BUSINESS RULE ENGINE-BASED FRAMEWORK FOR SaaS APPLICATION DEVELOPMENT

Zhang Xiuwei[1,2,3], He Keqing[1], Wang Jian[1], Wang Chong[1] and Li Zheng[1]
[1]*State Key Lab of Software Engineering, Wuhan University, Wuhan, China*
[2]*Computer School, Wuhan University, Wuhan, China*
[3]*94005 Troops of PLA, Jiuquan, China*

Keywords: Business Rule Engine, SaaS, Multi-tenancy, Personalized Customization.

Abstract: Software as a Service (SaaS) is becoming a new direction of software industry in the new cloud computing era. In order to satisfy business policy changes and personalized requirements from different tenants in SaaS applications, business rule management must support multi-tenancy and online customization. This paper proposed a framework based on business rule engine, decoupling of business logic rule from SaaS application, which makes SaaS application more robust and maintainable. It takes business rule as an independent and online maintainable part of SaaS application, which could allow tenants to safely upgrade, delete or create rules during runtime. Finally, a practical case study of absence approval-process in attendance management system evaluates the effectiveness of the framework.

1 INTRODUCTION

With the emergence of Cloud Computing and maturity of Service Oriented Architecture (SOA), the SaaS delivery model has gained popularity, due to advantages such as lower start-up cost and reduced time to market. A SaaS vendor owns and takes the responsibility of maintaining a single application for multiple tenants who may have similar but also varying requirements (Kapuruge, et al., 2011). The service vendor delivers software functionalities with one single instance software application running for all of its tenants. The most ideal case for SaaS vendors is that every tenant feel comfortable using a completely standardize offering. However this ideal case usually does not happen in enterprise software application area. Normally, this one instance is used by different tenants having different personalized requirements in terms of data, process rules, and business rules (Kwok, et al., 2008). Typically, business data and logic integrate with other functionalities. In SaaS application, configurability, multi-tenancy and scalability are the three key attributes to evaluate the maturity of SaaS application.

In today's business environment of relentless change, software configurability is inevitable since changes generated by business policies and operations need to be propagated onto the support software application. A software system is directly related to the business system within which it operates and is thus a manifestation of some business requirements for operational control and support of decision making (Wan-Kadir and Loucopoulos, 2003). Typically, business rules have been bundled in program code or in database structures, so it is very hard to upgrade. No matter in large enterprises or small and medium enterprises (SME), the business rules change very fast and need to be adjusted timely. Business Rule Group (BRG) believes that rules are a first-class citizen of the requirements world. Traditional information management systems for business process configuration are not easy to maintain and are difficult to expand. This problem becomes increasingly prominent. There may even be a situation where business rules changes can cause the entire system to change (Liu, et al., 2010).

In SaaS application, many tenants are running on one instance with the availability of 24*7. It is unimaginable to modify or upgrade the business rule for one tenant by suspending the whole application. Meanwhile, it is a horrible disaster for one tenant to modify its own business rules and affect other tenants' rules. In order to dealing with this kind of

Xiuwei Z., Keqing H., Jian W., Chong W. and Zheng L..
BUSINESS RULE ENGINE-BASED FRAMEWORK FOR SaaS APPLICATION DEVELOPMENT.
DOI: 10.5220/0003919603450354
In *Proceedings of the 2nd International Conference on Cloud Computing and Services Science* (CLOSER-2012), pages 345-354
ISBN: 978-989-8565-05-1

situation, the SaaS application's business rules need to be customized with a flexible method, which enables organizations of any size to build, execute, manage, and evolve its own rule-oriented applications. A rule engine can be viewed as a sophisticated interpreter of if-then statements. It can reach a conclusion from a set of facts feed into it and trigger an appropriate action. So we can use the characteristic of rule engine to separate the business logic out of the SaaS application to support online customization and multi-tenancy with the isolated rule file with specific tenant who has personal customization of business rules. Therefore rule independency and isolation is an essential part in the development of SaaS application. In this paper, a business rule engine-based framework was proposed to help the development of SaaS application with the personalized customization of business rules, which is convenient for tenants to change the business rules on-the-fly and minimize the downtime of the application during the business rule upgrading or modification.

In this paper, we only focus on the business rule's online customization and multi-tenancy support, other parts customization like process data, UI elements, localization, performance monitor and so on are out of the scope. The next section identifies the related work and section 3 provides a clear and concise description of the background. Section 4 demonstrates our framework and provides explanation for our framework. Section 5 presents the implantation representing our case study and is used to exemplify the potential of our approach. Section 6 draws conclusions from our work and identifies the possibilities for future work.

