On-Demand Business Rule Management Framework for SaaS Application

Xiuwei Zhang[1,2,3], Keqing He[1], Jian Wang[1], Chong Wang[1], and Zheng Li[1]

[1] State Key Lab of Software Engineering, Wuhan University, Wuhan, China
[2] School of Computer, Wuhan University, Wuhan, China
[3] 94005 Troops of PLA, Jiuquan, China
xiuweizhang@163.com,
{hekeqing,jianwang,cwang,zhengli_hope}@whu.edu.cn

Abstract. SaaS (*Software as a Service*) is becoming a new direction of software industry in the new cloud computing era. SaaS applications and services must be able to react in a fast and flexible way to ever changing business situations, policies and products. In order to satisfy policy changes and other personalized requirements from different customers (or *tenants*), business rule management of SaaS needs to support multi-tenancy and online customization. This paper proposed a business rule engine based framework for managing and decoupling of business logic rule from SaaS application. It takes on-demand business rule management 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 in Attendance Management System (AMS) evaluates the effectiveness of the framework.

Keywords: Business Rule Engine, SaaS, Decision Table, Personalized Customization.

1 Introduction

With the emergence of Cloud Computing and maturity of Service Oriented Architecture (SOA), SaaS delivery model has gained popularity due to advantages such as lower start-up cost and reduced time to market. SaaS is the best way to adopt advanced technology and the most effective business model in the Cloud Computing era. Typically in SaaS application, configurability, multi-tenancy and scalability are the three key attributes to evaluate the maturity of SaaS application. SaaS vendor owns and takes the responsibility of maintaining a single application for multiple tenants who may have similar but also varying requirements [1]. 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, such one instance is used by different tenants with different personalized requirements in terms of data, process rules, and business rules (BR) [2]. In order to satisfy their maintainability of flexible business policy, we must decouple the close relationship between business data and business logic.

I. Ivanov et al. (Eds.): CLOSER 2012, CCIS 367, pp. 135–150, 2013.

Business Rule Group (BRG)[1] believes that rules are a first-class citizen of the requirements world. No matter in large enterprises or small and medium enterprises (SME), business rules change very fast and need to be adjusted timely. Software system is directly related to the business process within which it is a manifestation of some business requirements for operational control and support of decision making [3]. Nevertheless, many business rules have been bundled in program code or in database structures, so it is very hard to upgrade and expand [4]. For SaaS application, this problem becomes increasingly prominent because different tenant has different rule variation. Many tenants are running on one instance with the availability of 24*7. There may even be a situation where one tenant business rule changes may affect other tenants and even cause the entire system to change. In order to dealing with this kind of situation, the business rules of SaaS application need to be customized in a flexible way, which enables any tenants to build, execute, manage, and evolve its own rule-oriented applications. Rule engines allow the separation of business rules from the applications that use them and enable the maintenance of business logic without having to resort to code changes and software modification. 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 business rule engine to separate the business logic out of the SaaS application to support online customization and multi-tenancy with the isolated rule file. Each tenant can individually configure and upgrade his own 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 management of business rule for SaaS application, which is convenient for tenants to change business rules on-the-fly and minimize the downtime of the application during the business rule upgrading or modification. Tenants with non-IT profession can on-line update business rule in a simple spreadsheet and deploy them with a few clicks. It makes SaaS application more robust and maintainable.

In this paper, we only focus on the business rule's online customization and multi-tenancy support. 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 management of software system is not a new issue. 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

[1] http://www.businessrulesgroup.org/home-brg.shtml

write lots of "*if x, then y*" statements over a wide range of variables without resorting to spaghetti code [5]. Orriens [6] and Vasilecas [8] 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. Another one is discussed in work [7], 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 [9]. Many rule-engine tools and application development support environments was applied such as Blaze Advisor Builder, BRS RuleTrack, Business Rule Studio, Haley Technologies, ILOG Rules, Platinum Aion, etc [10].

