Research and Application of Rule Engine in Scheduled Outage Management

Bo Wu, Zhimin Guo
Equipment Condition Assessment Center, HAEPC ELECTRIC POWER RESEARCH INSTITUTE
Zhengzhou, 450052, China
E-mail: 516wb@163.com

Abstract—**With the rapid development of economy, the requirement for electricity supply is growing dramatically, and the consumers are claiming for higher power supply quality with less outage. Scheduled outage makes a major part of power outage, and it is mainly caused by power grid construction and power equipment maintenance. Scheduled outage has a lot of concerns, such as grid structure, potential outage loss, critical consumers or important occasions, and such factors are changing rapidly with the quick development of distribution network. As a result, the validation and optimization of scheduled outage becomes complicated. Rule engine[1] can describe business rules by pre-defined language, separate such rules from functional code[2], and execute them when trigger condition is satisfied. The application of rule engine can increase system adaptability to business variation, and decrease the system coupling and maintenance cost. With the application of rule engine in scheduled outage management, outage plan can be effectively validated and flexibly optimized, and the outage loss and consumer complaints could be significantly decreased.**

Keywords- distribution network; business rule; rule engine; scheduled outage

I. INTRODUCTION

As the quick development of economy, the requirement for electricity supply is growing dramatically. Distribution network links every consumer to power grid, therefore the distribution network is developing rapidly and scheduled outage becomes inevitable due to the construction and maintenance of distribution network[3]. At the same time, consumers are claiming for better service quality, higher power supply reliability and less power outage. As the result, the scheduled outage must be carefully validated and optimized to shorten the outage interval and minimize outage loss. This article focuses on how to validate and optimize the outage plan, and extract such validation logic as adaptable business rules, in order to promote power supply quality and shorten outage intervals.

An outage plan can be influenced by a lot of factors. First of all, repeating outage should be avoided. That means maintenance work for grid facilities in the same distribution line, corresponding substation and transmission line should be arranged at the same time. Secondly, outage loss should be minimized, which will ask the outage time to be arranged at low point of load. Third, the gird structure would also influence the scheduled outage. For example, the current and voltage should not exceed limit when transferring load in ring network, and power sources for same consumer cannot be cut at the same time. Even more, the geographical location and human resource can also influence outage plan. In sum, scheduled outage management can be treated as a multi-objective and multi-constraint problem[4,5].

The objectives for scheduled outage are:

- Avoid repeating outage arrangement for same line;
- Minimize outage loss for consumers and power supply company[6];

And the constraints are:

- Current cannot exceed the limit when load is transferred;
- Voltage cannot fluctuate beyond the limit;
- Human resource and facilities are sufficient;
- Power sources for same consumer cannot be cut at the same time;

Another problem for scheduled outage management is such factors are always changing. For example, the power source for an important consumer could be changed due to grid construction, the time and location of an important conference could be rearranged, and in both these cases, the outage plan should be modified accordingly. Such situation asks the objectives and concerns to be more flexible and configurable.

All the factors which are affecting outage plan can be described as business rules for scheduled outage. Rule engine can describe such business rules by pre-defined language, separate such rules from functional code, and execute them when trigger condition is satisfied. There are several Business Rule Management[7,8] systems in rule engine market, which implements Rete[9,10] or LEAPS[11] algorithm. The application of rule engine in scheduled outage management can increase the adaptability of outage plan to business variation, decrease the coupling of business logic and functional code, and optimize the outage plan effectively.

II. SORM ARCHITECTURE

Enterprise can benefit a lot when using business rules in information system. Business rules can significantly reduce the cost of requirement analysis, business logic development, integration testing and deployment. The development cycle becomes much shorter to adapt the rapid change of business logics. The developer can pay more attention to business rules by ignoring the function implementation, as the business rules are described close to the natural language, easy to be comprehended and maintained.

Considering the requirement for scheduled outage management, we designed Scheduled Outage Rules Manager (short for SORM). SORM is interoperating with application work flow component very closely. Figure 1 shows the architecture of SORM.

