UTM
UNIVERSITI TEKNOLOGI MALAYSIA
RESEARCH UNIVERSITY

International Journal of Innovative Computing

Journal Homepage: http://se.cs.utm.my/ijic

Dynamic Parameterized States Tracking for Reusable Workflow Routing

Khor Swee Eng[1], Tan Chin Teong[2] and Chin Chee Kheen[3]
Software Development Lab - KHTP
MIMOS Berhad
{[1]se.khor, [2]ct.tan, [3]ck.chin}@mimos.my

***Abstract*— Dynamic parameterized states tracking workflow system is a workflow solution which aims to decouple business logic and application code in order to minimize rigid or spaghetti code in any kind of workflow system. This approach, allows programmer to focus on the development of the application itself, and let the business process owner to deal with the application data, business rules and logic. Business process owner can setup, configure and deploy workflow process independently without changing the workflow engine application code. This workflow engine is featured with dynamic functions calling, task aging checking and reminder sending. These specific features allow process owner to customize their flows seamlessly with their business requirement using this workflow engine. The proposed workflow solution can be library package or Service Oriented Architecture (SOA) based, and can further configure to be multitenant workflow solution. In this paper, the design of proposed workflow solution is discussed. Actual live systems, Intellectual Property Management System (IPMS) and Human Resource Information System (HRIS) which have already using the proposed workflow solution are also showcased in this paper**.

Keywords: Workflow Engine, Routing, Business Process Management, Workflow Management, Reusability, State Change.

I. Introduction

Workflow system, as modeling technology for business processes management (BPM), is the key component for daily operations of millions of enterprises across the globe. The main objective of workflow system is to improve business effectiveness and efficiency while striving for innovation, flexibility and accountability [1]. Workflow Management Coalition (WfMC) has proposed a framework, which includes five categories of interoperability and communication standards as describe in TC003v11 WfMC Workflow Reference Model [2]. These five interfaces are Process Definition Tools, Administration and Monitoring Tools, Workflow Engine, Workflow Client Applications and Invoked Applications [3]. The idea of this framework is to define scope for the implementation of complicated workflow system into different common core set at the same time satisfies individual business requirements. With this framework, formal separation between the development and run-time environment is achieved and thus enabling a process definition generated by one modeling tool, to be used as input to a number of different run-time products [4].

At present a variety of different tools have been created to analyze, model, describe and document a business process. A quick search of "workflow software solution" on Google results in thousands of software related to workflow; namely K2 Workflows, Free Workflow, Office Routing Plus, Progression, Microsoft Workflow Foundation etc. Creating, executing and maintaining workflow software poses unique challenges. Most of the commercially available software is usually complicated and expensive [5]. On top of that, business process owner needs to go through steep learning curve before he/she can start using this software. For example, Microsoft Workflow Foundation requires users to familiarize with the designer view, to understand rules set editor and to perform multiple rules evaluations [6] before he/she can start creating simple approval activity, which consists of two processes and one step. The featured rich software may favor sophisticated processes like forward chaining rule sets, but it becomes an obstacle for users who seek simple activity conditions and fast implementation of business logic without going through heavy learning cycle for the particular software.

The motivation of this paper is to showcase a design of a lightweight workflow engine, which can serve majority of the business process management. Based on the design, business process owners are only required to understand of their business logic [7]. They can configure their workflow process without going through complicated GUI and rigorous training. At the same time, they are able to enjoy the flexibility to change business process on-the-fly. In this kind of design, business process model data, which stored in database is independent from the runtime workflow engine.

In the following section, we first discuss the design and architecture of our dynamic parameterized workflow engine. In section III, the application and implementation of the workflow engine is described with real-life applications developed and hosted in MIMOS. Lastly, section IV discusses about the advantages and section V gives a conclusion.

II. Worflow System Design Principle

The core design principle of the workflow solution is reusability. While keeping the basic workflow engine capability, which is controlling flow between activities, it should be easy enough for the developers to integrate the application business logic. This is to ensure the decoupling of application logic from strongly tight with the workflow engine to prevent any future changes of routing behaviors. There are a few benefits, one of main reasons is maintainability. Business world always subjects to change and it finally leads to application logic change. The application logic change always involves a complex code revamp where finally incurs poor maintainability of the system .

The two important resuabilities components that are focused are :

A. Workflow Process Reuse

Worfklow process reuse is referring to the workflow tree configuration that has been created can be stored and further customized and replicated for any applications. The configuration of the workflow decision tree works as a template that is generalized for any applications usage. It can be adopted for any applications if it fits their need.

B. Business Logic Reuse

Business logic reuse is aiming at allowing the application to update, create, delete their application logic for a specific state/activity easily. A business logic will be triggered to complete certain processing once a decision is made. This is translated to a transition of a workflow decision tree. It can significantly help to reduce system changes time.

