

Abstract

i-ems(Intelligent Electronic Mail Sorter) is an interface of e-mail client to organise e-mail items with automatic induction. We need to consider how we apply automatic induction in organising e-mail items so that user can willingly accept it. E-mail items are a simple database with small number of fields and they are everyday task. If we could sort a huge number of e-mail items efficiently, we would be able to save a lot of time and money. The future research beyond this project will cover AI and smart clustering so the result of this project should be extensible for these.

Thunderbird is an open source software we can modify or add its functionalities. We can add sorting functionality to Thunderbird by implementing a Thunderbird Extension. This saves us much time because we do not need to implement the other functionalities for an e-mail client. We need prediction rules for e-mail sorting and these are managed by prediction manager. E-mail items are organised by there rules and user can execute user commands on GUI. He can either accept suggested predicted folder provided by the prediction manager or assign the target folder for e-mail items manually.

We have an installable package which works with Thunderbird regardless of platforms. Installation of this package adds a functionality of sorting e-mail items to Thunderbird.

1 Introduction

1.1 Overview

Email clients such as Outlook provide their own filtering rules along with interfaces. These filtering rules are used to facilitate the process of organising email items in inbox automatically. But these process are not apparently recognisable to users, thus making them confused. For instance, users can miss an important email item because it has been moved into spam folder automatically by the filtering rules and they can not recognise such process.

The aim of *i-ems*(Intelligent-Electronic Mail Sorter) is to create a prototype interface which suggests the better way in organising email items. This is very interesting subject because we can experiment various induction strategies based on Machine Learning to find a better methodology for users.

1.1.1 Background

Email sorting

Email sorting is a popular subject because email is a simple database and used by people everyday. We can investigate a variety of methodology to effectively deliver this database to users. Whittaker and Sidner[6] scrutinised inbox management made by users. They investigated 21 workers and found their inbox had 2482 mail items on average and only 858 of them were classified. The rest of them just remained in inbox unorganised. Even though email items in inbox is a simple database, it will be expensive to manage as the size of inbox increases. Mackay[4] suggested an approach to show classified folders for email items to be verified by users. If there is an incorrect classification, users can simply adjust its target folder so that it can be moved to a folder as they want. This has advantages that user can recognise the process of email classification clearly and it provides an intelligent way to correct wrong classification at a lower cost. We have taken a similar approach regarding the future induction research of email filtering rules. Our prototype application will sort each message in the inbox according to its predicted classification.

Thunderbird

Thunderbird provides developers to modify or add its functionalities in the form of add-on called Thunderbird extension. A Thunderbird extension includes XUL, Javascript and XPCOM. Each of them has a unique role and purpose. Thunderbird extension follows the common rules of Mozilla extension but has several specific rules only available in Thunderbird.

XUL

XUL(pronounced zool) is XML User interface Language. This was created to suggest a cross-platform interface environment written in XML. As a result, it has advantages that XML has and a cross-platform development environment would provide. It inherits DOM structure at runtime so we can use a lot of features from DOM including dynamic binding.

XUL Overlays

Thunderbird extension works via run time binding technique named XUL Overlay. Thunderbird extensions created by developers merge into Thunderbird at runtime. XUL overlays are defined in XUL files along with JavaScript files. We can merge, insert, remove or modify Thunderbird by using XUL Overlays.

JavaScript

XUL is a good tool used for the definition of the UI. But XUL defines only appearance and has no functionality which can interact with users. In a Thunderbird extension, interaction with users is implemented by Javascript. Javascript is a small object-oriented language which creates event handling of the UI defined by XUL. Javascript also provides a way accessing Thunderbird components made by XPCOM.

XPCOM

XUL defines the appearance of UI and Javascript implements the event handling of it. But Thunderbird is more than a simple UI. For example, it downloads email items through transmission with email server. These functionalities are implemented in XPCOM(Cross-platform Component Object Model) and Javascript can access XPCOM via XPCOM interfaces.

