

SaaS-Flow: Software as a Service Workflow Management System

Eng. Mustafa Gamal, Ass. Prof. Manal Ismail, Dr. Hesham Kesik
Faculty of Engineering, Helwan University, Helwan, Egypt
mustafamgamal@gmail.com, manal_shoman@yahoo.com, h_keshk@hotmail.com

Abstract

This paper introduces SaaS-Flow workflow system. SaaS-Flow is a Workflow Management System whose architecture is based on Service Oriented Architecture and Component Based Architecture. This paper proposes a Workflow Management System architecture that enables providing workflow as a service to subscribers over the web with the concept of Software as a Service SaaS. The proposed Workflow Management System enables organizations to host their business processes to SaaS-Flow without the need to install SaaS-Flow in their local servers. It enables a workflow execution over internet in disconnected and heterogeneous system environments.

1. Introduction

Workflow process is traditionally defined in office terms: moving the paper, processing the order, issuing the invoice. The same principles apply to filling the order from the warehouse, assembling documents, and people to repair a complex system, or manufacturing complex device [1].

Workflows describe business processes as the coordinated execution of simple activities (tasks) by human or automatic executors (agents). It also described as software systems that support the automatic execution of workflows [2].

Most of e-governmental solutions are based on a Workflow Management Systems WfMS. Workflow process may pass through different organization to complete its executions. Those governmental organizations use different operating systems and may support different development environments. Some of them may have workflow management system in their servers and others may not. Also some of governmental processes may need intervention of nongovernmental organizations to complete its execution.

This paper proposes a Workflow Management System Architecture that addresses problems of executing processes on such systems. It proposes a WfMS that work with concept of Software as a Service SaaS [3]. SaaS is a model of software deployment where an application is hosted as a service provided to customers

across the Internet without the need to install and run the application on the customer's own computer.

The proposed WfMS enables organizations to host their business processes to the WfMS without the need to install the WfMS in their local servers. Its design is based on component architecture and service oriented architecture (SOA). The proposed architecture enables workflow execution remotely over the web in disconnected and heterogeneous system environment. Heterogeneous system means a system with different operating systems or different development environments.

The paper is organized as follow: section two is Workflow concepts, section three Related Work, section four presents the proposed SaaS-Flow System Architecture, section five explain execution and deployment example, then the Conclusion in section six.

2. Workflow concepts

A workflow system can be described as a cube with three dimensions [2]; *Organization*, *Business process*, and *Information*. *Organization model* contains the organization structure, the users (executors of tasks) and their position in the organization, and the roles of each executor in the organization. Roles are used to authorize a given executor to perform a specific task. *Information model* contains the information needed to perform process activities; this includes the application form and any supporting documents. *Process model* contains the structure of the business process.

WIDE4 project [4] defines *process* as a coordinated set of *activities* which are connected in order to achieve a common goal. According to a predefined sequence of execution, each *process* can be instantiated in several instances of that *process*, called *Jobs*.

Next section explains related work and the value added by the proposed system.

3. Related Work

WfMSs has been developed using a lot of development methodologies. Each has its benefits and drawbacks. For example, the architecture of *WIDE* [4] is based on database technology. This enables the engine to utilize the power of the DBMS it uses. But at the same

time this design makes the system design less flexible and hard to extend.

Micro Workflow [5], and *Microsoft Workflow* [6] are example of systems that based on techniques specific to object systems and compositional software reuse. Both engines have more focus on software developers. Process modelling is based on object oriented inheritance which makes process structure update (specially while execution) complex. Also it didn't use standard workflow language to model processes.

Liaison Workflow [7] is another example of a component based and object oriented architecture. Liaison consists of five major components, and uses a software bus to connect these components. The software buss uses CORBA as its communication protocol. Application not written in java will hardly integrate with the engine. Also the system uses a special process modelling language called Valmount [8], which lower its flexibility to integrate with other process modelling solutions.