In SaaS application, there is 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 multi-tenant architecture and each tenant should have their own rule customization.
- Not to affect other tenants, SaaS providers could not suspend the system when some tenants want to modify or upgrade the business rules.
- The rule customization will be executed by administrator of tenant, not by developers of SaaS provider.
- The business rule should be support Web-based online modification.
- The customization of the business rules should be simplified and friendly.

The above differences between SaaS applications and traditional software have raised many researches in this new area. Guo [11] proposed a multi-tenant supported framework to support better isolations among tenants in many aspects such as security, performance, availability, administration, etc. Zhang [12] 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 [13] 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 literature [14] who researched the configurability like user interface, workflow, data and access control from the different aspects of SaaS. From the customization and configuration perspective, Sun [15] 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 literature [16], a flexible business process customization framework for SaaS was proposed to solve problems caused by orchestrating SaaS business process through BPEL specifications. Kapuruge [1] 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. 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 translate statements as running computer code [17]. Business rule 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.

Fig. 1 shows the basic architecture of business rule 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 behaviors of how an enterprise, and the applications that support it, performs. Rules assert influence over business or system behavior by recommending actions to be undertaken. A rule provides its invoker a directive on how to proceed. Further, business 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 [18].

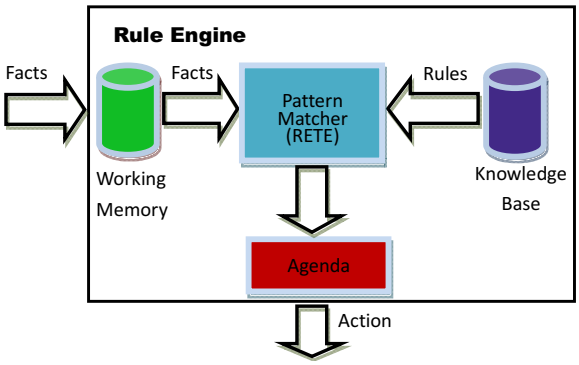

Fig. 1. The architecture of rule engine

3.2 Decision Table

Decision table is a tabular representation used to describe and analyze decision situations, where the states of a number of conditions determine the execution of a set of actions. Many variations of the decision table concept exist which look similar at first sight [8]. 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. Similar to 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 [19]. A column in the entry portion of the table is known as a rule. Values 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. Fig. 2 depicts the basic principle of the decision table. It uses available information on frequency of outcome of the various cases and whether core minimization or run time minimization is the more important. A further development in programming languages will be to hand this information along with the decision table to a compiler which will then be responsible for this. Thus decision tables not only offer a clearer way of stating the logic of a program but also provide the notational means of extending the scope of automatic programming [23].

4 Rule Engine Based Framework for SaaS Application

The SaaS application is one packaged business application with Web-based user interface for multiple tenants operating on the SaaS platform. With the increase in complexity of SaaS application, business rules have become harder to express hence require additional simple and friendly way to represent. 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 direct customization of