Referring to Fig.1, the workflow model consists of three components, namely workflow settings, workflow engine and workflow output. The workflow settings are basically a module for configuration and setting purpose of the generic workflow process. The workflow engine, the core component is responsible for the runtime routing, scheduling and etc. Last, the workflow output in-charge of the dynamic business library invocation and transactions data storage.

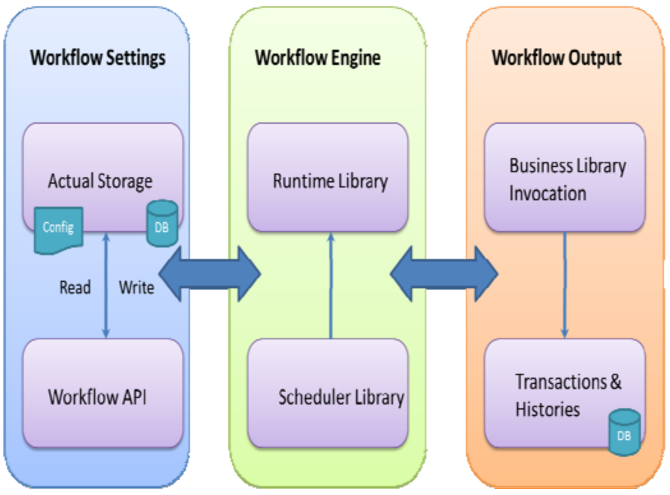

Fig. 1. Workflow System Components

Typically, a user of the workflow API with their system will initialize the configuration of customized workflow processes through the Workflow API. In the configuration, the states or activities of the workflow will be stored into the database. The necessary configuration setting is to tell the basic logic flow of their intended workflow process. The creation of the states or activities will also bind to a specific namespace, class and method to be invoked when the activities is routed. This design makes workflow engine to be dynamically adjusted and tuned to any complicated business process. The following is a simple example of a customized workflow process.

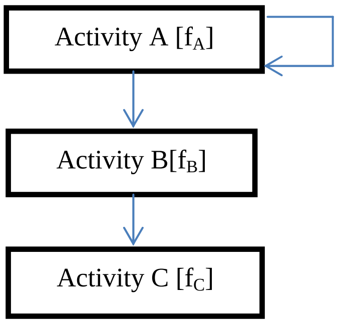

Fig. 2. Example of customized Workflow Process

In Fig. 2 example, we have shown that three different activities which are named A, B, C are created. The arcs are referring to the routing decision by the activities. The arc, A→ B explains the transition from activity A to activity B by a specific decision. Each of the arcs can code for different decisions. Concurrently, each of the activities is bound to a pre-developed method or function where it contains certain business logic. The workflow engine will help to route from activity A to activity through the correct decision (from the arc A→ B) and the method in activity B, which is [f_B] will be automatically invoked. The namespace, class and the method are already preset to bind with the activity, in the runtime; the workflow engine will invoke the method through reflection.

In the design, there is a scheduler service, which is running as a windows service at the back end of the operation service. This scheduler provides functionality for activity aging routing and reminder service. Aging and reminder service is to allow the workflow process to be automatically processed after certain timeframe. The windows service at the background will check for all workflow instances duration compared with last activity, if the duration exceeds the preset timeframe, its aging or reminder function will be called.

For the workflow output, any of the transaction data, it will be logged in the database. The handy API provides functionality for developer to check for the current status and even the list of histories.