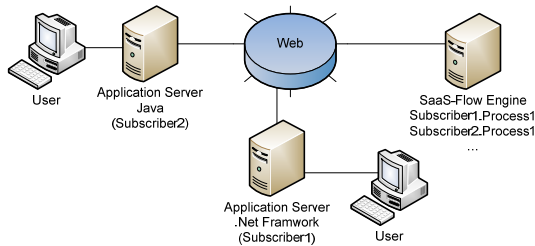


Figure 1: Organization can subscribe and host their process to SaaS-Flow engine.

Commonly known WfMSs don't focus on building a workflow engine that work with the concept of SaaS [3]. This paper proposes '*Software as a Service workflow management system*' (SaaS-Flow) architecture. A WfMS that try to address the drawbacks of previously mentioned systems and at the same time focus on building workflow engine that works with the concept of SaaS [3].

Figure 1 shows how organization can subscribe and host their process to SaaS-Flow engine. SaaS-Flow exposes its Task Manager (responsible of handling users tasks) to users through XML webservice [9] interface to enable them to fully integrate their applications with the engine.

SaaS-Flow Architecture is flexible enough to allow the clients to host an instance of the Task Manager of the SaaS-Flow WfMS at their servers. In this scenario, SaaS-Flow will send tasks to Task Manager using XML webservice. This will allow clients to receive tasks that require to be done by their organizations coming from any other governmental organizations hosting SaaS-Flow engine. Also it will allow clients to handle tasks disconnectedly from engine. Figure 2 shows this scenario.

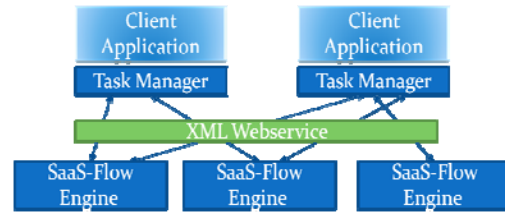


Figure 2: Task Manager collects tasks of the client from different SaaS-Flow engines.

Next section describes the architecture of SaaS-Flow system.

4. SaaS-Flow architecture

SaaS-Flow architecture is based on Component Architecture mixed with Service Oriented Architecture SOA technologies. A component-based software development is the process of assembling existing software components in an application such that they interact to satisfy a predefined functionality [10]. SOA [11] is a way of designing a software system to provide services to either end-user applications or other services through published and discoverable interfaces.

The use of component design makes it easy to expose the workflow engine functionalities through a set of services. Also it provides a great way to partition the engine into a set of component. These components communicate with each other through a set of predefined contracts (*Interfaces*).

Figure 3 shows SaaS-Flow system architecture. It is Client/Server architecture. Both client and server are considered to run two different operating system and use different development languages. *The client side* as shown contains the **Client Application**, which is the application controlled remotely by the engine and used by the user to accomplish a specific task, and optionally **Task Manager**. *Server Side* contains **SaaS-Flow engine**, **Pluggable Agents**, **Task Manager**, **Communication Layer**, and **Data Repository**.

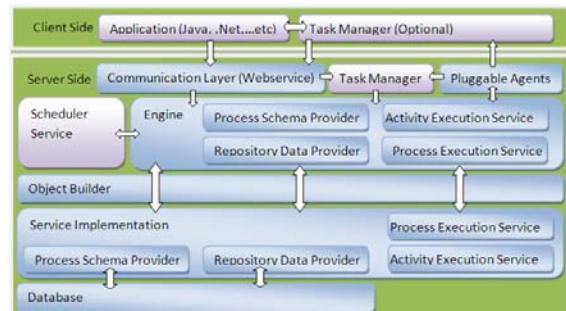


Figure 3: SaaS-Flow System Architecture

The **Communication Layer** is used to expose engine functionality through an XML webservice technology [9]. **Data Repository** contains workflow process schema, execution information, and complete history of previously executed processes. The function and the role of **Engine**, **Pluggable Agents** and **Task Manager** in process execution will be briefly described in the next subsections.

Another services not mentioned in figure 3 and lie at server side are *Process Interpreter* and *Membership Provider*.

