and controllable manner. The formal model presented in this paper is based on the understanding of existing business workflow as an event-driven and constantly evolving process of incremental development, execution and control. This model operates on two levels, namely Workflow, or Process level (Figure 1) and Rule, or Control level (Figure 2). The business rules are building blocks that control workflows and they are made up of event, condition and action components, or the famous "When <event> If <condition> Then <action>" structure, whereas the workflows are made up of business processes (directed structures), process steps (primitive procedures), process flows (material and information links between processes), roles, etc. For instance, if some events are observed during execution of a working process then the corresponding business rules which depend on these events are invoked and lead to actions which in turn perform the transition to a new step which may execute other processes or amend the parameters of the current process. The model uses business rules to glue together processes from start to finish in a workflow (Figure 1).

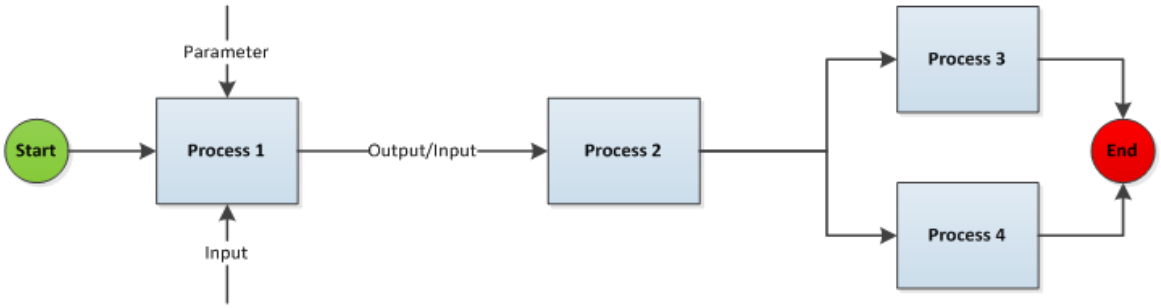


Figure 1: Workflow Level

The rule level (see Figure 2) provides a level of abstract "independence" from the process level, suggesting that the rules can be changed without affecting the part of the current workflow which has already been completed. The rule level automates complex business processes to perform business logic without writing a new code.

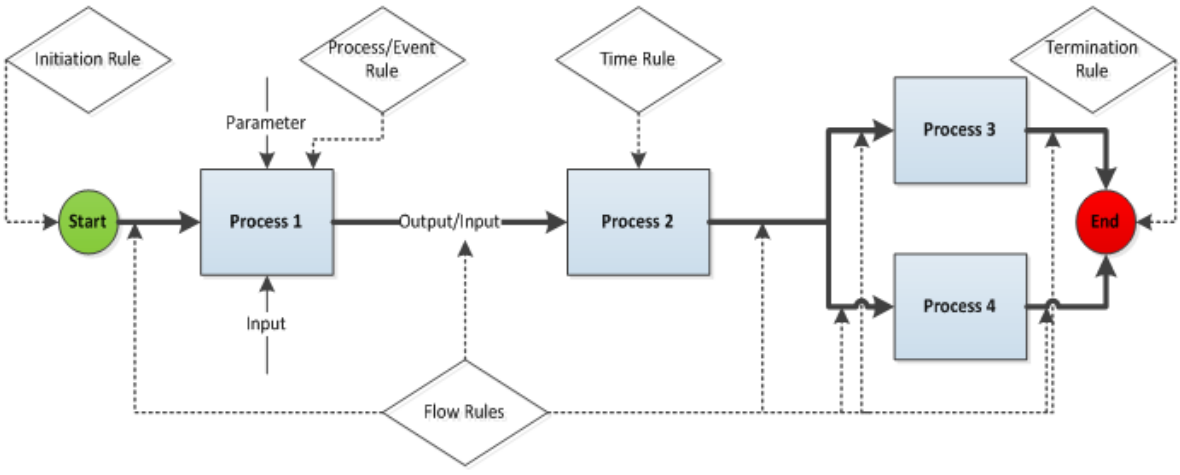


Figure 2: Rule Level

The business rules apply at various stages of execution of the workflow - **Initiation**, **Execution** and **Termination** rules (Figure 3). Based on the different role they play in relation to the workflow they can be organised in a kind of taxonomic hierarchy: Execution rules are divided into **Flow** and **Process** rules, Flow rules are divided into **Sequence**, **Fork** and **Join** rules and Process rules are classified into **Time based** and **Non-Time-based** rules.

Additional rules known as **Data** rules (not covered in this paper) may be considered when some conditions are applicable directly to the input and output data in order to maintain the integrity of the flow.

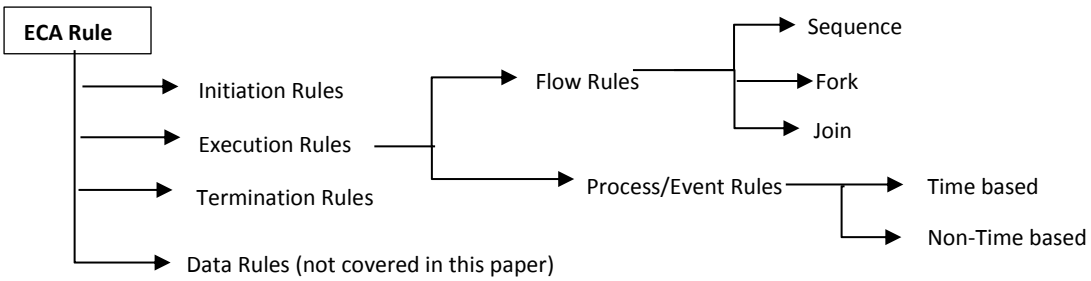


Figure 3: Rules Classification

An ontology of the rule-based framework

This section presents the basic ontology of objects used to construct the workflows and the rules which govern them. It has been developed in a purely logical manner. All examples have been illustrated using DFD diagrams.

I. Objects

The objects are the building blocks for describing business processes, rules and workflows.

Example 1

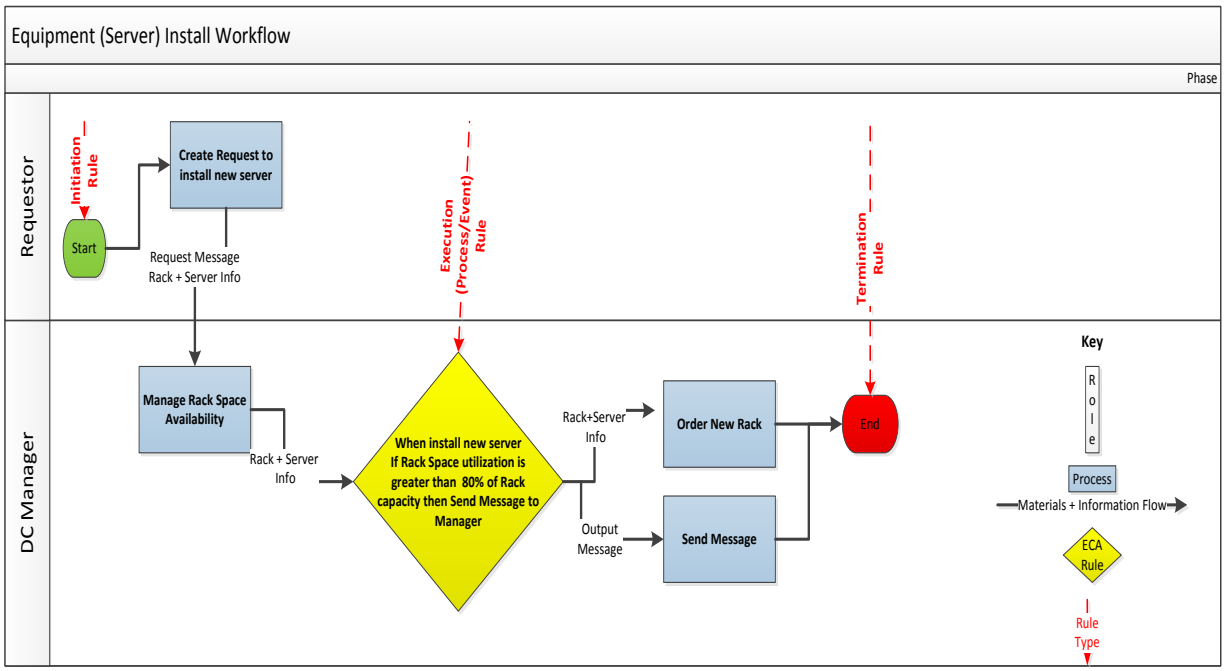


Figure 4: Business Workflow with Roles and Associated Rules

Consider the workflow in Figure 4. It defines Business Rule "When install new equipment (Server), if Rack Space Utilization is greater than the 80% of Rack Capacity then send message". Analysing the above example, the following concepts can be identified:

- 4 Processes: (Create Request to install new Server, Manage Rack Space Availability, Send Message and Order New Rack). Identify different work units that need to be accomplished.
- 2 Roles: Requestor, DC Manager which has not been covered in this paper.
- 1 Flow: Capturing data/material and information in and out the processes. Rack Capacity, Rack Utilization, New Equipment and even the Request are examples of Information and Material flows.
- Initiation Rule:
  - Start event - Workflow can be manually or automatically started by the use of initiation or triggering events. The business processes can be started only by Initiation rules after a suitable triggering event.
- Execution Rule:
  - Event - triggers or kick starts the rule: "When Install new equipment"
  - Condition - criteria for the rule to execute: "If rack utilization is greater than 80% of rack capacity"
  - Action - can be performed within the workflow or externally by the users of the workflow. In the example above, action response after the condition is satisfied is "send message".
- Termination Rule:
  - End event - Workflow can be manually or automatically ended by the use of termination event trigger. The workflow termination is always based on termination rule, invoked by suitable termination event AFTER the process is finished, or on process execution rule DURING execution in the case of emergency

In Example 1 above, the Execution Rule is used to check rack space availability. The decision to install new server onto a rack depends on the rule. Through the event "When Install new equipment", the rule links two processes "Manage Rack Space Availability" and "Order New Rack". The event "When Install new equipment" is observed in relation to process "Create Request to install new server" then the rule which depends on this event is invoked and lead to an action which performs the transition to "Order New Rack" process.