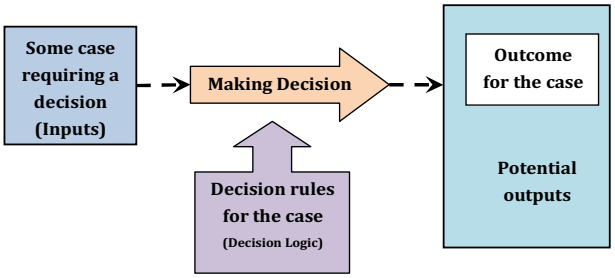

Fig. 2. The basic principle of decision table

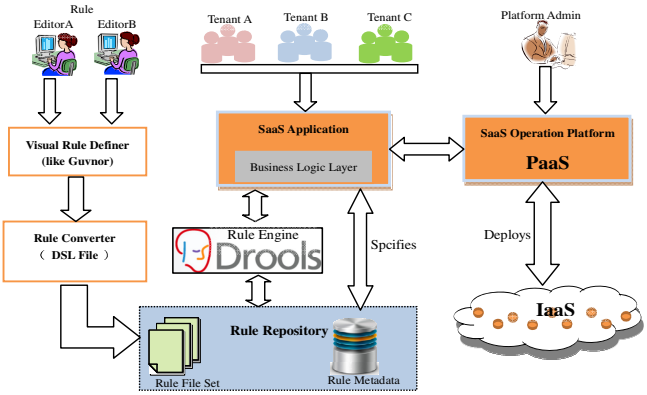

Fig. 3. The business rule engine-based framework

business rules by tenants is one of our objectives since it relieves, in many cases, the SaaS providers from doing such heavy customization tasks each time when a new tenant subscribe to the application [21]. The architecture of the proposed framework is shown in Fig. 3. The essence of this framework is to separate business rules from application, and make the business rules management as an independent and maintainable part, to 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 demand or business changes.

4.1 Basic Units of the Framework

The biggest challenge of business rules management is tracking them down, and organizing a more effective management approach. In each case there is need for business rules management. Business rules management comprises the definition, storage, and application of the many rules used in business operations to provide organizations with greater automation, more responsiveness to change and less expensive distribution and maintenance of their business guidelines [5]. Rules management offers a solution to meet the requirements of changing business rules. The proposed framework includes the following major interrelated parts: BR definer, BR Converter, BR engine, BR repository, SaaS application and SaaS deployment system.

- **The 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.
- **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 tenant 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. A rule repository is a central place where managers, analysts, and software developers can define, share, and maintain the business rules of a company. This component contains two major parts: rule set and rule metadata. The former is used to store the information of business rules including decision table, "When...Then" based rule file, and DSL (Domain Specific Language) file and so on. The stored business rules in the repository are determined based on the target system's specifications. While the latter mainly includes the tenant customization and configuration information for specific tenants. Metadata is stored in the repository as management information to support multi-tenancy.
- **SaaS application** includes basic functionalities and business logic layer. We have separated the business policy out of code and take it 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 operation 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 paper focus on the relationship between Business Rule Management (BRM) and SaaS application.

4.2 Capability of the Framework

SaaS application based on this framework will be supported with the following capabilities, which also are the basic features of SaaS application.

- **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 each Tenant ID with the corresponding 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 are used to resolve 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. A good rules management system allows the business logic of a system to be specified external to the system itself. Rules can be changed directly by rule maintainers and editors. Many rules management system provide the whole lifecycle management from designing rules, deleting rules to editing and deployment of rules. The business rule lifecycle of SaaS including rules creation, edition, activation, deletion, etc, is illustrated in Fig. 4

- **Rules Creation.** The creation of business rules is done by rule editors. A new rule is available for editing and deleting. Only approved new rules can be deployed.
- **Rules Edition.** Rule edition is the modification of the condition part or the action part of a rule. To keep track of rule 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 its time is constrained to function between 01 January 2008 and 31 December 2010. An editor may manually deactivate a rule especially when the regulatory or policy changes. Rule editor may also reactivate a manually deactivated rule as they needed.
- **Rules Deletion.** Rules that are no longer in use in the system can be removed from the system by deletion. New rules and annotated unused rules can also be deleted.
  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. The deployment process of business rules includes at least the following steps [22]: (1) extract the rules in scope for the execution; (2) package the rule elements into a ruleset –a deployable artifact; (3) deploy the ruleset to the

target environment; (4) notify the engine of a new ruleset; (5) let management stack inside the rule execution environment loading the ruleset; (6) trigger the engine API to parse the ruleset; and (7) send business transactions to fire the rules.