2 RELATED WORK

Business rule customization of software is not a new issue. And many researchers have done a lot in traditional applications. Initially, rule based software tools originate from work carried out in the artificial intelligence (AI) research community. Companies were faced with the need to combine domain expertise with the flexibility to write lots of "if x, then y" statements over a wide range of variables without resorting to spaghetti code (Gichahi, 2003). Orriens (2003) and Vasilecas (2009) have two main views in dynamic business rule driven software system design. One of them is to design predefined executable processes and execute them by using rules in software system, where processes and execution rules are derived from business rules using transformations. And another one discussed in the work (Vasilecas, 2009), where business rules and facts describing current business system state are loaded into inference engine of the software system and transformed into software system executable data analysis process according to the results of logical derivations. Computer scientists and programmers began developing rule languages and the corresponding engines that could handle the conditions and actions needed to satisfy the wide range of rules. The most successful approach for doing this has proven to be the Rete algorithm (Forgy, 1982). Many rule-engine tools and application development support environments was applied like Blaze Advisor Builder, BRS RuleTrack, Business Rule Studio, Haley Technologies, ILOG Rules, Platinum Aion, etc (Karami and Iijima, 2010).

In SaaS applications, there are still lots of differences in business rule customization with traditional applications. These differences include:

- The business rule customization or configuration for SaaS applications should support multitenant architecture and each tenant should have their own customization.
- Not to affect other tenants, SaaS providers could not suspend the system when some tenant wants to modify or upgrading the business rules.
- The customization will be executed by administrator of tenant, not by developers of SaaS provider.
- The customization of the business rules should be simplified and friendly.

Above mentioned differences between SaaS applications and traditional software have raised many researches in this new area. Guo (2007) proposed a multi-tenant supported framework to support better isolations among tenants in many aspects such as security, performance, availability, administration etc. Zhang (2007) proposed a SaaS-oriented service customization approach, which allows service vendors to publish customization policies along with services. If tenant's customization requirement is in agreement with policy after being verified, vendors will update service accordingly. This approach will inevitably burden service providers because of tenants' reasonable customization requirement increments. Gong (2009) developed ECA process orchestration architecture to create flexible processes. This architecture based on both knowledge rules (separating knowledge from processes) and event-condition-actions (ECA) mechanisms to provide the highest level of flexibility. Configurability of SaaS

issue was addressed in Ref (Nitu, 2009) who researched the configurability like user interface, workflow, data and access control from the different aspect of SaaS. From the customization and configuration perspective, Sun (2009) explored the configuration and customization issues and challenges to SaaS vendors, clarifies the difference between configuration and customization. A competency model and framework has been developed to help SaaS vendors to plan and evaluate their capabilities and strategies for service configuration and customization. In the Ref (Shi, et al., 2009), a flexible business process customization framework for SaaS was proposed to solve problems caused by orchestrating SaaS business process through BPEL specifications. Kapuruge (2011) discussed the challenges arising from single-instance to multi-tenancy, and presented an approach of Serendip4SaaS to define business processes in SaaS applications.

To the best of our knowledge, no related work has combined the rule engine and decision table with the SaaS application for multi-tenancy support and online customization. Compared to them, our work was focused on the perspective of business rule customization and configuration. In our framework, each tenant can update their personalized business rule in SaaS application by online selecting and modifying corresponding rules. Rule engine was utilized as the essential part to improve the flexibility and multi-tenancy for SaaS application, which makes business rule as an independent and maintainable part of application.

3 BACKGROUND

3.1 Business Rule Engine

In business, a lot of actions are triggered by rules: “Order more ice-cream when the stock is below 100 units and temperature is above 25° C”, “Approve credit card application when the credit background check is OK, past relationship with the customer is profitable, and identity is confirmed”, and so on. Traditional computer programming languages make it difficult to translate this “natural language” into a software program. Business rule engine enables anybody with basic IT skills and an understanding of the business to turn statements as running computer code (Browne, 2009). A business rules engine is a software system that executes one or more business rules in a runtime production environment. It will test data objects quickly in the workspace, pick out rules which meet requirement from loading rule sets, and generate an instance of rule execution. Figure 1 shows the architecture of rules engine. Pattern matcher decides which and when rules will be implemented. The implementation sequence of rules picked from pattern matcher is arranged in agenda so that execution engine can execute the rules or other actions in order. The underlying idea of a rule engine is to externalize the business or application logic. Business rules are expressions that describe and control the processes, operations and behaviour of how an enterprise, and the applications that support it, performs. Rules assert influence over business or system behaviour by recommending actions to be undertaken. A rule provides its invoker a directive on how to proceed. Further, rule policies provide a generalized mechanism for specifying frequently changing practices, freeing system components from the burden of maintaining and evaluating evolving business and system environments (Jeng, et al. 2004).