Following the terminology of the object models of Grady Booch (1994) and Umeshwar Dayal, et al. (2005), we refer to Process, Flow (Material, Information) and Rule (Event, Condition and Action) as first class objects.

II. Object Properties

Informally speaking, the business rules and workflows can be constructed in terms of object characteristics. The object properties provide information about the characteristics of the objects. For example, the object "Process-21" may have as properties process id, name, status, creation date, etc. From the viewpoint of the conceptualization of our ontology, object properties can be classified onto one of the following types:

- **Identification properties** - example are process id, name, type, context, scope, etc.
- **Qualitative description properties** - these are categorical or nominal properties, which can be described only qualitatively – for example, current status, deviation, trend, etc.
- **Quantitative description properties** – these properties can be described using a fixed value, which can be estimated qualitatively or specified quantitatively- for example, the number of closed processes in a chemical plant.

Sun, Bo and Fox (2014) describe object properties as a common approach to specify characteristics or attributes of a real-world object instance, which in turn helps to understand how to interact with the object. An object property value may be of different primitive type, including numeric, nonnumeric (strings/text/etc.), Boolean, etc. Property may have a single or multiple values. By introducing property characterisation for each object, our model can fulfil the requirements for flexibility and maintainability of the formulation of Business Rules and the versatility of the Process Workflow.

Since the objects are building blocks of both the process workflows and the business rules which govern them, the object properties are the main vehicle for analysing the dependencies between the business rules themselves. They will be the bridge between the process ontology and the algorithm for propagating the changes in the business rules. The primary role of qualitative and quantitative property measures is to accurately describe object properties rather than the usual identification and classification. The more sophisticated properties, the more elaborated dependencies we can formulate. Some object properties may be used to identify, name and categorize the objects. Others may be used to quantify and qualify the objects. There are circumstances where qualitative and quantitative properties are also used for identification of an object. We can even introduce properties for "potentially active" characterisation of the objects, like reflexive regularities, directed constraints and associative interdependencies between the properties of several objects. For instance, Business Rules may involve an array of object properties with objective estimation based on value measurement along with highly subjective value

judgments based on qualitative estimations. Finally, using the object properties we can organise them into groups and hierarchies which enables the use of object-oriented technology.

III. Rules

The structure and the classification of business rules (Figure 3] is based on the famous Event-Condition-Action paradigm, Bry, et al (2006). Our framework considers the following types of rules:

Initiation Rules (IR)

We propose Initiation Rule (IR) to formally depict rules that specifically initiate a process. Depending on the conditions of the rule, the process can be launched and thus continue the workflow execution. Some Initiation rules are driven by events only, hence known as Start Event. As an example, Figure 5 presents Equipment Installation workflow of an organization with three processes "**Create Request**", "**Send Message**" and "**Order New Rack**". In the background, the initiation rule "**When receive request start message then start**" looks up and assigns "**Create Request**" process whenever the rule is invoked. The rule is invoked when request message is received**.**

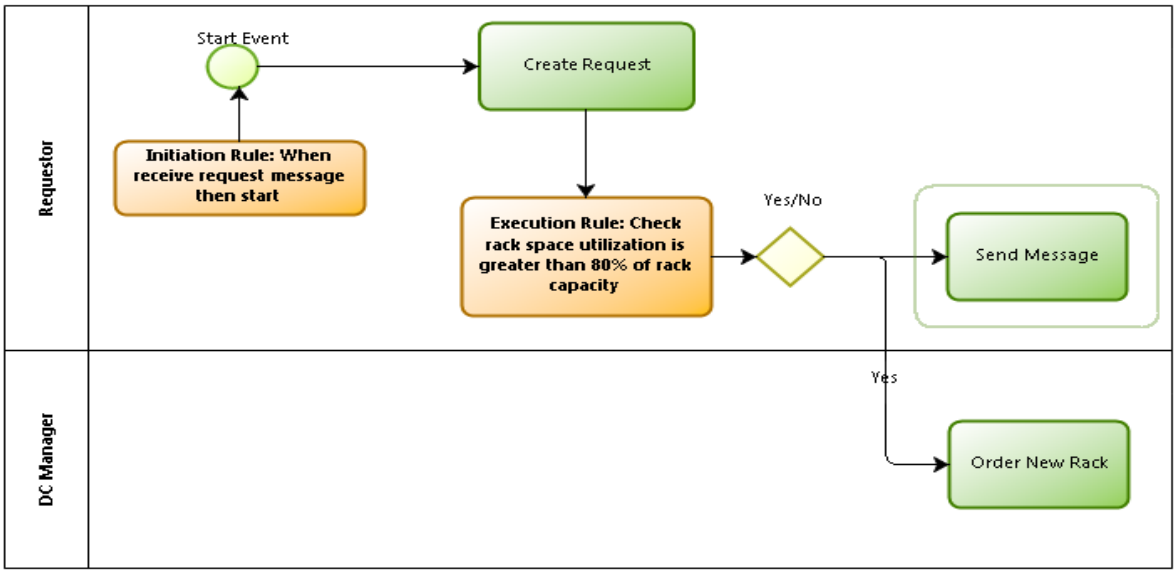


Figure 5: Initiation Rule – "When receive request start message then start"

Event or Process Rules

We propose the class Event Rule to group rules that are specifically defined on Processes during the execution of a workflow. An example of such an event rule is the rule which requires the drivers to stop when road traffic light colour changes to red (Figure 6).

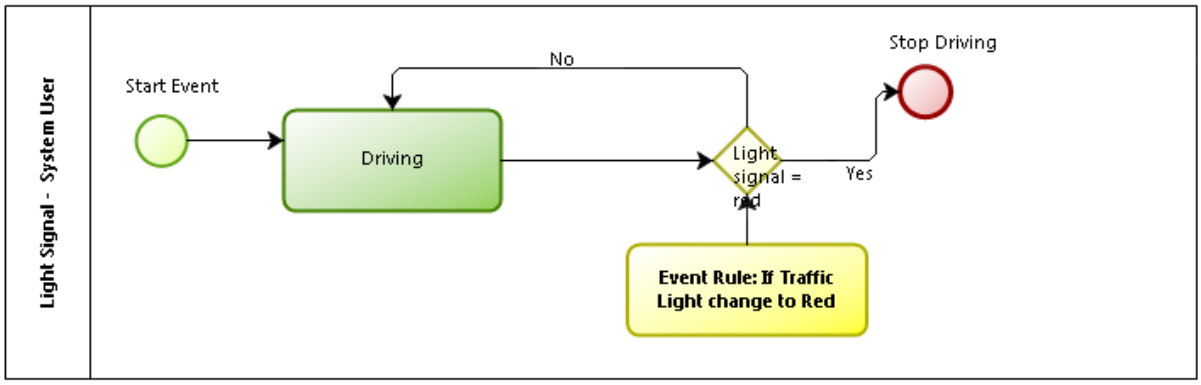


Figure 6: Event Rule - "When light colour change to red, stop driving"

Flow Rules

We propose the class Flow Rule (FR) to formally depict rules that specifically define the flow of workflow processes. All workflows depend on flow rules to progress from one process to another. In other words, flow rules determine the start process and the transition through a chain of processes until the workflow ends. Flow rules can move the workflow along a single chain of processes or split it into multiple pathways, thus forming an acyclic graph. For instance, a path can be established between **"Create Request"** and **"Approve Request"** processes to connect the two related processes in a workflow. Important flow patterns that will be covered in this research include sequence, parallel split and merge. From this perspective Flow Rules define the transition pattern and allow to order the business processes in the workflow dynamically at runtime.

