THE UNIVERSITY OF NEW SOUTH WALES

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

Software Engineering Thesis

Towards Personal Process Management

Ernest Lie

z3186778

elie464@cse.unsw.edu.au

Supervisor: Helen Hye-Young Paik

hpaik@cse.unsw.edu.au

Prashanthan Ranjan

z3219211

prashr@cse.unsw.edu.au

Assessor: John Shepherd

jas@cse.unsw.edu.au

29 May 2012

Abstract

As business processes continue to be used extensively by large organizations, the importance of considering the individuals involved in these processes is constantly being over looked. This paper aims to define of the concept of a personal process and the idea of personal process management.

In this paper, we also propose a system designed to help users manage their personal processes. The system will involve process designers to specify and design personal processes, and end-users, which select and download these processes for personal use. The End-users however will effectively be downloading an interactive to-do list derived from the process, which provides context aware suggestions such as which task is closer or more important in the process. As users continue to use their smartphones to perform most tasks, a mobile–based application has been developed to implement this system.

Acknowledgments

We would like to thank Helen Paik, our supervisor, for her constant guidance throughout this project. She has provided us with invaluable insight into this research topic and her constant stream of suggestions and possible improvements has certainly motivated us to take our solution to the next level.

We would also like to thank John Shepherd, our assessor, for taking the time to provide useful feedback through the course of this thesis. His constructive criticism was always taken onboard in order to better the final solution.

And last but not least, we would like to thank our respective families and friends for supporting us through this ordeal and for making sure that we saw it through to the end.

Contents

List of Figures

1 Introduction

1.1 Motivation

Business processes are a set of structured tasks set out to achieve a business goal. Personal processes on the other hand are repetitive day-to-day activities relevant at a personal level. At the moment there are currently many tools that support business process execution, but none for personal processes. We hope that this will change as a result of our thesis.

The lack of support for personal processes can be attributed to the different nature of personal processes. Personal processes differ to business processes in the sense that many of the tasks that personal users prefer to do cannot be automated. This means that personal processes are by nature, more manual and are mostly partially automatic in their execution. Personal processes are also in a sense, much more flexible in the order of how we would like to do things. Users may prefer to do things out of sequence while business processes are much more strict and rigorous in how they are defined. These two concepts mean personal processes are very loosely structured, highly variable and that applying the automated tools currently out there for business processes would not be suitable on a personal level. As a result, there is very little support provided for those individuals attempting to complete personal processes. This forms the basis of the motivation for our thesis.

1.2 Goals

It is well known in the workflow and business processes area that supporting flexible processes is extremely difficult and is still an open research problem. We recognise that there are different types of personal processes depending on how much control or input the user has over them. For example, a process to cook a meal could vary to great extent between users as users may choose to perform steps in the recipe in a different manner or ignore steps completely. In contrast, a personal process can also involve for example a job application, which users cannot control everything

exactly with their own freedom as it is still dependent on the business process that underlines the job application process. These issues need to be considered when designing a personal process management system.

The aim of this project is to create a tool that will allow users to effectively manage personal processes. The tool will convert a personal process to a list of tasks that the user needs to complete to achieve the goal and will help the user in completing these tasks by using context aware suggestions, task filtering and sorting options. It will allow users to share and re-use knowledge that is gained from completing personal processes to benefit future users of the same personal process.

1.3 Thesis Structure

1.3.1 Background

This chapter details the background information that is necessary for this thesis. It includes a review of two papers that deal with personal workflows and personal process management. It also details features of two popular task management applications that were used in this thesis.

1.3.2 Design

This chapter establishes the design approach that was followed for this thesis. It includes an overall system view, a brief on the system architecture and the main components such as the Process Manager, YAWL Parser, Repository Browser and Analytics Manager.

1.3.3 Implementation

This chapter puts forward an implementation that aims to demonstrate the main concepts presented in this thesis. It details the design pattern used, the architecture and the resulting final outcome.

1.3.4 Evaluation

This chapter presents an assessment of the thesis and the final system. It includes details of the evaluation plan and a summary of the results gained from the evaluation runs.

1.3.5 Conclusion

This chapter sums up the main topics presented in this thesis. It also provides brief conclusions to the other chapters in this report.

1.3.6 Future Work

This chapter details the possibilities of additional work that can be undertaken in this research area which may improve the system that is presented. It goes into detail about the main areas that if improved on could have a significantly positive impact on the final system.

2 Background

This section looks at some of the related work which we drew inspiration from in order to complete our solution. It includes a review of two papers that deal with the concept of personal process management and two application that deal with task management. It also includes our justification for choosing the process modelling tool that our system will be developed to work with.

2.1 Related Work

2.1.1 *Personal Workflows: Modelling and Management* (San-Yih Hwang and Ya-Fan Chen), Conference Paper, 2003

San-Yih Hwang and Ya-Fan Chen propose a model for specifying and querying personal processes. It allows for a mobile user to specify their own personal processes on the system and then run queries on these personal processes so as to be able to assist them with planning their activities while on the move.

One of the key ideas gained from this paper is the fact that personal workflows can be represented as a set of tasks using a graph data structure. Each node in the graph has associated metadata that will help the system determine interdependencies between tasks.

The paper clearly identifies issues specific to personal process management as opposed to business process management. It clearly identifies that personal processes are related by executable location and time, and that, unlike business processes, users generally do not impose rigid rules to the control flow of these tasks. It goes on to state that the any personal workflow management system should only remind or provide suggestions to the user, and not to impose the execution of the task onto the user as evidenced on business-oriented workflow management systems.

We hope to make use of the principle of using graphs to represent the personal process and

improve upon this system by providing context aware recommendations on tasks to the user. This current system only shows a list of tasks to complete to the user and although contains contextual information for each task, it does not go ahead and provide recommendations to the user based on the context.

2.1.2 *Personal Process Management: Design and Execution for End-Users*(Ingo Weber, Hye-Young Paik, Boualem Benatallah, Coren Vorwerk, Liangliang Zheng, and Sungwook Kim), Technical Report, 2010

This paper is the backbone of where most of our ideas come from. It attempts to define what a personal process is, how it differs from business processes, the technologies required to effectively come up with a solution, the architecture and more.

One of the main things we took from this paper is the definition of a personal process. The paper defined that a personal process is a business processes as experienced and described by a single person. It also notes that personal processes differ in business processes in that they are not as automated, and require more flexibility and simplicity. They attempted to create their own personal process modelling language called PPML, which is quite relevant and notes the control structures that it finds more important than others. It also described its architecture and solution: FormSys, an execution program which executes PPML through a variety of tools used externally (JQuery, Acroform, Intalio) and models some scenarios on how they can solve personal processes.

Our project extends from their core idea, but we adjust our approach to a mobile and to-do list environment instead. One of the key ideas from the paper that inspired this is that it stated to cover the interleaved parallel routing control flow pattern (where one user completes the task, but must complete all tasks at least once) a checklist would be appropriate. We will also be using a pre-defined modelling language (YAWL) instead of PPML as PPML does not contain a toolset

that can easily generate a description file of a designed process, which we will need to convert to a To-Do list.

2.1.3 Remember the milk

Remember The Milk (RTM) is a web-based task management application with an open API that is targeted towards providing the services through as many means as possible. It can therefore be easily integrated into other web, mobile and desktop applications.

One of the key aspects of the application is that it allows the user to specify locations that can then be assigned to tasks so that the user is able to view tasks by location on a map. It also allows the user to specify an estimated completion time for each task along with many forms of context.

Some criticism we offer is that multiple tasks cannot be viewed on the map for comparison, nor does it allow users to create subtasks - showing control flow can be improved.

2.1.4 Checkmark

Checkmark is a to-do list manager app on the android mobile platform. It allows users to create simple lists, group tasks into sub-tasks, group lists into different categories (entertainment/shopping etc.), and even assign simple hyperlinks to applications like Gmail and Google Maps if an email task was required, or a task had a particular location assigned to it. Bonus functionality includes the ability to sync to Google Tasks, Calendar (their own separate Checkmark Calendar) and you can create your own templates.

What we picked up from this app the most are the interface designs, how a page links to another, and the ability to assign a location and email context to a task was certainly useful.

There are some limitations of this app however:

- There is no view for viewing all the tasks with a location simultaneously so you can see which task is closer. You can't see which tasks are closer to a deadline at present.
- A due date can be assigned to a task, but only by a certain date. It would be useful if a user can assign a time deadline as well, or a time frame. The key idea we would like to extend on this is that users may want to design not only a checklist, but impose certain conditions on which tasks to do first, and which ones are the most convenient

2.2 Modelling tools: BPMN vs. YAWL

Business Process Modelling Language (BPMN) and Yet Another Workflow Language (YAWL) are both business process-modelling languages. They are used to model business processes and have an execution engine underneath that attempts to implement the process.

BPMN is well known. It is in fact a standard supported by many well-known software vendors such as IBM, HP and Apple. YAWL is as described, another workflow language, and although not a standard, it covers a huge scope of business processes or workflow patterns that can be created.

We chose YAWL and the reasons for this are:

- We found it is more simple to use, and simple to understand
- YAWL has formal semantics supporting it so that interpretation should be precise and processes verified for implementation
- YAWL is known to have more comprehensive support for different workflow patterns (control flow)
- A BPMN to YAWL plugin is available
- The XML output of YAWL, which is necessary for our translation engine to convert into a mobile language, is easier to understand and control. YAWL also has less constructs(e.g.

tasks, gateways, endpoints, events etc.)

Disadvantages:

- Not a standard, an academic project
- The joins and splits code in XML is not very intuitive

Differences:

- BPMN has connector chains (multiple gateway chaining together), while YAWL doesn't. YAWL includes joins and splits in the task objects.
- YAWL requires processes to have only one start and one end condition. In BPMN, multiple start and multiple end events are allowed.

