PBCP: A Process-based Bussiness Collaboration Platform

Shiping Chen[1,2,*], Tsz Lam Chim[2], Zhijie Li[2], Bo Yan[1], Hendra Wijaya[1] and Surya Nepal[1]
[1]*Commonwealth Scientific and Industrial Research Organization (CSIRO) Data61, Sydney, Australia*
[2]*School of Electrical and Information Engineering, The University of Sydney, Sydney, Australia*

Keywords: Multi-party Collaboration, Business Process, Rule-based Business Engines, Event-driven Architecture.

Abstract: Business expertise becomes more specialized and focused, which leads to a large number of small and medium enterprises (SME) to deliver a specific service in context of a business collaboration. This paper presents such a process-based business collaboration platform to facility this kind of multi-party business collaboration. The platform features with a process-oriented user interface, event-driven business process backend, which makes it elegant and generic to be used for different multi-party collaboration applications.

1 INTRODUCTION

Business expertise becomes more specialized and focused, which leads to a large number of small and medium enterprises (SMEs) to deliver a specific service, such as home loan brokers, solicitors and building inspectors. On the other hand, some businesses require multiple service providers to work together to complete a complicated business process. As a result, there is a strong demand for a collaboration platform to facilitate such a multi-party collaboration as an innovative online service, Blaire Palmer (2014).

Most of web-based collaboration service are either yellow-page-based simple information repositories or complicated and expensive enterprise-level CRM/ERP systems as Salesforce (1999). We envision a simple, lightweight yet capable secure collaboration platform that can attract clients and SMEs to work together by following natural business processes for dynamic business collaborations Surya Nepal and Shiping Chen (2011).

In this paper, we explore software architecture and middleware technologies for building process-based business collaboration platforms (PBCP) for small and medium business. In particular, we identify common functionalities and requirements for building and operating process-based business collaboration systems. A XPDL-based front UI (User Interface) is designed and implemented by reusing and extending the standard XML Process Definition Language (XPDL) in Brunt, J. and K. Swenson. (2006). A simple business engine is implemented on top of jBPM v6.2.0 (2015).

2 REQUIREMENTS FROM MOTIVATION EXAMPLE

We use the following real-world application scenario as a motivation example to specify the problem and capture the key functionalities and requirements of this type of collaboration applications.

2.1 Motivation Example

Alice wanted to spend $1,000,000 to buy house in Sydney. First, she contacted a number of banks to enquire their home loan interest rates. Then she decided to borrow $800,000 from Bank-A (pay the rest with her saving), because Bank-A's home loan interest is the lowest. She got a pre-approval of her home loan application from Bank-A before her property hunting.

Next Alice found out $1,000,000 property near her workplace via a property agent. She put a $1000 deposit to secure the property. Then she hired a solicitor to help the purchase. The solicitor reviewed the sale contract, checked the property registration, had Alice signed the contract, and exchanged the contract with the vendors' solicitor.

According to the local regulation, Alice has 12 days, also called cooling period, to terminate the contract with reasonable reasons after signing the

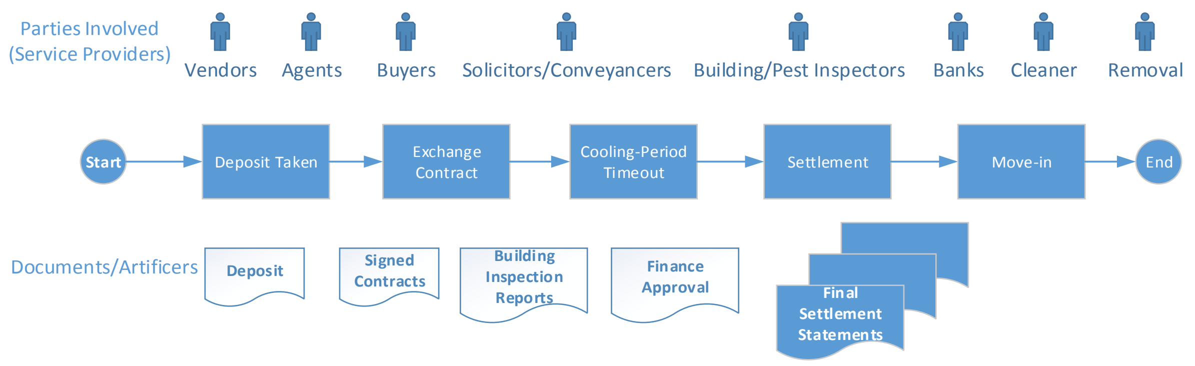


Figure 1: "Buying a property" as the motivating example.

contract. In practice, Alice would hire a property specialist to conduct building and pest inspections to ensure there were no major issues with the property. At the same time, Alice had her home loan application fully approved from Bank-A after the property being valued.