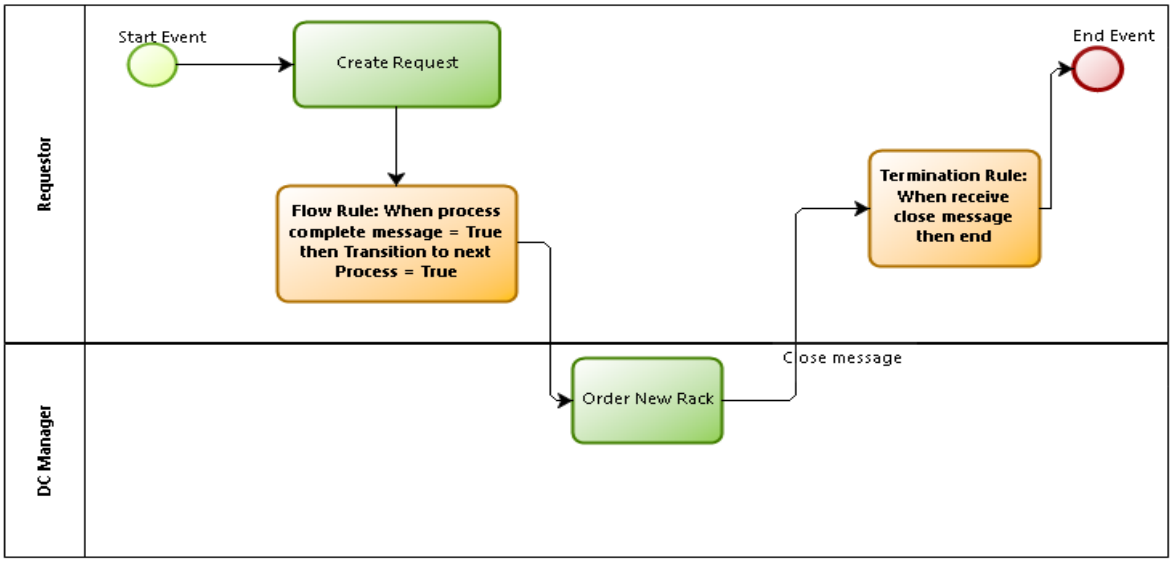


Figure 7: Flow Rule

Termination Rules (TR)

We propose the class Termination Rule (TR) to formally depict rules that specifically trigger the end of a workflow. Some Termination Rules are driven by events only, hence known as End Event. Figure 7 presents Equipment Installation workflow of an organization with three processes "**Create Request**", "**Send Message"** and "**Order New Rack**". In the background, the termination rule "**When receive closing message then end**" looks up and ends processes whenever the rule is invoked. The rule is invoked when the request message is received.

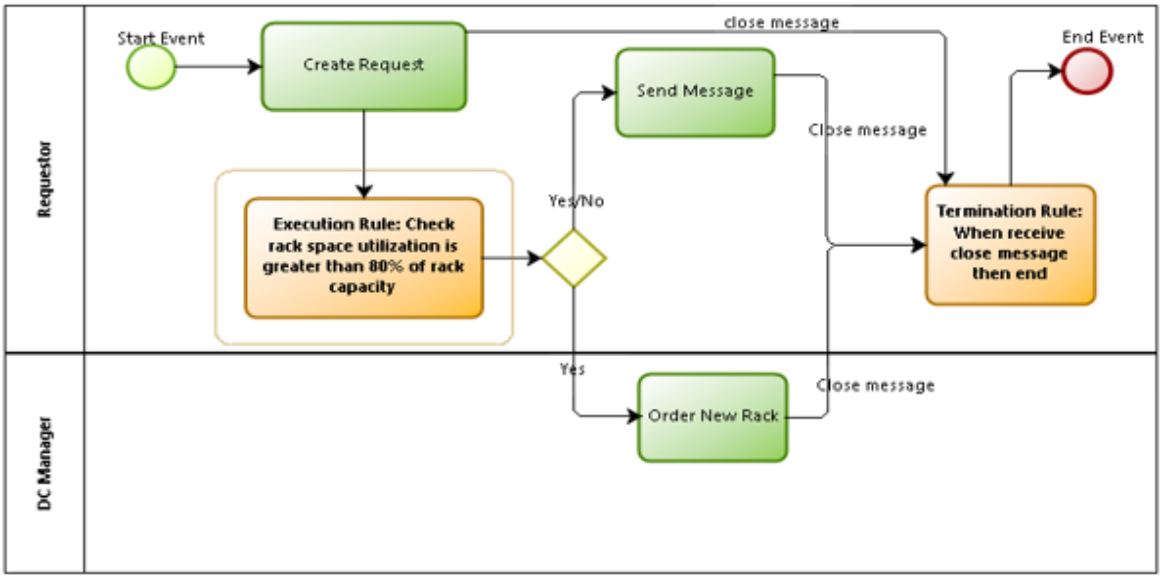


Figure 8: Termination rule

Relationships and dependencies between Business Rules

Business rules formal description

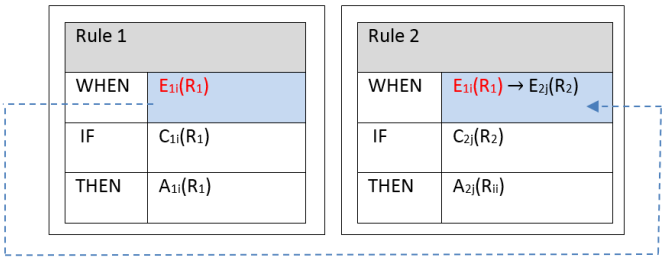
Consider a Business Rule set R containing a collection of rule samples controlling a particular workflow. A Rule set R has one or more related rules that has been put together to guide the movement of processes in the workflow. For instance, R may be made up of Initiation Rule, Flow Rule, Event or Process Rules and Termination Rule.

Let every Rule in R be expressed in terms of $\{R_i,| i= {}_{1,\ldots,\ n}\}$. Each Rule definition R_i consists of a collection of Event (E), Condition (C) and Action (A). We refer to E, C and A to represent sets of Events, Conditions and Actions respectively containing fragments of the Rule R. Now, let E be expressed in terms of $\{E_i,| i= {}_{1,\ldots,\ n}\}$. And C be expressed in terms of $\{C_i,| i= {}_{1,\ldots,\ n}\}$. Also A be expressed in terms of $\{A_i,| i= {}_{1,\ldots,\ n}\}$. In this research, we will use notation $E_{1i}(R_1)$, $C_{1i}(R_1)$ and $A_{1i}(R_1)$ where $E_{1i} \in E_1$, $C_{1i} \in C_1$ and $A_{1i} \in A_1$ to represent Business Rule basic definition. Note that for simplicity reasons, if a part of the Business Rule has no importance in a discussion then it will be omitted. For example, $C_{1i}(R_1)$ and $A_{1i}(R_1)$ will represent a Business Rule that contains Conditions and Actions only.

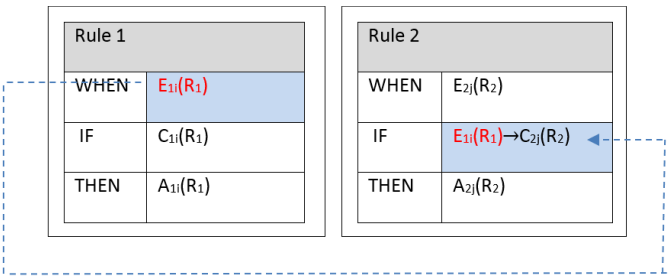
Relation between the business rules

The existence of a dependency between two rules expresses that communication occurs between components (Event, Condition, and Action) of the Business Rule. For example, one Business Rule action may trigger conditions of other Business Rules or condition of one Business rule may depend on an event of another Business Rule. Therefore, Business Rules relationships can be described by analysing Business Rule components relationships. We consider the relationship between two rules to be represented by the symbol $\xrightarrow{Relate\ to}$. For example, $R_1 \xrightarrow{Relate\ to} R_2$ means Rule 1 relates to Rule 2. If one of R_1 action activates event for R_2, we declare as $A_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2)$. Business Rules relationships can be analysed and declared in one of the following possible six ways:

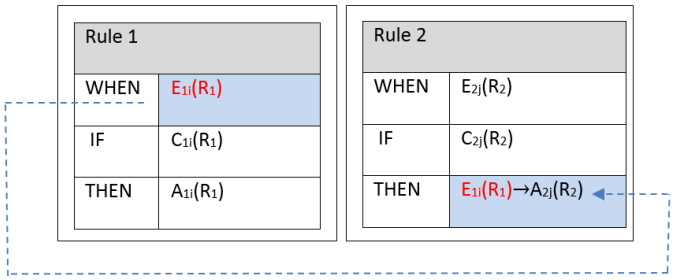
i. $E_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2)$



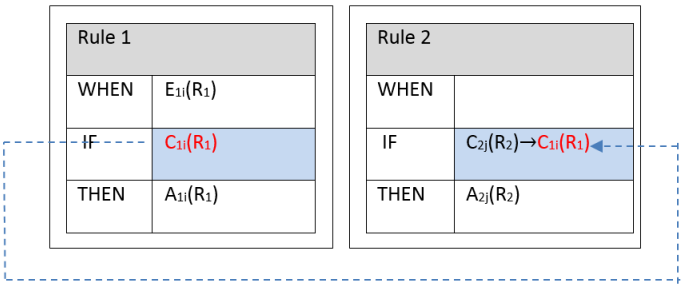
ii. $E_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2)$



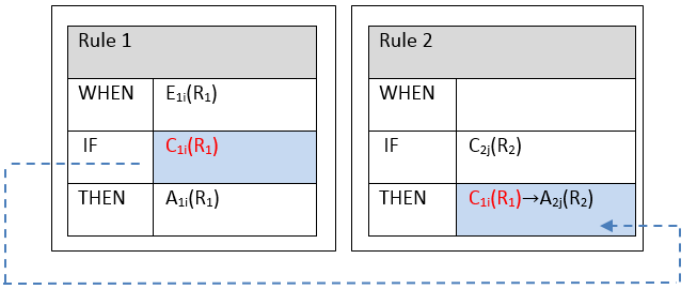
iii. $E_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2)$



iv. $C_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2)$



v. $C_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2)$



vi. $A_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2)$

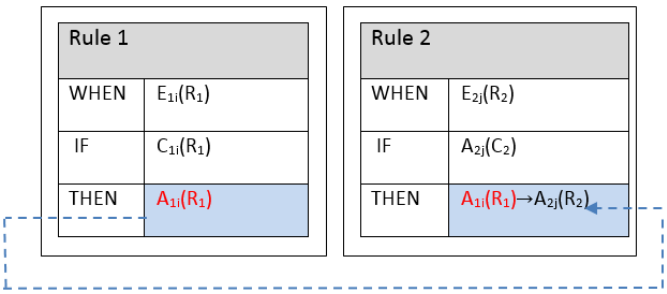


Figure 9: Structural Rule Dependencies

These relationships are defined based on Objects and Objects properties involved in Condition, Event and Action components of the Rules. Moreover, relationship can be defined in terms of qualitative and quantitative characteristics of the object parameters. We examined six ways (i-vi) of representing rule relationships based on the partial order relationship. However, it is far simpler and natural to apply the tree structure to model and picture the relationships between rules. Therefore, tree structure and patterns to show relationship are introduced next.