Install manifest and Chrome manifest

Thunderbird extension is installed as an add-on. Thunderbird has an extension manager which detects new extensions at startup so we need to provide proper information for it. We need to make install.rdf including creator information and Thunderbird version supports. Also we should create chrome.manifest which describes the way how XUL overlays are conducted. These must be included in the install package in the format of *xpi*.

1.1.2 Purpose

The main purposes of the current project are as follows.

System requirement

The i-ems interface approach will have to work in Thunderbird email manager and will be easy to install.

Prediction rules

Prediction rules are used for organising email items in the inbox. The prediction rules will analyse the content of each email item such as sender. The i-ems will provide a prototype of the prediction rule. The prototype will be designed in a systematic format so that we can add or modify the prediction rules in the future project. We will enhance it in the future project to investigate the intelligent induction rules of email classification.

Prediction

Learning rules(applying the prediction rules) should be performed in a quick and efficient way. The i-ems will organise the inbox with the applied prediction rules. Email items of the inbox will be organised based on the first matching prediction rule. The interface will display the list of the email items grouped by the prediction rules.

Organising email items

The users can archive the email items directly into the predicted folder or they can move them into another folders they

think appropriate. The i-ems interface should provide a quick and efficient way for user to do this selectively.

User manual

A user manual will be included describing install and usage of the i-ems.

1.1.3 Statement of scope

Research on induction of prediction rules is not within the scope of this project so only an extensible prototype of the rules will be created in the form of arrays. However very simple approach will be implemented so that we will be able to evaluate whether it satisfies the requirement well.

1.1.4 Participants

The participants of this project are described below.

Role	Name	Contact Details
Course coordinator	Dr. Peter Strazdins	Peter.Strazdins@cs.anu.edu.au
Supervisor	Dr. Eric McCreath	ericm@cs.anu.edu.au
Developer	Yoon Suk Chang	edwin.gruhner@gmail.com

1.2 Introduction to *i-ems*(Intelligent-Electronic Mail Sorter)

(This will be created as the whole structure of i-ems is specified.)

System interaction described using Modelling here

Show how XUL extension structure is constructed : *.xul, *.js, *.rdf

How to install. How XPI is installed.

Implemented GUI with figures and explanation

Implementation issues

...

1.3 Report Overview

(This will be done as all the documentation is finalised.)

2 Plan / Management

The Plan / Management specified the whole schedule given as a Gantt chart, details of the the task to complete, a risk analysis for the project and an overview with diagnosis/the actual time taken.

2.1 Gantt chart (Scheduled)

In the first few weeks, the following Gantt chart was created to help plan the tasks. Gantt chart was created as follows :