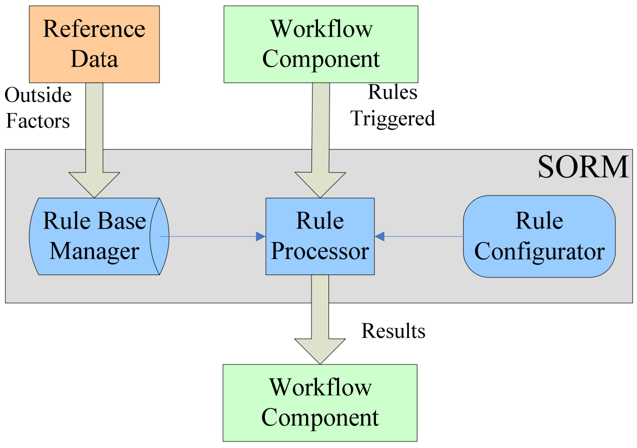


Figure 1. SORM Architecture

SORM includes 3 major parts: **Rule Base Manager, Rule Configurator,** and **Rule Processor**. The **Reference Data** provides outside factors information which can affect outage arrangement.

The whole business rules processing is independent of the **Workflow Component**. The Workflow Component invokes the API of SORM, and then SORM queries out the rule sets for the client and runs the rules. Afterwards, SORM returns the result to the Workflow Component.

III. Rule Base Manager

Rule Base Manager is the administration of scheduled outage business rules. It contains 3 parts: Rule Base, Rule Compiler and executable Binary Rule Base. Rule Base Manager Architecture is demonstrated in Figure 2.

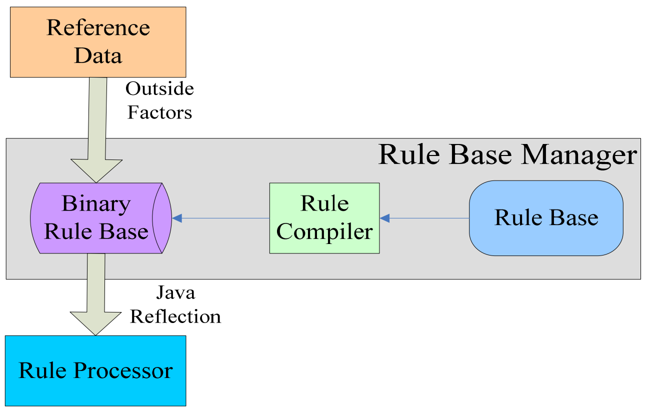


Figure 2. Rule Base Manager Architecture

Rule Compiler translates the business rules described in Rule Base into binary Java class files and saves them into Binary Rule Base. Rule Processor can call the executable rules in Binary Rule Base through Java reflection mechanism.

A. Rule Base

The business rules are saved in plain text, organized by the different concerns and constraints in outage plan. The grammar for such business rule is very simple: every rule has two parts - Predicate and Action, using IF and THEN-ELSE key words to express. One rule can invoke another rule in order to improve rule reusability. The Basic Rules contain only the predication for one or more context attributes and the action, For example, BASIC_Rule1_CalculateAttr3 is called a Basic Rule. Basic Rules are with the smallest granularity, also they the most common unit in the business rule reusing. See the following sample rule set:

```
OBJECT OutageObject IRules {
        bool attr1;
        bool attr2;
        bool attr3;
        String errorCode;
        void  setError();
}
RULE BASIC_Rule1_CalculateAttr3 {
        VALUEPROPERTY vp;
        IF (OutageObject.attr1 ==  true &&
              OutageObject.attr2 == true )
        THEN {
             OutageObject.attr3 = true;
         }
         ELSE {
        OutageObject.attr3 = false;
         }
}
RULE BASIC_Rule2_ValidateAttr3 {
        VALUEPROPERTY vp;
        IF (OutageObject.attr3 != null )
        THEN {}
        ELSE {
             OutageObject.setError();
```

```
            OutageObject.errorCode = “ErrorCode1”;
        }
}
RULECP RuleCP_ClaculateAttri3 {
        VALUEPROPERTY vp;
        IF (BASIC_Rule1_ ClaculateAttri3.vp==true &&
           BASIC_Rule2_ValidateAttr3.vp == false)
        THEN{}
}
```

It is mandatory to use key word OBJECT to declare all the variables at the beginning, OBJECT means the constrain information for the outage plan. OutageObject is a hash map essentially, it implements the IRules interface (this interface defines all the basic methods for running the rules). In OutageObject, the attr1, attr2 are the attributes corresponding to the outage constrain information. For example, the classification of a consumer, or the power supply of a certain area can both be described as an attribute of an OutageObject.

The definition for each rule starts with the key word RULE or RULECP, and the specific rule name follows them. RULE represents a Basic Rule, whereas RULECP means a compound rule, which contains one or more Basic rules in its declaration.