Since nothing went wrong during the cooling period, the contract entered to non-conditional stage, in which both the buyer and vendors were preparing for the final settlement of the deal, including booking cleaning and removal services etc.

The whole business process of "buying a property" is illustrated in Figure 1. And we can find a considerable number of this class of multi-part collaboration applications in our real life.

2.2 Functionalities vs. Requirements

Based on the above motivation example and the other similar applications, we can capture the following basic functionalities/features for this class of multi-party business collaborations applications:

- A collaboration can be defined as a simple business process.
- The multiple business entities can dynamically participate in a particular business process.
- A participator can join in a collaborative process via (but not limited to):
  - Queuing
  - Invitation
  - Auction
  - Recommendation
- The interaction between participators can be either automatic machine-to-machine, or semi-automatic human-two-machine, or completely human-to-human.
- The collaborators can share some documents as the collaboration is going on, such as contracts, deposit receipts, building inspection reports, etc.

To support the above features and functionalities and consider the (performance and cost) overheads, we identify three generic requirements for middleware technologies of building and operating the process-based collaboration systems as follows:

R1. A process-based generic frontend User Interface, which drives the interaction between end-users and the service providers and visualizes the excitation of the whole business process

R2. A lightweight business process engine that can decouple the business rules/logics with the engine execution.

R3. A secure database and/or file system to store the collaboration process and data

In this paper, we address the top two requirements and leave the last requirement as our future work.

3 RELATED WORK

Workflow has been developed for decades ago. It is used to describe the business procedure by utilizing a series of *tasks*. Those tasks are also handled by different roles in a business process. With the development of the computer technology, the workflow processes can be handled by the workflow application which can automatically or semi-automatically execute the task in a more efficient way Xiao, Y., et al (2004). The Workflow Management Coalition (WFMC) defined a basic workflow reference model in 1994. It contains composing, function and interface of the workflow for further development in Xiao, Y., et al (2004).

For many organizations, the workflow models have been widely used to deal with the collaborative processes using heterogeneous workflow middleware Armin Haller, M.M., *et al.* (2009). In order to define a complex collaborative business process, the internal and external workflow processes should be aligned as a whole, which may cause the low flexibility and interoperability problems. Therefore, a new workflow approaches which are the XML Process Definition Language (XPDL) and Business Process Model and Notation (BPMN) are created to develop the workflow process model presented in Brunt, J. and K. Swenson. (2006).

3.1 XML-based Business Process Languages

Many XML-based business process languages have been developed in the past two decades to address all aspects of enterprise business process Xiao, Y., et al (2004). It is with many working examples and the acceptance from large commercial companies to use these languages to tackle the workflow related challenges Swenson, K. (2007). .Some of the most popular languages include BPML (Business Process Modelling Language) introduced by Business Process Management Initiative, BPEL (BPEL4WS) proposed by IBM, Microsoft and BEA, and also XPDL (XML Process Definition Language) developed by Workflow Management Coalition Shapiro, R. (2002).