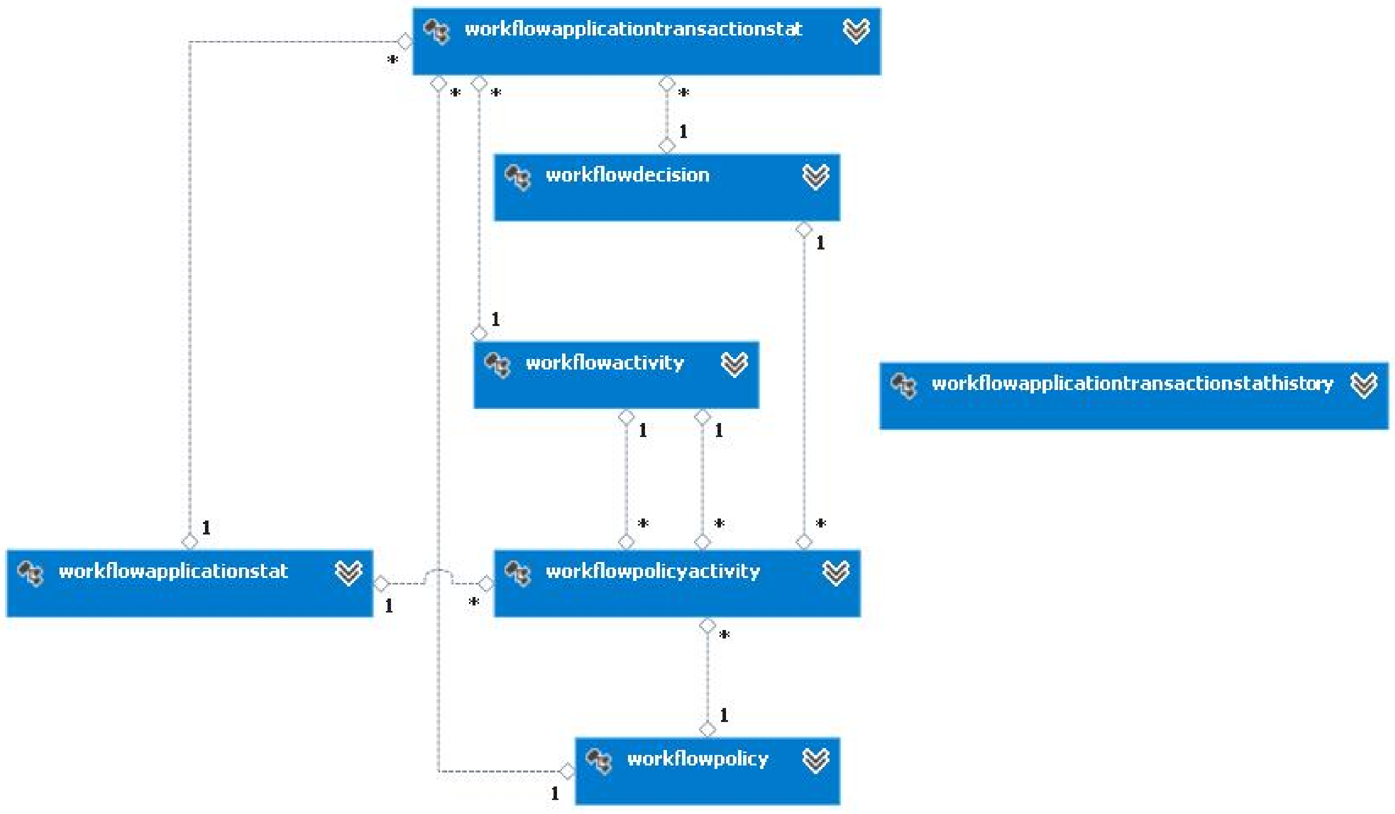

Fig. 3. Entity Relationship Design for the Workflow System

The above Fig. 3 illustrates the ERD for the whole workflow system, which will provide business logic separation for any system. Through the API that we have provided, the user can supply the necessary workflow setting data so that the workflow engine can help to do the routing and tracing. Below we explain the data withhold by each of the entity:

Workflow Policy – Workflow policy is a generic and unique naming for a specific workflow process, for example, Leave Application flow, Finance Approval flow, etc.

Workflow Activity – Workflow activity is a unique and generic activity or state, which will form up a workflow policy.

Workflow Decision – Workflow decision is a set of decision that can be linked to an activity to perform a routing.

Workflow Policy Activity – Workflow Policy Activity is relationship table to describe the ‘how’ process of routing for the create workflow policy. It will hook-up with the workflow activity, decision and application stat which finally form a unique policy.

Workflow Application Stat – It stores the naming for the triggered workflow instance.

Workflow Application Transaction Stat – Each of the workflow instances is a workflow application. This entity will store all the current transaction status.

Workflow Application Transaction Stat History – All historical transaction data is stored here.

III. Application Integration

The proposed approach has been successfully applied to a few development projects in MIMOS. These applications are IPMS (Intellectual Property Management System), HRIS (Human Resource Information System) and TalentXchange (Talent Management System). All the abovementioned systems do share same characteristic which require a flexible and robust customizable approval flow process. Fig. 4 refers to a deployed workflow system in MIMOS applications.

