Master Thesis

Semi-Automatic Linking of Resources in End-User Scenarios

Author(s):
Ratnala, Pradeep Kumar

Publication Date:
2012

Permanent Link:
https://doi.org/10.3929/ethz-a-010056901

Rights / License:
In Copyright - Non-Commercial Use Permitted
Semi-Automatic Linking of Resources in End-User Scenarios

Master’s Thesis

Pradeep Kumar Ratnala
<ratnalap@ethz.ch>

Prof. Dr. Moira C. Norrie
Matthias Geel

Global Information Systems Group
Institute of Information Systems
Department of Computer Science

24th April 2012
Abstract

The success of hyperlinks in the World Wide Web (WWW) fostered many Personal Information Management (PIM) systems to incorporate the concept of associations in the desktop environment for navigation and retrieval purposes. These systems associated resources like documents, images, web pages, contacts etc. either implicitly by analyzing the content and metadata of resources or by mining the user activities according to specific access patterns. However, not much research has been done to study the scenarios where users need to explicitly create associations between resources and how such functionality can be provided to users. Also the existing systems were not tightly integrated with the desktop environment and demanded the end-users to interrupt their workflow to use the application’s user interface, in order to benefit from any associations. The aim of our thesis is to address these problems.

In this thesis, we present an approach to allow end-users to associate personal resources and benefit from these associations, in a lightweight way and as a part of their workflow. We analyzed the typical workflows of end-users and elicited a set of use cases which identify the exact points in the user workflow where links can be created and used. We designed and implemented a system which realized our use cases. Our system is integrated with the desktop environment and creates implicit associations as well as allows end-users to explicitly associate resources in specific scenarios. A user study showed that our system improved the productivity of the users with tasks involving personal resources.
## Contents

1 Introduction .......................... 1  
  1.1 Goals of the Thesis .................. 2  
  1.2 Structure of the Thesis ............. 3  

2 Background .......................... 5  
  2.1 Associations between Resources in PIM Systems .... 5  
    2.1.1 Content-based Associations .......... 5  
    2.1.2 Activity-based Associations ......... 7  
  2.2 Retrieval of Resources based on Associations ....... 8  
  2.3 Desktop Integration Techniques .......... 9  

3 Approach .......................... 11  
  3.1 System Design Goals ................. 13  
  3.2 Roadmap ................................ 14  

4 Scenario Analysis and Use Case elicitation ............ 17  
  4.1 User Interviews ...................... 17  
  4.2 Inference of Use Cases .............. 20  
    4.2.1 Document Links ...................... 21  
    4.2.2 Associations between Web Pages .... 21  
    4.2.3 Associations between Messages and Saved files .... 22  
    4.2.4 Temporal Associations ............. 22  
    4.2.5 Associations between Downloaded files and Web pages .... 23  
    4.2.6 Associations between Uploaded files and Web pages .... 24  
    4.2.7 Linking Embedded Files and Documents .... 24  

5 Feasibility Study ..................... 25  
  5.1 Range of Technologies .............. 25  
  5.2 Selection of Technologies .......... 27  
    5.2.1 Applications and Extensions .......... 27  
    5.2.2 Programming Languages .............. 29  
    5.2.3 Database .......................... 30  
    5.2.4 Web Browser ....................... 30  

6 Implementation ....................... 31  
  6.1 System Architecture .................. 31  
  6.2 Data Model .......................... 32  
  6.3 Document Linker ..................... 33
Psychology studies [15] have revealed that our memory works in the context of associations or links. Also studies have proved that people prefer to use association based navigation rather than keyword search to directly jump to their targets [19]. We can see the power of links through the success of World Wide Web where links serve as means of navigation.

Surprisingly, the desktop environment provided by the most of the operating systems supports very limited functionality for associating resources. The only kinds of associations that are supported by default are creation of shortcuts or symbolic links. These shortcuts merely act as placeholders by pointing to files, folders and applications. They are static links created by the users and primarily help in easy navigation. There is no support to link semantically related files or associate files implicitly based on user behavior.