It is important to note that both languages have their own issues - a list of BPMN issues and YAWL issues can be found under the 2 links:
BPMN -`http://www.omg.org/bpmn/BPMN_2-0/BPMN_2-0_Issues.htm`
YAWL - `http://code.google.com/p/yawl/issues/list`
As we will be dealing with a single person process, a lot of the issues do not apply to us and so we will not be focusing on their analysis in the scope of our project.

3 Design

This chapter establishes the design approach that was followed for this thesis. It includes an overall system view, a brief on the system architecture and the main components such as the Process Manager, YAWL Parser and Analytics Manager.

3.1 Overall System View

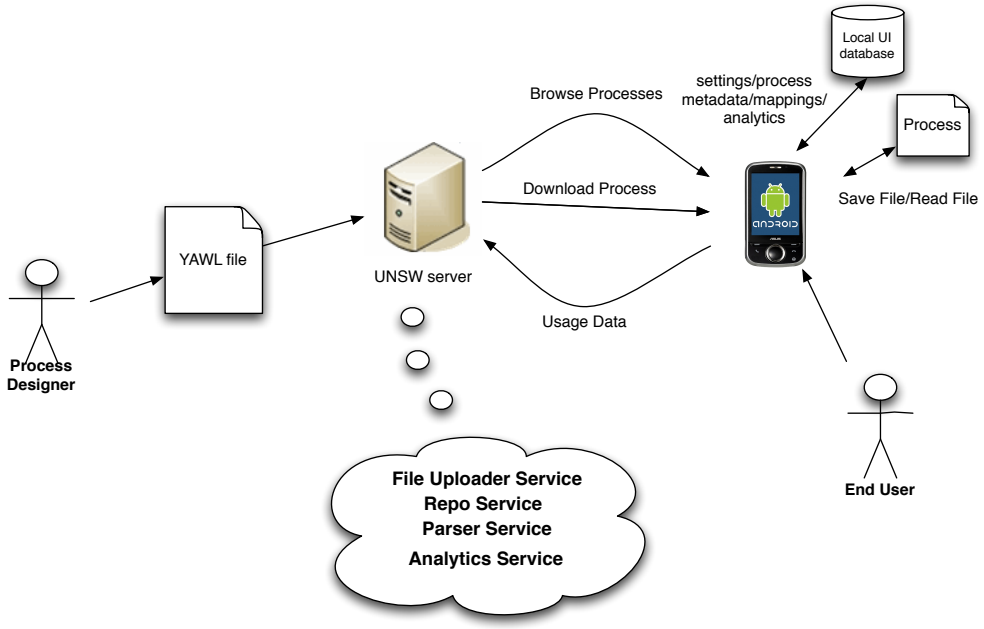

Figure 3.1: An overall system view of the flow and the modules involved.

This diagram provides an overall system view depicting the users of the system, the various components involved and the flow of data throughout the system. There are two main users in the system. A *Process Designer* is responsible for designing the process using the YAWL modelling language. The process would incorporate the context that is associated with each task as well. The *YAWL file* is then uploaded on to the *UNSW server* using the *File Uploader Service*. Some

information about the process such as name and description is also added. The file is stored on the server and gets detected by the *Repo Service*. The *End User* is then able to browse the list of processes available in the repository using the application. Once they have chosen to download a particular process, the *Parser Service* obtains the YAWL file from the *Repo Service* and decodes it into a process object and sends it to the mobile device. The *Process* is stored as an XML file in the phone's filesystem. The user's settings along with analytics are stored in the *Local UI database* Once the user has completed the process, the analytics that was stored on the phone's database is now synced with the server using the *Analytics Service*.

3.2 System Architecture

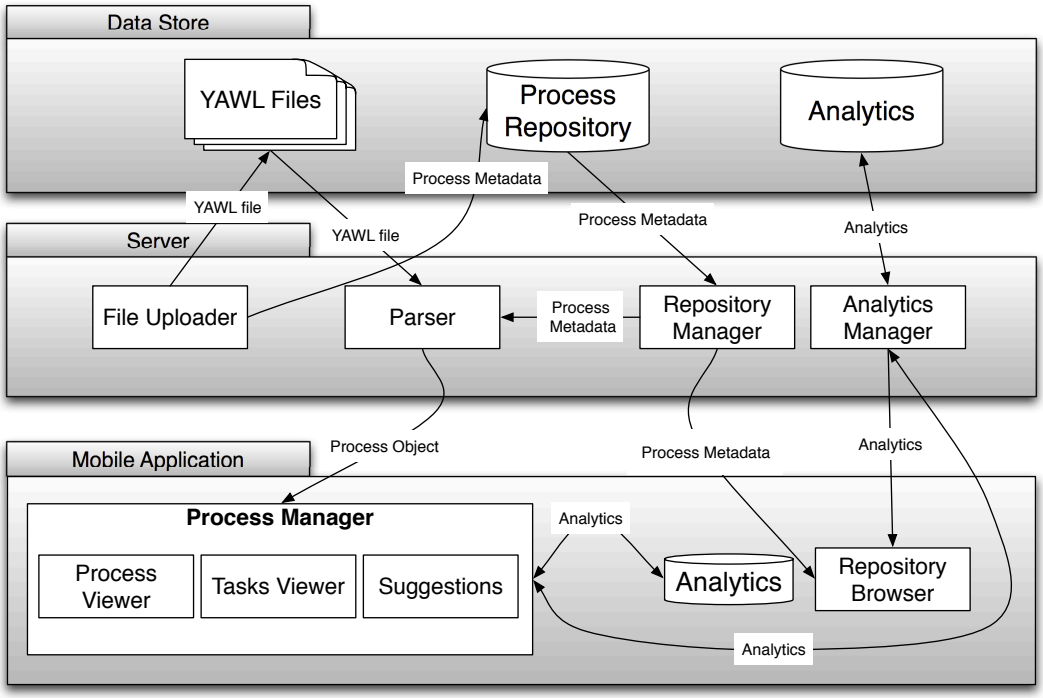

Figure 3.2: High level view of the the system architecture including database, server and mobile components.

This diagram provides a more detailed view of the three layers that form the system. The *Data Store* layer contains the *Process Repository* and *Analytics* databases along with the *YAWL Files*. The *Server* layer contains the *File Uploader*, *Parser*, *Repository Manager* and *Analytics* web services. The *Mobile Application* layer contains the *Process Manager* and *Repository Browser* components. The key components in this diagram are described in detail below

3.2.1 YAWL Parser

The Parser component is a web service stored on the server. Essentially, its task is to convert a file created by the process designer, into the appropriate Process object that is readable by the Process Assistant mobile application. As the parser is an interface, it can attempt to parse various different files, but in our implementation, as we have decided to use YAWL as our modelling language, we have implemented a YAWL parser.

The Parser component firstly requires a file to convert. This can be stored locally, but the way we have implemented it, we wrap the Parser into a web service that connects to our Repository Manager web service to be supplied the location of the file. The file is then converted into the Process object. We also add metadata such as the file–size, process description and id into the Process object and we obtain this data by connecting to the Repository Manager web service once again. When the Parser web service is called, the Process object that we create is serialised into XML and sent across.

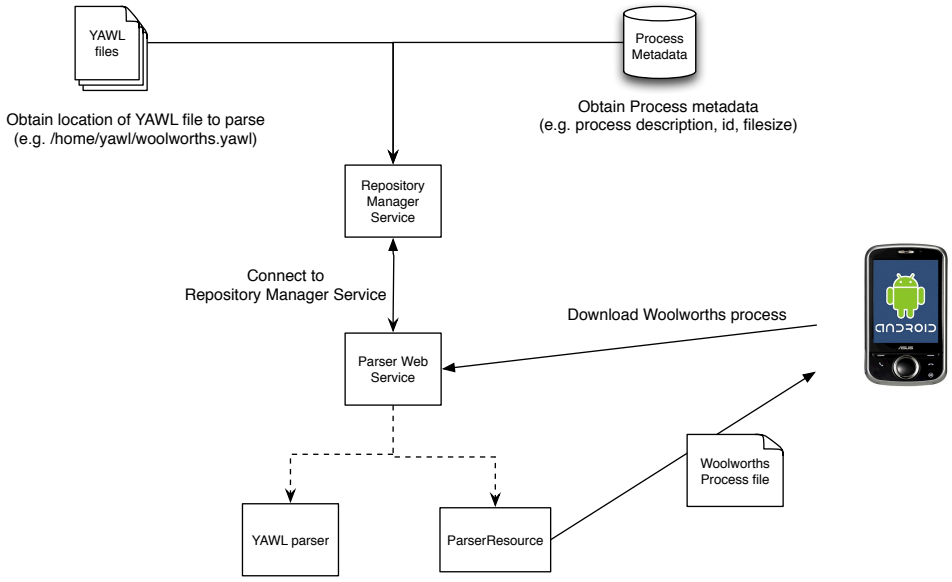

Figure 3.3: Parsing of Woolworths YAWL file

In terms of how our YAWL parser works, it is done simply by Document Object Model(DOM) XML parsing. As YAWL files are XML files, DOM parsing can be used. The main functionalities of the YAWL parsing includes:

- converting YAWL Task elements into Process Task elements
- converting YAWL input and output conditions into Process Vertex elements
- converting YAWL flowsInto elements into Process edges
- updating Process Task information(e.g. task description, co-ordinates, start and end dates) by reading YAWL documentation tag
- updating Process split and join codes by looking at the flowsInto elements

Figure 3.4: YAWL to Process conversion elements

3.2.1.1 Documentation tags

The documentation tag is the feature in YAWL that is used to add contexts to tasks. The following is the various tags that can be used to add context to tasks.

des: This is the description of the task and can include links to resources if the user needs more information about the task.

lon: This is the longitude of the task's location.

lat: This is the latitude of the task's location.

int: This states whether the an internet connection is required to complete this task.

std: This is starting date of the task. i.e. the date from which the task is available.

stt: This is starting time of the task. i.e. the time from which the task is available.

end: This is ending date of the task. i.e. the date from which the task is no longer available.

ent: This is ending time of the task. i.e. the time from which the task is no longer available.

dur: This is the duration of the task. i.e the amount of time the user would require to complete this task.