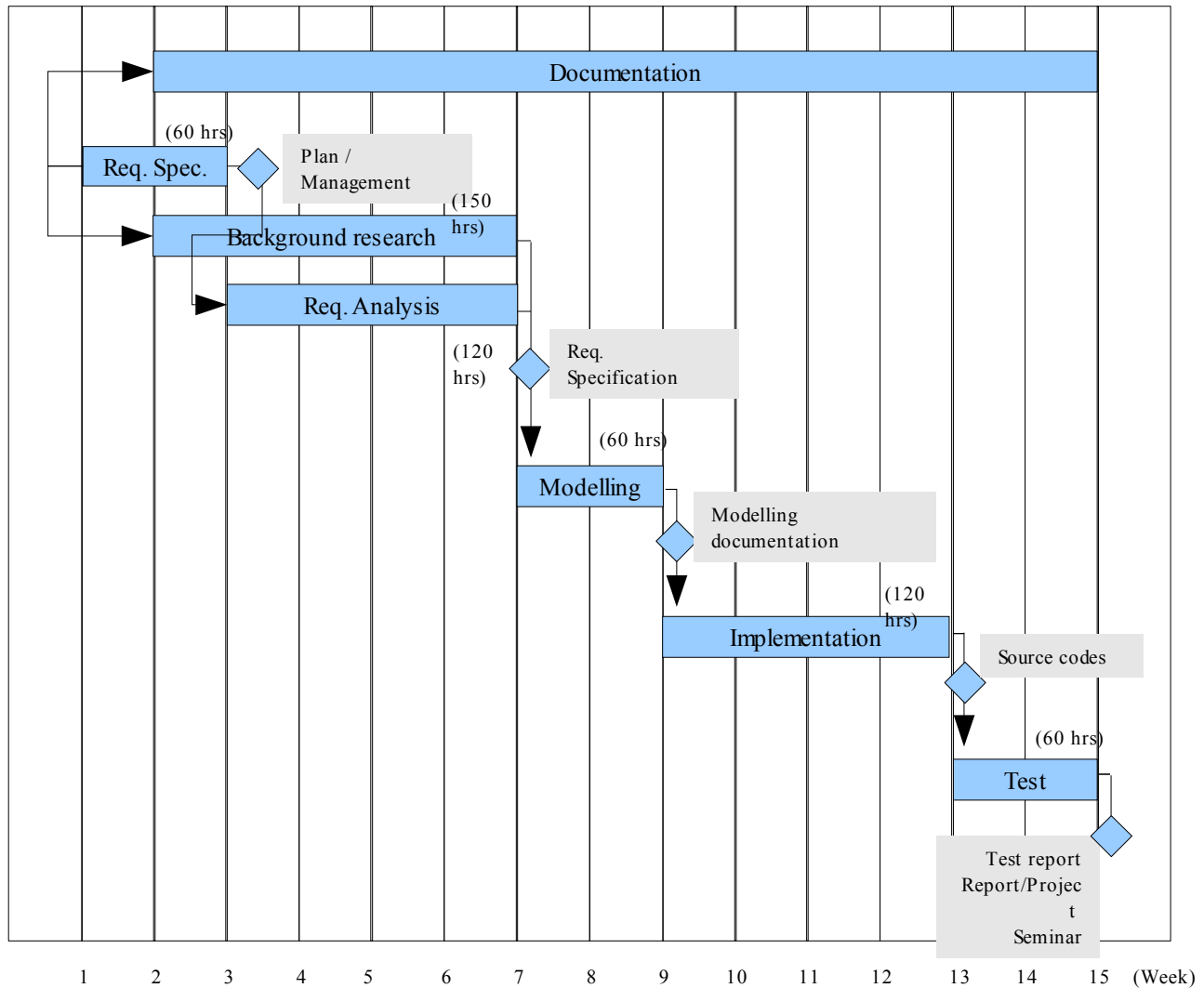


Figure 1. Gantt chart (Scheduled)

- It was decided that the writing of the report would not be left to the end, rather it would be “spread-over” the entire semester. This aimed to improve the quality of the report.
- The whole period of this project was 15 weeks.
- Given it was a 12 unit course this would namely involve 20 hours a week. However, I planned to spend more time

than this to improve quality of the final project. This was supposed to involve at least 30 hours a week

- The period of Requirements Specification and Background research was longer than any other projects because this project requires a lot of background knowledges which was complex and difficult to combine. Also the documentation of these technologies on the Mozilla developer's site was a mess. This project involved the collection of a large number of technologies including : XUL, JavaScript, XPCOM, Install manifest, Chrome manifest, Thunderbird XUL content structure, and Thunderbird component structure.
- The period of Modelling and implementation was shortened to meet the deadline of 15th week.
- Details of each task were included in later section, Task details.

2.2 Task details

The project included a lot of tasks which include :

1.	Documentation of Requirement Specification/Analysis
	Index will be created.
	Abstract will be written.
	Introduction will be written.
✓	Project schedule will be established and Planning/Management will be written.
✓	Software Requirements Specification will be written.
✓	Requirement and Detailed requirement will be written.
	Acceptance Test Plan will be specified.
2.	Documentation of Planning/Management
	Scheduled Time table, Actual Time table will be written.
✓	Risk will be described.
3.	Documentation of Modelling/Design
	Directory/file structure will be written.
	Class diagram will be written.
	Object diagram will be written.
	State diagram will be written.
	Scenario will be written.
	Directory/File structure will be written.
	Package Structure will be written.
4.	Documentation of Implementation
	Source codes will be described with detailed comments for the future project.
5.	Documentation of Test
	The result of Boundary test will be documented.
6.	Documentation of User manual
	A user manual will be included describing install and usage of the <i>i-ems</i> . This project is for researchers with high understanding of software development. Therefore only technical issues of <i>i-ems</i> will be included. -1/2 page about install, 2pages about usage
	The format and process of prediction rules will be written in detail.