Process Interpreter is used by *Process Schema Provider* to interpret process written in standard languages such as XPDL [12] and BPEL [13]. SaaS-Flow architecture allows developers to create interpreter for each modeling language and plug it to system. Those interpreters are associated to their correspondences language using XML [14] configuration file. This configuration file is used by *Process Schema Provider* to choose the right interpreter to interpret incoming processes and register it to system.

Membership Provider responsible of registering and authenticating members to system. Communication layer isolate security token from incoming webservice requests and provide them to *Membership Provider* to authenticate them and return member identity.

SaaS-Flow architecture executes process activities as a set of pluggable agents (services) [15]. Software developer has to associate each process activity with its corresponding agent (service) that developed to satisfy the required action from this activity.

SaaS-Flow architecture considers human tasks as remote services executed by human. That is why it is fully separate Task Manager Component from the engine. Also SaaS-Flow handles human tasks as a service call to agent. This separation frees the engine from considering how the activity will be executed. All activities are the same; they execute a service and inform finish to engine to continue the flow of process execution.

4.1. Engine

The maestro of the work, it is responsible of process execution by organizing work between all working services. Engine is hosted as a singleton object [16] and can be accessed manually through console application. Other applications can communicate with the engine through the communication layer. If the application in the same local area network of the engine, the application can communicate with the engine using COM+ [17] technology, otherwise the application can access the engine remotely through its webservice interface.

The engine receives the process in the form of Process Object Model (POM) for execution. POM is an object oriented structure developed for SaaS-Flow to model business process in both **Static** (Process) and **Execution** (Job) views. The engine executes process's activities as a set of pluggable agents as it will be described in the next section.

As shown in figure 4, Engine uses the following services:

- Process Execution Service and Activity Execution Service: used by engine to execute the process.
- Process Schema Provider: this service handles Process storage in its static view.
- Repository Data Provider: this service handles storage of Process execution data such as saving state and tracking information.
- Scheduler Service: monitors execution times, delays and trigger events related to these delays.

All mentioned services are pluggable components. A special class is used to realize plug-in pattern which is ObjectBuilder. Next subsection describes pluggable agents and its role in process execution.

4.2. Pluggable Agents

Pluggable agents are a set of agents that are executed in relation to the execution of a certain Activity. After business owner design process, Process software developer associates its activities to agents. The activity agent is a working code developed by software developer to perform the task required from this activity.

When Activity Execution Service executes activity, it loads instantiation information of its correspondent agent from the Repository. In turn Object Builder creates an instance of the agent. Then Activity Execution Service passes callback delegate and task parameters to the agent then executes it. The callback delegate is used by the agent to inform engine for its status updates.

Developers can create his own Agents by implementing IAgent interface. Then associate it to its related activity through system portal. Agents can be reused for activities that need similar execution action.

There are two types of tasks, automated tasks; that need no human intervention, and human interaction task; that needs human intervention to complete the task. For the automated tasks agents finishes its work and callback engine to inform finish. For the human interaction task type, the agent calls Task Manager to register user task. After user finishes the task, Task Manager informs engine about task finish.

A reusable agent called UserInteractionAgent is used to register user interaction tasks to Task Manager. It calls

Task Manager locally or remotely over the web to register user task.

4.3. Task Manager

Task Manager is used in SaaS-Flow system to handle tasks that require user interaction. It acts as an interface between the engine and client applications. User application communicates with Task Manager through XML webservice interface [9] to query for user tasks and inform for task status.

Task Manager is fully isolated from the engine. It is called by pluggable agents when there is a user task need to execute. Also Task Manager inform engine for task completion through SaaS-Flow communication layer. This separation allows Task Manager to be hosted elsewhere. Also this will facilitate collaboration of two different SaaS-Flow WfMS as they are one system.

Next section provides an example of a process executed by SaaS-Flow WfMS.

5. Execution and Deployment Example

Figure 4 provides an example of a cross organizational process that passes through several organizations to complete. As shown in the figure, the process must pass through Internet Service Provider which hosts the Student Portal, Faculty of engineering at Mattaria, Faculty of engineering at Helwan, and Helwan University. The process starts from student portal where student request transfer from Faculty of engineering at Mattaria to Faculty of engineering at Helwan and finishes when student transferred to Helwan.