These tags are not required so the application will generate the context of the task based on the tags that ARE provided by the process designer.

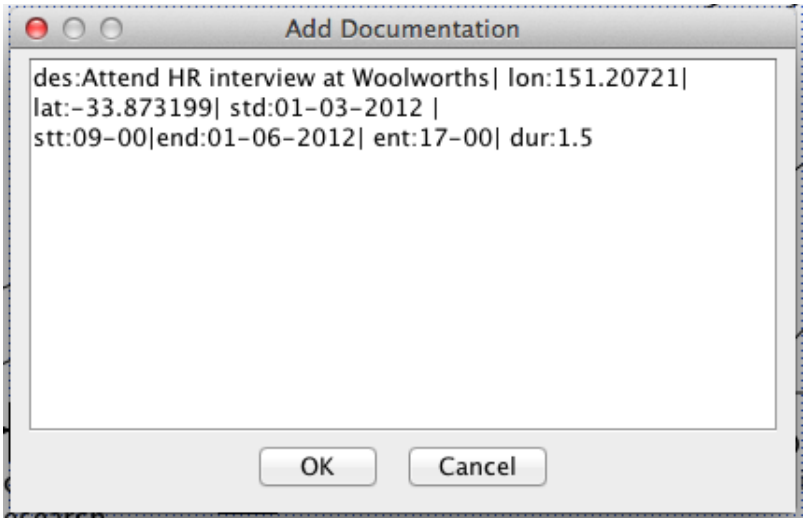

Figure 3.5: YAWL Documentation tag editor view depicting the context of a task

3.2.2 Process Manager

The Process Manager is the backbone of our system. It is the engine that allows the user to manage and interact with the Process objects that define the personal processes of our system. It aims to display to the user 3 views: Processes, Tasks and Suggestions.

Processes: Processes are the processes designed by process designers. In this view, we show a list of the processes the user has downloaded into his/her process list. The view also includes for each process: the list of tasks associated and process information such as a description, author

and upload date.

Tasks: Tasks are sub elements of processes that describe a task a user is to complete. The user in this view can interact with all the tasks from every process they have downloaded as well as view individual task information such as its location, description, duration and start and end dates. Tasks can also be sorted by distance or time in this view for better manageability for the user.

Suggestions: Suggestions are a set of tasks displayed for the user that we have suggested, based on a set of rules.

To drive these views the Process Manager is expected to do the following:

- Read the Process File objects
- Read the phone database for the cache of which tasks we are currently up to
- Read the phone database for the current state of tasks
- Read the user settings for the application
- Display information to the user
- Write any interactions to the associated databases(including analytics)
- Submit analytics data when the user has completed a process

Users interacting with a view would be able to confirm tasks or ignore them(and complete them later), as well as being able to change the view settings either through sorting or filtering. The views also co-exist and so interactions displayed on 1 view would be reflected automatically in the other views. In addition to the above functionality, the Process Manager displays Google Maps views with respect to the current location of the user and the tasks or processes they chose.

3.2.2.1 Suggestions

The Suggestions component of our system has been designed to be modular - it is an interface whereby different implementations of suggestions can be adopted and used. Our current implementation involves the following:

- filtering tasks so that it should be time valid(current time + task duration is within task end date)
- filter tasks so that it should be internet valid(if a user has no internet, show only tasks that don't require internet connection)
- sort the tasks based on the closest deadline
- sort the tasks based on the closest location

Should there be a want to change and improve the suggestions component, such as incorporating past history or other contextual information, this component can easily be modified to return a different list of tasks to display than our current implementation.

3.2.3 Repository Browser

The Repository Browser is a component on the mobile application that allows the user to view details about the processes available for download in the repository. All details about the process that was entered by the process designer when the file was uploaded to the system using the *File Uploader* web application is shown to the user. The browser also connects to the *Analytics Manager* service to obtain comments and the average rating provided by other users for the particular process that the user has chosen to view. If the user chooses to download the process, a check is performed to make sure that the process is not currently in the user's local process list and has not been completed. If this is the case, the user is warned about this condition and requests confirmation to go ahead. If the user chooses to go ahead, database entries regarding the process in the application are deleted and the process manager is signalled to download the process using the *Parser* service.

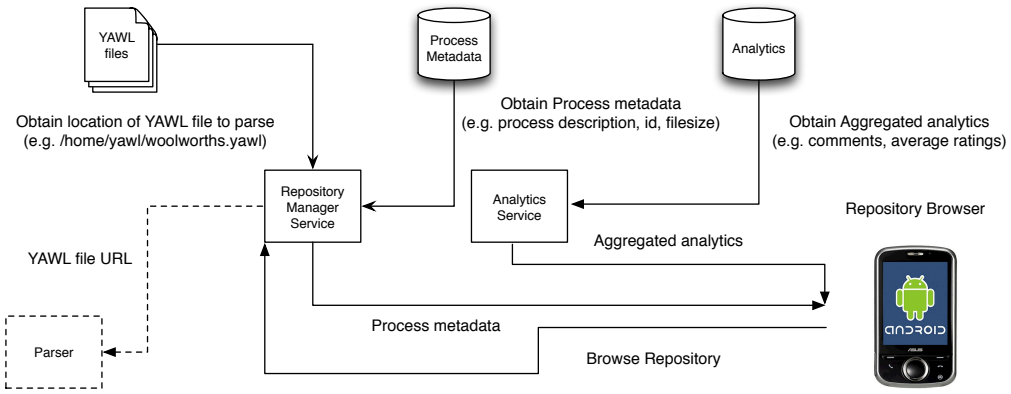

Figure 3.6: Repository viewer elements

3.2.4 Analytics Manager

The *Analytics Manager* is the component that is responsible for collecting usage data about how the user uses the application and the processes. When a task appears on the to-do list of the user, the timestamp is recorded along with the task in the analytics database located on the mobile. Once the user has confirmed the task, another timestamp is recorded, so it is possible to calculate how long the task took and by repeating this for all the tasks the process, the total time taken to complete the process can be calculated. If the user chooses to ignore the task, no end time is recorded for the task so it is also to find out which tasks the user has ignored. Once the user completes the process, the application asks the user to provide a rating for the process and some comments. This information is also stored in the analytics database. The database is then synced with the analytics database using the *Analytics Manager* service. We believe this information will prove useful to process designers when estimating duration of tasks and general pattern that most users will follow when completing the task. The rating and comments is also fed into the *Repository Manager* so that users can see the average rating and comments for a process before they choose to download it.

4 Implementation

This chapter puts forward an implementation that aims to demonstrate the main concepts presented in this thesis. It details the design pattern used, the architecture and the resulting final outcome.

4.1 MVC Design Pattern

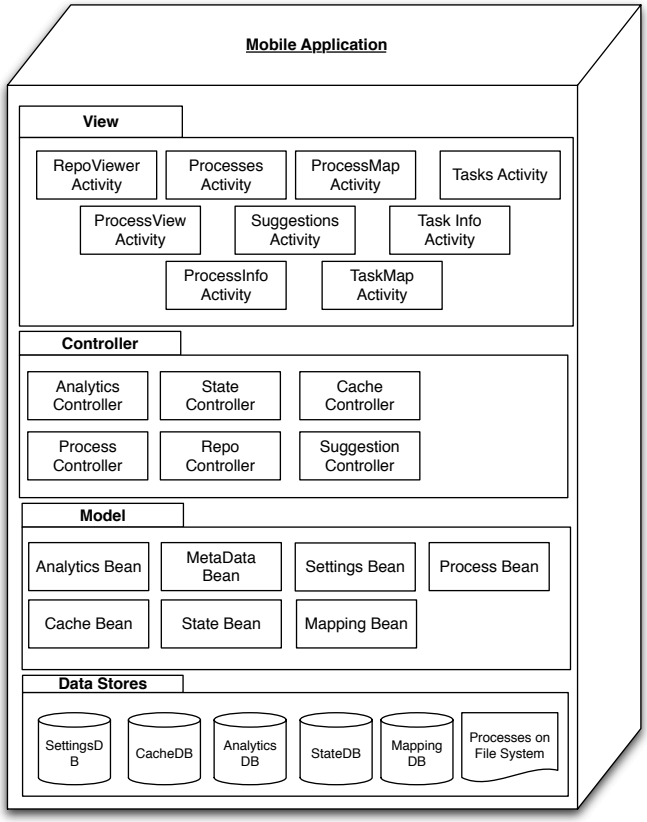

Figure 4.1: System Architecture view of the mobile application

The Model–View–Controller (MVC) design pattern has been used in this application as it allows the data to be logically independent from how it is displayed to the user. This allows the interface

to constantly change and evolve without affecting the logic and flow of data within the application. The following describes the layers involved, the roles they play and how they interact with each other.

4.1.1 Data Stores

The *Data Stores* layer contains the components that store application related data in databases and the local filesystem. It comprises of the following:

SettingsDB : This database stores the user's current view settings. This includes the selected sorting options, filter options and task view options.

CacheDB : This database stores the currently available list of valid tasks for the user. This is stored for performance reasons as searching through each of the processes to determine the valid tasks would take quite a while.

AnalyticsDB : This database stores data related to how the user is using the processes within the application. It stores the process ID, task ID, task start time, task end time, comments and ratings.

StateDB : This stores the state of all the tasks such as whether they are ignored or confirmed.

MappingDB : This database maps a processID to the process xml located on the local filesystem. This forms part of the Hybrid Architecture.

Process on File System : This is collection of process objects that have been converted to XML files and stored on the phone's filesystem.

4.1.2 Models

The *Model* layer contains the data structures that represent the application's state and are used to transfer data around the application. It consists of the following:

CacheBean : This is transfer the information regarding the currently available valid tasks.

MappingBean : This is used to link a process ID with its associated xml file.