	Detailed explanation of how to develop the <i>i-ems</i> for the future research of inductive sorting.
	Referencing guide to Mozilla developer's website on <i>XUL</i> development will be written. The order of references will be determined in the form of index regarding the understanding of the future developers. This will include a detailed explanation on the contents.
	Tips for future development will be written including the problems I will have had and how to solve them.
	Discussion, Conclusions, Future Work and Bibliography will be written.
7.	Documentation of Additional
	Index will be added.
	References will be added.
	Notebook will be filed.
8.	Requirement Specification
	Tasks will be defined .
	Software Requirements Specification will be prepared.
	Requirement and Detailed requirement will be prepared.
	The total period of project will be estimated.
9.	Background research
<i>V</i>	Relevant documents will be downloaded.
<i>V</i>	XUL(XML User Interface Language) content will be researched.
<i>V</i>	JavaScript for event handling will be researched.
	JavaScript for accessing XPCOM component (especially mail component) will be researched.
<i>V</i>	Install manifest will be researched.
<i>V</i>	Chrome manifest will be researched.
<i>V</i>	RDF (Resource Description Framework) and Templates will be researched.
<i>V</i>	Thunderbird XUL content structure will be investigated.
	Thunderbird component structure will be investigated.
	Thunderbird Javascripts will be investigated.
	Sample source codes will be examined and experimented with.
10.	Modelling / Design
	Class diagram will be developed.
	Object diagram will be developed.
	State diagram will be developed.
	Scenario will be developed.
	Directory/File structure will be developed.
	Package Structure will be developed.
11.	Implementation
11-1.	Packaging
<i>V</i>	Directory/File structure for the <i>XUL</i> extension will be defined created at the beginning of development. - Install.rdf, chrome.manifest, *.xul, *.js
	All files included in the project will be binded into a <i>xpi</i> file.

11-2. Prediction rules	
	The format of prediction rules will be defined. - Prediction name, prediction rules, priority
	Sample prediction rules will be added in the form of arrays. - pruleSender, pruleFriend
	Classes will be created for dealing with email item, UI and prediction rules. - emailItem, emailClient, iemsUI, predictionRule, predictionItem
	Each prediction rule will bind with a folder. - Prediction rule = {folder, {rule 1, rule 2, rule 3, ...}}
	The default folder will be created after install along with its prediction rule. - Sender, pruleSender
11-3. Prediction	
	User can perform Learning rules in a quick and efficient way through UI. - Toolbar buttons, Menus, Context menus
	UI will request the prediction class for analysis of email items. - iemsUI.learnPrediction -> predictionRule.findMatchingRule(emailItem)
	The prediction rules will be tested in the prediction class by the content of each email item. - Sender, Email title, Email text
	Email items of the <i>inbox</i> will be organised based on the first matching prediction rule and displayed on UI. - predictionRule.findMatchingRule(emailItem)
	The interface will display the list of the email items grouped by the prediction rules. - Format : [Prediction]-[Email item title]
11-4. Organising email items	
	The users can <i>Archive</i> the email items directly according to suggested predictions. - Toolbar buttons, Menus, Context menus
	The users can move email items into another folders they think appropriate. (MoveTo) - Toolbar buttons, Menus, Context menus
	Email items will move into the designated folders. - emailItem.setActualFolder()
	Selected prediction and actual folder will be stored for statistics. - emailItem.setPrediction(), emailItem.setActualFolder
	UI will show the feedback of organisation of email items. - Message box with OK button
	Drag and drop will be implemented. - No interaction is required unlike MoveTo.
	Synchronisation with Mail server will be implemented.
12. Test	
	Testing and Debugging will be performed.
	Code and Unit Test will be conducted.