Business rules dependency tree

Structuring of the rules into an AND-OR tree according to their dependencies would allow implementing of more efficient algorithms for search of the rules. Furthermore, the different patterns of inclusion of the rules in the trees will be used inside the algorithms to control the flow of execution of the rules as the business processes progress at real-time. In addition, we can describe behaviour and flow dependency pattern of rules. For each dependency pattern, we can provide a visual representation of the rule dependency. It is important to understand that although

trees make it easier to understand the relationship between rules, they will need to be translated into rule language for workflow interpretation. Hence algorithms will be defined in additional to rule relationships definitions. The tree is constructed using nodes starting with the root of the tree going down to its leaves. The nodes will represent the Business Rules and edges represent relationships between rules. Tree structures such as decision trees are widely used to describe rules order and priorities; a tree can be made up of a large number of rules presented in analytical and visual manner Gizil Oguz, et al, (2008). In our model, navigation through tree nodes is attained by establishing relationships between rules components. As the name suggests AND/OR Tree, the relationships will be of two kinds. AND relationships, which group several rules that can be invoked simultaneously, and OR relationships, which group several rules that can be invoked alternatively. Variations of AND/OR relationships exist, including Direct AND Dependency, Direct OR Dependency, Indirect AND dependency and Indirect OR Dependency. Each pattern is illustrated in Figure 10a to 13c below. Our discussion in the rest of this section is confined to business rules relationships patterns and formal definitions.

i. Direct AND Dependency patterns

A) Rule's Event Relationships

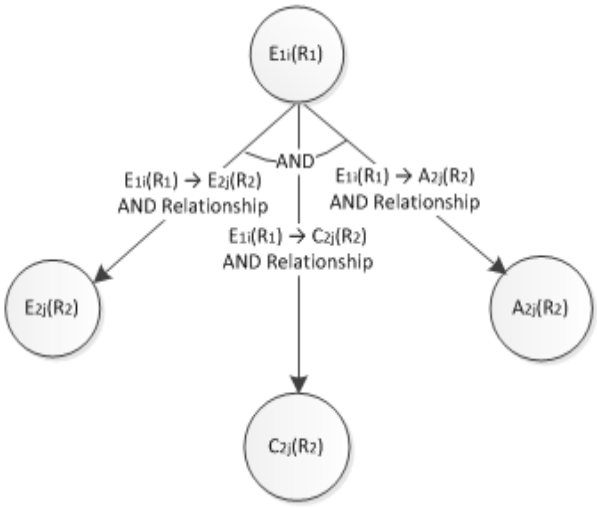


Figure 10a: A Strong Direct AND Tree exemplifying Rule's Event relationships

The above tree represents a direct AND dependency where each node corresponds to the root node/rule $E_{1i}(R_1)$. The following patterns are depicted:

- Direct edge ($E_{1i}(R_1)$, $E_{2j}(R_2)$), with $E_{1i} \rightarrow E_{2j}$, means that the event of rule R_1 must influence the result of rule R_2's event. This is $E_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2)$ relationship.
- Direct edge ($E_{1i}(R_1)$, $C_{2j}(R_2)$), with $E_{1i} \rightarrow C_{2j}$, means that the event of rule R_1 must influence the result of rule R_2's condition. This is $E_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2)$ relationship.
- Direct edge ($E_{1i}(R_1)$, $A_{2j}(R_2)$), with $E_{1i} \rightarrow A_{2j}$, means that the event of rule R_1 must cause change to rule R_2's action. This is $E_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2)$ relationship.
- We can also depict the following possible combination of AND patterns:
  - $E_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \wedge E_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2)$
  - $E_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \wedge E_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2)$
  - $E_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \wedge E_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2)$
  - $E_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \wedge E_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \wedge E_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2)$

B) Rule's Condition Relationships

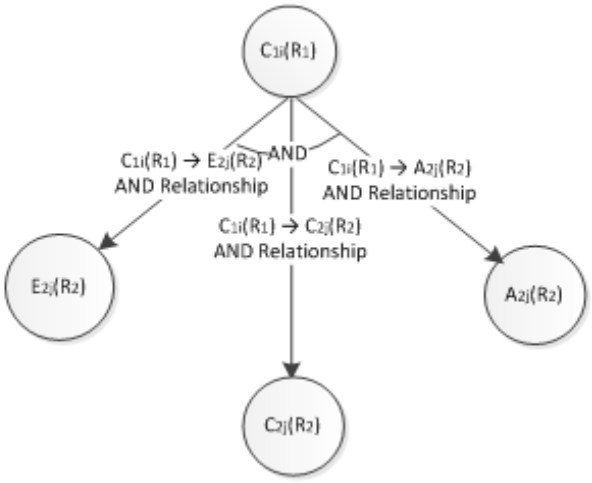


Figure 10b: A Strong Direct AND Tree exemplifying Rule's Condition relationships

This tree represents a direct AND dependency where each node corresponds to the root node/rule $C_{1i}(R_1)$. The following patterns are depicted:

- Direct edge ($C_{1i}(R_1)$, $E_{2j}(R_2)$), with $C_{1i} \rightarrow E_{2j}$, means that the condition of rule R_1 must influence or trigger rule R_2's event. This is $C_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2)$ relationship.
- Direct edge ($C_{1i}(R_1)$, $C_{2j}(R_2)$), with $C_{1i} \rightarrow C_{2j}$, means that the condition of rule R_1 must influence the result of rule R_2's condition. This is $C_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2)$ relationship.
- Direct edge ($C_{1i}(R_1)$, $A_{2j}(R_2)$), with $C_{1i} \rightarrow A_{2j}$, means that the condition of rule R_1 must cause change to rule R_2's action. This is $C_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2)$ relationship.
- We can also depict the following possible combination of AND patterns:
  - $C_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \wedge C_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2)$
  - $C_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \wedge C_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2)$
  - $C_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \wedge C_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2)$
  - $C_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \wedge C_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \wedge C_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2)$

C) Relationships between Rule's Action and the Conditions/Events of another rule

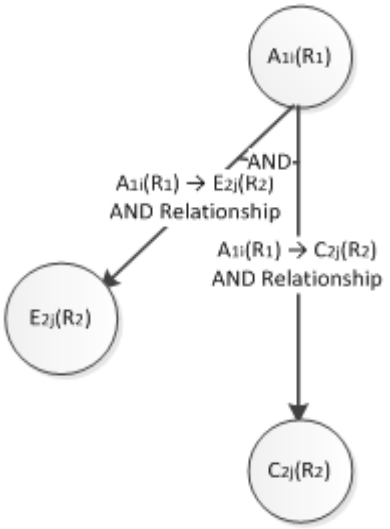


Figure 10c: A Strong Direct AND Tree exemplifying Rule's Action relationships

This tree represents a direct AND dependency where each node corresponds to a root node/rule $A_{1i}(R_1)$. The following patterns are depicted:

- Direct edge ($A_{1i}(R_1)$, $E_{2j}(R_2)$), with $A_{1i} \rightarrow E_{2j}$, means that the action of rule R_1 must influence the result of rule R_2's event. This is $A_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2)$ relationship.
- Direct edge ($A_{1i}(R_1)$, $C_{2j}(R_2)$), with $A_{1i} \rightarrow C_{2j}$, means that the action of rule R_1 must influence the result of rule R_2's condition. This is $A_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2)$ relationship.
- We can also depict the following possible combination of AND pattern:
  - $A_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \wedge A_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2)$

Consider patterns identified from Figure 10a-c. Such dependency patterns only appear when there is a strong relationship between one of more rules. The patterns are based on an AND join, one node (rule) is directly joined to another node (rule) through related components (event, condition, action). The relationship may include relation between objects, quantitative estimation of a property, and qualitative estimation of a property as well as relation between properties of objects/ components (event, condition, and action). A combination of nodes (rules) can also be linked through an AND join.

ii. Direct OR Dependency patterns

A) Rule's Event Relationships

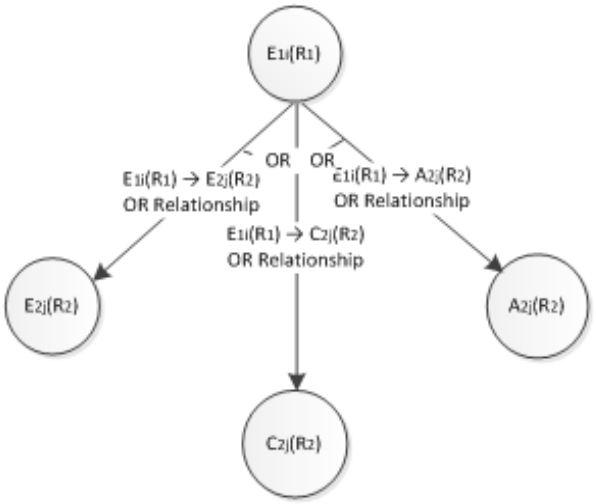


Figure 11a: A Weak Direct OR Tree exemplifying Rule's Event relationships

This tree represents a direct OR dependency where the following possible combination patterns are depicted when $E_{1i}(R_1)$ is a root node/rule:

- The execution of $E_{1i}(R_1)$ may or may not trigger the execution of $E_{2j}(R_2)$ depending on additional events, conditions or actions from the class, or guarded by external events so each of these cases introduces different degree of "weakness"
- The execution of $E_{1i}(R_1)$ may or may not trigger the execution of $C_{2j}(R_2)$ depending on additional events, conditions or actions from the class, or guarded by external events.
- The execution of $E_{1i}(R_1)$ may or may not trigger the execution of $A_{2j}(R_2)$ depending on additional events, conditions or actions from the class, or guarded by external events
- We can also depict the following possible combination of OR patterns:
  - $E_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \vee E_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2)$
  - $E_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \vee E_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2)$
  - $E_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \vee E_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2)$
  - $E_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \vee E_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \vee E_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2)$

B) Rule's Condition Relationships

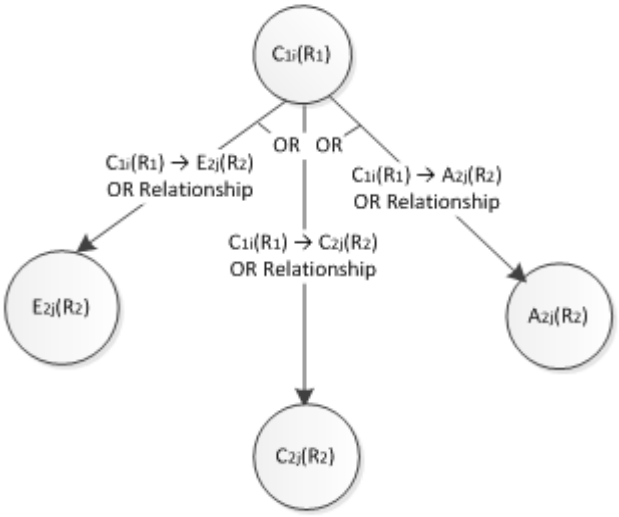


Figure 11b: A Weak Direct OR Tree exemplifying Rule's Condition relationships

This tree represents a direct OR dependency where the following possible combination patterns are depicted when $C_{1i}(R_1)$ is a root node/rule:

- The execution of $C_{1i}(R_1)$ may or may not trigger the execution of $E_{2j}(R_2)$ depending on additional events, conditions or actions from the class, or guarded by external events so each of these cases introduces different degree of "weakness"
- The execution of $C_{1i}(R_1)$ may or may not trigger the execution of $C_{2j}(R_2)$ depending on additional events, conditions or actions from the class, or guarded by external events
- The execution of $C_{1i}(R_1)$ may or may not trigger the execution of $A_{2j}(R_2)$ depending on additional events, conditions or actions from the class, or guarded by external events
- We can also devise the following possible combination of OR patterns:
  - $C_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \vee C_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2)$
  - $C_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \vee C_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2)$
  - $C_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \vee C_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2)$
  - $C_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \vee C_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \vee C_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2)$

C) Relationships between Rule's Action and the Conditions/Events of another rule

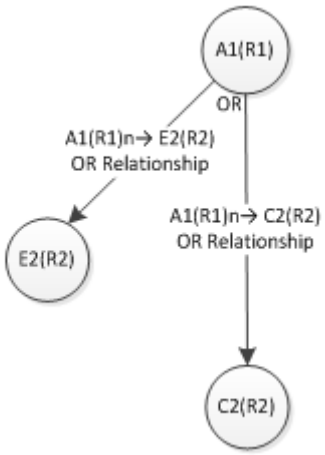


Figure 11c: A Weak Direct OR Tree exemplifying Rule's Action relationships

The above tree represents a direct OR dependency where the following possible combination patterns are depicted when $A_{1i}(R_1)$ is a root node/rule:

- The execution of $A_{1i}(R_1)$ may or may not trigger the execution of $E_{2j}(R_2)$ depending on additional events, conditions or actions from the class, or guarded by external events so each of these cases introduces different degree of "weakness"
- The execution of $A_{1i}(R_1)$ may or may not trigger the execution of $C_{2j}(R_2)$ depending on additional events, conditions or actions from the class, or guarded by external events
- We can also devise the following possible combination of OR patterns:
  - $A_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \vee A_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2)$

Consider patterns identified from Figure 11a-c. Such dependency patterns only appear when there is a weak relationship between one of more rules. These dependency patterns are based on an OR join, one node (rule) is directly joined to another node (rule) through related components (event, condition, action). The relationship may include relation between objects, quantitative estimation of a property, and qualitative estimation of a property as well as relation between properties of objects/ components (event, condition, and action). A combination of nodes (rules) can also be linked through an OR join.

iii. Indirect AND Dependency patterns

A) Rule's Event AND-Relationship

The tree below represents indirect AND dependency where nodes are indirectly connected to the root node/rule $E_{1i}(R_1)$ through nodes/rules (X,Y,Z).

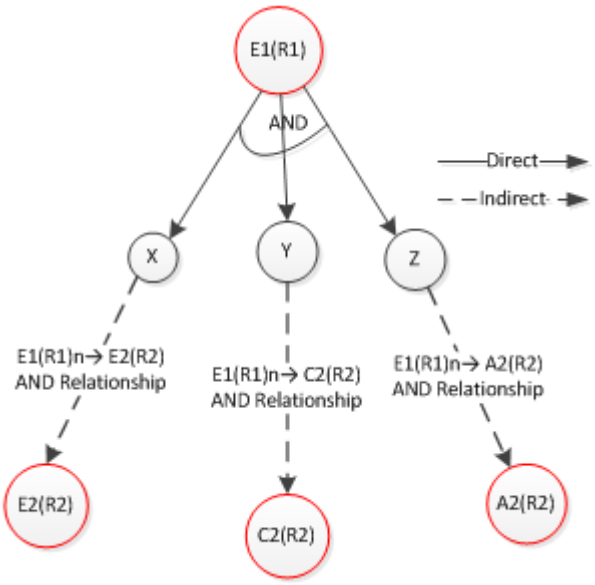


Figure 12a: Strong Indirect AND Tree exemplifying Rule's Event AND-Relationships

The following relationship patterns are depicted:

- Edge $(E_{1i}(R_1), X)$; $(X, E_{2j}(R_2))$, with $E_{1i} \rightarrow X$; $X \xrightarrow{AND} E_{2j}$, means that the event of rule R_1 is indirectly influencing the result of rule R_2's event through rule X. The relationship consists of pairs $E_{1i}(R_1) \xrightarrow{Relate\ to} X$ and $X \xrightarrow{Relate\ to} E_{2j}(R_2)$. By the transitivity relation property, Di Nola A (1991) $E_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2)$
- Edge $(E_{1i}(R_1), Y)$; $(Y, C_{2j}(R_2))$, with $E_{1i} \rightarrow Y$; $Y \xrightarrow{AND} C_{2j}$, means that the event of rule R_1 is indirectly influencing the result of rule R_2's condition through rule Y. The relationship consists

of pairs $E_{1i}(R_1) \xrightarrow{Relate\ to} Y$ and $Y \xrightarrow{Relate\ to} C_{2j}(R_2)$. By the transitivity relation property, Di Nola A (1991) $E_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2)$.

- Edge $(E_{1i}(R_1), Z)$; $(Z, A_{2j}(R_2))$, with $E_{1i} \rightarrow Z$; $Z \xrightarrow{AND} A_{2j}$, means that the event of rule R_1 is indirectly causing change to rule R_2's action through rule Z. The relationship consists of pairs $E_{1i}(R_1) \xrightarrow{Relate\ to} Z$) and $Z \xrightarrow{Relate\ to} A_{2j}(R_2)$. By transitivity relation property, Di Nola A (1991) $E_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2)$.
- We can also depict the following possible combination of AND-relationship patterns:

  - $E_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \wedge E_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2)$
  - $E_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \wedge E_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2)$
  - $E_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \wedge E_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2)$
  - $E_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \wedge E_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \wedge E_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2)$

C) Rule's Condition AND-Relationship

The tree below represents an indirect AND dependency where nodes are indirectly connected to the root node/rule $C_{1i}(R_1)$ through rules (X,Y,Z).