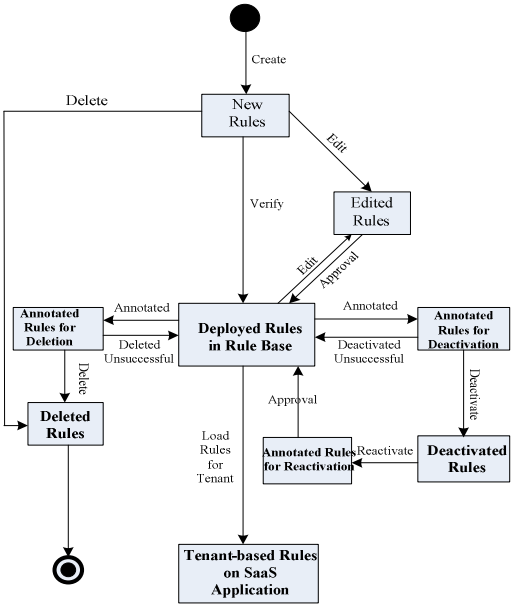

Fig. 4. Lifecycle of business rules in SaaS application (based on [5])

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 is to develop a multi-tenancy supported AMS application with 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. The Process of Absence Approval enables employees to enter absence requests in the system. The request passes through an approval procedure in which the request is checked by employee's superiors to see if the employee's absence can be approved according to company rules. In most enterprises, the approval process for employee

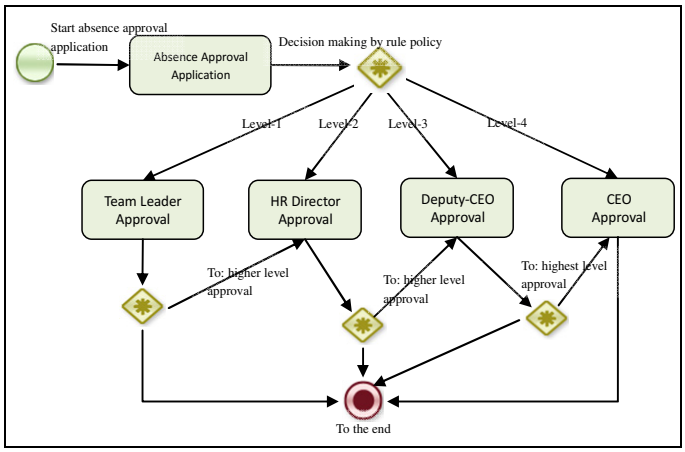

Fig. 5. Absence approval process of tenant C

who applies for the absence of sickness, personal reason or salary holiday has different rules. Here we take 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 such as 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 Fig.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 locate 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 different approval process and rule policies of three tenants A, B, C respectively.

- **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).*
- **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 as mentioned above. Here we take Tenant C's rules as a case to demonstrate how to fill these rules into a decision table. The concrete steps are described as follows.

— Step1, Definition of the Terms

Here we draw up a list of all condition statements and actions that are mentioned above. 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 above context.

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, now we can proceed by defining the rules, analyzing each line in the regulation and translating it into a rule. Absence approval rule of Tenant C is also taken as an 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 Director 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 of the Decision Table

After specifying the decision rules, it needs to fill them into the appropriate combinations in the decision table as shown 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 no action will be 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 5[2] as business rule 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 [17]: (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; (4) fusion (event processing/temporal reasoning), providing event processing capabilities. Drools expert is used as a rule engine and Guvnor as a visual business rule definer which allow browsing and editing the rule set. Generally,

[2] `http://www.jboss.org/drools/`

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 is adopted as business rule representation style in our application.

Fig.6 is the snapshot of the executable Drools decision table for absence approval process of Tenant C. We can update business rule in a simple spreadsheet and deploy them with a few clicks. In this decision table, the first three rows 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 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 can also express rules as follows:

LHS (< rule name >) = < preconditions >

RHS (< *rule name* >) = < *conclusions* >

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

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

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 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 modify it, then upload 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 in Fig.7.