	Install will be tested on various platforms including Linux, Windows and Macintosh.
	Acceptance Testing will be performed.
	Maintenance will continue until delivery to the client.

2.3 Risk analysis

1.	Background research
	The documentations on Mozilla Developer's Website are not organised well. Therefore it is hard to estimate the exact period of Background research.
	There is a lot of background knowledge to be researched including : XUL content making, XPCOM, JavaScript, Install manifest, Chrome manifest, Thunderbird XUL content structure and Thunderbird component structure
2.	The internal structure of Thunderbird
	Thunderbird is a huge program and it may impose a lot of work to look inside.
	The structure of source codes in Thunderbird is complicated.
3.	Modelling
	The period of Modelling is shorter than the other projects.
4.	Implementation
	The period of Implementation is shorter than the other projects conducted within this semester.

* Requirement may be rediscussed in case of a problem arising.

2.3 Gantt chart (Actual)

The following Gantt chart was created as the project progressed :

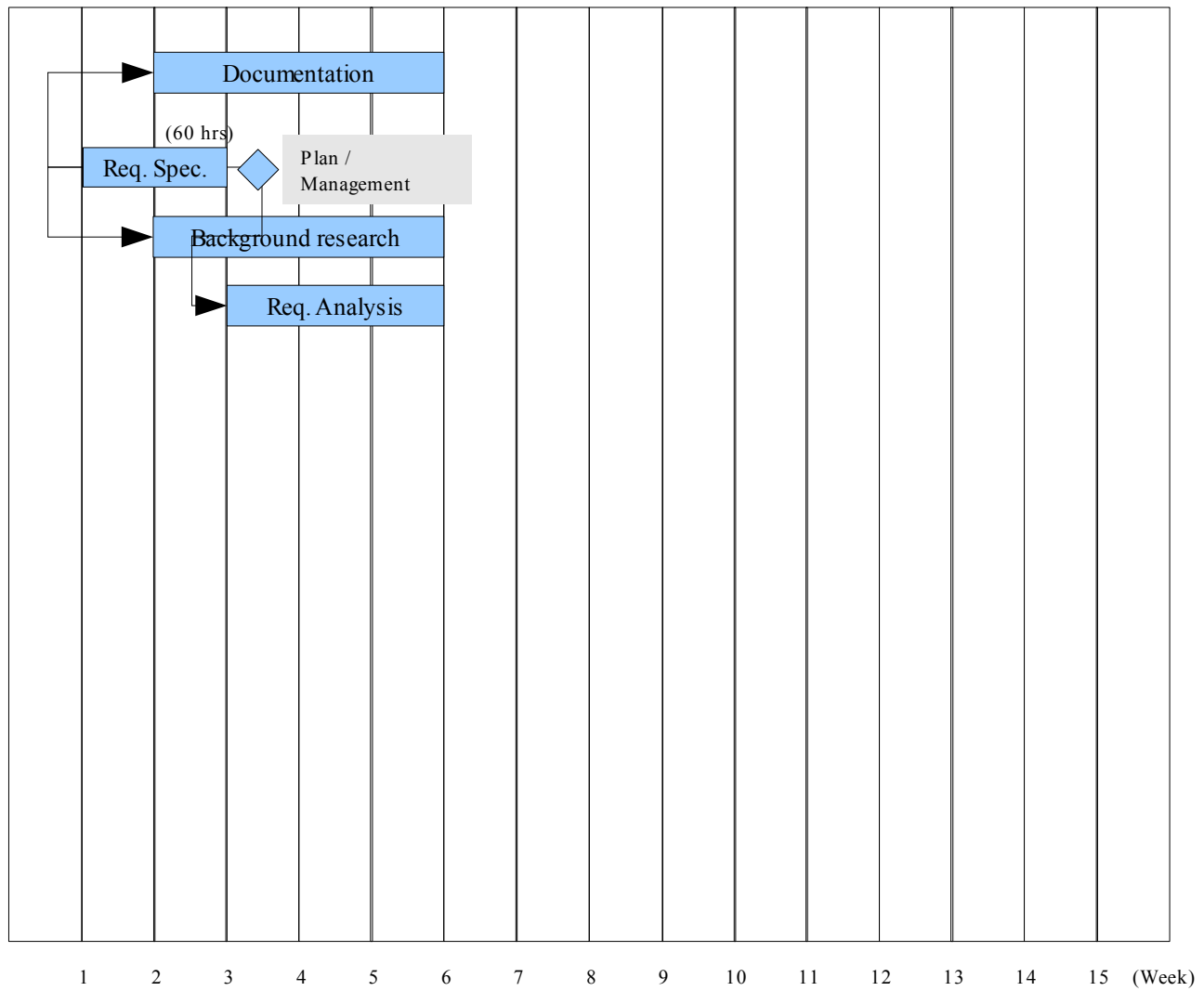


Figure 2. Gantt chart (Actual)

3 Requirements

The requirements of *i-ems*(Intelligent-Electronic Mail Sorter) for Thunderbird were discussed with the client early in the first week of the project. This section brings together an overview of the requirement including the : key issues summary, and constraints. Appendix A describes a detailed and complete list of the requirements.

3.1 Key Issues

The client was interested in extending the functionality of *Thunderbird* by adding an add-on for organising the *inbox* in more intelligent way. The system needed an extension to help organising the inbox. Several key issues were discussed and were as follows:

1.	Filtering rules of email programs
	Many email programs provide users with filtering rules which can do various mail management tasks : automated forward, deletion, replies and so on. These rules can be expressed in terms of combined keywords and directives. The rules are evaluated for each email item in order and the first rule that applies to the item triggers the email manager to move it into the associated folder. Setting up the rules can be time-consuming and users can have misplaced messages which require additional manual tasks. Generally, many users avoid filtering.