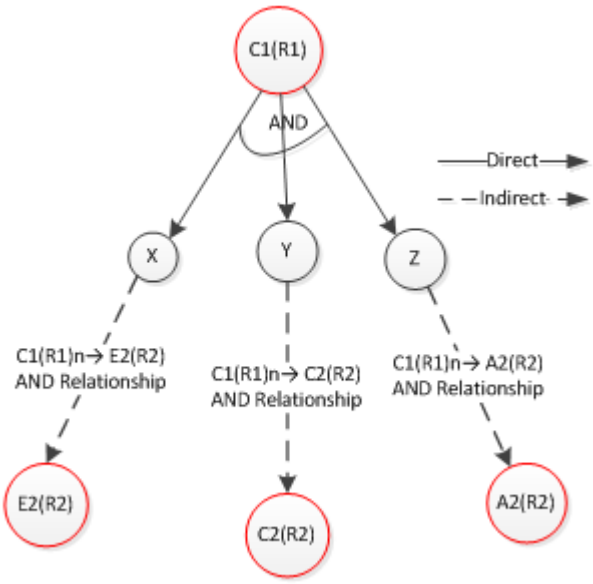


Figure 12b: Strong Indirect AND Tree exemplifying Rule's Condition AND-Relationships

The following patterns are depicted:

- Edge $(C_{1i}(R_1)\ X)$; $(X, E_{2j}(R_2))$, with $C_{1i} \rightarrow X$; $X \xrightarrow{AND} E_{2j}$ means that the condition of rule R_1 is indirectly influencing or triggering rule R_2's event through rule X. The relationship consists of pairs $C_{1i}(R_1) \xrightarrow{Relate\ to} X$ and $X \xrightarrow{Relate\ to} E_{2j}(R_2)$. By the transitivity relation property, Di Nola A (1991) $C_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2)$.
- Edge $(C_{1i}(R_1)\ Y)$; $(Y, C_{2j}(R_2))$, with $C_{1i} \rightarrow Y$; $Y \xrightarrow{AND} C_{2j}$ means that the condition of rule R_1 is indirectly influencing the result of rule R_2's condition through rule Y. The relationship consists

of pairs $C_{1i}(R_1) \xrightarrow{Relate\ to} Y$ and $Y \xrightarrow{Relate\ to} C_{2j}(R_2)$. By the transitivity relation property, Di Nola A (1991) $C_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2)$.

- Edge $(C_{1i}(R_1), Z)$; $(Z, A_{2j}(R_2))$, with $C_{1i} \rightarrow Z$; $Z \xrightarrow{AND} A_{2j}$, means that the condition of rule R_1 is indirectly affecting rule R_2's action through rule Z. The relationship consists of pairs $C_{1i}(R_1) \xrightarrow{Relate\ to} Z$ and $Z \xrightarrow{Relate\ to} A_{2j}(R_2)$. By transitivity relation property, Di Nola A (1991) $C_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2)$.
- We can also depict the following possible combination of AND-relationship patterns:
  - $C_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \wedge C_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2)$
  - $C_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \wedge C_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2)$
  - $C_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \wedge C_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2)$
  - $C_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \wedge C_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \wedge C_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2)$

D) AND-Relationships between Rule's Action and the Conditions/Events of another rule

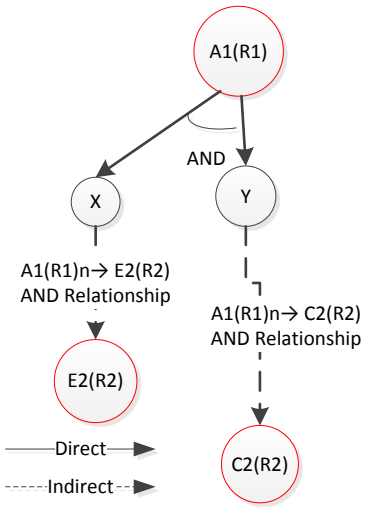


Figure 12c: Strong Indirect AND Tree exemplifying Rule's Action AND-Relationships

The above tree represents indirect AND dependency where nodes are indirectly connected to the root node/rule $A_{1i}(R_1)$ through rules (X,Y). The following patterns are depicted:

- The execution of $A_{1i}(R_1)$ indirectly triggers the execution of $E_{2j}(R_2)$ through additional events, conditions or actions of the X rule. The relationship consists of pairs $A_{1i}(R_1) \xrightarrow{Relate\ to} X$ and $X \xrightarrow{Relate\ to} E_{2j}(R_2)$. By the transitivity relation property, Di Nola A (1991) $A_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2)$.
- The execution of $A_{1i}(R_1)$ indirectly triggers the execution of $C_{2j}(R_2)$ through additional events, conditions or actions from Y rule. The relationship consists of the pairs $A_{1i}(R_1) \xrightarrow{Relate\ to} Y$ and $Y \xrightarrow{Relate\ to} C_{2j}(R_2)$. By the transitivity relation property, Di Nola A (1991) $A_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2)$.
- We can also devise the following possible combination of AND-relationship patterns:
  - $A_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \wedge A_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2)$

The "Indirect AND Dependency" pattern (Figure 12a-c) is such that rule nodes flow into two or more edges; the edges proceed and merge into a rule node where a connection or relationship is to be established, hence indirectly connected through intermediate nodes. This dependency pattern is based

indirect AND connections between nodes or rules on the same path. There must be at least one indirect rule from the nodes with an AND connection.

iv. Indirect OR Dependency patterns

A) Rule's Event OR-Relationship

The tree below represents indirect OR dependency where nodes are indirectly connected to the root node/rule $E_{1i}(R_1)$ through nodes/rules (X,Y,Z).

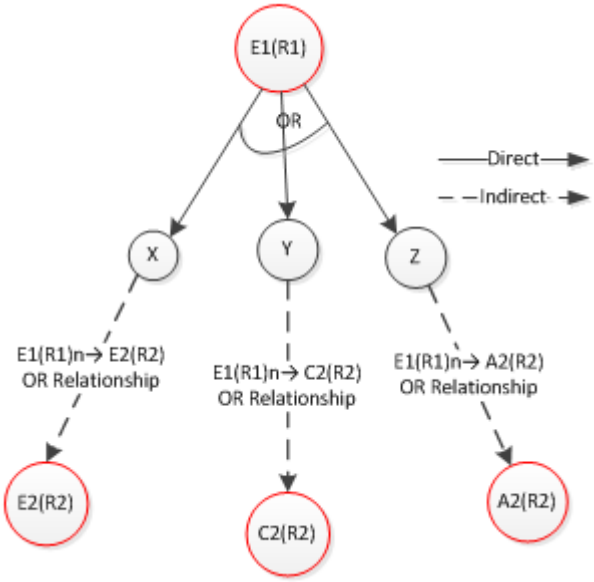


Figure 13a: Weak Indirect OR Tree exemplifying Rule's Event OR-Relationships

The following patterns are depicted:

- Edge $(E_{1i}(R_1), X)$; $(X, E_{2j}(R_2))$, with $E_{1i} \rightarrow X$; $X \xrightarrow{OR} E_{2j}$, means that the event of rule R_1 may or may not indirectly influence the result of rule R_2's event. The relationship consists of pairs $E_{1i}(R_1) \xrightarrow{Relate\ to} X$ and $X \xrightarrow{Relate\ to} E_{2j}(R_2)$. By the transitivity relation property, Di Nola A (1991) $E_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2)$
- Edge $(E_{1i}(R_1), Y)$; $(Y, C_{2j}(R_2))$, with $E_{1i} \rightarrow Y$; $Y \xrightarrow{OR} C_{2j}$, means that the event of rule R_1 may or may not indirectly influence the result of rule R_2's condition. The relationship consists of pairs $E_{1i}(R_1) \xrightarrow{Relate\ to} Y$ and $Y \xrightarrow{Relate\ to} C_{2j}(R_2)$. By the transitivity relation property, Di Nola A (1991) $E_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2)$.
- Edge $(E_{1i}(R_1), Z)$; $(Z, A_{2j}(R_2))$, with $E_{1i} \rightarrow Z$; $Z \xrightarrow{OR} A_{2j}$, means that the event of rule R_1 may or may not indirectly cause change to rule R_2's action. The relationship consists of pairs $E_{1i}(R_1) \xrightarrow{Relate\ to} Z$ and $Z \xrightarrow{Relate\ to} A_{2j}(R_2)$. By the transitivity relation property, Di Nola A (1991) $E_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2)$.
- We can also depict the following possible combination of OR patterns:
  - $E_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \vee E_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2)$
  - $E_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \vee E_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2)$
  - $E_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \vee E_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2)$
  - $E_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \vee E_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \vee E_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2)$

B) Rule's Condition OR-Relationship

The tree below represents indirect OR dependency where nodes are indirectly connected to the root node/rule $C_{1i}(R_1)$ through rules (X,Y,Z).