StateBean : This is used to handle the state of the tasks.

AnalyticsBean :This is used to handle the statistics of the tasks and processes on the application.

MetaDataBean : This stores information regarding a process such as name and description.

SettingsBean : This handles the user's application settings.

Process Bean : This stores the process object as graph. It includes all the tasks and the flow of the process along with additional details about the tasks themselves.

4.1.3 Controllers

The *Controller* layer translates user action into operations on the model and the data stores

Analytics Controller : This manages the analytics database which records statistics about how the processes and tasks are used.

State Controller : This manages the state database that deals with the state of the tasks and processes.

Cache Controller : This manages the cache database that stores the list of currently available and valid tasks.

Process Controller : This handles all functions associated with managing processes and tasks. It has the ability to find valid tasks, apply filters to tasks, sort tasks based on the user's choice, provide information about processes and tasks and also connects to the Parser Service to download new processes.

Repo Controller : This allows the application to connect to the Repository Service in order to browse the processes on the repository and view more details about them.

Suggestion Controller : This is used to provide a list of best possible tasks to complete based on the context of the tasks and the current context of the user.

4.1.4 Views

The *Views* layer provides the interface elements through which the user interacts with various aspects of the system.

RepoViewerActivity : This is the view the user sees when they browse the repository showing a list of all the processes on the repository.

ProcessesActivity : This view shows the processes downloaded onto the phone and the tasks associated with them. The user has the option to confirm and ignore tasks and access the options menu for sorting and filtering of tasks.

ProcessMapActivity : This shows a map of all the task's locations associated with a process. It shows the user's current locations along with markers for each of the task with a name and description.

Tasks Activity : This shows a list of all the tasks that are available with options provided for sorting and filtering

ProcessViewActivity : This is shown when the user selects a process from the repository. It shows additional details about the process such as a brief description, date created and file size.

SuggestionsActivity : This is the view that shows the application suggested tasks to the user

TaskInfoActivity : This provides additional information about the task such as the description and availability of the task.

ProcessInfoActivity : This shows additional information about the process such as a brief description.

TaskMapActivity : This shows a map of the task's location along with the name and description and the user's current location.

4.2 Hybrid Architecture

To process the user interactions with our mobile application and drive the Process Manager component, we need to save the state of our Processes. As our application is mobile, this has to be done optimally and how we resolved this is through a hybrid architecture.

We figured that because we receive from the Parser web service a Process object(a graph), it was not optimal to save the whole graph into the android database. Eventually a user would have many Processes, and storing many of these into Android would take up large amounts of space, longer amounts of read/write time, and that a better solution would be to just save the interactions the user makes. So how did we do this?

We created a new database on Android called stateDB, mapping each Task to a state(isConfirmed, isIgnored). When the user confirms or ignore a task through the mobile app, we write to the stateDB. When loading our process to the user interface, we read the original Process file, read the stateDB, and allocate the state to each Task. Hence, our application uses both a File system and database to optimise performance.

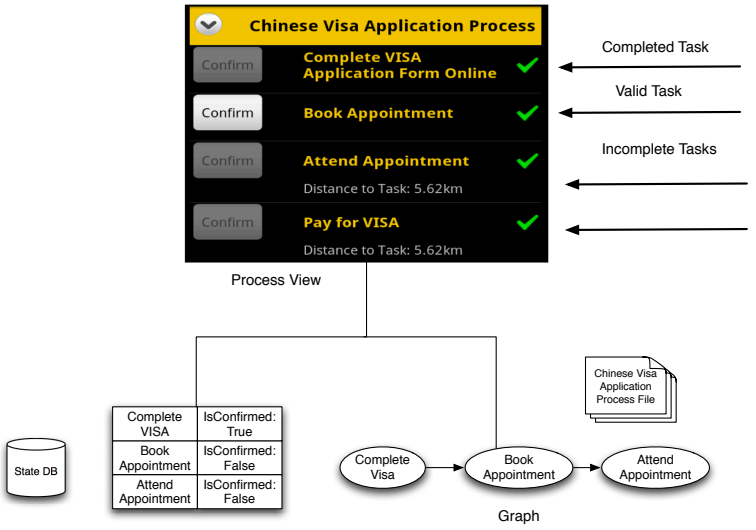

Figure 4.2: Hybrid Architecture of StateDB and Process File

4.3 Process Flow of Mobile Application

This section describes the process flow of our Mobile Application. The application was implemented on Android and we this section will describe our implementation using Android terminology such as views and activities. The application has 2 main components - The Processes Manager and the Process Repository:

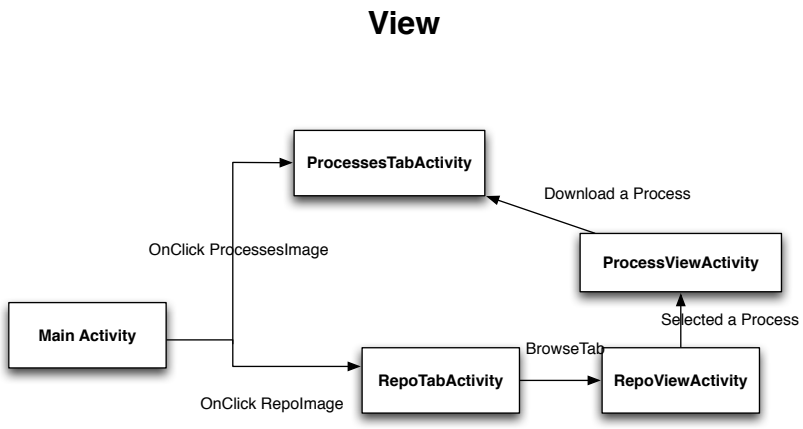

Figure 4.3: Process Flow of Downloading Processes

The app first starts off on the main screen - MainActivity. If the user clicked the 'Download processes' image, the user will be taken to the RepoViewActivity, which displays the list of processes stored in our repository. Clicking on a single repository will bring him/her to the ProcessViewActivity where he/she can find more information about the Process.

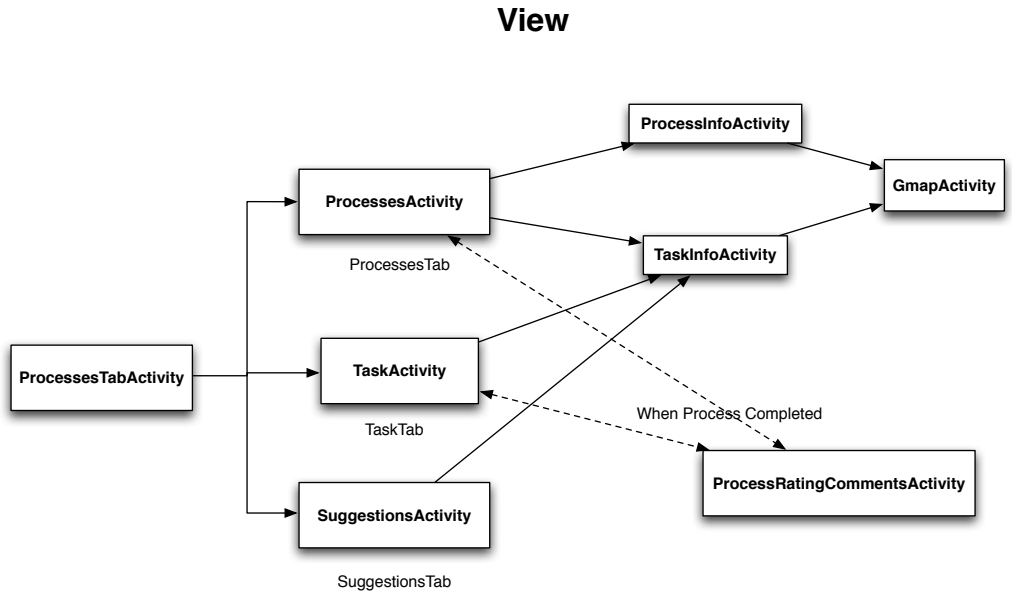

Figure 4.4: Process Flow of Viewing Processes List

If the user clicked the 'View Process List' image, the user will be taken to to the ProcessTabActivity consisting of 3 tabs: Processes, Tabs and Suggestions. Should the user long-press the tasks in any tab, the user will be taken to the Task Info Activity for more information about the task, and in the case of the Processes Tab, should the user long-press a process, he/she will be taken to the Process Information Activity displaying similar meta-data information about a Process. In either case, the user can view a Google Maps page displaying the tasks or single task with respect to the current location for location information if it is available.

4.4 Graph Traversal Algorithm

The Process objects used in our mobile application are essentially graph data structures. To process the graph, we need to traverse it so that we could for example, find what stage we are up to in the Process to display valid tasks. To do this, we chose a breadth first search algorithm - but modified to process the splits and join constructs of YAWL.

```
public List<Task> getValidTasksforProcess(int ProcessID) {

   List<Task> validTasks = new ArrayList<Task>();

   Queue<Vertex> queue = new LinkedList<Vertex>();
   queue.add(getProcess(ProcessID).getVertex(START_NODE)); // add start node

   while (!queue.isEmpty()) {
      Vertex v = (Vertex) queue.element();
      queue.remove();

      if (v.getClass().equals(Task.class)) {
         Task task = (Task) v;
         updateSplitsJoins(task); //check task split/join

         //check for splits and joins
         if(splitJoinFlag == AND){

            //snip: process AND split
         }

         else if(splitJoinFlag == OR){
            //snip: process OR split
         }

         //check for splits and joins
         else if(splitJoinFlag == XOR){
            //snip: process XOR split
```

```
        }

        //no split or join code
        else{
           if (psc.isConfirmed(ProcessID, task.getId())) {

              for (Integer i : task.getOutvertices()) {
                 if (i != END_NODE) {
                    if (!queue.contains(getTask(ProcessID, i))) {
                       queue.add(getTask(ProcessID, i));
                    }
                 }
              }
           } else {

              validTasks.add(task);
           }
        }
     }

     //snip
       }
     return validTasks;
  }
```

As seen in the code, we create a queue. We add the root node of the graph(the start node) to the queue, and find adjacent nodes. If the adjacent nodes are completed, we continue to find more children, otherwise, we add the node to the list of tasks. If the node, previous node or node ahead have any splits or joins, we process them differently.