The comparison between these XML-based languages has been presented by Shapiro, R. (2002). In the article, he firstly identified and introduced them and then clearly compared each of the components of these languages side by side in a table format. Within the comparison, the author has summarized that each of the languages are targeting to a slightly different user group. For BPML and BPEL, they are mainly focusing on issues important in defining web services, whereas XPDL focuses on issues relevant to the distribution of work. He supported his view by pointing out that activity attribute in XPDL is capable of specifying the resources and applications required to perform the activity. This would be a slight advantage for XPDL to be implemented in a collaboration system in comparison to BPML and BPEL Tran, H., et al. (2008). However, the author did not make any further elaboration on the decision of languages except stating their usages. Also it is important to note that some of these languages have been evolved and revised since the release of the paper in 2002, especially for the mentioned language XPDL 1.0, which current version XPDL 2.2 is now supporting to present BPMN 1.x and 2.0 in XML file format as well. Besides those changes, this article remains to be a good reference for languages comparison. The advantage of using XPDL over other languages can also be seen in the presentation slides prepared by Brunt, J. and K. Swenson. (2006), where it has expressed that XPDL 2.0 is particularly good in terms of extensibility. The language itself allows the developers to handle and store vendor specific features without affecting the compatibility. This can also be seen and further described in XPDL version 2.2 process definition. Developers are allowed to use "Extended Attributes" to extend the functionality of this specification to meet individual product needs.

Other than the possibility to extend the language, XPDL 2.0 and upwards is also capable to serialize BPMN (Business Process Modelling Notation) to an XML file, as stated in White, S.A. and S. BEYOND, (2003). This functionality is important as BPMN is a graphical notation representation of workflow processes which allow business people easily to understand and develop. In conjunction with that, a lot of existing applications are developed to translate from other XML processing languages to XPDL. Yuan, P., et al. (2008) have successfully demonstrated the possibility to develop a tool called WFTXB to interpret files from BPEL to XPDL with a good transformation result. The authors of the article decided to analyse the structure of the two languages and finding the similarity between them in order to write the transformation algorithms in pseudo-code.

XPDL is also one of the well accepted language of choice compares to other XML process languages. A lot of enterprises have chosen to use this as their language as it is easier to understand. And this is also presented in Yuan, P., et al. (2008) and to support this argument.

3.2 Business Process Model and Notation (BPMN) Approach

In addition to business process representation, BPMN also specifies how business processes should be executed in a standard way. The popular BPMN-based workflow engine is JBPM which is developed by Red Hat. The JBPM provides mature business process analysis. Following the JBPM engine, companies can use internal business workflow editor to design the specific workflow for their business.

However, using jBPM is seen like too big. It is because jBPM has comprehensive description for the business, which leads to the large size library. If the project is designed to be a light weight system, the large szie library will also cause the presure to the system capacity and running budget.

3.3 Challenge vs Solutions

One of the biggest challenge to implement a collaboration system for multi-parties business process is to apply and design one single workflow for everyone to be working on. This presents the difficulty to abstract other business processes for the current company within the workflow model. To get around this issue, one could simply define the business work flow process separately for different users. But this will create two problems as Armin Haller, M. M., *et al.* (2009) identified, choreography interfaces need to be created and synchronised with the workflow model manually, and the consistency of choreography interfaces to the workflow model is not guaranteed. Hence, this option is not feasible and there should be a way to address this issue directly.

There are several researches done on this specific topic over the years. Chebbi, Dustdar et al. (2006) have presented their solution of this firstly by allowing the clients define their own private workflow process and then interconnect them to present a whole holistic view for the large collaboration system. This solution allows for partial visibility of workflows and their resources to the required users, and this is important to solve the issue. The presented idea seems to be a good solution if the system is too complicated to connect in the first time or some of the components is not available in the beginning. However, this algorithm has not been deployed and tested as the time when the paper is written. Another solution is presented one year after the last one by Tran, H., et al. (2008). In this proposed idea, it is a reversed concept of the previous where this time it first accepts a large fully developed workflow as an input, and then generates executable views for each party as an output after processing has been done. However, in this very specific solution, only BPEL/WSDL language can be used within the toolchain provided. It can be a good solution and a reference to solve this very specific issue if the chosen language is BPEL.

A similar approach is suggested with the use of XPDL as input language by Armin Haller, M.M., *et al.* (2009). Within the paper, the authors presented the whole process of the separation of views into 3 main steps, XPDL to m3po, compaction rules processing, and finally mapping the individual views to an executable choreography interface model. By performing these 3 steps, it can translate the large XPDL workflow into separated executable code to run in other machines. However, it is important to note that the output of this approach is no longer in XPDL format anymore. Also the performance of this translated executable file or the extensibility provided by XPDL should be examined before using this approach.