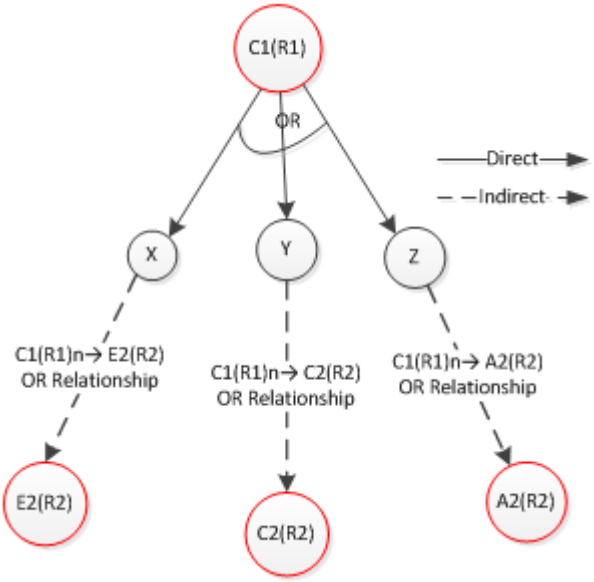


Figure 13b: Weak Indirect OR Tree exemplifying Rule's Condition OR-Relationships

The following patterns are depicted:

- Edge ($C_{1i}(R_1)$ X); (X, $E_{2j}(R_2)$), with $C_{1i} \rightarrow X$; $X \xrightarrow{OR} E_{2j}$ means that the condition of rule R_1 may or may not indirectly influence or trigger rule R_2's event. The relationship consists of pairs conclude $C_{1i}(R_1) \xrightarrow{Relate\ to} X$ and conclude $X \xrightarrow{Relate\ to} E_{2j}(R_2)$. By the transitivity relation property, Di Nola A (1991) $C_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2)$.
- Edge ($C_{1i}(R_1)$ Y); (Y, $C_{2j}(R_2)$), with $C_{1i} \rightarrow Y$; $Y \xrightarrow{OR} C_{2j}$ means that the condition of rule R_1 may or may not indirectly influence the result of rule R_2's condition. The relationship consists of pairs $C_{1i}(R_1) \xrightarrow{Relate\ to} Y$ and $Y \xrightarrow{Relate\ to} C_{2j}(R_2)$. By the transitivity relation property, Di Nola A (1991) $C_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2)$.
- Edge ($C_{1i}(R_1)$, Z); (Z, $A_{2j}(R_2)$), with $C_{1i} \rightarrow Z$; $Z \xrightarrow{OR} A_{2j}$, means that the condition of rule R_1 may or may not indirectly cause change to rule R_2's action. The relationship consists of pairs $C_{1i}(R_1) \xrightarrow{Relate\ to} Z$ and $Z \xrightarrow{Relate\ to} A_{2j}(R_2)$. By the transitivity relation property, Di Nola A (1991) $C_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2)$.
- We can also depict the following possible combination of OR patterns:
  - $C_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \vee C_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2)$
  - $C_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \vee C_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2)$
  - $C_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \vee C_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2)$
  - $C_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \vee C_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \vee C_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2)$

C) OR-Relationships between Rule's Action and the Conditions/Events of another rule

The tree below represents an indirect OR dependency where nodes are indirectly connected to the root node/rule $A_{1i}(R_1)$ through rules (X,Y).

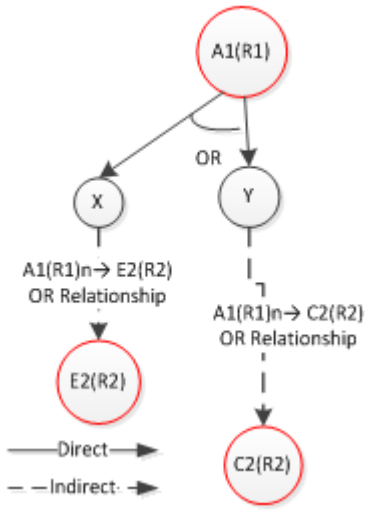


Figure 13c: Weak Indirect OR Tree exemplifying Rule's Action OR-Relationships

The following patterns are depicted:

- The execution of $A_{1i}(R_1)$ may or may not indirectly triggers the execution of $E_{2j}(R_2)$ depending on additional events, conditions or actions from X rule. The relationship consists of pairs $A_{1i}(R_1) \xrightarrow{Relate\ to} X$ and $X \xrightarrow{Relate\ to} E_{2j}(R_2)$. By the transitivity relation property, Di Nola A (1991) $A_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2)$.
- The execution of $A_{1i}(R_1)$ may or may not indirectly trigger the execution of $C_{2j}(R_2)$ depending on additional events, conditions or actions from Y rule. The relationship consists of pairs $A_{1i}(R_1) \xrightarrow{Relate\ to} Y$ and $Y \xrightarrow{Relate\ to} C_{2j}(R_2)$. By the transitivity relation property, Di Nola A (1991) $A_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2)$.
- We can also devise the following possible combination of OR patterns:
  - $A_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \vee A_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2)$

The "Indirect OR Dependency" pattern (Figure 13a-c) is such that rule nodes flow into two or more edges; the edges proceed and merge into a rule node where connection or relationship is to be established, hence are indirectly connected through intermediate nodes. This dependency pattern is based on an indirect OR connections between nodes or rules on the same path. There must be at least one indirect rule from the nodes with an OR connection.

All relationship patterns presented in (i) to (iv) are formally defined as follows:

(1) Direct AND Dependency

$$\forall R1, R2\ R_1 \xrightarrow{AND} R_2 \longleftrightarrow$$

$$E_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \vee$$

$$E_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \vee$$

$$E_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2) \vee$$

$$E_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \wedge E_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \vee$$

$$E_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \wedge E_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2) \vee$$

$$E_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \wedge E_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2) \vee$$

$$E_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \wedge E_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \wedge E_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2) \vee$$

$$C_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \vee$$

$$C_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \vee$$

$$C_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2) \vee$$

$C_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \wedge C_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \vee$

$C_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \wedge C_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2) \vee$

$C_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \wedge C_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2) \vee$

$C_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \wedge C_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \wedge C_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2) \vee$

$A_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \vee$

$A_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \vee$

$A_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \wedge A_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2)$

(2) Direct OR Dependency

$\forall R1, R2\ R_1 \xrightarrow{OR} R_2 \longleftrightarrow$

$E_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \vee$

$E_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \vee$

$E_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2) \vee$

$E_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \vee E_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \vee$

$E_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \vee E_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2) \vee$

$E_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \vee E_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2) \vee$

$E_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \vee E_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \vee E_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2) \vee$

$C_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \vee$

$C_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \vee$

$C_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2) \vee$

$C_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \vee C_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \vee$

$C_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \vee C_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2) \vee$

$C_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \vee C_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2) \vee$

$C_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \vee C_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \vee C_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2) \vee$

$A_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \vee$

$A_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \vee$

$A_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \vee A_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2)$

(3) Indirect AND Dependency

$\forall R1, R2\ R_1 \xrightarrow{AND} R_n \wedge R_n \xrightarrow{AND} R_2 \longleftrightarrow$

$E_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \vee$

$E_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \vee$

$E_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2) \vee$

$E_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \wedge E_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \vee$

$E_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \wedge E_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2) \vee$

$E_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \wedge E_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2) \vee$

$E_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \wedge E_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \wedge E_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2) \vee$

$C_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \vee$

$C_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \vee$

$C_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2) \vee$

$C_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \wedge C_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \vee$

$C_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \wedge C_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2) \vee$

$C_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \wedge C_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2) \vee$

$C_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \wedge C_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \wedge C_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2) \vee$

$A_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \vee$

$A_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \vee$

$A_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \wedge A_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2)$

(4) Indirect OR Dependency

$\forall R1, R2\ R_1 \xrightarrow{OR} R_n \wedge R_n \xrightarrow{OR} R_2 \longleftrightarrow$

$E_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \vee$

$E_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \vee$

$E_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2) \vee$

$E_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \vee E_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \vee$

$E_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \vee E_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2) \vee$

$E_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \vee E_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2) \vee$

$E_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \vee E_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \vee E_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2) \vee$

$C_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \vee$

$C_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \vee$

$C_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2) \vee$

$C_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \vee C_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \vee$

$C_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \vee C_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2) \vee$

$C_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \vee C_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2) \vee$

$C_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \vee C_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \vee C_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2) \vee$

$A_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \vee$

$A_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2) \vee$

$A_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2) \vee A_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2)$

The relationship is defined by directly linking the objects and indirectly relating the quantitative and qualitative estimation of their properties. Although the relationship patterns are different in terms of their semantics, they also bear some similarities in terms of the appearance of different components of the rules in the structures representing their use in real time. For example, in Figure 14 we can identify the following patterns of dependency between rules: rules on the same path (also known as chained rules, shown in the same color), rules on the same level (alternative rules), rules with the same parents (alternative chains), directly related rules, indirectly related rules, etc.

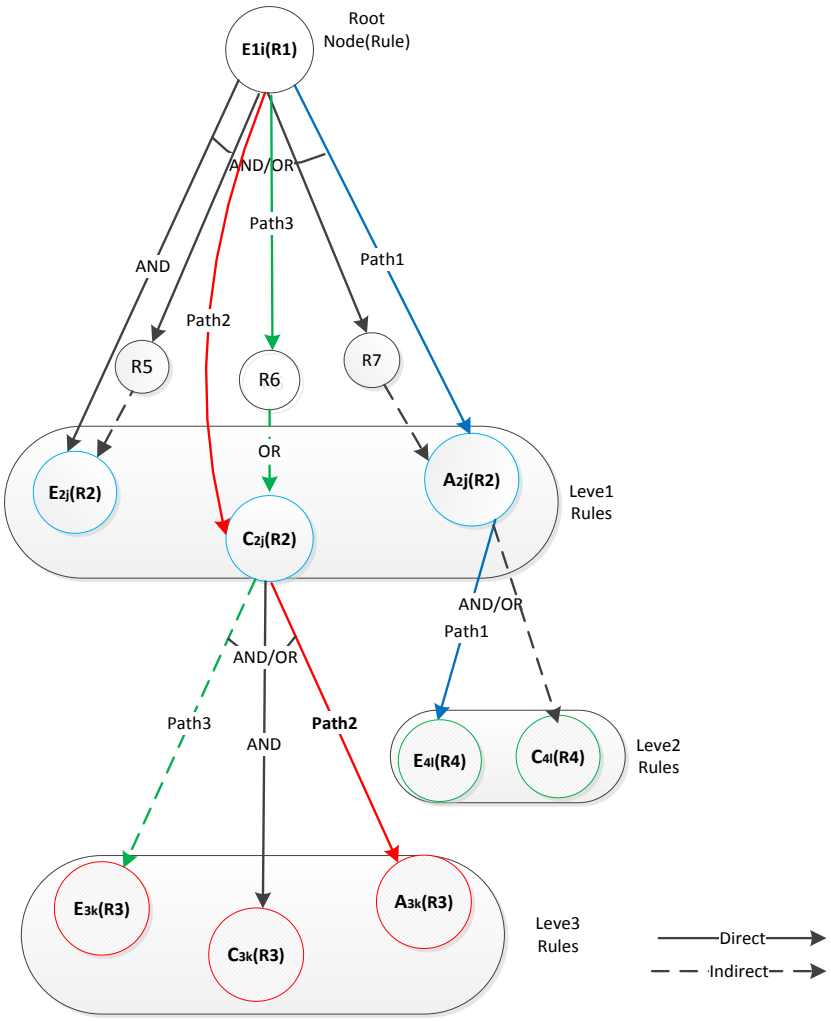


Figure 14: AND/ OR Tree - Rule Relationship patterns

The AND/OR Tree (Figure 14) combines all relationship patterns presented earlier. Now, let us describe various dependency patterns found in the AND/ OR Tree:

Precedence based dependencies: The relationship between Rules is formed by using either successor or predecessor rules. This relationship can be identified when within the same root, parent-child or sibling nodes are related.