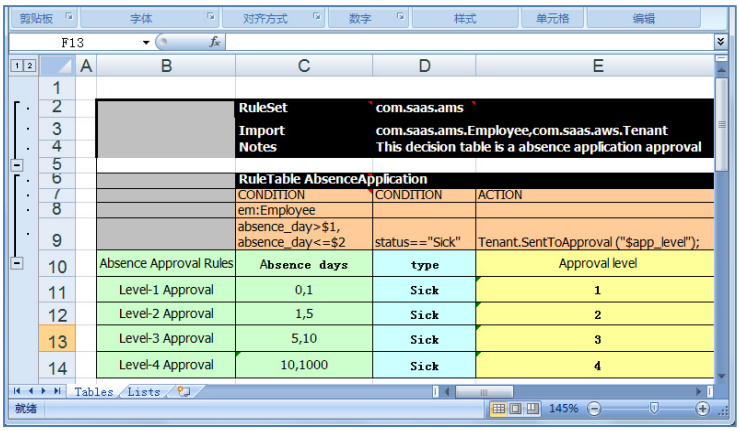

Fig. 6. Snapshot of Drools based on decision table

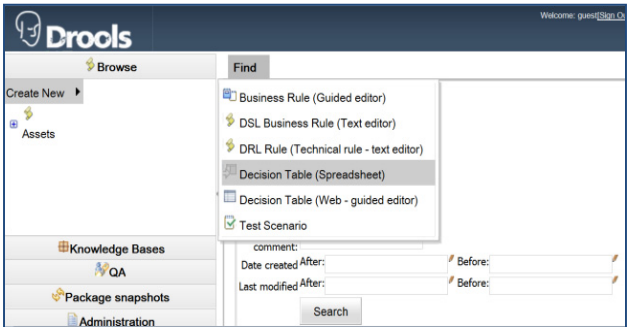

Fig. 7. Snapshot of decision table creation in Gnuvor

5.4 Prototype Application

The SaaS application of AMS prototype is developed following the proposed framework which has successfully integrated business rule engine into SaaS application. AMS is a SOA-based multi-tenant application. It allows tenant to manage their employee attendance and presence in a work setting to maximize the motivations and minimize the loss. And AMS is one of SaaS applications in Cloud Service Supermarket. Fig.8 shows a snapshot of the AMS prototype system, which is successfully deployed on the SaaS platform of Cloud Service Supermarket [20].

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. These advances allow SaaS application to be more transparent, flexible and cost reduction.

Fig. 8. Snapshots of SaaS application of AMS

Although BR engine based application has 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, is still in its earlier stage. On the one hand, business rule isolation for multi-tenant is not completely resolved by Guvnor. So there are more work still needs to be done on visual definer for specific SaaS application. On the other hand, performance evaluation work still need to be done in the future to make sure that the multi-tenant request response time is in a reasonable and tolerable ranage.

Acknowledgements. This Research Project Was Supported by the National Natural Science Foundation of China under Grant No. 60970017, No. 61202031, No. 61273216, and No.61100018, National Science & Technology Pillar Program of China under Grant No.2012BAH07B01, the Fundamental Research Funds for the Central Universities under Grant No.201121102020004, the Central Grant Funded Cloud Computing Demonstration Project of China Undertaken by Kingdee Software (China) Co., Ltd.

References

1. Kapuruge, M., Colman, A., Han, J.: Achieving multi-tenanted business processes in SaaS applications. In: Bouguettaya, A., Hauswirth, M., Liu, L. (eds.) WISE 2011. LNCS, vol. 6997, pp. 143–157. Springer, Heidelberg (2011)
2. Kwok, T., Nguyen, T.N., Lam, L.: Software as a Service with multi-tenancy support for an electronic contract management application. In: 2008 IEEE International Conference on Services Computing, pp. 179–186 (2008)
3. Wan-Kadir, W.M.N., Pericles, L.: Relating evolving business rules to software design. Journal of Systems Architecture (50), 367–382 (2003)