In this paper, we also use XPDL as business process representations between business process design and tool and our collaboration system. However, we use XPDL heavily for multiple purposes in our design as follows:

- First, we use XPDL to represent the design of a business process and its initial states.
- We use XPDL to interact with users to get users' inputs.
- Once getting use inputs, we use the XPDL as a data container to carry the use inputs to send back to the backend business engine.
- According to the pre-defined business logic, the business engine would either conduct a set of actions (e.g., send a number of emails) or updating some states of the business process in XPDL to send back to the frontend.

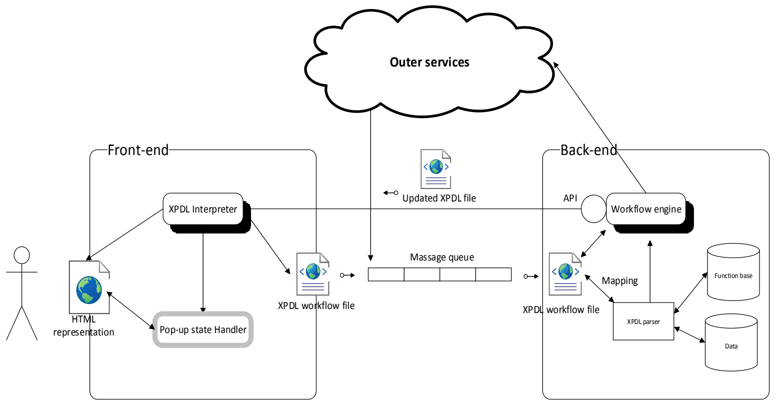


Figure 2: Overall architecture design of process based business collaboration platform.

4 PBCP ARACHECTURE DESIGN

The architecture of the system shown in Figure 2 is an overview of the entire platform. It includes three main components, front-end, back-end and outer services, and also some connection mechanisms in between each of these components.

4.1 Frontend UI

In the front-end, it is a simple model which proves the possibility of using XPDL to display the content in an interactive diagram format which combines with the functionality of displaying and collecting extra data. It is currently designed to show the content within a webpage due to ease of implementation, however, it is possible to use the same concept and apply to the design of mobile application as well. On the other hand, the back-end is receiving the workflow language XPDL and process it with its own design engine. On the far left hand side, it represents all the modules within front-end component, including HTML representation, XPDL interpreter, and pop-up state handler. All of these work together to form the front-end user experience. XPDL interpreter is responsible to interpret the useful information from the XML-based file to readable objects for the front end system, and vice versa to convert it back to XPDL before it being transferred to the back-end system. After the file is being interpreted, it sends the information to HTML representation module.

In here, it takes advantage of existing library for web application to render these information into an interactive diagram with CSS and JavaScript. Besides that, XPDL interpreter also passes state details to the handler for rendering the information as a separate pop-up window when the user clicks on the diagram. All of these are powered by the web 2.0 technology mentioned in the previous “Background” section. And finally, the pop-up state handler is one of most important part of the front-end component. Since the design of the collaboration platform is to be generic, this part of the implementation is to leave as much future design space for pop-up window as possible, while still allowing the handler to retrieve important information like form inputs at the end. For the current design, the business process can define both the pop-up window to be rendered internally as view in the web application, or dynamically retrieve the webpage externally with an URL and replace part of the HTML code to submit the result to the system.

4.2 Backend Business Process Engine

Back-end component is shown on the far right side of Figure 2. The main module of the component is workflow engine where it parses the XPDL file that it receives or stores in the database and perform the corresponding function as it matches up the rule engine inside. There are various ways to perform this task as suggested by previous people in their related work. However, this is out of the scope of this thesis as it focuses only on the front-end part of the system. Besides the workflow engine, it also contains database and function base to store all the information that it receives from the front end and also outer services. Once the back-end finishes processing the data and performing the related task in the workflow, it provides an API for others to retrieve and get the latest business workflow status from the system.