VALUEPROPERTY vp defines the return value of a rule. The expression after the key word IF is the rules predicate, it supports any combination of logic operators. If the result of IF expression is true, vp is assigned true value implicitly, the actions in THEN are fired, otherwise, vp is assigned false value implicitly, and the actions in ELSE are fired. If there is nothing to do in ELSE, it can be omitted.

B. *Rule Compiler*

Rule Compiler translates plain text rules into binary executable rules. There are three steps while compiling.

First, it translates the text rules into Java file temporarily. Second, it calls Java compiler to compile this temporary Java file to generate binary Java class. At last, it saves the binary Java class into database.

C. *Binary Rule Base*

After compiling, the business rules have been turned into binary rules and they are saved in database, as the result, the Binary Rule Base is created. The Rule Processor can directly execute these binary rules through Java reflection mechanism.

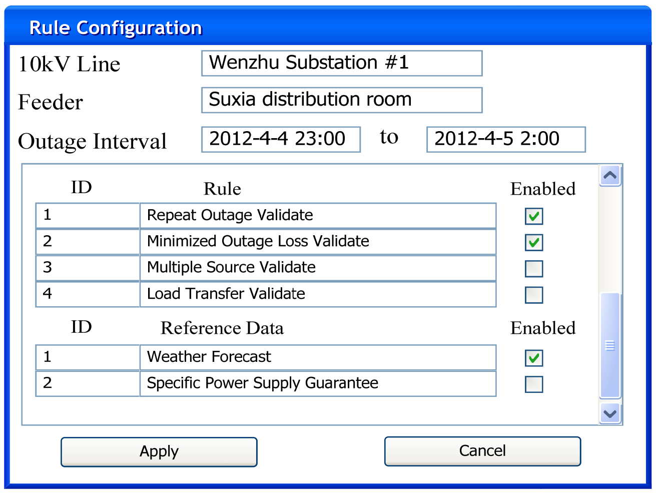
D. *Reference Data*

In rule processing, it needs to access the Reference Data to get the outside factors information which can affect an outage plan. For example, an important conference would require specific power supply guarantee, which needs all the power source available. Another typical Reference Data is weather condition, because storm and rain would affect overhead line maintenance work. Such outside factors would be defined as Reference Data to validate and optimize scheduled outage.

SORM uses Web Services to communicate with Reference Data. SORM defines an abstract class DataAccess, which offers some methods to access the interfaces of RDS. The subclasses of DataAccess define the concrete methods for their special requirements. But in rule processing, rules do not call these DataAccess classes to access the RDS. Although the rule grammar allows calling outside Java method, it would make the rules tight coupling with outside component. So SORM defines a ReferenceDataObject class, which implements the IRules interface. This object will be imported as a parameter with context information when running the rules. While the rules need to access the RDS, they use ReferenceDataObject to call RDS interface directly.

IV. RULE CONFIGURATOR

As mentioned before, scheduled outage management can be treated as a multi-objective and multi-constraint problem, and all these objectives and constraints can be described as a business rule. As the result, we need rule configurator to manage such rules in order to make outage plan adaptable and optimized.



10kV Line	Wenzhu Substation #1
Feeder	Suxia distribution room
Outage Interval	2012-4-4 23:00 to 2012-4-5 2:00

ID	Rule	Enabled
1	Repeat Outage Validate	✓
2	Minimized Outage Loss Validate	✓
3	Multiple Source Validate	☐
4	Load Transfer Validate	☐

ID	Reference Data	Enabled
1	Weather Forecast	✓
2	Specific Power Supply Guarantee	☐

Figure 3. Rule Configurator

Figure 3 is a demo UI for rule configurator, User can choose line and interval for a single scheduled outage event, and apply specific business rules. All rules are predefined and optional. For example, in Figure 3, Suxia distribution room would be arranged an outage to maintain breaker and transformer, the outage time are scheduled during 23:00 to 2:00 the day after, and there are 2 rules applied to validate and optimize this outage: Rule 1 would check whether there are outage plan for the same line in the following 3 months and whether the substation would arrange outage for maintenance; Rule 2 would check the outage interval and time to minimize outage loss; Rule 3 and Rule 4 are not enabled because the load of Suxia distribution cannot be transferred due to grid structure. At the same time, reference data describing outside factors could also be applied to a scheduled outage event.