Existing PIM tools addressed this issue by incorporating the concept of associations to relate personal information resources in their systems. These associations are used primarily to provide the user a better way to navigate and retrieve files. Some of these systems used content or metadata similarity to associate resources while others created links by analyzing user activity and the contexts in which the user works. However, in most of the cases, associations are created implicitly by these systems without any user intervention. Particularly, not much research has been done to understand the contexts in which users would like to explicitly associate resources and the workflow the user would follow in such contexts.

Also, most existing systems are standalone applications with their own sophisticated user interfaces. To associate resources or to use the existing associations for navigation or retrieval purposes, the user needs to explicitly invoke the application’s user interface and interact with it. This type of approach has a disadvantage that it does not integrate well with the user’s workflow as the user has to interrupt his existing workflow and switch to the PIM tool’s user interface. Considering the already complex workflows we have in our computing lives, one may be less motivated to open another application to find associations even though
they may eventually be useful.

Whenever a user works with a resource, there is a need to make the user aware of any associations that exist for that resource. If there are no notifications, there are chances that the user may never know that two resources are related unless he explicitly queries the system. Many PIM systems did not provide any mechanism to notify the user about any existing associations. The user has to explicitly invoke the application and search or navigate for a resource to find any related resources.

The aim of this thesis is to fill these gaps. We want to allow the users to associate resources and get benefit out of them in a lightweight way and part of their workflow. We want these associations to be created implicitly or manually by the users depending on the scenario. In addition, we want to provide an unobtrusive notification mechanism through which the users get notified about any existing associations.

Firstly, we exploit the scenarios in the user workflow where users would benefit from relations between resources. Specifically, we identify the use cases which describe the exact point in the workflow where an association can be created and how the user would use these associations to retrieve resources. To give an example of a relevant use case: A user works with a few documents always together. In this case, all these documents could be associated together. By creating associations, we enable the user to access the other associated documents whenever he opens one of them. Secondly, selected use cases need to be implemented as prototype applications which together define the system. We intend to explore technical solutions such as web browser plugins, file explorer integration or separate desktop applications. Finally, the implemented prototypes need to be evaluated by the end-users.

1.1 Goals of the Thesis

The main goal of this thesis is to design a system which enables link creation between personal information resources in a lightweight way and as part of user’s workflow. Also the users should be able to easily access the associated resources, thereby getting benefited by the process of creating links. To this end, the thesis should consist of following main steps:

- **Scenario Analysis**: An analysis should be done on how end-users use and manage their personal resources. Various factors like the kind of applications used, the different kinds of user interfaces interacted with and typical user workflows have to be studied.

- **Use case Elicitation**: Based on the findings of the first step, use cases should be designed which identify the scenarios where resources can be associated by the users in their workflow. Also the use cases should identify how the links that are created will be used again by the user. Finally, they should also detail how the user gets benefited by going through this whole additional process of associating resources.

- **Feasibility Study**: The viability of the implementation of the designed use cases has to be studied. For this, we analyze the various possible technologies available and their suitability as a choice to implement the use cases.
• **Implementation**: Proof-of-concept prototypes should be implemented for all the use cases considered. These prototypes together should work in an integrated environment which defines our system.

• **Evaluation**: The designed prototypes should be evaluated by users in order to understand the perceived user experience in using the concept of associations between personal resources.

### 1.2 Structure of the Thesis

We start in chapter 2 with a discussion of background information and work related to this thesis. In chapter 3, we introduce our approach to create associations between personal resources in a lightweight way. In this regard, we discuss our system design goals and roadmap for our work in designing and developing the system. In chapter 4, we analyze the scenarios where link creation is possible for end-users and follow up with the design of multiple use cases. Chapter 5 discusses the feasibility study we did for the implementation of the use cases. Chapter 6 provides a detailed description of our system architecture, including implementation details of the individual components. In chapter 7, we provide the details of the evaluation of our implemented prototypes which we did through a user study. The final chapter concludes this thesis and outlines future work.
1.2. STRUCTURE OF THE THESIS
This chapter covers related work in associations between resources in Personal Information Management (PIM) systems. Firstly, we study how existing systems have introduced the concept of associations to define relationships between resources. Secondly, we consider how these systems enabled end-users to use these associations for navigation and retrieval of resources. We analyze various techniques used by existing PIM systems in creating associations/links between personal resources. In addition, we discuss the interesting user interfaces provided by these systems to associate resources and benefit from them.