4.5 Final Outcome

The final outcome of our research and development is our mobile application and system called Process Assistant. This section will describe the end-to-end process of our whole system.

Firstly, Process designers design a process using the YAWL toolkit. The process flow and dependancy between tasks are joined through arrows and designers can also make use of Split and Join features. To add context to the task, designers will add a documentation tag with set codes to specify contextual information such as location and time. Please see 3.2.1.1 for more information on how designers can specify this. Designers should also aim to design simple processes for users, complicated processes containing loops and many levels of splits and joins will not work.

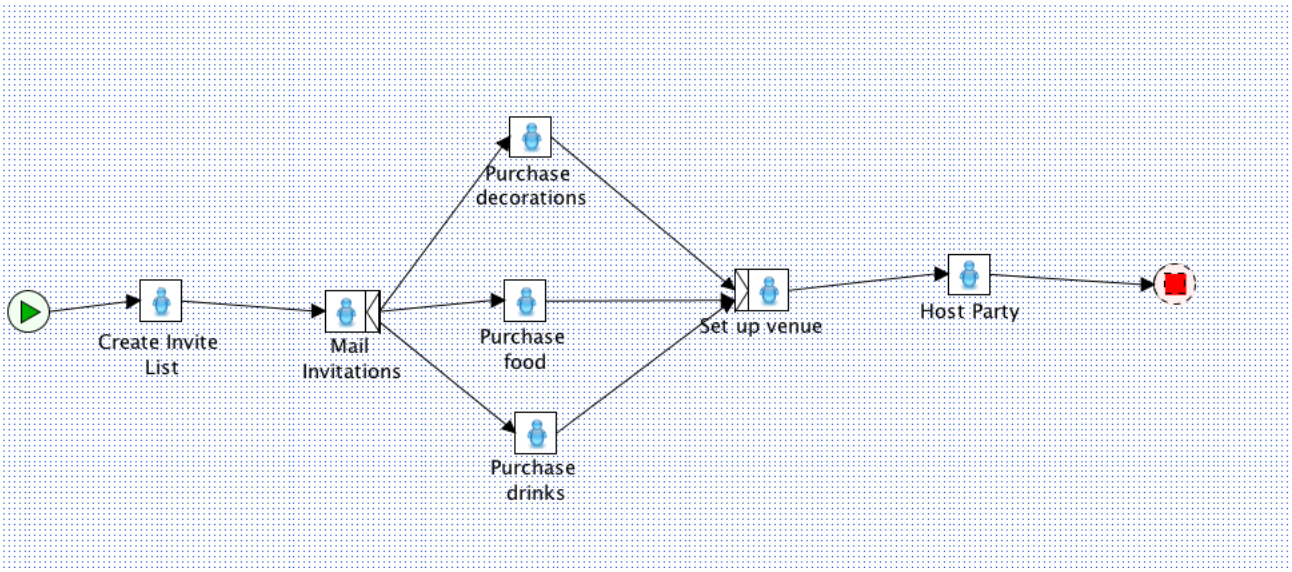

Figure 4.5: YAWL editor and Process designing

Process designers then connect to our repository File uploader to upload their process designs. Connections to our server are done via RESTful web services and so the designer simply calls

a URL(http://mashsheet.cse.unsw.edu.au:8081/FileUploader/). Designers are asked to fill out metadata about the process, which is deposited into a local database. Also, designers are required to provide unique process names within the scope of our prototype, as there is no validation checking provided.

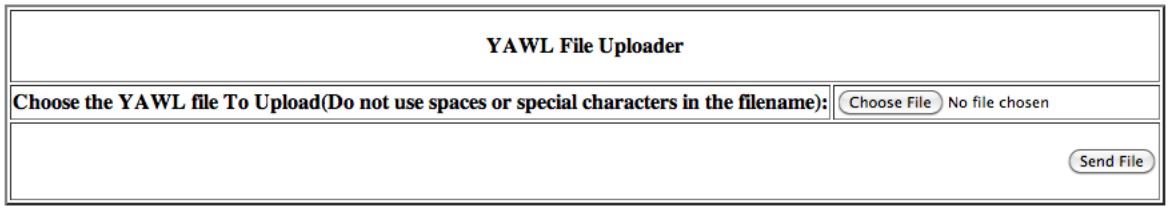

Figure 4.6: File uploader

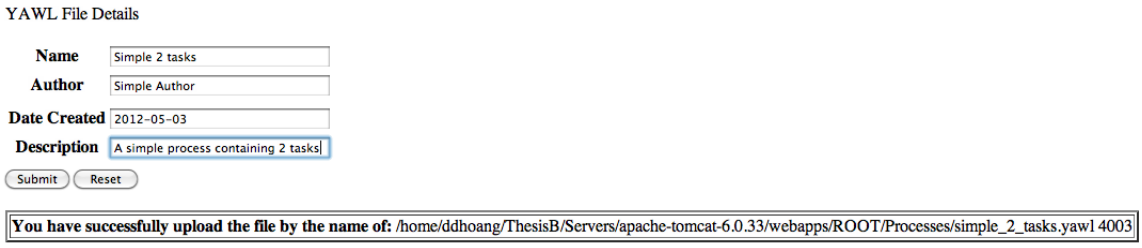

Figure 4.7: File uploader Process Metadata

End users with an Android mobile can then browse the repository through the Process Assistant mobile application. They require an internet connection to do this and so the Repository Manager service is invoked to display a list of processes. Pressing on a process will give more information about that process including the average rating and a comments thread which is handled by the analytics component of our system. Users downloading a process that is already saved in their process list will be prompted with a popup warning that re-downloading the process will delete the status of the existing process. When a user downloads a process, the parser component or web service is invoked and a XML file containing the Process object will be downloaded to the phone under

their home/Thesis directory(where home is the default directory of a user's external storage device)

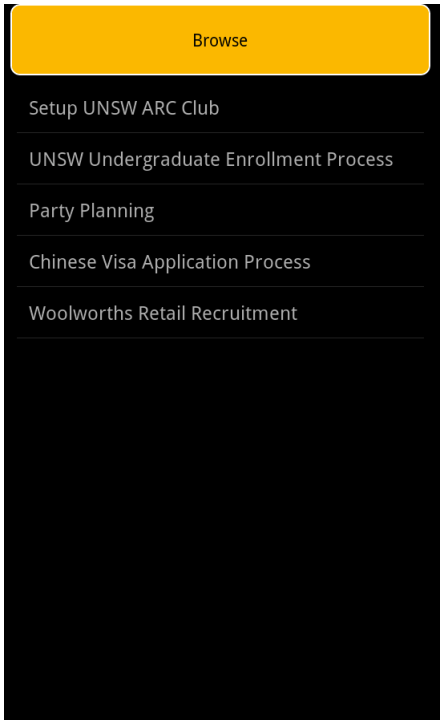

Figure 4.8: Browse repository processes

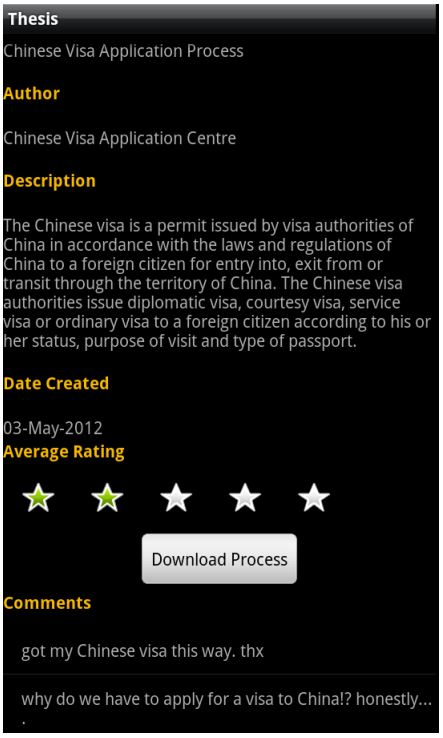

Figure 4.9: Repository View of a Process

When the user clicks the Process List, assuming it is not empty, it will display the list of downloaded Processes. This component is the Process Manager showing 3 tab views- Processes, Tasks and Suggestions. The tab views automatically refresh each other, so confirming tasks or completing processes in 1 tab will refresh the other tabs. Please see the Process Manager section 3.2.2 for a more detailed description.

Figure 4.10: Processes List

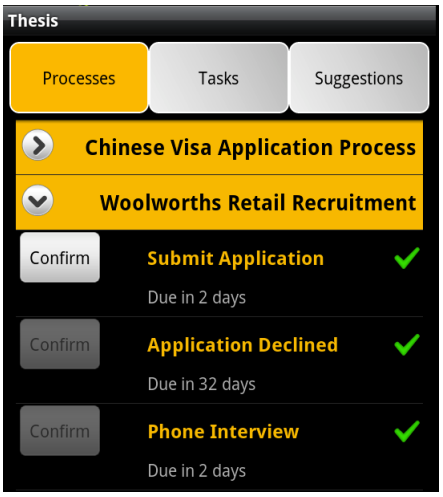

Figure 4.11: Processes List with Tasks

In the Processes tab, users see their list categorised by Processes. On Android, this view is an ExpandableListActivity. When users touch a Process, it expands to list out the associated tasks. Users can also long-press Processes to view more information about each Process. Users clicking on a Task, would bring up the confirm button, and Long-pressing a Task would bring about 2 options: Task information and Ignoring a Task. The user can also change the settings on how they would like to view the task list and this setting will be saved and refreshed with the other tabs. By clicking the options Menu on Android, users can filter their tasks by time or internet, as well as changing the view to display valid tasks, all tasks or valid and upcoming tasks.