4. Liu, C., Dong, X.P., Yang, Z.Q.: Research of modern enterprise intelligent system based on rule engine and workflow. In: 2010 Intelligent Computing and Intelligent Systems (ICIS), pp. 594–597 (2010)
5. Gichahi, H.K.: Rule-based process support for enterprise information portal (2003), http://www.sts.tu-harburg.de/pw-and-m-theses/2003/gich03.pdf
6. Orriëns, B., Yang, J., Papazoglou, M.P.: A framework for business rule driven service compostion. In: Benatallah, B., Shan, M.-C. (eds.) TES 2003. LNCS, vol. 2819, pp. 14–27. Springer, Heidelberg (2003)
7. Vasilecas, O.: The framework for the implementation of business rules in ERP. Informacijos Mokslai (49), 146–157 (2009)
8. Vanthienen, J.: Ruling the business: about business rules and decision tables (2009), http://www.econ.kuleuven.be/tew/academic/infosys/members/vthienen/download/papers/br_dt.pdf
9. Forgy, C.: Rete: A Fast Algorithm for the many pattern/many object pattern match problem. Artificial Intelligence (19), 17–37 (1982)
10. Karami, N., Iijima, J.: A logical approach for implementing dynamic business rules. Contemporary Management Research 6(1), 29–52 (2010)
11. Guo, C.J., Sun, W., Huang, Y., et al.: A framework for native multi-tenancy application development and Management. In: The 9th IEEE International Conference on E-Commerce Technology and The 4th IEEE International Conference on Enterprise Computing, E-Commerce and E-Services, pp. 551–558 (2007)
12. Zhang K., Zhang X., Sun W., et al. 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, pp.123-130 (2007)
13. Gong, Y.W., Janssen, M., Overbeek, S., et al.: Enabling flexible processes by ECA orchestration architecture. In: ICEGOV 2009 Proceedings of the 3rd International Conference on Theory and Practice of Electronic Governance, pp. 19–26 (2009)
14. Nitu.: Configurability in SaaS (software as a service) applications. In: Proceedings of the 2nd India Software Engineering Conference ISEC 2009, pp. 19–26 (2009)
15. Sun, W., Zhang, X., Guo, C.J., et al.: Software as a Service: Configuration and Customization Perspectives. In: IEEE Congress on Services, SERVICES 2008, pp. 18–25 (2008)
16. Shi, Y.L., Luan, S., Li, Q.Z., et al.: A flexible business process customization framework for SaaS. In: WASE International Conference on Information Engineering, ICIE 2009, pp. 350–353 (2009)
17. Browne, P.: JBoss Drools business rules. Packet publishing. Birmingham-Mumbai (2009)
18. Jeng, J.J., Flaxer, D., Kapoor, S.: RuleBAM: A rule-based framework for business activity Management. In: 2004 IEEE International Conference on Services Computing, pp. 262–270 (2004)
19. Vasilecas, O., Smaizys, A.: Business rule based data analysis for decision support and automation. In: International Conference on Computer Systems and Technologies, CompSysTech 2006, pp. 191–196 (2006)
20. Zhang, X.W., He, K.Q., et al.: SaaS service super-market building model and service recommendation approach. Journal on Communication 32(9A), 158–165 (2011) (in Chinese)
21. Ghaddar, A., Tamzalit, D., Assaf, A., Bitar, A.: Variability as a service: outsourcing variability management in multi-tenant SaaS spplications. In: Ralyté, J., Franch, X., Brinkkemper, S., Wrycza, S. (eds.) CAiSE 2012. LNCS, vol. 7328, pp. 175–189. Springer, Heidelberg (2012)
22. Boyer, J., Mili, H.: Agile business rule development. Springer, Heidelberg (2011)
23. King, P.J.H.: Decision tables, pp. 135–142 (1967), http://comjnl.oxfordjournals.org/content/10/2/135.full.pdf+html