2.	Thunderbird
	<i>Thunderbird</i> is a widely used email manager working on Linux, Windows, and Macintosh. Because of its flexibility and extensibility, it will be a good base for this project. There are a lot of documentations on the <i>Mozilla developer's website</i> but they are not organised well because <i>Thunderbird</i> is an open source software project.
3.	Future project
	This project can be extended to very large one involving Machine Learning, User Modelling, Text Classification and Inductive Logic Programming in the future research project.

3.2 Summary of requirement

The client described detailed outline of the *i-ems*(Intelligent-Electronic Mail Sorter) project. His requirements were summarised:

1.	System requirement
	The <i>i-ems</i> interface approach will have to work in <i>Thunderbird</i> email manager and will be easy to install.
2.	Prediction rules
	Prediction rules are used for organising email items in the <i>inbox</i> . The prediction rules will analyse the content of each email item such as <i>sender</i> . The <i>i-ems</i> will provide a prototype of the prediction rule. The prototype will be designed in a systematic format so that we can add or modify the prediction rules in the future project. We will enhance it in the future project to investigate the intelligent induction rules of email classification.
3.	Prediction

	Learning rules (applying the prediction rules) should be performed in a quick and efficient way. The <i>i-ems</i> will organise the <i>inbox</i> with the applied prediction rules. Email items of the <i>inbox</i> will be organised based on the first matching prediction rule. The interface will display the list of the email items grouped by the prediction rules.
4.	Organising email items
	The users can archive the email items directly into the predicted folder or they can move them into another folders they think appropriate. The <i>i-ems</i> interface should provide a quick and efficient way for user to do this selectively.
5.	User manual
	A user manual will be included describing install and usage of the <i>i-ems</i> .