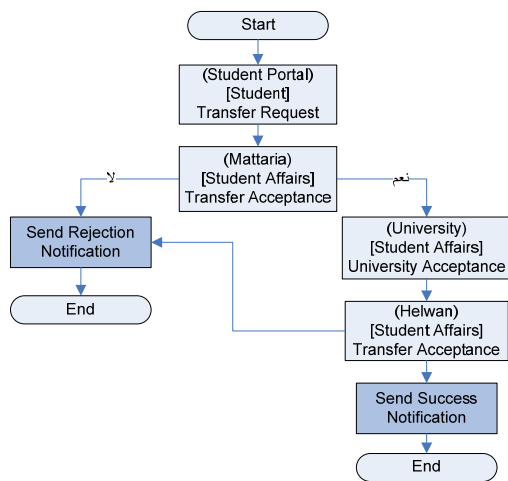


Figure 4 Student Transfer process

Each of those organizations has its own servers and applications. Student web portal is hosted on Linux

server. The portal is written in PHP[18]. Other organizations are using windows server and their applications are using .Net Framework[16]. The SaaS-Flow is hosted at the University campus.

Organizations involved in the process needs to register to SaaS-Flow through its portal. Each organization will have its credentials to use the SaaS-Flow.

The student Transfer process built using Together Workflow Editor [19]. It produces process in XPDL [12] format. "Student Transfer" process file is uploaded through SaaS-Flow portal. The portal passes the request to the Process Schema Provider through communication layer. Process Schema Provider chooses the right interpreter to interpret the process to POM to be ready for execution.

All organizations' applications are communicating with SaaS-Flow through its communication layer using webservice. Credentials are sent with each request to authenticate request.

The following steps describe process execution mechanism:

1. When student log to the student portal, portal requests SaaS-Flow to get all available processes.
2. Communication layer receives the request and ask Membership provider to authenticate it and returns Membership identity of the organization. Communication layer pass the request with the identity to Process Schema Provider to fulfil the request.
3. *Process Schema Provider* gets all process allowed based on the provided identity and returns it to caller through communication layer. Portal shows the resulted processes.
4. When student picks "Student Transfer" process; the portal passes the request to Task Manager which transfer request to the engine.
5. The engine gets the targeted process from process provider and starts its execution by passing it to *Process Execution Service*.
6. *Process Execution Service* creates a job for the process with unique id and passes the first activity to *Activity Execution Service*.
7. *Activity Execution Service* loads the corresponding agent of the current activity, instantiate it using Object Builder, then executes it and passes *Task* object to it.
8. *Human Interaction Agent* serializes the Task and sends *register user task* request to Task Manager.
9. Portal shows the task on the student task page by requesting user tasks of currently logged user from the Task Manager.
10. When student open task, the portal sends *open task* request to Task Manager.

11. The portal uses task information to open the application (web page) for that task.
12. After student finishes the task. The portal sends *finish task* request to Task Manager which transfers the request to the engine to rout for next task.
13. The engine asks Process Execution Service to route and execute next activity which will be executed at the university campus the same way described in previous steps.

There are two automated activities in this process: *Send Success Notification* and *Send Rejection Notification*. Both uses the same automated agent called "Notify User" agent. When activity execution service executes this agent, the agent sends notification to the targeted user and then callback the engine to inform finish through callback delegate.

The engine finishes the process when there is no next activity to execute.

6. Conclusion

This paper described the architecture of SaaS-Flow system. SaaS-Flow is a workflow management system whose architecture is based on component based and service oriented architecture. SaaS-Flow is a WfMS that works with the concept of SaaS. Clients don't need to host the engine to be able to use it in their applications. They host their processes at the engine without the need to host the engine itself.

The system is client/server architecture. It is based on three major components: engine, Task Manager and pluggable agents.

SaaS-Flow architecture executes process activities as a set of pluggable services. It considers human tasks as remote services executed by human. Task Manager is called as a remote service using pluggable agent. This separation provides great flexibility for portability and interoperability between workflow engines.