When users confirm a Task, the next set of Tasks defined by the Process appear in the list. Depending on the splits and joins defined by the Process, some tasks might automatically skip some others as a result of a Exclusive OR split/join, or require you to complete a set of tasks first before progressing for a AND split/join. Users can also ignore tasks to get to the next stage but these will be flagged and highlighted in red, in which the user can complete them later. Note that you cannot ignore tasks that are not valid(i.e. you are not up to that task yet or that task has already been completed)

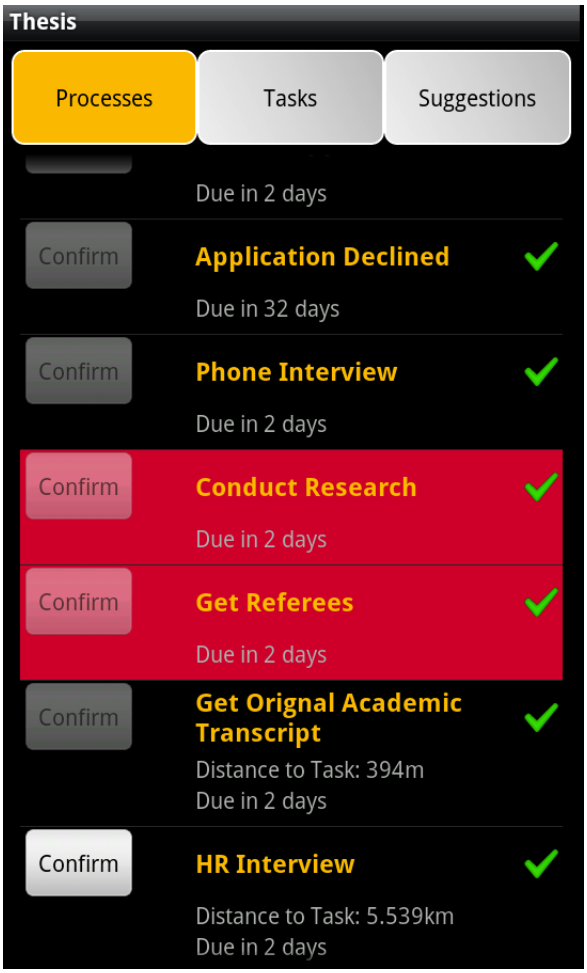

Figure 4.12: Ignoring tasks in Process Assistant

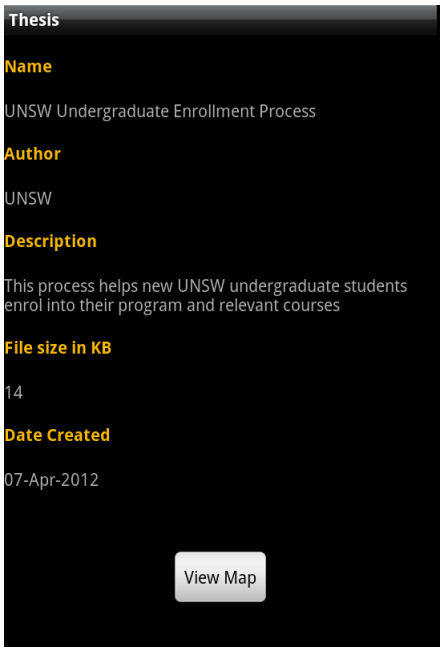

Figure 4.13: Process Information View

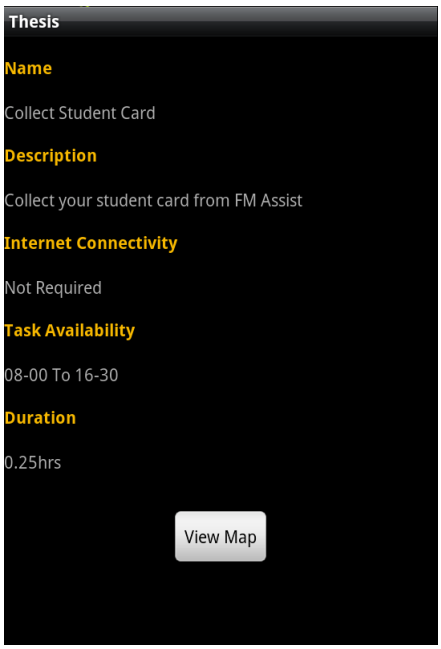

Figure 4.14: Task Information View

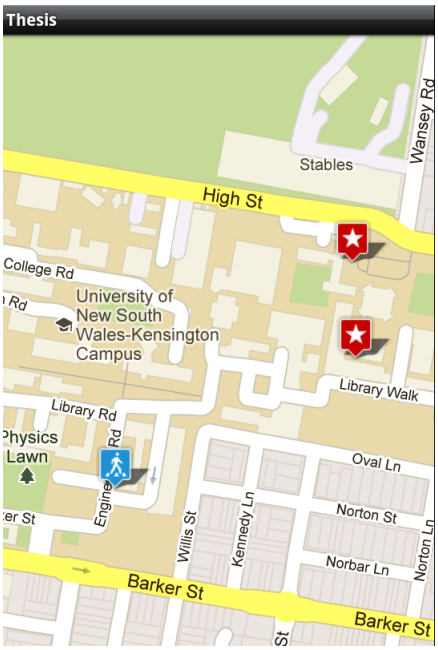

Figure 4.15: Google maps view of UNSW enrolment process

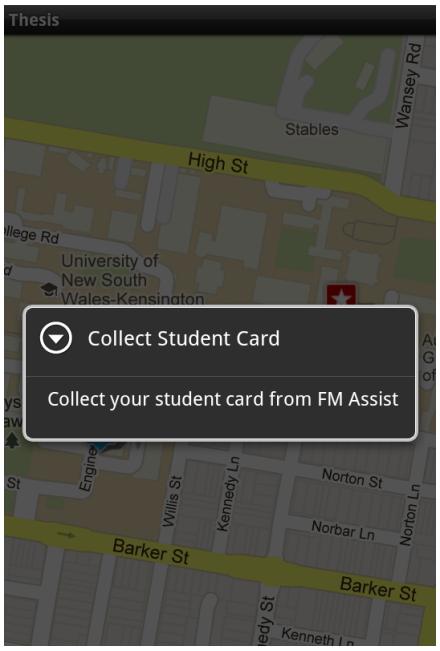

Figure 4.16: Google maps view of Collect student Card Task

Process and task information can be accessed if the user long presses a Process or Task. These views describe the process or task in further detail, as well as containing a link to a Google Maps View. The google maps view will contain icons of where the current user is and where the list of tasks/individual task is with respect to the user's location. To find the location of the current user we rank it the following way:

1. The system will first attempt to find the location based on GPS signal of the phone
2. If that information is not available, it will attempt to find the location based on the WiFi positioning system protocol
3. Lastly if none of the above connections are available, it will attempt to find the location based on the phone's 3G signal

Note that Android phones will also attempt to save past locations and will give you the last known locations and may not be refreshed unless multi-threading is incorporated to poll and refresh the location request.

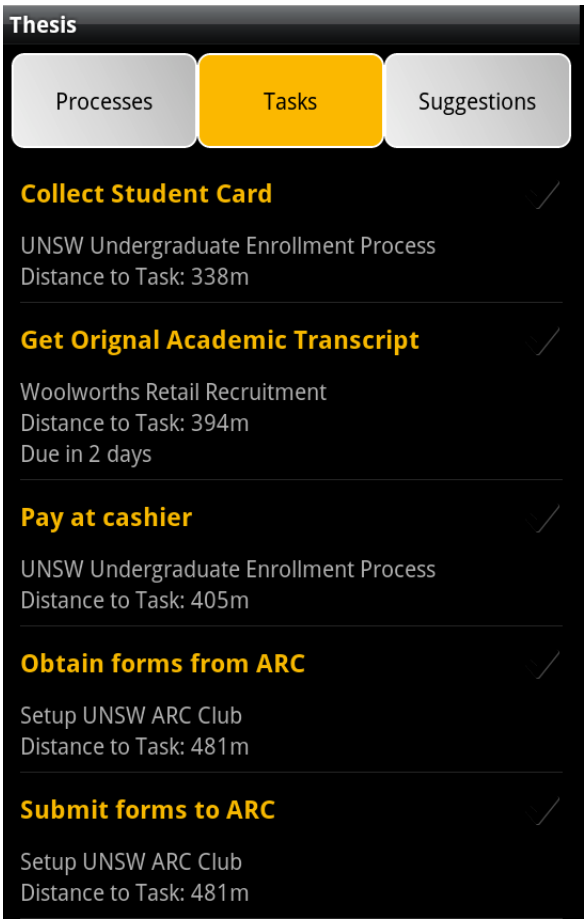

Figure 4.17: Tasks view

The tasks tab displays all tasks for every process. Tasks contain a description displaying what Process they are part of as well as the distance and due date if they exist. This view has additional functionality by the way of being able to sort tasks by closest distance, or closest due date by accessing the options menu.