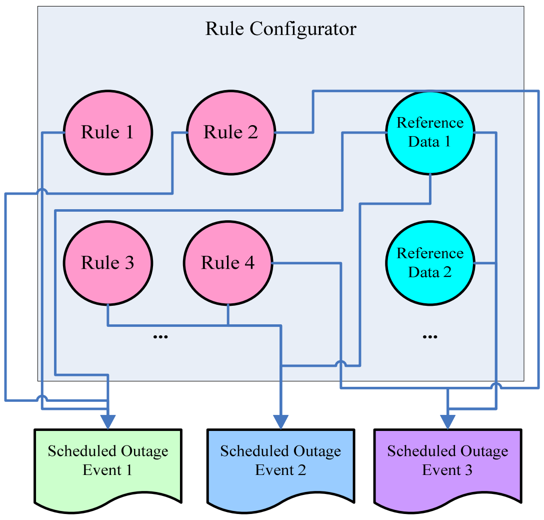


Figure 4. Rule – Outage relation

Rules are independent from outage events: a rule can be applied to various outage events and a scheduled outage can use several suitable rules. Figure 4 demonstrates the role of rule configurator and the relation between rule and outage. Rule configurator extracts scheduled outage validation and optimization rules from every discrete scheduled outage event, and applies such rules flexibly.

V. Rule Processor

Rule processor offers an interface to workflow component to call the binary rule sets. There are five main processing steps:

- Input scheduled outage information;
- Load rule configuration information;
- Load the enabled binary rules;
- Get ReferenceDataObject;
- Execute the enabled rules.

First, scheduled outage information should be set. Scheduled outage information must include substation, 10kV line, feeder, opened switch, outage start time and outage end time. If the load can be transferred, the load transfer information is also necessary, which includes closed switch, transfer line, transfer substation.

Second, appropriate rules and reference data are configured for this scheduled outage. Enabled binary rules and reference data object are loaded. Rule processor uses Java reflection mechanism in place of directly calling. Java reflection mechanism allows program using Reflection APIs to get the internal information of a known class in runtime.

At last, enabled rules are executed to validate the scheduled outage event. Scheduled outage which passed the rule validation can be published as outage plan.

VI. Conclusion

In this paper, we proposed a solution of rule engine for scheduled outage management. Scheduled outage management can be treated as a multi-objective and multi-constraint problem, and all these objectives and constraints can be described as a business rule. We implemented SORM to extract such rules from single outage event into rule language, compile them into binary rules and process such rules in outage plan work out. SORM also can administrate the business rules dynamically.

References

[1] Chisholm M. “How to Build a Business Rules Engine” Morgan Kaufmann, 2003

[2] Tao Xiaojun, Zhu Min. “Pattern of Building Enterprise Services Based on Rule Engine”. Computer Technology And Development, China. Vol.18, No.2, Feb. 2008.

[3] Brethaue G, Gamalea T, Handschin E, et al. Integrated maintenance scheduling system for electrical energy systems. IEEE Transations on Power Delivery, 1998, 13(2), pp.655-660.

[4] Janjic A.D, Popovic D S. Selective maintenance schedule of distribution networks based on risk management approach[J]. IEEE Transactions on Power Systems, 2007, 22(2), pp.597-604.

[5] Sawa T, Furukawa T, Nomoto M, et al. Automatic scheduling method using Tabu search for maintenance outage tasks of transmission and substation system with network constraints, Proceedings of Power Engineering Society 1999 Winter Meeting, New York(NY, USA), 1999, pp.895-900.

[6] Rolim J G, da Silva Filho C R. An intelligent tool for maintenance scheduling of distribution systems. International Conference on Electric Utility Deregulation and Restructuring and Power Technologies, 2000 Proceedings DRPT 2000, pp.215-220.

[7] Xiaoming Feng, Mani Subramanian. “Incorporating Business Rule Engine Technology in Control Center Applications”. IEEE Energy2030 Atlanta, GA USA, pp.17-18, November, 2008.

[8] http://en.wikipedia.org/wiki/BRMS

[9] Production Matching for Large Learning Systems. Robert B. Doorenbos. January 31, 1995. CMU-CS-95-113.

[10] Charles L. Forgy. “Rete: A fast algorithm for the many pattern/many object pattern match problem”. Artificial Intelligence. 1982. Volume 19, Issue 1, pp.17-37.

[11] Don Batory. “The LEAPS Algorithms”. Technical Report 94-28 of Department of Computer Sciences, University of Texas at Austin. 1994.