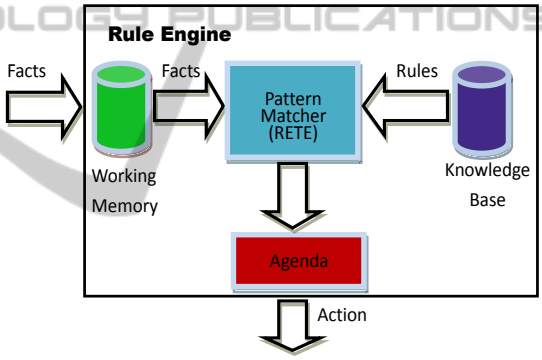

Figure 1: Basic architecture of rule engine.

3.2 Decision Table

A decision table is a tabular representation used to describe and analyze decision situations, where the state of a number of conditions determines the execution of a set of actions. Many variations of the decision table concept exist which look similar at first sight (Vanthienen, 2009). Decision tables are best suited for representing business rules that have multiple conditions. Adding one condition is done by simply adding one row or column. Like the if/then rule set, the decision table is driven by the interaction of conditions and actions. The main difference is that in a decision table, the action is decided by more than one condition, and more than one action can be associated with each set of conditions. If the conditions are met, then the corresponding action or actions are performed (Vasilecas, 2006). A column in the entry portion of

the table is known as a rule. Values which are in the condition entry columns are known as inputs and values inside the action entry portions are known as outputs. Outputs are calculated depending on the inputs and specification of the program. Figure 2 depicts the basic principle of the decision table.

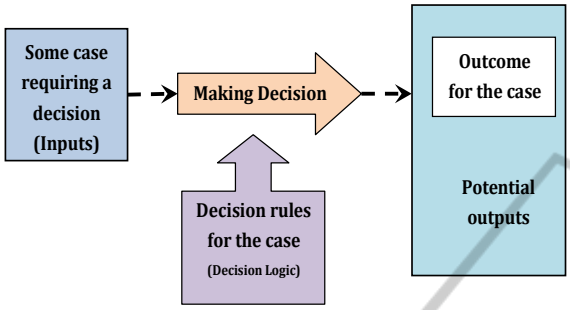

Figure 2: The basic principle diagram of decision table.

4 FRAMEWORK

The SaaS application operating on the proposed SaaS platform is one packaged business application with web-based user interface to multiple tenants. Based on the features of business rule engine, we design and implement a framework for development of SaaS application with an online business rule customization. The architecture of framework is shown as Figure 3. The essence of this framework is to separate business logics and business rule, and make the business rule as an independent and maintainable part, support multi-tenancy. The objective of this framework is to reach a flexible and competitive scenario in which it would be easier and faster to react when changes in demand or business appear.

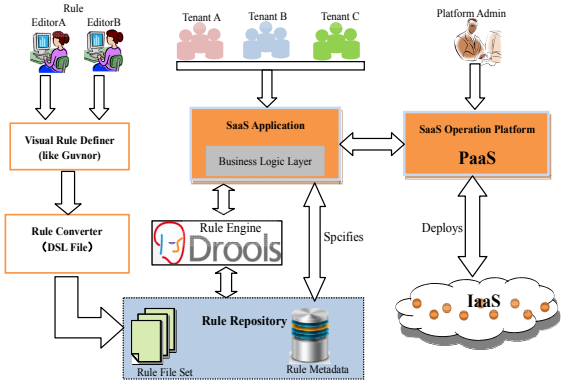

Figure 3: The business rule engine-based framework.

4.1 Basic Units of the Framework

The proposed framework includes the following major interrelated parts: BR definer, BR engine, BR repository, SaaS application and SaaS deployment system.

BR definer acts as a web-based tool or sub-system that helps visually manage and create new business rules, where the business policy can be changed online by manager, business analysts, and software developers.