4.3 External Services

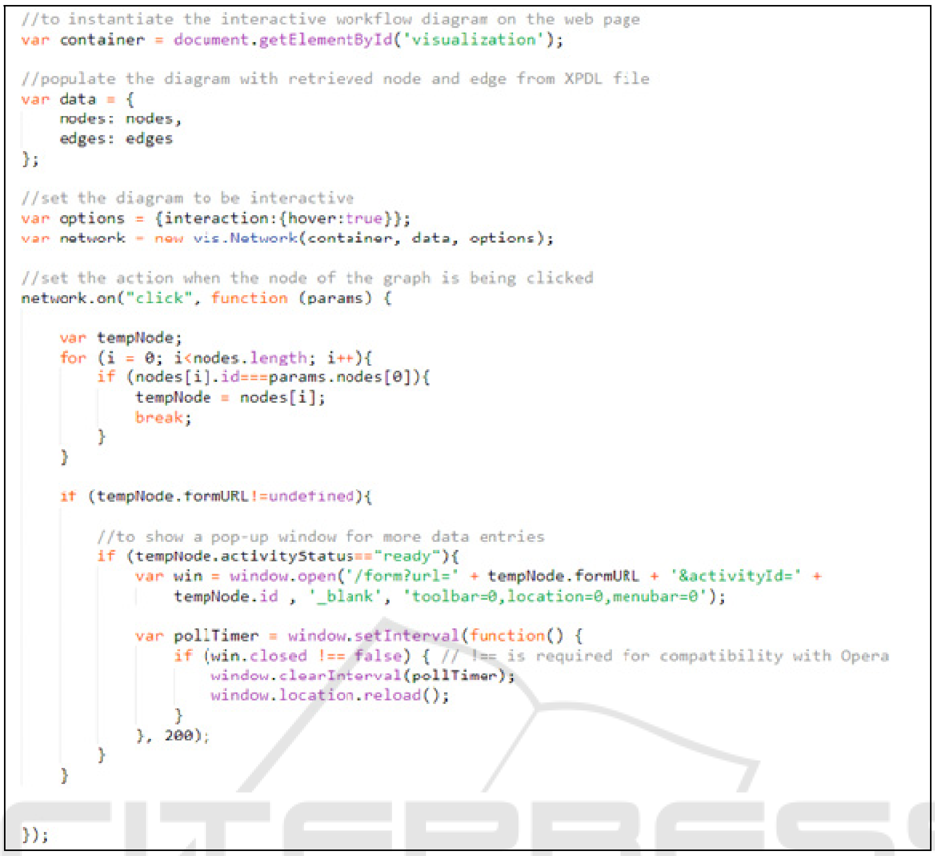
To allow the extensibility of the platform, the designed solution is generic to any business process workflows. Therefore, an outer services component is drawn in Figure 2. This is to represent the functions or actions provided by business partners to be performed after the mapping of rules engine. When these actions are finished performing, it would then transfer back the back-end for further processing and storing

4.4 Communication Channel

There are two main communication channels for the entire platform, which are message queue and also API. Both of them are provided by the back-end server to send and retrieve XPDL file. Message queue is chosen since the server requires buffer to handle the requests in an organized manner while not losing the data during the transfer. On the other hand, API is chosen for front-end to retrieve data since the time the customers go online can be different. And API is good for that use to retrieve the information stored in the system when it is needed.

5 PROTOTYPING AND TESTING

In this section, we provide technical details in implementing the above design as a proof concept prototype. We also evaluate this prototype in terms of usefulness and performance.



```
//to instantiate the interactive workflow diagram on the web page
var container = document.getElementById('visualization');

//populate the diagram with retrieved node and edge from XPDL file
var data = {
    nodes: nodes,
    edges: edges
};

//set the diagram to be interactive
var options = {interaction:{hover:true}};
var network = new vis.Network(container, data, options);

//set the action when the node of the graph is being clicked
network.on("click", function (params) {

    var tempNode;
    for (i = 0; i<nodes.length; i++){
        if (nodes[i].id===params.nodes[0]){
            tempNode = nodes[i];
            break;
        }
    }

    if (tempNode.formURL!=undefined){

        //to show a pop-up window for more data entries
        if (tempNode.activityStatus=="ready"){
            var win = window.open('/form?url=' + tempNode.formURL + '&activityId=' +
                tempNode.id , '_blank', 'toolbar=0,location=0,menubar=0');

            var pollTimer = window.setInterval(function() {
                if (win.closed !== false) { // !== is required for compatibility with Opera
                    window.clearInterval(pollTimer);
                    window.location.reload();
                }
            }, 200);
        }
    }

});
```

Figure 3: Interactive workflow diagram code in JavaScript.