2.1 Associations between Resources in PIM Systems

In 1945, Vannevar Bush postulated Memex, "a device in which an individual stores all his books, records, and communications, and which is mechanized so that it may be consulted with exceeding speed and flexibility" [4]. Bush has envisioned two fundamental features such a device should provide: annotations and links. In recent years, various Personal Information Management (PIM) systems have extended Bush’s vision and introduced the concept of associations in their systems to relate personal information resources. Based on how these associations are created by these systems, we can broadly classify them into two categories - Content-based associations and Activity-based associations. Content-based associations are created by analyzing the content and metadata of resources. Activity-based associations are created based on observing specific user events.

2.1.1 Content-based Associations

Many of the early PIM systems used content-based associations to relate resources. These systems created associations based on similarity of the content or metadata provided by contextual cues such as time, author, thumbnail etc.

Haystack[14] is one of the early projects which emphasized on associations in impro-
2.1. ASSOCIATIONS BETWEEN RESOURCES IN PIM SYSTEMS

Haystack stores arbitrary objects of interest to the user and records arbitrary relations between them. Haystack uses a uniform resource identifier (URI) to identify any object of interest, for example a document, a person, a task etc. and used these URIs as endpoints between associations. Also it provides a user interface which presents this information to the users and allows them to manipulate the objects and associations in a meaningful fashion. For retrieval of objects, Haystack uses the associations to provide an associative browsing model similar to the World Wide Web.

MyLifeBits [13] was a project by Microsoft Research which is very similar to Haystack but concentrated more on supporting many visualizations of data. The project implemented a digital store for storing digital media including documents, images, sounds and videos. MyLifeBits used the concept of links to annotate data where a link indicates that one resource annotates other. A resource can link to any number of other resources and can be linked by any number of others.

Compared to MyLifeBits, Stuff I've seen Seen (SIS) [11] covered a wide range of information sources. SIS presented a system for personal information retrieval and reuse based on the fact that knowledge work involves finding and re-using previously seen information. SIS provided a unified index of information that a person has seen on their computer in different information sources like email, files, web, and calendar. As the information has already been seen before, rich contextual cues such as time, author, thumbnails and previews can be used to search for and present information.

Semex is Personal Information Management system proposed by Yuhan et al., which extended keyword search with search by associations [5]. Semex is a link based system and enables browsing by associations. Semex addressed the issues of data integration, data cleaning and data analysis which were not addressed by Haystack. Semex creates a database automatically from information extracted from multiple types of data sources. To create associations between heterogeneous data from multiple data sources, Semex runs a reconciliation algorithm. The algorithm reconciles and integrates data from different sources and constructs content-based associations by extracting metadata. Semex also provides an interface that combines intuitive browsing and a range of querying options.

Chaue et al. proposed Feldspar (Finding Elements by Leveraging Diverse Sources of Pertinent Associative Recollection), a system that supports associative retrieval of personal information on the computer [6]. Feldspar stores the associations between the resources as an association graph separately. The user interacts with this association graph through Feldspar’s user interface, which serves as both a query construction tool and query results tool. Feldspar supports seven kinds of information sources - email, person, file, folder, webpage, event and date. Feldspar uses Google Desktop to create the database for indexing and keeping track of the information items.
2.1.2 Activity-based Associations

The systems which we have examined till now establish associations based on similarity in content or metadata. Now we discuss some systems which create associations by analyzing user activity and the contexts in which the user works. Some of these systems have used only activity-based associations for relating resources while others have used both content-based associations as well as activity-based associations.

Elin et al. proposed Ivan which discovers and visualizes relations among documents based on users interaction with them [17]. Ivan used a simple set of heuristics to create useful relationships between documents that have been touched by the user. Ivan monitors events like when a user switches back and forth between documents or when a user cuts, copies and pastes from one document to the other. Document relationships are derived from the logged event data using algorithms which calculate the strengths of relationships based on the user activity. Ivan’s focus was only on creating associations between documents and it did not consider other personal information resources.