BR Converter is an essential auxiliary tool of rule engine and responsible for convert the visualized rule from definer to BR engine understandable language. It also can translate the decision table to a specific executable language.

BR engine is a central component which is responsible for computation and evaluation of the business rules according to the user's invocation and request. It can automatically assert the business rules for specific tenant according to the rule load metadata from repository.

BR repository is a repository that stores the rule-related information and supports the flexibility of rule expression. This component contains two major parts, rule files set is used to store the information of business rules including decision table, "When...Then" based rule file, and DSL file (Domain Specific Language) and so on. The stored business rules in the repository are determined based on the target system's specifications. Rule Metadata is another important part of repository which includes the tenant customization and configuration information for specific tenants. Metadata was stored in repository as management information to support multi-tenancy.

SaaS application includes basic functionalities and business logic layer. And we have separated the business policy out of code and as an independent part for upgrading and modification.

SaaS deployment system includes SaaS operation platform (Platform as a service) and IaaS (Infrastructure as a Service). In SaaS platform, administrator will be responsible for management and deployment of SaaS application. IaaS as a basic part for SaaS deployment including hardware and storage part and so on. We will not explain more details about the SaaS deployment system because this article is focused on the connection between Business Rule Management(BRM) and SaaS application.

Rule editors can configure various business rules in terms of workflow, activity type, and business policy by using the Rule Definer tool. Tenant's

business rule configured information is stored separately in tenant-specific metadata repository. Rule engine-based framework generates polymorphic service for individual tenant using tenant-specific metadata at runtime. Through the polymorphic service, tenant users feel as if they are using their own business application while service instance is shared by every tenant.

4.2 Capabilities of the Framework

SaaS application based on this framework will be supported with the following capabilities, which also was the basic requirement for SaaS application development.

Support of Business Rules Management
Enterprises run their businesses with repeatable business processes driven by general business rules for specific situations and customers. These capabilities allow enterprises to execute business functionality using independent rule services made up of executable, declarative rules, rather than being forced to integrate the logic as code into a system.

Support of Online Maintenance
Current Enterprise applications require a new application maintenance paradigm that can deliver faster, easier application modification. Business rule changes are first identified by the users of the system. The fastest and safest way to empower these users is to give them the tools they need to make the application changes themselves. This can be achieved by giving them access to easy-to-use rule maintenance that allows them to maintain the policies, procedures and rules for which they are responsible.

Support of Multi-tenancy Customization
As the number of tenants with subscribed SaaS application grows, specific personalized business rules are needed for most tenants. In this framework, we bind Tenant ID with rule files and store the metadata in repository. In order to support multi-tenancy, the most important part is the safety of specific rules with specific tenants. In this framework, the metadata of rules resolved this problem.

4.3 Lifecycle of Business Rules in SaaS

In business world, some rule policies are changed periodic and others are altered disorderly depending on market competition and development. Rule life-cycle in SaaS is illustrated in figure 4 including rules creation, edition, activation, deletion, et al. The whole lifecycle of business rule should be considered in SaaS development.

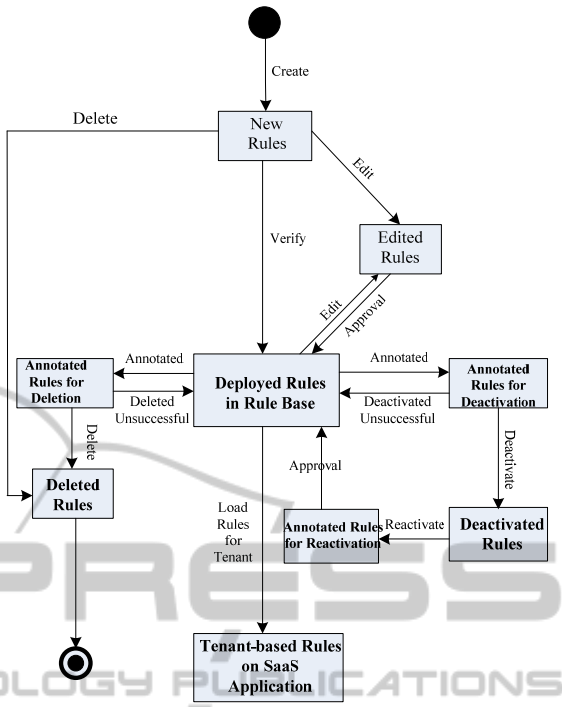

Figure 4: Lifecycle of business rules.