The use of XML webservice enabled exposing engine's functions to external systems independent of their location or working environment. The use of Component architecture facilitates extending system functionality without the need to rebuild the whole system.

SaaS-Flow engine supports *separation of concerns*. Where the *application developer* can build and test their application without the need to communicate with the engine. At the same time *Business Process Designers* can build their processes and test it without the need to communicate with the application. That is done through the separation of process execution from application execution. Furthermore, this SaaS-Flow workflow engine architecture can also be used as a general reference for

workflow management system architecture because of its generic design, flexibility and generality.

7. References

- [1] **Plesums, Charlie.** "Workflow in the World of BPM Are They the Same?" *Workflow Handbook 2005*. WFMC, 2005, pp. 17-22.
- [2] **Fabio Casati, Silvana Castano, Mariagrazia Fugini, Isabelle Mirbel, Barbara Pernici.** "Using Patterns to Design Rules in Workflows." *IEEE Transactions on Software Engineering*, 2000. pp. 760 - 785.
- [3] **Mangan, David Greschler and Tim Mangan.** "Networking lessons in delivering 'Software as a Service'---part I". *International Journal of Network Management*. September/October 2002, pp. 317-321.
- [4] **F. Casati, P. Grefen, B. Pernici, G. Pozzi, and G. S'anchez.** "Wide workflow model and architecture." Centre for Telematics and Information Technology (CTIT), University of Twente, Netherlands. April 1996. pp. 96-19, Technical Report.
- [5] **Manolescu, Dragos A.** "Workflow Enactment with Continuation and Future Objects". *Proceedings of the 17th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*. ACM, Seattle, Washington, USA, 2002. pp. 40-5.
- [6] **Andrew, Paul, Conard, James and Woodgate, Scott.** "Presenting Windows Workflow Foundation, Beta Edition." Sams, 2005 Sept.
- [7] **Chung, Jojo M.L.** "The Liaison Workflow Engine Architecture." *IEEE Computer Society*, 1999. The Proceedings of the Hawai'i International Conference On System Science.
- [8] **Leung, D.K.C. Chan and K.R.P.H.** "Valmont: a Language for Workflow Programming." *IEEE Computer Society Press*, Washington, DC, USA : 1998. Proceedings of the Thirty-First Annual Hawaii International Conference on System Sciences-Volume 7. p. 744.
9. **David Booth, Hugo Haas, Francis McCabe, and Eric Newcomer.** W3C Working Group Note: Web Services Architecture. W3C. Feb 11, 2004. <http://www.w3.org/TR/ws-arch/>.
10. **Gill, Nasib S.** "Importance of Software Component Characterization for Better Software Reusability." *ACM, 2006. ACM SIGSOFT Software Engineering Notes*. Vol. 31, pp. 1-3.
- [11] **Johnston, Alan W. Brown and Simon.** "Using Service-Oriented Architecture and Component-Based Development to Build Web Service Applications." Rational Software Corporation, 2002.

[12] **Workflow Management Coalition.** "Workflow Standard, Process Definition Interface-XML Process Definition Language." Workflow Management Coalition, 2008 Oct.

[13] **OASIS.** "Web Services Business Process Execution Language Version 2.0." OASIS, 2007.

[14] Extensible Markup Language (XML) 1.1 (Second Edition). W3C. Aug 16, 2006. <http://www.w3.org/TR/xml11/>.

[15] **Martin Fowler, David Rice, Matthew Foemmel, Edward Hieatt, Robert Mee, Randy Stafford.** "*Patterns of Enterprise Application Architecture.*" Addison Wesley, 2002.

[16] **Lowy, Juval.** "*Programming .NET Components.*" O'Reilly, 2005.

[17] **Mulder, Chris Peiris and Dennis.** "Pro WCF Practical Microsoft SOA Implementation." Microsoft Press, 2007.

[18] **PHP.Net.** PHP Manual. *PHP.* <http://www.php.net/manual/en/>.

[19] **Together Workflow Editor.** <http://www.together.at/together/prod/twe/index.html>.