Beagle++ [8] proposed a desktop search prototype based on content-based associations and activity-based associations. Beagle++ enhances desktop search based on both resource specific and semantic metadata collected from different available contexts and activities performed on a computer. It focused on three main working contexts - email exchanges, file procedures (i.e., create, modify, etc.), and web surfing. Various activity specific heuristics are used to create links between desktop resources like local files, emails and cached web pages. Beagle++ relies on the notification events provided by the operating system kernel. The metadata generation is triggered by modification events generated upon occurrence of file system changes. Some of the events used are: a new file is copied to hard disk or stored by the web browser, a file is deleted or modified, a new email is read, etc. As Beagle++ is primarily a search prototype, it uses a page rank algorithm to rank the resources based on contextual information and displays the search results accordingly.

iMecho [7] extended Beagle++‘s approach by deriving semantic associations through mining from special patterns of user activity sequences. File access patterns and implicit user tasks are exploited to create many more associations. iMecho is an associative memory based desktop search system and enhances the conventional full-text keyword search with semantic associations derived from user activity contexts. iMecho creates three different kinds of associations - Content-based associations, explicit activity-based associations and implicit activity-based association. Content-based associations are created by analyzing the content and attributes of desktop resources, similar to other systems we discussed in the previous section. Similar to Beagle++, explicit activity-based associations are created when specific user events are observed. Implicit activity-based associations are novel to iMecho which are discovered by user access pattern analysis and resource lineage analysis. The personalized ranking scheme uses the associations along with users’ personal preferences to rank results by both relevance and importance. Also iMecho provided faceted search and association graph navigation that helps users to refine and associate search results generated by conventional full-text keyword search.
2.2 Retrieval of Resources based on Associations

A system that creates associations also needs to provide a way to use these associations. Most of the systems we have considered were developed in the context of search and retrieval of resources. In this section, we describe how these systems enabled end-users to use associations for these purposes. We discuss in the context of user interfaces provided by these systems and the typical user workflow in using them.

An early project which supported attribute-based document management and retrieval is Presto [10]. Presto presented a generic browser for document spaces which is more like a traditional PC desktop and gives a generalized overview of the document space, supporting organizational tasks and launching other applications. Systems such as Timescape [18] and the Lifestreams [12] system managed resources by associating time information to the resources. They supported a time-based retrieval of resources. MyLifeBits supported multiple query visualizations. It supported a detail view, thumbnail view, timeline view and clustered-time view. Detail view displays a list of the resources including each property. Thumbnail view shows miniature images of the resources in a grid information. Timeline view displays thumbnails on a linear time scale. Clustered-time view clusters thumbnails by similar time and arranges them in time order. The user interface allowed the users to switch between these multiple views.

The systems that we discussed till now provided hard-coded layouts of information. Haystack [1] provided a user interface with layouts that flex on the properties and relationships of the object being shown. This is achieved through a recursive rendering architecture in which each object is asked to render itself and recursively make the same request to other objects to which it is related. The user interface is defined by a set of views which are defined by a set of view prescriptions. A view prescription is a collection of RDF statements which guide how a display region should be divided and the related objects should be displayed. The view prescriptions can also be customized by users to handle new types of information, to add new properties or add new ways of looking at old information. Search functionality is provided as the primary way to retrieve resources. To provide a natural way of retrieving resources, Haystack also combines search with navigation. When a text query is entered into a search field, a collection of results is displayed. Each of these results in the collection can be used as a starting point for navigating further.

SIS [11] provided an interface which allows users to specify queries and to view and manipulate results. Contrary to the systems before where user needs to specify several properties and then launch a query, SIS provided a dynamic filtering mechanism. SIS provided filtering widgets in its interface and whenever any of these widgets are manipulated a query is launched. Semex provided a user interface where the user can either formulate a keyword query or a more specific selection query. Semex displays all the information about the object and enables the user to further browse the information by association.