Rules Creation
The creation of rules is done by rule editors. A new rule is available for editing but has to be approved for deployment.

Rules Edition
Rule edition is the modification of the condition part or the action part of a rule. To keep track of changes, only new or deployed rules can be edited. Deactivated rules must be reactivated before they can be modified.

Rules Deactivation and Reactivation
A rule can be manually or automatically deactivated. For example, a rule is automatically deactivated on 1 January 2011, if it is time constrained to function between 01 January 2008 and 31 December 2010. An editor may manually deactivate a rule especially when the regulatory changes. Rule editor may reactivate a manually deactivated rule.

Rules Deletion
Rules that are no longer in use in the system can be removed from the system by deletion.

Rules Deployment
Rules are deployed into the repository will be reacted immediately by making a snapshot of isolation for the deployed rules in SaaS application.

5 CASE STUDY

5.1 Motivation

In order to evaluate the proposed framework, we will illustrate a business rule online customization process via an example. We take Attendance Management System (AMS) as the domain we do experiment. AMS is an easy way to keep track of attendance for enterprises, school activities, church groups, and community organizations. It has become as the necessity application for workforce performance monitoring and evaluation. The objective of this case was developing a multi-tenancy supported AMS application with the online customization. In order to show variation of business rule for specific tenant, we demonstrate a roadmap of rule policy from elicitation, presentation to implementation by the process of absence approval for sickness in AMS.

In most enterprises, the approval process for employee who applied for the absence of sickness, personal reason or salary holiday has different rules. Here we show a simplified absence of sickness approval process in AMS as a case to show the variation of rules for different tenants. The approval process of absence policy for sickness depends on the absence days and other conditions like total absence days in month, total absence days in year, duration time and so on.

A simplified approval process depending only on condition of absence days is depicted on Figure 5. The whole approval process divides into four situations, if the absence days not exceed the Level-1's limit. Only Level-1 approval is needed. If the absence days over Level-1 and located in the Level-2's scope. The approval process will need both Level-1 and Level-2. Normally Level-2's approval will executed after Level-1 approval passed except for emergency situation. Level-3 and Level-4's approval have the similar approval procedure.

The following italic description outline the three tenants A, B, C who has different approval process and rule policies.

Tenant A. *Absence days for sickness less than or equal one day will be approved by team leader (Level-1). From one day to five days absence will be needed both team leader and Human Resource Department approval (Level-2). And more than five days will be permitted by Manager (Level-3).*

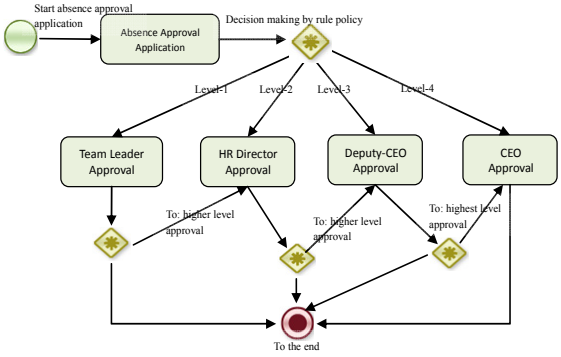

Figure 5: Absence approval process of tenant C.

Tenant B. *Absence days for sickness less than or equal two days will be approved by team leader (Level-1). And more than two days will approve by Human Resource Department (Level-2).*

Tenant C. *Absence for sick leave less than or equal one day will be approved by team leader (Level-1). From one day to five days absence will be needed both team leader and Human Resource Department approval (Level-2). From five days to ten days absence will be approved by team leader, Human Resource Department and deputy-CEO approval (Level-3). And more than ten days need to be permitted by team leader, HR director, deputy CEO and CEO (Level-4).*

5.2 Representation of Business Rule

Different enterprises have their own rule policies for absence approval like above mentioned. Here we take Tenant C's rules as a case to demonstrate how to fill these rules into decision table.

Step1, Definition of the Terms
Here we draw up a list of all condition statements and actions that are mentioned in the text. It is clear that this example only uses absence days as the condition to determine which level of approval will be executed. The following table lists all related occurrences of these terms in the text.

Table 1: Rule condition statement and action statement.