- The *i-ems* Interface will follow the main ideas in the paper written by Crawford, E. and Kay, J. and McCreath, E.[1]
- The *i-ems* will be part of Thunderbird email manager after installation.
- Users are allowed to judge prediction performed on prediction rules.
- Users can select Archive or MoveTo by clicking on XUL contents.
- Statistics of prediction will be shown beside Predicted folders.
- Applied prediction rules will be shown for selected email items.
- The final result of the project will be similar to the following appearance :

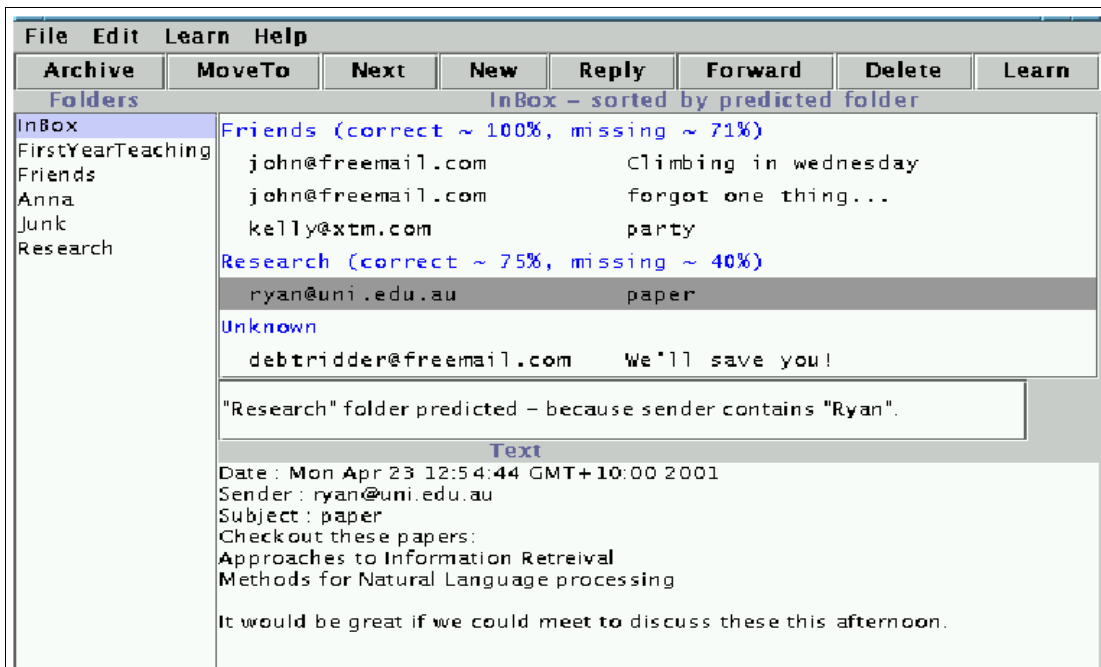


Figure 1. A screen shot of the *i-ems* interface (from Crawford, E. and Kay, J. and McCreath, E[1])

3.3 Constraints

Several constraints were identified in the initial scoping. The constraints related to initiation and implementation issues and were as follows :

1.	Development environment
	This system must be implemented based on the <i>Thunderbird</i> APIs. <i>Thunderbird</i> APIs named <i>XUL</i> extensions are written in <i>XUL</i> and <i>Java script</i> . Also it needs to be packaged in an installable format named <i>XPI</i> . It will involve understanding <i>XUL</i> extensions with which we can modify or enhance the functionality of <i>Thunderbird</i> . Doing it this way means <i>Thunderbird</i> running on Linux, Windows or Macintosh can simply add this extension.
2.	Modelling
	In its nature, this project is restructuring of <i>Thunderbird</i> . Internal classes of it will be inherited by new classes so that new functionalities can be added or extended.
3.	Prediction rules
	Creating rules based on Machine Learning will be researched in the future. A class for dealing with prediction rules is required so that it can be modified or enhanced easily. Research on induction of prediction rules is not within the scope of this project so only an extensible prototype of the rules will be created in the form of arrays. However very simple approach will be implemented so that we will be able to evaluate whether it satisfies the requirement well.
4.	Prediction
	Matching classification rules will automatically organise messages with prediction of target folders. If users are happy with the result, they will simply click on the <i>Archive</i> button at the <i>i-ems</i> tool bar. If not, they will select the <i>MoveTo</i> button followed by their selection of the folder. Some messages will be organised into <i>Unknown</i> category if no prediction has been made. Default folder such as <i>Sender</i> will be created after installation.
5.	Finance, time and resource
	There were no financial constraints on this project however there were strict time and resource constraints by the nature of this project. This project had to be completed before the submission date of the academic semester ends. In addition, all work had to be created only by the author.