Edward et al. proposed Phlat [9] a system which optimizes search and browsing through a variety of associative and contextual cues. Search for personal information resources should deal not only with the content but also the details associated with content. The idea of Phlat
is to develop a user interface which should provide the simplicity of web search but should also exploit the associations between information sources that users may remember. For this, Phlat proposed an innovative user interface where searching and browsing functionalities are combined together. The Phlat user interface comprises of a query area which contains the query text typed by the user and a filter area which comprises a set of buttons to select various orthogonal properties.

Most of the PIM tools did not support orienteering where the users specify and navigate to their destination in multiple relatively small steps. Feldspar is motivated by this idea and is the first tool to support multi-step associative retrieval of personal information on computer. Feldspar contributed an intuitive user interface that allows users to find information by interactively and incrementally specifying multiple levels of associations as retrieval queries. The user interface provides a query area for constructing the query visually and interactively. The user can incrementally construct a query and immediately see the updated query results in a result area.

iMecho provided a user interface which used faceted search and association graph navigation to browse and retrieve resources. Faceted search enables users to navigate in a multi-dimensional information space by combining text search with a progressive narrowing of choices in each dimension. The association graph shows all directly related resources to a specific resource. The user can browse through the graph by selecting any of the associated resources which again updates the graph with the related resources of the selected resource.

In the previous section we discussed Ivan, a tool which relates documents based on user activity monitoring. Though Ivan is only a document linking tool, it presented a simple and innovative user interface. Ivan presented itself as a simple side panel on the desktop with items representing open and related documents. When an item is clicked the systems opens the corresponding document and brings it into focus. As a result, the side panel gets updated again with the related documents to the currently focused document. The motivation behind providing such an interface is that it requires little training and learning.

## 2.3 Desktop Integration Techniques

In recent years, some PIM systems have explored techniques to integrate with the desktop environment to assist users in managing their resources. Many of these systems are activity management applications that enable end-users to manage applications and documents that are actively used for user tasks. In this section, we present an overview of some of them and discuss how these systems have provided ways to integrate with the desktop environment of end-users.

Jakob et al. presented activity-based computing which allows end-users to aggregate services and data in coherent sets called ‘activities’ [3]. They implemented their approach by extending the Windows XP operating system. The core user-interface components are the Activity Bar, Activity Icon, Activity List and the Activity Zoom. The Activity Bar replaces the Windows XP Taskbar and is the main user interface component. The Activity Icon, which is part of the windows’ title bar integrates activity-support with native Windows application
windows. The Activity List can be accessed from the Activity bar and shows the list of all the activities of the user. The Active Zoom component supports spatial 2D layouts of the windows.

Giornata [20] demonstrated how the traditional desktop interface can be re-framed to support activity management, while retaining the simplified interaction with applications and files. In Giornata, each activity in the system is associated with a corresponding virtual desktop. Activities can be annotated with optional tags. Giornata’s interface closely integrates with the file and window management components of Apple OS X. Giornata augments the visual stack of OS X window manager by inserting two additional layers: explicit interaction layer and implicit interaction layer. The explicit interaction layer lies on the top of all other layers and allows the users to manage activities and provides the persistent visibility of the Contact Palette. The Contact Palette component is attached to one side of the display space and provides a persistent display of contacts relevant to the current task. The shared files region represented the resources that are automatically shared with the contacts associated with a particular activity. In this way, Giornata implicitly supported associations between contacts and files. The implicit interaction layer is anchored to the desktop wallpaper and provides a quick overview of the activity state. Giornata also allows users to navigate among open activities using a similar interface as available in operating systems like Windows and OS X.

Gerard et al. proposed TAGtivity [16], a system that enables end-users to tag resources in the context of their ongoing work in the PC environment. TAGtivity had two primary components as its user interface: TAGtivity Manager and TAGtivity Toolbar, which facilitated the creation and management of tags. The TAGtivity Manager is the main application window which supported a range of tag management functions. In addition to the application window, TAGtivity has implemented extensions to MS Office applications and Internet Explorer to facilitate tagging from these applications. The TAGtivity toolbar which is located at the bottom of each of these application windows provides the tag information of the currently opened document or the webpage and also allows to retrieve the associated documents through tags. Through integration with the file system, TAGtivity also enables users to associate files and folders with tags. When a user drags a resource from the Windows explorer onto a tag in the application window, the resource gets associated with the tag. In addition, the context menu of the Windows explorer is extended to display the associated tags of files and folders.
In this chapter, we introduce our approach for this research work. In this regard, we discuss our research goals and present a contextual overview of our system. We place our solution in the context of previous related research and explain the benefits of our design decisions.