5.1 Prototyping

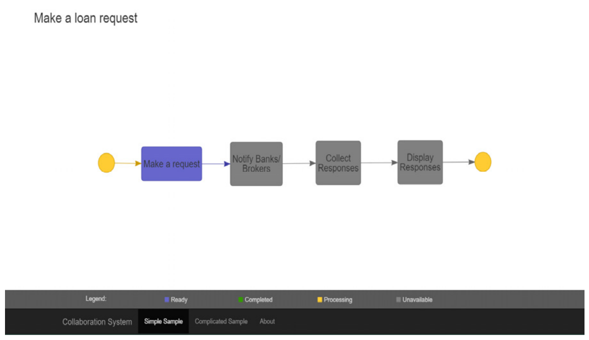
XPDL Interpreter is one of the main controller within the web application. The usage of the interpreter is to retrieve XPDL file from the server using API call, and process it and send towards both view and other controller for controlling pop-up window.

For the implementation of this module, the issue of unable to read directly from the XPDL file needs to be solved. For this challenge, an external library called “xml2js” is used, to first convert the XML-based language file to JavaScript object. It is then possible for the controller to process this information and retrieve the data inside. Due to the interchangeability of XML and JavaScript object, the integrity of the data in XPDL is preserved. And vice versa, when the pop-up window has finished collecting the data from the user, these information will be injected back to the JavaScript object and convert to XPDL file and send back to the server using message queue.

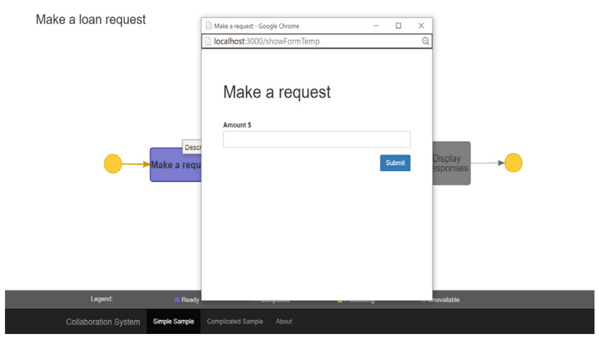
During the interpretation process, the interpreter is able to identify the selected components within the XPDL file. These include activities, and external attributes within the components. All of these information will then be passed on to the controller.

Figure 3 shows the skeleton of the JavaScript code to implement the interaction between users and a business process. The JavaScript code first gathers all the important information and create nodes and edges to represent activities and associations in XPDL file as shown in Figure 4 (a). Then it instantiates the diagram with these data and also enables the diagram to be interactive with click functionalities. When the corresponding activities are being clicked, a pop up window will be showing up and the controller of the pop-up window will be run and render the information as shown in Figure 4 (b).

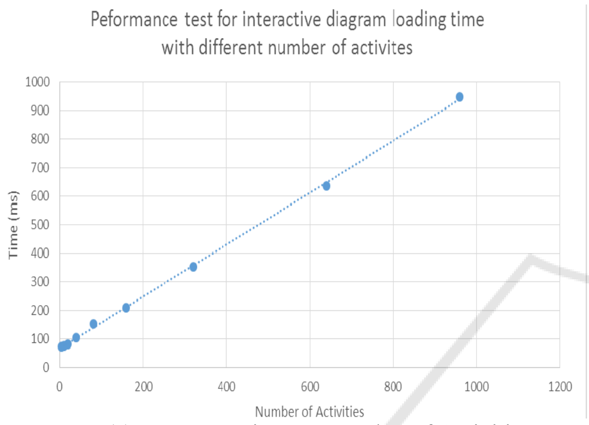
The backend was implemented using a RESTFul web service (Tomcat v7.0), a message queue (ActiveMQ v5.12.1), and a business process (JBMP v6.2.0). When the RESTful web service serves as a gateway to receive requests (e.g., XPDL), a generic event process engine is implemented using JBMP to process all event messages. A message queue is used to store the requests from all parties (The frontend web UI, as well as the outer services) and feed the process engine.