8 Appendix

A Requirements in detail

This section describes in detail user requirement for *i-ems*(Intelligent-Electronic Mail Sorter).

A.1 System specification

1. Thunderbird	
	The version of Thunderbird is 2.0 or higher to support better functionalities.
2. Implementation of User Interface	
	XUL(XML User Interface Language) will be used for definition of the objects(=contents) in the UI.
	JavaScript will be used for definition of methods for the objects created by XUL.
3. Implementation of Data classes	
	JavaScript will be used for defining classes and their methods for data.
	JavaScript will be used for accessing Thunderbird mail components.
4. Packaging	
	Install manifest(install.rdf) and Chrome manifest(chrome.manifest) will be used for definition of the package structure.
	<i>xpi</i> file format will be used for platform-free installation.

A.2 Development

1. Packaging	
<i>V</i>	Directory/File structure for the <i>XUL</i> extension will be defined at the beginning of development. - Install.rdf, chrome.manifest, *.xul, *.js
	All files included in the project will be binded into a <i>xpi</i> file.
2. Prediction rules	
	The format of prediction rules will be defined.
	Sample prediction rules will be added in the form of arrays.
	Classes will be created for dealing with email item, UI and prediction rules.
	Each prediction rule will bind with a folder.
	The default folder will be created after install along with its prediction rule.
3. Prediction	
	User can perform Learning rules in a quick and efficient way through UI.
	UI will request the prediction class for analysis of email items.
	The prediction rules will be tested in the prediction class by the content of each email item.
	Email items of the <i>inbox</i> will be organised based on the first matching prediction rule and displayed on UI.
	The interface will display the list of the email items grouped by the prediction rules.

4.	Organising email items
	The users can <i>Archive</i> the email items directly according to suggested predictions.
	The users can move email items into another folders they think appropriate. (MoveTo)
	Email items will move into the designated folders.
	Selected prediction and actual folder will be stored for statistics.
	UI will show the feedback of organisation of email items.
	Drag and drop will be implemented.
	Synchronisation with Mail server will be implemented.
5.	Test
	Boundary test of prediction will be performed over the project.
	The result of the boundary test will be included.
	Install will be tested on various platforms including Linux, Windows and Macintosh.

A.3 Documentation

1.	Preparation
	The introduction of the project will be written.
	The requirements will be written and finalised through discussion with the client.
	Project schedule will be established and Planning/Management will be written.
2.	Main documentation
	Abstract will be added.
	Index will be added.
	Requirement Specification/Analysis will be added.
	Modelling/Design documentation will be written.
	Test result will be documented.
	User manual will be documented.
	Referencing guide to Mozilla developer's website on <i>XUL</i> development will be written.

D References

- [1] Crawford, E. and Kay, J. and McCreath, E. An Intelligent Interface for Sorting Electronic Mail In IUI02, January 13-16, San Francisco, California, USA, 2002.
- [2] Crawford, E. and Kay, J. and McCreath, E. Automatic Induction of Rules for e-mail Classification In ADCS2001 Proceedings of the Sixth Australian Document Computing Symposium, pages 13-20, Coffs Harbour, NSW Australia, 2001
- [3] Ducheneaut, N. and V.Bellotti E-mail as habitat: an exploration of embedded personal information management In Interactions, pages 30-38, Volume 8, Number 5, 2001, ACM Press.
- [4] Mackay, W.E. Trigger s and barriers to customizing software In CHI'91 Conference on Human Factors in Computing Systems, pages 153-160, New Orleans, Louisiana, 1991.
- [5] R.Segal and M. Kephart Mailcat: An intelligent assistant for organizing e-mail In Proc. Of The Third International Conference on Autonomous Agents, pages 276-282, Seattle, WA, 1999.
- [6] S. Whittaker and C. Sidner. Email overload: Exploring personal information management of email. In CHI, pages 276-283, 1996.