In the previous chapter, we have discussed various Personal Information Management (PIM) systems which have used associations or links to relate resources. These associations are used to convey relationship between resources based on annotations as well as to provide a means of navigation and information retrieval. Some of these systems constructed these associations based on content and metadata while others analyzed user activity. In most of these cases, the associations between resources were created implicitly by the systems and do not require any user intervention. Hence, users were not provided with the flexibility to create explicit relations between resources manually. But we can think of some scenarios in our daily computing tasks where manually associating resources can be quite essential. For example, let us say a user wants to associate two documents because he knows that they are related in some way. However, the documents do not share any relatedness through the content or metadata. Also, the user had never used these documents together in his computer before. In this scenario, the systems which implicitly create associations fail to create an association between these documents. In our approach, we want the user to have the flexibility to create manual associations in scenarios like this. Our aim is to support implicit associations as well as manual associations in specific scenarios.

We have seen that the existing systems provided interesting user interfaces to use these associations for search and retrieval of resources. Systems like iMecho have provided very advanced interfaces like faceted search. In most of these cases, the tool runs as a standalone application on the computer. To search or retrieve a resource using associations, the user needs to invoke the tool separately. This is an extra burden for the user as he needs to interrupt his usual workflow to do a context switch. The user has to launch the PIM tool, run his query or navigate through the associations to retrieve the resources. Then again he has
Figure 3.1: Contextual overview of our approach

to switch back to his usual workflow to continue on what he had been doing. Similarly, in the case of systems like Haystack, which allow the user to arbitrarily link resources, the user needs to invoke the user interface to create these associations. In our approach we want the user to be able to associate resources as well as use the associations as a part of his workflow. The interaction required to retrieve a resource based on associations should be minimal and should not deviate the user from his workflow.

Earlier PIM tools did not provide any way of notifying the users if a resource is associated with other resources. The user needs to explicitly invoke the application and search for the resource to know its associations. The other problem is that the user will never come to know that a resource has associations, unless he explicitly wishes to know. But this may be essential in some scenarios. Consider an example, where a multimedia file is embedded in some other documents. If the user forgets that the document is being referenced by other documents, there are chances that he may unknowingly move it to another location, edit or delete it. In this case, even though the resources may be associated by the system, the user is not notified about the associations involved. In our approach, we want to provide notification mechanisms which notify the user about any existing resources whenever the user handles a resource.

We aim to create associations implicitly and allow the users to link resources manually in some scenarios. Also, we want the users to create and use these associations as
part of their workflow. A standalone application is not a solution for us as we face the same problems as the previous systems which we have discussed. Instead we want to have a set of standalone components/applications which run independently and collectively provide the functionality of associations. A contextual overview of our approach is illustrated in Figure 3.1. Also wherever possible we want these components to be integrated into the existing user applications as well as the desktop environment. By doing so our system components are tightly integrated with the user environment and require minimal deviation from the user’s workflow to create and use associations. This is another aspect of our approach that differentiates from the existing works. The components interact with the integration layer to persist and retrieve the links. The integration layer contains a data store to store the associations. In addition, it provides multiple interfaces for the components to interact with the data store. The standalone components together with the integration layer define our system. In addition, we provide a unobtrusive notification mechanism which alerts the user whenever he works with a resource that has associations.