Condition Statement	Action Statement
Absence Days	*Permission level*
Absence Days <=1	*Team leader (L-1)*
1<Absence Days<=5	*HR Director(L-2)*
5<Absence Days<=10	*Deputy CEO(L-3)*
Absence Days >10	*CEO(L-4)*

Step 2, Verification of the Decision Rules
Based on the text of the regulations and conditions, the condition states and the actions, we now can proceed by defining the rules, analyzing each line in the regulation and translating it into a rule. Absence approval rule of Tenant C was built here as example.

- Absence days for sickness less than or equal one day will be approved by team leader.

  Rule 1: *Absence Days <=1*

  Action: Level-1 Approved (team leader)

- From one day to five days absence will need both team leader and Human Resource Department approval.

  Rule 2: *1< Absence Days <=5*

  Action: Level-1(team leader) and Level-2 (Human Resource Director) approval.

- From three days to ten days absence will be approved by team leader, Human Resource Department and deputy-CEO approval.

  Rule 3: *5< Absence Days <=10*

  Action: Level-1, Level-2 and Level-3(deputy-CEO) approval.

- And more than 10 days will be permitted by team leader, HR Director, deputy-CEO and CEO.

  Rule 4: *Absence Days* >=10

  Action: Level-1, Level-2, Level-3 and Level-4(CEO) approval.

Step 3, Filling the Decision Table
After specifying the decision rules, it is needed to fill them into the appropriate combinations in the decision table as depicted in Table 2. The key point to keep in mind is that in a decision table, each row is a rule, and each column in that row is either a condition or action for that rule. "※" indicates actions in the combination will be activated, and "○" means action will not activated by rules.

Table 2: Decision table for absence approval rule.

Absence Days (ADs)	<=1	1<Ads<=5	5<Ads<=10	>10
Team Leader Approval	※	※	※	※
HR Director Approval	○	※	※	※
Deputy-CEO Approval	○	○	※	※
CEO Approval	○	○	○	※

Step4, Optimization of the Rule Condition
Once a complete validation of the decision table is finished, the table could be reduced to its minimal format. The order of the conditions might influence the number of columns in the contracted table. For this case, the above condition is already the optimal one.

5.3 Implementation

In this case, we take Eclipse IDE as the development environment and java-supported rule engine Drools as BR engine. Drools introduce the business logic integration platform that provides a unified and integrated platform for rule, Workflow, and Event Processing. Drools 5 is now split into four main subprojects: (1) Guvnor (BRMS), a centralized repository for Drools; (2) expert (rule engine); (3) flow (process/workflow), providing workflow or process capabilities to the Drools platform; and (4) fusion (event processing/temporal reasoning), providing event processing capabilities (Browne, 2009).

Drools expert was used as a rule engine and Guvnor as a visual business rule definer which allow browsing and editing the rule set. Generally, decision table is a useful way to represent conditional logic in a compact format. This format is also readily readable and editable by non technical users and will be suitable for most employees to understand. Spreadsheets may not be perfect, but popular and well-understood. So we can use them to hold the data that we supply to the business rules. Then use spreadsheets to hold the actual rules in a decision table format. Drools decision tables can utilize a spreadsheet (such as Excel, CSV) as the means to capture decision logic in a user friendly way. Because of the convenience of decision table and supportability of Drools, the decision table was adopted as business rule representation style in our application. The following figure is the snapshot of the executable Drools decision table for absence approval process of Tenant C.

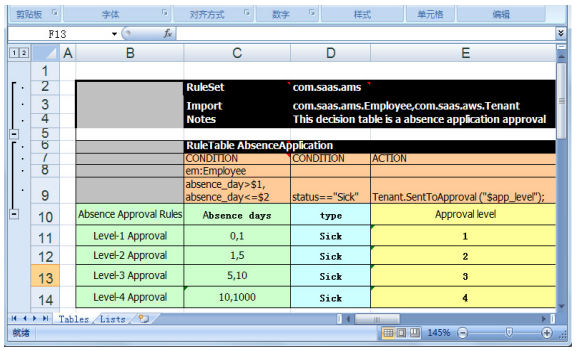

Figure 6: Snapshot of Drools based on decision table.