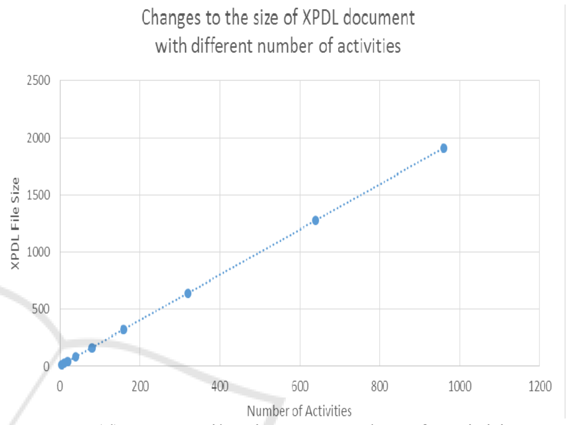
(a) A simple home loans business process



(b) Interaction within an activity



(c) Response Time vs. Number of Activities



(d) XPDL File Size vs. Number of Activities

Figure 4: Prototype UI and performance testing results.

5.2 Performance Testing

We also conduct some testing to evaluate the performance and scalability of the XPDL render in terms of message sizes and processing time. The testing results are shown in Figure 4 (c) and (d). As can be seen, both the size of the XPDL files and the response time of processing these XPDL files are reasonable as the number of activities in a business process increases till 1000 activities, which is very rarely in real life. This means our solution can scale for most of business process-based collaboration applications.

6 CONCLUSION

In this paper, we present a novel business process-based collaboration platform. This work is well motivated with real-world applications in context of multi-party collaboration. We abstracted the software architecture of this kind of systems as a business process-based UI and a generic event process engine. A design and implementation are provided as a proof of concept. We demonstrated and tested the prototype to show the usefulness and scalability of our technology. We believe our technologies can be easily reuse for different collaboration applications with less efforts than using classical web applications, due to our generic software architecture abstraction.

REFERENCES

Blaire Palmer (2014) Collaborating with SMEs - Advice and Recommendations for Large Enterprises, *Whitw Paper That People Thing*

Salesforce (1999): www.salesforce.com/au

Surya Nepal and Shiping Chen (2011): Dynamic Business Collaborations Through Contract Services. IJSSOE 2(4): 60-82

Xiao, Y., et al (2004) Research of Web services workflow and its key technology based on XPDL. *IEEE International Conference on Systems, Man and Cybernetics*

Armin Haller, M. M., *et al*. (2009) Brahmanada Sapkota, Manfred Hauswirth, From workflow models to executableWeb service interfaces. *2009 IEEE International Conference on Web Services*

Xin, J., X. et al. (2007). The design and implementation of XML-based workflow engine. in Software

Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*, 2007. SNPD 2007. Eighth ACIS International Conference on*. 2007

Swenson, K. (2007) *The Tipping Point for XPDL*, http://social-biz.org/2007/03/19/the-tipping-point-for-xpdl/.

Shapiro, R. (2002) *A technical comparison of xpdl, bpml and bpel4ws*. Cape Visions

Brunt, J. and K. Swenson. (2006) *XPDL 2.0 and BPMN 1.0 Tutorial*. http://www.fujitsu.com/downloads/INTSTG/webinars/Fujitsu-Interstage-XPDL-BPMN-Webinar.pdf.

White, S. A. and S. BEYOND, (2003) *XPDL and BPMN*. Workflow handbook: p. 221-238.

Yuan, P., et al. (2008) WFTXB: A Tool for Translating between XPDL and BPEL. *10th IEEE International Conference on High Performance Computing and Communications, HPCC'08*.

Chebbi, I., S. et al. (2006) The view-based approach to dynamic inter-organizational workflow cooperation. Data & Knowledge Engineering. 56(2): p. 139-173.

Tran, H., et al. (2008) View-Based Reverse Engineering Approach for Enhancing Model Interoperability and Reusability in Process-Driven SOAs. *In:* MEI, H. (ed.) *High Confidence Software Reuse in Large Systems.* Springer Berlin Heidelberg.

ActiveMQ v5.12.1 (2015): http://activemq.apache.org/

jBPM v6.2.0 (2015) open source: http://www.jbpm.org/