3.1 System Design Goals

Our system is specifically designed to support the following goals, each of which will be explained in further detail below:

- Lightweight Link Creation
- Desktop Integration
- Unobtrusive Notification Mechanism
- Bi-directional Links

**Lightweight Link Creation**

Our first design goal is to provide a lightweight link creation mechanism that enables users to associate resources easily and efficiently. Links should be created either implicitly based on user workflow or explicitly by the user with a minimal user action, based on the exact scenario. In the case of automatic link creation, links should be implicitly created by the system. For example, when a user opens and edits a set of documents together every time, the system should automatically associate these documents together. Through implicit link creation, we allow the users to forget the task of link creation but have the benefit of using associations when needed. As the whole process of link creation takes place in the background for automatically created links, we strictly confirm to our goal of lightweight link creation.

We attempt to create links automatically in most of the scenarios and limit the effort required by the user to minimum. But in some cases, it sounds more meaningful to allow the users to decide if they wish to create an association. In these cases, we intend to provide the flexibility to the users to associate resources by themselves, thereby granting them more fine-grained control. In the case of manual link creation by the users, we want the users to create the links in a lightweight way. The link creation task should be as simple and easy as possible. Our software components should not provide any unnecessary overhead by demanding the users to interact with complicated interfaces.
3.2. ROADMAP

**Desktop Integration**

We plan to provide the functionality to associate resources through a set of application components that allow the users to create links within their workflow. They extend existing user applications with this feature. These components together define our system which integrates directly with the desktop environment of the end user. It is our intention to provide these individual software components instead of a separate standalone application which imposes additional effort to the users to launch it every time to create links. These components run independently in their own context but collective store and retrieve resources from a single data store.

Also the ability to use the associations should be incorporated into users’ workflow. Just like the way resources are associated, the user should not do anything out of context to use these associations. As our system integrates with the existing desktop environment, it should provide an easy access mechanism to access the links at any point within the user workflow. Existing applications that are used by the end-users for handling resources should be extended with features to support retrieval of resources based on associations.

**Unobtrusive Notification Mechanism**

Our system should provide a notification mechanism to notify the users whenever a resource that has associations is used. The notification should subtly remind the user about any existing associations of this resource. However, our goal is to provide an unobtrusive mechanism to assist the users in making aware that a link exists. In this way, we will be able to draw the user’s attention on the existence of a link and the benefit he may get out of using the link.

**Bi-directional Links**

Our last goal is to have bi-directional links. For example, if we create a link from a resource A to resource B, we also create a link from resource B to resource A. This is a very useful feature especially for navigation and retrieval, as it enables one to navigate using the link in both the directions i.e. from resource A to resource B and vice versa.

To get a clear picture about the bi-directionality of links, let us consider an example. Microsoft Outlook provides the functionality to add hyperlinks to any file or folder location in a calendar appointment. But as the link is unidirectional, it would only be possible to navigate from the calendar appointment to the local file. If we make this link bi-directional, we can allow the users to navigate not only from the calendar appointment to the local file but also from the local file to the calendar appointment. This simple example scenario clearly shows the power of bidirectional links.

### 3.2 Roadmap

We broadly divide our approach for this research work into three main phases - study and design, implementation and evaluation. Each of the main phases has been divided into sub phases: scenario analysis, use case elicitation, feasibility study, implementation, evaluation and analysis. The roadmap for our work is shown in the figure 3.2.
Followed after the background study, we start with the scenario analysis phase. In the scenario analysis phase, we want to find out where exactly the concept of links can be introduced into a user’s workflow. For this, we need to understand how end-users handle their personal resources. We are interested in the kind of workflows the end-users follow while using personal resources. We want to know about any specific applications used by them to manage resources and the kind of user interfaces they had to interact with. A good way to get these details is to find them out directly from the end-users. So we conduct semi-structured user interviews.

From the findings of the user interviews, we analyze the scenarios where associations between resources would be advantageous to the users. But as discussed in the previous section, our main goal is to allow the users to create and use links as part of their workflow in a lightweight way. Confirming to these criteria, we design a set of use cases which identify the possible points in the user workflow where resources can be associated. Also the use cases should explain how the user would benefit by using the associations in each case.

We follow up with a study to find out the feasibility of implementation of the use cases. Also for each of the use cases, we consider different options for implementation right from choosing the appropriate technologies to defining the user interfaces. While designing the user interfaces, we need to ensure that they are well integrated with the desktop environment. Finally, proof-of-concept prototypes should be implemented for the most promising use cases. These prototypes together define our system.