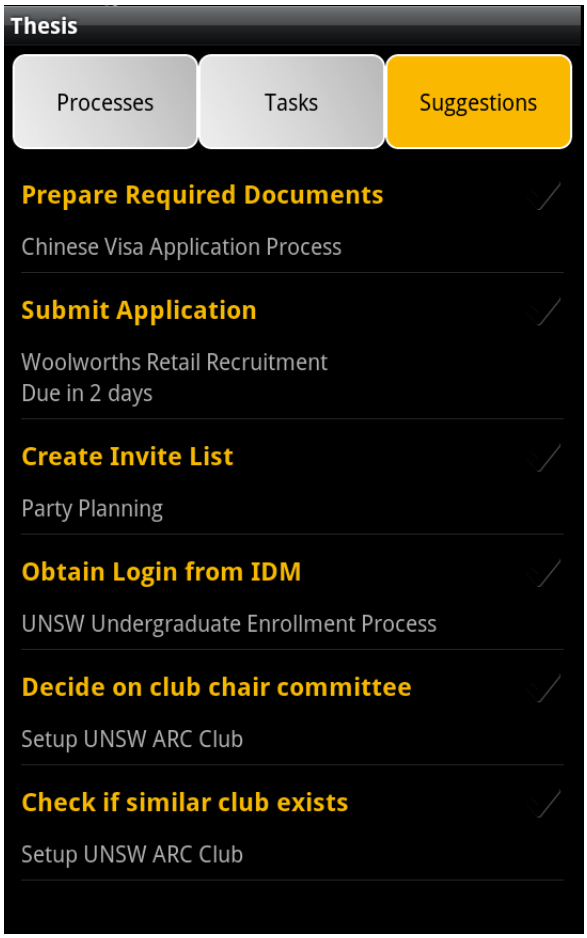

Figure 4.18: Suggestions view

The suggestions tab displays a set of tasks that we recommended via our set of rules. Please see 3.2.2.1 for more details.

When a user has completed a task, he/she would be taken to the feedback view which allows them to enter a rating as well as a comment for the process. The feedback will automatically by uploaded into our server as part of the system's data collection, and in real-time, any user who browses the repository for the same process will see the updated comment and rating with aggregated results. Feedback is optional.

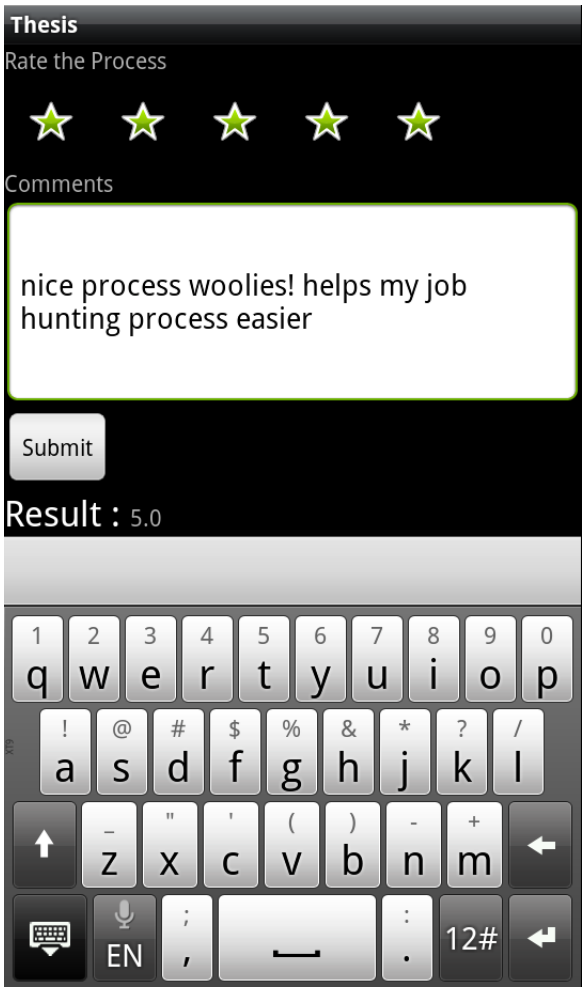

Figure 4.19: Feedback view

4.6 Issues and Limitations

As our system is a prototype, there will inevitably exist quite a few limitations to our application. At the same time, we had encountered several roadblocks during the course of design and implementation of the system, and we will describe them in detail in this section.

4.6.1 Performance

The performance of the system was a major issue throughout the course of development. As the team was inexperienced in mobile development as well as Android development, we had suffered poor performance several times when attempting to implement our design. When we had initially implemented our Hybrid architecture, we had not realised that the File object de-serialisation on the Android system would take quite a significant amount of time, and this increased quite significantly as more Processes are downloaded to the library and Processes contain more tasks. Our controllers(especially Process Controller) had too many read operations to the File system, especially as we had 3 tabs with its own Process Controller object and so read operations tripled. To counteract this, we decided to initialise it once, and Parcel the object thrice to the different tabs, moving more of the processing into memory.

As we have a hybrid architecture, to find out what stage we are up to in Processes, we would have to read an original Process file, check each task to see if it is completed via the stateDB, and continue to do this until there are no more completed tasks. This had to be done for each process, and the delay would increase as users downloaded more processes and completed more tasks. We had implemented our design using a Breadth First search algorithm. To improve the performance, we implemented a cache database to record the current node we were up to in a Process so that we do not have to start from the beginning all the time.

Another example of when we suffer poor performance is when we try to connect to our Analytics server. As we require the user to submit feedback after he/she completes a process, on Android, we require to use the startActivityForResult(intent, FLAG) method, which jumps from one screen to another and goes back to the original after it has been completed. When doing this however, we return to a tab screen without the Process Controllers initialised(we do this in the TabActivity), as Android unfortunately doesn't save the memory objects. The Process Controllers are required to update the Analytics to the server, so we are forced to initialise(and suffer a read operation to the File system) our processes again. Then we require to refresh the other tabs and again this will

suffer from some delay. Overall, with a slow and unreliable internet connection, this process may perform quite poorly. We acknowledge the design can be improved to improve performance but did not have the time to do so.

4.6.2 Process Logic

As the focus of our thesis project is on Personal Processes, we have decided to forego the possibility of complicated Process Logic that may exist in some Business Processes. For example, a Task branches with an AND split, and one of the next tasks branches further to a XOR split, while the other branches to a OR split - this feature is not handled by our system. Conditional logic(e.g. an IF gateway) is also not handled by our system unfortunately, as well as other advanced workflow patterns mentioned in the workflow patterns initiative(http://www.workflowpatterns.com/).

4.6.3 Automation

Automation is another limitation of our system. Our processes do not synchronise with other Web services to complete certain functionality automatically. An example could be entering a form for applying for a University club directly through our application, and receiving confirmation from an external web service which handles this process and allowing us to proceed to another stage. This feature would prove useful and may be a viable option for future work.

5 Evaluation

This chapter presents an assessment of the thesis and the final system. It includes details of the evaluation plan and a summary of the results gained from the evaluation runs.

5.1 Plan

5.1.1 Aim

In order to evaluate the final application, the key components of the application had to be evaluated separately in order to judge how significant they were in defining the application. The goal of this thesis as mentioned earlier is to help users with managing their personal process saving them time in the process.

The components of the application that were deemed to be most significant in saving the user's time is the Process Repository and the Suggested Tasks view. The inclusion of a process repository saves time as it means that the user is not required to conduct their own research on each process to derive the associated tasks and their contexts. The suggested tasks component saves the user's time as it takes care of the decisions that a user would have to make when completing multiple processes and in attempting to follow the best possible pat through the processes. Since this is a mobile application the user interface is key factor in determining its success. An easy-to-use efficiently designed interface would allow the users to interact with the application seamlessly.

5.1.2 Method

A questionnaire was designed to evaluate these components of the application. The first exercise required participants to derive the enrolment process for a new undergraduate student at UNSW. Participants were required to list out the required steps in order to do this and were allowed to use whatever means necessary (except for questioning the facilitator) in order to derive the required tasks. The next step required participant to pick three tasks from a list of five tasks provided. The

participants were required to pick the tasks and the order in which they would complete them as efficiently as possible. This exercise was timed as well. The time taken for these tasks would reflect on the time taken by most users to research processes and in deciding on which tasks complete if our application did not exist. The last exercise required the participants to provide feedback on the application itself. The participants were shown how to use an Android device, the features of the application and were given a brief overview of the thesis itself to put the application into context. This would provide a review of the application's user interface and the functionality provided.

Evaulation Plan

1. Defining a Process

Please derive the process of enrolling at UNSW as an undergraduate. Please note that you just have to list the steps of WHAT you need to do, not HOW. You may use whatever resources you need.

2. Deciding on a task

You have been given 5 tasks: Please choose your top 3 tasks in the order of which you would like to complete them. It is 2pm 20th May 2012, you are at UNSW. Please rank them in order of how you would complete them as EFFICIENTLY as possible.

i. You need to obtain the forms required to setup a student club at UNSW. The ARC club office closes at 5pm
ii. You need to go to an interview today at North Sydney at 3pm .
iii. You need to collect your passport at the Chinese Visa Application Centre which closes at 5pm (Sydney CBD)
iv. You need to purchase decorations for a party(Annandale)
v. You need to collect your new student card at FM Assist UNSW.

Figure 5.1: Page 1 of the questionnaire used for evaluation

3. Mobile Application

Please Rate the following:

Learning curve of app

- Rating: /10
- Like

- Dislike

Flexibility and ease of use

- Rating: /10
- Like

- Dislike

How often would you use this app? (no. of times per week)

What processes would you like to see?

What do you like & dislike about our app the most?

Any further suggestions:

Figure 5.2: Page 2 of the questionnaire used for evaluation

5.2 Results

This section provides a summary of the evaluation responses received. Please refer to the Resources chapter to gain access to the original responses.

Defining a Process : As this was a free text question, a variety of answers was received for the list of tasks necessary to enrol at UNSW as a new undergraduate student. The majority of participants chose to navigate to the UNSW website and locate the instructions for enrolling as a student. A few participants who have been through this process before attempted to recall the process from their experience of it. A major flaw with this approach is that the process has changed since they last completed it and therefore their response provided for this question was missing these critical changes. A common characteristic of the responses received was that they were quite vague in that the steps listed would not suffice in providing an individual with no knowledge of this process the required information in order to complete the process. As this was a largely online–based process with few physical tasks, one response simply stated that the user should navigate to the online application page and follow the steps listed there.

A graph of the time taken by participants is provided below. The average time was 6 minutes and 23 seconds, the least time taken was 1 minute and 36 seconds and the most time taken was 12 minutes and 13 seconds. Predictably the participant that took the least time provided a highly vague and inaccurate list of steps while the participant that took the most time provided a detailed and somewhat accurate list of instructions. However even this set of instructions would not be sufficient for an individual with no prior knowledge of the process. Therefore it can be deemed that the inclusion of a centralised process repository in our application would save the user at least 12 minutes, if not more for processes where the instructions were not so easily available.