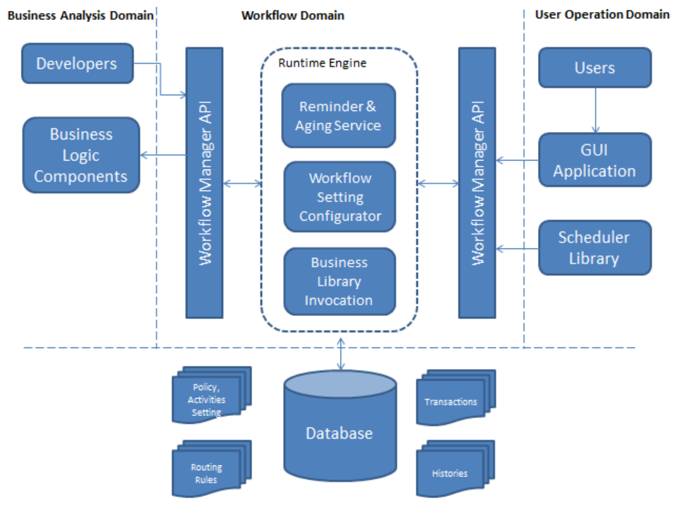

Fig. 4. Dynamic Parameterized Workflow System

This system implements independent logic components. They are loosely coupled from the main application. By this de-coupling technique, the business logic components can be invoked effectively and repeatedly. Re-configuration of routing rule can be done through configurator and saved in database without any interference or re-coding of the application. Below, the steps to integrate with an application are explained.

1) As a start, developers create all the necessity policies, activities and the routing rules.
2) Developers then are required to implement the business logic components correspond to the activities.
3) During runtime, the application users interact with GUI application as per normal, the application then invoke workflow manager with necessary parameters.
4) Business Library Invocation in the Runtime Engine then route the application to the corresponded business component based on the routing rules that previously set by the developers. As for aging and reminder items, the invocation is done through a scheduler service.
5) The scheduler interfaces with Workflow Manager API.
6) Aging & Reminder Service is specifically designed to handle the scheduler items.
7) The service then interacts with the Business Library Invocation to invoke business logic components similar to (4). As this is an automation service, no user interaction is needed throughout the whole workflow invocation cycle.

IV. Discussion And Advantages

There are several applications have been developed using the workflow system. By using the SDK provided by the workflow system, developers can easily integrate their configuration workflow logic into the enterprise applications. The development time is shortened and at last, the applications developed can be easily maintained if there is a need for future changes and modifications. The often change request of the enterprise applications due to the business rules changes can lead to lesser code changes as well.

Advantages of the proposed workflow system are listed as follows:

A. **Multi-Tenant** - A complete re-usable workflow system that can support multiple applications even tenants concurrently. We now can have a centralized workflow repository that will fulfill most of the application needs.

B. **Self-Maintainability** - Some of the process flow change can even be modified by the application owners themselves. Since changes of existing activity route required no development effort (only routing rules setting) it is possible for the application owner to manage the flow without involving developers.

C. **Traceability** - Historical data are ready instantly. Application history is critical in workflow oriented application. Application developers can leverage this capability effortless from this dynamic workflow system.

D. **Code Simplicity** - In this system, only 2 functions are needed to invoke the workflow. All interaction with business application is done through the Workflow Manager API.

E. **Performance** - Direct Assembly function invocation for speed and system responsiveness. Unlike other workflow system, which implemented through http protocol, incurred unnecessary network overhead throughout the innovation cycle.

V. Conclusion

By implementing our system above, we noticed that the development effectiveness had improved drastically. The developers were confident to modify the code with minimal undesirable effect. Those changes incurred only little to no development effort. Stability would no longer be an issue should there are flow changes in the future. Owners and developers were clearer with all the business flows and activities that implemented in the system. The workflow system improvement and enhancement could be planned independently and could be executed concurrently, with no inter-dependencies between these two development parties.

Repeating changes of business flow by owners are definite challenges for the developers, as not only that they need to revise the code over and over again; they also can't have overall clarity of the business flows. In the long run, the system will become complicated, fragile and prone to errors, as a simple change in the code might result unpredicted behavior of the system. Instead of implementing a gigantic if-else rules engine in the application, it should be independently coded and implemented, as well as should be maintained modularly.

Acknowledgment

We would like to thank MIMOS for sponsoring the research.

References

[1] D. Georgakopoulos,, M. Hornick, A. Sheth, An overview of workflow management:from process modeling to workflow automation infrastructure distributed and parallel databases, vol 3, pp 119¨C153. Boston: Kluwer 1995.

[2] http://www.wfmc.org/wfmc-standards-framework.html

[3] Workflow Management Coalition Workflow Standar, Process Definition Interface – XML Process Definition Language (Document Number WFMC-TC-1025), 2012.

[4] H.B Li, D.C Zhan, Reusable workflow system design and development, IJCIM, vol 15, no 3,pp39-49,2007

[5] H. Gruber, C. Huemer, Profitability Analysis of Workflow Management Systems, IEEE Conference on Commerce and Enterprise Computing, 2009.

[6] http://msdn.microsoft.com/en-s/library/aa480193.aspx#introwork_topic2

[7] J. L. Kmetz, Mapping workflows and managing knowledge: simply, sensibly, flexibly, and without software. Newark, CreateSpace Independent Publishing Platform, 2011.