In this decision table, the first three lines are the head information includes RuleSet, Import and Notes. RuleSet lets Drools know where the header table begins. Import lets Drools know which package these rules live in and other imported additional JavaBeans. Notes line is the comment information and ignored as it means nothing to Drools. The following part is the main body of decision table. The left part of the decision table is the CONDITION cells, which makes up the "*WHEN*" part of the rule. The right part of the decision table is ACTION cells which give the "*THEN*" part of the rules. In Drools, the "*WHEN*" part of the rules define the preconditions. The "*THEN*" part defines conclusions, decision, actions, or just a new fact deduced from the knowledge base. The < *preconditions* > is also referred to as the left-hand side (LHS) of the rule, whereas the < *conclusions* > is referred to as the right-hand side (RHS). So, we also can express rules as follow:

LHS (< *rule name* >) = < *preconditions* >
and
RHS (< *rule name* >) = < *conclusions* >

The first row of decision table could be rendered like the following DRL rules:

```
rule  "absence approval"
when
    em(absence_days>0&&absence_days<1);
then
    Tenant.sentToApproval (Level 1);
update (em);
end;
```

The fragments code for execution of decision table is demonstrated in Figure 7. It almost has the same executable code compare with the execution of the DRL rule files.

In order to support the online customization of business rule, it is necessary to use visual rule definer. Guvnor Editor is a user-friendly web editor which is powerful enough to modify rules. Tenants

```
KnowledgeBase kbase = readKnowledgeBase();
StatefulKnowledgeSession ksession =
kbase.newStatefulKnowledgeSession();
KnowledgeRuntimeLogger logger =
KnowledgeRuntimeLoggerFactory.newFileLogger(ksession, "AMS");
Employee employee = new Employee();
Tenant tenant = new Tenant();
ksession.insert(employee);
ksession.insert(tenant);
ksession.fireAllRules();
Logger.close();
private static KnowledgeBase readKnowledgeBase() throws Exception {
        KnowledgeBuilder kbuilder =
KnowledgeBuilderFactory.newKnowledgeBuilder();
        DecisionTableConfiguration config =
KnowledgeBuilderFactory.newDecisionTableConfiguration();
        config.setInputType(DecisionTableInputType.XLS);
kbuilder.add(ResourceFactory.newClassPathResource("SickLeave.xls"),
ResourceType.DTABLE, config);
        KnowledgeBuilderErrors errors = kbuilder.getErrors();
        if (errors.size() > 0) {
            for (KnowledgeBuilderError error: errors) {
                System.err.println(error);
            }
            throw new IllegalArgumentException("Could not parse
knowledge.");
        }
        KnowledgeBase kbase =
KnowledgeBaseFactory.newKnowledgeBase();
kbase.addKnowledgePackages(kbuilder.getKnowledgePackages());
        return kbase;
    }
```

Figure 7: Fragment of Java code for implementation.

can fill in the rule name and rule description, set the priority of this rule and choose templates to define business rule in line with their requirements. The modification of decision table will need to download the decision table and modified it, then uploads it with Guvnor. Otherwise, in order to keep the isolation of business rules for different tenant, we build the tenant-based security policy on the login page with different password for different tenant to prevent the violation of the rules modification. The visual rule definer of Gnuvor is shown as Figure 8.

Figure 8: Snapshot of decision table creation in Gnuvor.

5.4 Prototype Application

The SaaS application of AWS prototype was developed followed the proposed framework which has successfully integrated business rule engine into

Figure 9: Snapshots of SaaS application of AMS.

SaaS application. Figure 9 shows a snapshot of the AMS prototype system, which was successfully deployed on the SaaS platform of RGPS Cloud Service Supermarket (Zhang et al., 2011). The testing work and performance evaluation is now ongoing and will be completed in other paper.

6 CONCLUSIONS

In this paper, we have overviewed BR engine based framework and separated three main components used for such a SaaS application development. Depending on the proposed framework, it may be possible to ensure different level of agility by an instant deployment of changes in the business policy and immediate reaction to the changes on the market or competition by changing existing business rules. Such advances allow SaaS application to be more transparent, flexible and cost reduction. Although BR engine based application have more complex development process in an initial phase, but such a system is more efficient in further maintenance and modifications for numbers of tenants with frequently changing regulations and business policy. Although the proposed approach is convenient and effective to modify the rule file and manage the requirement changes by Rule Engine, it also brings lots of extra performance consumptions. The consumptions mainly include the following parts: the time of compiling rule files, the time of rule matching and the time of rule conflict resolution and the time for management of rule metadata.

The work presented here, still in its earlier stage, on the one hand, business rule isolation for multi-tenant was not completely resolved by the Guvnor. So still need to do more work on visual definer for specific SaaS application. On the other hand, the performance evaluation work still need to be done in the future to make sure that the multi-tenant request response time to in a reasonable and tolerant time. Moreover we also need to do more experiment to verify the efficiency of the framework.