Level based dependencies: Nodes (Rules) at the same levels are connected by using root nodes. The relationship between Rules can be defined based on the levels in the tree. The relationship can form multilevel dependencies as well. Furthermore, this pattern can form an AND or OR relationship pattern.

Path based dependencies: Nodes (Rules) on the same paths are connected consecutively from the top node to the leaf nodes.

Node based dependencies: Nodes (Rules) without the same root, parent-child or sibling can be related. The relationship can be defined solely based on individual node (rule) properties that invoke other Rules. Here the relationship may result in a non-tree structure and can be inefficient for a large set of rules since every node's relationship is to be checked. However, we argue that this is a reasonable relationship since those properties with dependence can be achieved by establishing rule property class dependencies.

Indirect node based dependencies: The dependency is established through intermediate nodes on the same root node. AND and OR Indirect node dependencies variations exist.

The AND/ OR Tree may be considered as a set of rule dependency patterns that have similar behaviour and shapes. We can distinguish a dependency pattern subset using $AOT_{pattern} \subseteq AOT$ to represent different dependency patterns as described in 1-5 above. A dependency pattern consists of a set of nodes $\{R_i, | i = 1, \ldots, n\}$. A distinguished node called rootR_i and a mapping (relationship): root$R_i \rightarrow R_i$ relating a node with its parent node. Any given rule

dependency pattern type may have a default meta-rule dependency established. Derived classes created from base classes through process of inheritance can inherit a dependency type and override the meta-rules based on relationships.

Implementation

Dependencies between rules are formulated using objects which are parameters of their conditions, events and actions, hence the building of the dependency tree can be done using an incremental algorithm as the rules are added to the repository. Since the events, conditions and actions of the rules can be implemented as separate objects, the algorithms for processing the dependency tree can be based on rules operating on the same objects (meta-rules). This approach allows to use the object-oriented technology of programming for indexing. We have implemented a pilot of the framework using the open source rule management systems **DROOLS,** Proctor, et al. (2011). To provide high level of adaptation, each component of the rules is implemented as an atomic Java class which can be executed directly in Drools classes. Business rule classes (Event, Condition, Action) are associated with a rule (Rule) and workflow class (Flow). Business rules classification is implemented using the class inheritance concept as discussed in Aliverti et al (2016). Figure 15 presents an example of Condition Class implementation:

```
//Condition class implementation - setting properties and methods
i.e. setter and getters
Package com.ABRIW.model;
Public class Condition {
        //Properties declarations
        Private String Condition_Object;
        Private String Condition_Property;
        Private String AND;
        Private String OR;
        //Methods
        Public String getConditionObject () {
                return ConditionObject;
        }
        Public void setConditionObject (String ConditionObject){
                this.ConditionObject = ConditionObject;
        }
        ...
}
```

Figure 15 Implementation of Condition Class

In additional to the representation of business objects in **DROOLS**, currently we are working on a series of algorithms for propagating the changes and logical analysis of the rules accounting the structural dependencies between them. Also under way is the implementation of two separate inference engines: a forward chaining inference engine for process workflow management, invoked by the events captured during execution, and a backward chaining inference engine for reorganising the rules, which propagates the changes and adapts to the changing state of control at real-time.

Conclusion and future work

The framework presented in this paper introduces a two-level model for business rules governed process workflows which is based on strict logical formalization of the business ontology. It allows the use of object-oriented technology for modelling workflows and business rules while providing a seamless mechanism for incremental indexing of the business rules by accounting their logical dependencies. This can also be used to propagate the changes amongst the rules in real time which can influence activities and operations in many business workflows today. Although our primary interest is to apply the framework to the business processes typical in the construction industry we believe our approach has much wider potential due to its strictly logical formalisation and domain independence. The framework could be applied to both large business process modelling tasks and small but very dynamic business processes like the typical digital business processes found in online banking or e-Commerce. It can be also interesting for adjusting rule-based policies in the case of changing conditions, typical in cyber security area. The effect of adding the capability to adapt automatically the access rights in order to account the new resources and new channels can have a huge impact in this area.

Acknowledgement

The work reported here has been partially funded by Vertiv.

References

Cognizant (2015), The Robot and I: How New Digital Technologies Are Making Smart People and Businesses Smarter by Automating Rote Work. White Paper [available online at https://www.cognizant.com/whitepapers/the-robot-and-I-how-new-digital-technologies-are-making-smart-people-and-businesses-smarter-codex1193.pdf]

Grady Booch et al. (2007), Object-Oriented Analysis and Design with Applications, 2nd edition. ISBN 0-8053-5340-2 Addison-Wesley

Umeshwar Dayal, et al. (1988), Rules Are Objects Too - A Knowledge Model for an Active, Object-Oriented Database Management System: International Workshop on Object-Oriented Database Systems (OODBS 1988), Lecture Notes in Computer Science, vol 334. Springer, Berlin, Heidelberg.

D. Jordan et al. (2007), Web Services Business Process Execution Language (WSBPEL) [available online at http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.pdf]

Chun Ouyang, et al. (2005), Formal Semantics and Analysis of Control Flow in WS-BPEL, BPM Center Report BPM-05-15 [available online at http://wwwis.win.tue.nl/~wvdaalst/publications/p294.pdf]

François Bryet al. (2006): Realizing Business Processes with ECA Rules. Benefits, Challenges, Limits, in: 4th International Workshop Principles and Practice of Semantic Web Reasoning PPSWR 2006. Lecture Notes in Computer Science, vol 4187. Springer [available at https://epub.ub.uni-muenchen.de/17311/]

Gizil Oguz (2008), Decision Tree Learning for Drools, Ecole ´ Polytechnique F´ed´erale de Lausanne [available online at https://infoscience.epfl.ch/record/126292/files/oguz-thesis_final.pdf]

Casati et al. (2000), Using Patterns to Design Rules in Workflows, IEEE Trans. Software Eng. vol. 26, no.8, pp. 760-785

Proctor, et al. (2011), Drools Expert User Guide, JBoss [available online at https://docs.jboss.org/drools/release/6.0.0.CR4/drools-expert-docs/html_single/]

Rowe, Stephens & Guo (2004), The use of Business Rules with Workflow Systems [available online at https://www.w3.org/2004/12/rules-ws/paper/105/]

Al Hilwa & Stephen D. Hendrick (2012), Competitive Analysis Worldwide Business Rules Management Systems, IDC. [available online at ftp://public.dhe.ibm.com/software/websphere/odm/2011_BRMS_MarketShare_Report.pdf]

Andrew Macdonald (2010), The value of IBM WebSphere ILOG BRMS, IBM. [available at https://www-01.ibm.com/software/integration/business-rule-management/jrules-family/]

Christian de Sainte Marie (2011), IBM Web Sphere ILOG BRMS, IBM.[available at https://www-01.ibm.com/software/integration/business-rule-management/jrules-family/]

Paul Haley (2013), Confessions of a production rule vendor, Commercial Intelligenece [available at http://haleyai.com/wordpress/2013/06/22/confessions-of-a-production-rule-vendor-part-1]

Browne, P (2009), JBoss Drools Business Rules. ISBN-10: 1847196063 ISBN-13: 978-1847196064

Jacob Feldman (2011), Creating, Testing, and Executing Decision Models with OpenRules, WordPress. [available at https://openrules.wordpress.com/]

Jérôme Boyer & Hafedh Mili (2011), Agile Business Rule Development: Process, Architecture, and JRules Examples 2011th Edition, ISBN 978-3-642-19040-7

Anantaram C, (2007), A Framework to specify Declarative Rules on Objects, Attributes and Associations in the object model, in Journal of Object Technology, vol. 6, pp. 91-106.

A. Goh et al. (2001), ECA Rule-Based Support for Workflows, Artificial Intelligence, vol. 15, pp. 37-46.

Geppert (1998), Defining the Semantics of Reactive Components in Event-Driven Workflow Execution with Event Histories, Information Systems, vol. 23, nos. 3/4, pp. 235-252.

Esteban Aliverti et al (2016), Mastering JBoss Drools 6, Packt Publishing, ISBN 139781783288625

Di Nola, A (1991) Transitive solutions of relational equations on finite sets and linear lattices. Lecture Notes in Computer Science, 173-182.