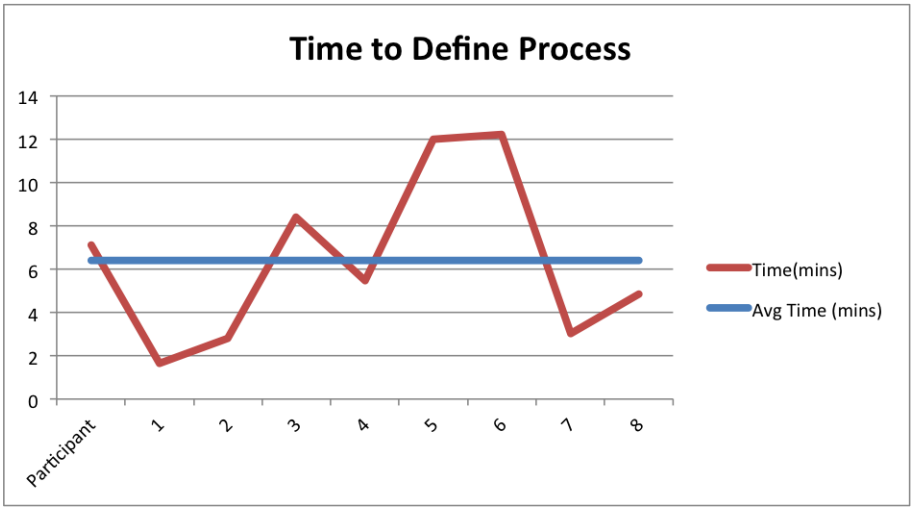

Figure 5.3: Time taken to define the UNSW undergraduate enrolment process

Deciding on a task : Very few participants paid attention to the fact that the order of tasks had to be performed as efficiently as possible. As a result they did not conduct any research on the context of the tasks provided. They instead chose tasks based on what they believed would have the highest priority to them based on what the task involved.

A graph of the time taken by participants is provided below. The average time taken was 1 minute and 6 seconds, the least time taken was 10 seconds and the most time taken was 3 minutes and 40 seconds. Only one of the participants derived the same order of tasks as those suggested by our application. However when questioned about this, the participant explained that it was more as a result of priority rather than the deadline of the task or the location of the task. The participant that took the longest time conducted the most amount of research for this task. The participant researched the location of the tasks to find the closes tasks, the travel time between tasks and took into account the deadlines of the tasks. Although the final result did not match that of our application, this method provided some useful insights. Our application prefers to suggest tasks with a closer deadline over tasks with a closer location. From this evaluation, the need for an option where the user is able to decide whether the deadline or location takes more precedence is visible. Also, a feature

that enables users to view the time required to travel from the user's current location to the location of the task may prove handy as some users seem to take this into account when deciding on what tasks to complete.

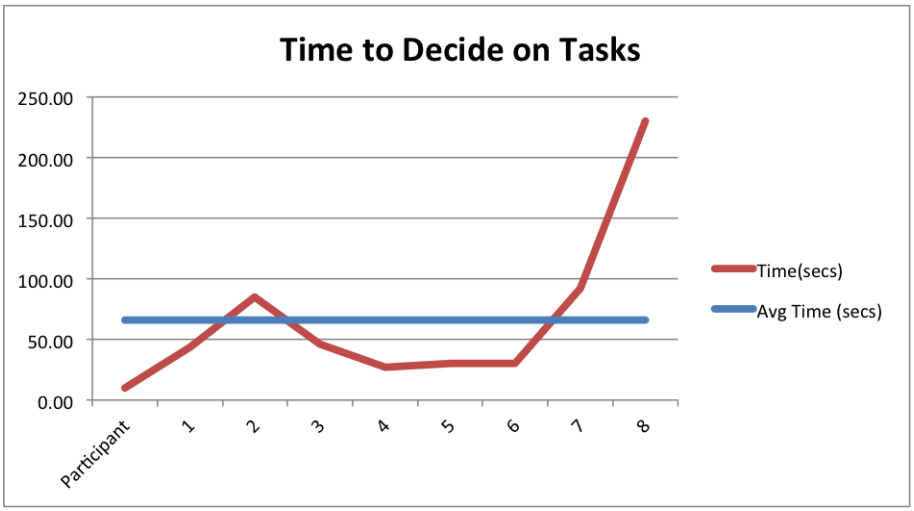

Figure 5.4: Time taken to decide on the best possible path to follow when completing a set of tasks

Mobile Application : In terms of the learning curve of the app, most participants who have used Android devices before were able to access the various option menus, context menus and seamlessly interact with the application. Other users however seemed to struggle and requested that a tutorial be included in the application that they can follow when they first use the application.

In terms of the flexibility of the application, participants like the idea of being able to filter and sort tasks and view the tasks on a map. The ability switch between the process view and tasks view proved useful as well.

On average, participants claimed they would use this application up to four times a week but would like to see more processes added to the repository namely job application processes,

holiday planning, cooking recipes and academic processes.

In terms of what users liked the most about our application, they liked how the application attempted to prioritise tasks and suggest these tasks to the user, detailed step-by-step instructions provided in order to complete a process, the maps feature and the broad audience that it is catered towards. They disliked the fact they were unable to undo their actions on a task, determine which tasks were already completed and that there was no help menu.

In regards to suggestions to improve the application, most participants requested a better, more intuitive interface and a help menu. Some participants requested a feature that allowed them to create their own process using the application and modify existing processes and the ability to change the rules that defined the suggested tasks.

6 Conclusion

The dramatic rise in the use of business process by almost all organisations has left the individuals involved in these processes with little to no help in attempting to complete these processes. Additionally the personal point of view of these business process has completely been ignored.

This thesis introduces the topic of personal processes and proposes a system that is designed to help manage their personal processes. An implementation of this system has also been outlined. The system allows process designers (businesses and personal users) to design process that are converted into task–lists for the end users. The system aids users in completing these tasks by providing various sorting and filtering options and goes to further to recommend tasks based on the context of these tasks and the context of the user.

The evaluation of this system has proved that it has been successful in achieving its core functionality. It saves the users, the time spent on researching a process to derive the associated list of tasks and helps them to complete their processes as efficiently as possible by suggesting tasks to the user.

7 Future Work

This chapter details the possibilities of additional work that can be undertaken in this research area which may improve the system that is presented. It goes into detail about the main areas that if improved on could have a significantly positive impact on the final system.

7.1 Dynamic Processes

Currently the order and flow of the process is explicitly specified when the process designer creates the process. However, realistically some processes are fully alterable by the user at the time of execution. Upcoming tasks in a process may rely on the user's choice for particular decisions in the process or a process may completely change depending on how the user completes it. For e.g.: a process to purchases a used car may have a stage where it requires the user to enter in the locations and other details for the cars they wish to have look at once they have conducted their search and the application would create a task for each of the results. Additionally the flow of the process may be affected by a decision the the user makes, so the process should specify the different paths possible and the application should be able to handle this. A more simple use would be that a user may feel the need to change the process during execution. They may wish to add or remove a task or change the flow of the process. The application should be made more flexible in order to handle and provide the user the necessary means to do so.

7.2 Smarter Suggestions

Currently the application suggests tasks based on the context of the tasks available to the user and the context of the user. The suggestions provided could be made more intuitive by taking into account the user' history and by analysing how they go about completing a process such as which tasks they usually ignore. It could also take into account any priorities that users assign to certain tasks so as to always recommend these tasks to the user. It could also use a cumulative

scoring system based on the context of the task such as the location, duration and due date of the task and then combine this score with the past trend of the user to recommended the best possible tasks to the user.

7.3 Rules Based Contexts

The context of tasks has to be explicitly described in order for the application to be able to process it. However, realistically not all forms of context can be explicitly described or in other case the the context of the task is actually relative to the context of the user or the current state of the process. In this case rather than having the process specify all possible context, a link to a web service could be provided instead. For e.g.: If a user is required to visit a post-office, a web service can be used to locate the closest post-office to the user. This would make the use of context-based tasks more useful and appealing as they would be more relevant to the user's context.

7.4 Application of Analytics

The analytics gained from users while they use the application is being stored on the server but is not being used in any way to enhance the application at the moment. If the analytics was able to be processed accordingly it could help improve the application. It could be used to improve the suggestions component to recommend tasks based on statistic gained for those tasks from users who have previously completed these tasks. It could also provide valuable insight for process designers about users complete their processes and therefore can design more accurate and better processes for users. However in order to be able to integrate analytics even further into the app, an additional component needs to be built that can analyse the analytics and provide the results in any form that is requested by the application, application developers or process designers.

8 Resources

The following resources are available as outputs of our thesis.

Project homepage : This contains a wiki for this project and instructions on how to install and configure the required web services and databases on a server. It also details the necessary changes to the code that would be required if the server has to be transitioned in the future.

Source code repository : This contains a repository with up-to-date source code for the web services, databases and mobile application.

Code documentation : This contains detailed Java–docs for the source code.

Evaluation results : This is a collection of the original responses received during the evaluation process.

The final archival location of the resources is still being finalised at the time of writing. For up-to-date information about the access to the resources, please contact Helen Paik (hpaik@cse.unsw.edu.au).

Bibliography

[1] Boualem Benatallah Corren Vorwerk Zifei Gong Liangliang Zheng Ingo Weber, Hye-Young Paik and Sung Wook Kim. Personal process management: Design and execution for end-users. 2010.

[2] Workflow Patterns Initiative. http://www.workflowpatterns.com/.

[3] Remember The Milk. http://www.rememberthemilk.com.

[4] Hwang San-Yih and Chen Ya-Fan. Personal workflows: Modeling and management. 2003.

[5] CSE Ohio State University. www.cse.ohio-state.edu/ rountev/421/lectures/lecture23.pdf.