ACKNOWLEDGEMENTS

This research project was supported by the National Natural Science Foundation of China under Grant No. 60970017 and No.61100018, the Fundamental Research Funds for the Central Universities under Grant No.3101034 and No.201121102020004, the Specialized Research Fund for the Doctoral Program of Higher Education No.20090141120020.

REFERENCES

Browne P., 2009. JBoss Drools business rules. Packt publishing. Birmingham-Mumbai.

Forgy C., 1982. Rete: A Fast Algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence,19 (1982)*, pp 17–37.

Gichahi H.K.,2003. Rule-based process support for enterprise information portal. [online]Available at< http:// www.sts.tu-harburg.de/ pw-and-m-theses/ 2003/ gich03.pdf> [accessed 29 Sep 2011].

Gong YW., Janssen M., Overbeek S., et al. 2009. Enabling flexible processes by ECA orchestration architecture. *ICEGOV 09 Proceedings of the 3rd international conference on Theory and practice of electronic governance*, pp.19-26.

Guo CJ., Sun W., Huang Y., et al., A framework for native multi-tenancy application development and Management. *The 9th IEEE International Conference on E-Commerce Technology and The 4th IEEE International Conference on Enterprise Computing, E-Commerce and E-Services(CEC-EEE 2007)*, pp.551-558.

Jeng JJ., Flaxer D., Kapoor S., 2004 . RuleBAM: A rule-based framework for business activity Management. *2004 IEEE International Conference on Services Computing*, pp.262-270.

Kang SJ., Kang SW., Hur SJ. A design of the conceptual architecture for a multitenant SaaS Application Platform. *Computers, Networks, Systems and Industrial Engineering (CNSI), 2011 First ACIS/JNU International Conference*, pp.462-467.

Kapuruge M., Colman A., Han J., 2011. Achieving multi-tenanted business processes in SaaS applications. In WISE 2011, LNCS 6997, pp. 143–157.

Karami N., Iijima J., 2010. A logical approach for implementing dynamic business rules. *Contemporary Management Research Vol 6(1)*, pp. 29-52.

Kwok T., Nguyen T. N., Lam L., 2008. Software as a Service with Multi-tenancy Support for an Electronic Contract Management Application. *2008 IEEE*

International Conference on Services Computing, pp 179-186.

Liu C., Dong XP., Yang ZQ.,2010. Research of modern enterprise intelligent system based on rule engine and workflow. *2010 Intelligent Computing and Intelligent Systems (ICIS)*, pp. 594-597.

Nitu, 2009. Configurability in SaaS (software as a service) applications. *ISEC'09, Proceedings of the 2nd India software engineering conference*, pp.19-26.

Orriens B., Yang J., Papazoglou M.P., 2003. A framework for business rule driven service compostion. *Technologies for E-Services, Vol(2819)*, pp.14-27.

Shi YL., Luan S., Li QZ, et al, 2009. A flexible business process customization framework for SaaS. *ICIE 09, WASE International Conference on Information Engineering*, pp.350-353.

Sun W., Zhang X., Guo CJ.,et al. Software as a Service: Configuration and Customization Perspectives. *IEEE Congress on Services, SERVICES 2008*, pp.18-25

Vanthienen J., 2009. Ruling the business: about business rules and decision tables. [online]Available at: <http://www.econ.kuleuven.be/tew/academic/infosys/members/vthienen/download/papers/br_dt.pdf> [accessed 25 Sep 2011].

Vasilecas O.,2009. The framework for the implementation of business rules in ERP. *Informacijos mokslai,Vol (49)*, pp.146-157.

Vasilecas O., Smaizys A., 2006. Business rule based data analysis for decision support and automation. *International Conference on Computer Systems and Technologies, CompSysTech'06*. pp.191-196.

Wan-Kadir ,W.M.N., Pericles L. 2003. Relating evolving business rules to software design. *Journal of Systems Architecture 50 (2004),* pp.367–382.

Zhang K., Zhang X., Sun W., et al., 2007. A policy-driven approach for software-as-services customization. *The 9th IEEE International Conference on E-Commerce Technology and The 4th IEEE International Conference on Enterprise Computing, E-Commerce and E-Services (CEC-EEE 2007)*, pp.123-130.

Zhang XW., He KQ., et al., 2011. SaaS service super-market building model and service recommendation approach. *Journal on Communication (In Chinese), 2011, 32(9A)*, pp.158-165.