After the implementation of the system, we want to evaluate our system with users to know the perceived user experience while using our prototypes. The feedback from the users
helps us to analyze how our system with the concept of semi-automatic links helps them to improve their productivity in tasks involving personal resources.
Scenario Analysis and Use Case elicitation

In the scenario analysis phase, we analyze how end-users work with their personal resources and manage them. Typical insights which we want to find out include the types of applications used by the end-users, the user interfaces that are provided by these applications and the workflow of users. So, we conducted semi-structured interviews with end-users. From the findings of the user interviews, we inferred a set of use cases which identify the possible places in the user workflow where link creation functionality would be advantageous. Firstly, we shall discuss about the user interviews that we conducted and then proceed to use case elicitation.

4.1 User Interviews

The primary goal of the interviews is to understand how users manage their personal resources like documents, media resources, web resources and how they use utility applications like email client and calendar. In particular, we want to find out how users store and retrieve their personal resources in their computers and the kind of workflow patterns they follow. The interviews were semi-structured and had both open ended questions as well as specific questions which were targeted against designing use cases. The complete set of questions can be found in Appendix A. We conducted interviews with twelve users who were chosen from our laboratory as well as from other departments. Most of the participants were using Microsoft Windows as their operating system.

Document Management

The participants were asked how they manage their documents in their computers. Specifically, we tried to find out from them how they store their documents and retrieve them. Interestingly, most of the participants used the folder hierarchy provided by the operating
Figure 4.1: Various options for storing and retrieving resources plotted against the number of participants using them

system to store their documents. To access the documents, some of them used shortcuts and search features, while majority of them mentioned that they use the default file explorer based navigation. Few users reported that they use applications like Mendeley\(^1\) for handling specific categories of documents like scholarly articles.

**Media Management**

We got a mixed response from the participants on how they manage their media resources like images, videos and music files. For images, file explorer based navigation seems to be the best alternative. Very few participants mentioned that they use applications like Picasa\(^2\) to manage their images. A couple of participants reported that they had used Picasa previously and now switched to file explorer based navigation. For music, applications like iTunes\(^3\) seemed to be a better choice compared to direct navigation through folder hierarchy. Also for the case of videos, most of the users mentioned that they use file explorer based navigation to access them. Also four participants mentioned that they do not store videos locally and often just watch and delete them.

**Folder Organization**

As many of the users mentioned folder hierarchy as their primary choice for organizing their documents and media resources, we were tempted to ask the users how they organize their

---

\(^1\)http://www.mendeley.com/

\(^2\)http://picasa.google.com

\(^3\)http://www.apple.com/itunes
resources across the folder hierarchy. Specifically, we tried to find out how they group their content into folders and organize them across the hierarchy. From the responses of users, the depth of categorization of resources into folders seems to depend on the type of content stored underneath the folders. For examples, most of the users followed a deep hierarchy for storing images, while for videos they followed a relatively flat hierarchy. Grouping the content based on the topic or event seems to be the favorite choice for organizing resources across the folder hierarchy. Interestingly, seven of the twelve participants mentioned that they associate chronological information either to store or to retrieve the resources. As the operating systems do not support any default mechanism to store the resources based on chronological information, the users tried to either add time or date information to the folder name or arrange their folders by time or date. Some participants mentioned that they only access their resources based on chronological information which they do it by searching for the file and then sorting the search results based on date and time.

Document Associations

To understand the specific scenarios where associations between documents would be meaningful, we asked the participants how often they work with related documents. As expected, many of the users responded that they do so very frequently. We tried to find out some example scenarios from the users where they had to work with multiple related documents at the same time. Some of the scenarios that were mentioned included reading scholarly articles, preparing a lecture, preparing a report etc. But the participants mentioned that to open these documents together they had to access each of the documents individually through folder navigation.

Web Browser and Bookmarks

Most of the users responded that they use bookmarks. Many of them used categories in the web browser’s bookmark manager to categorize their bookmarks. Surprisingly, very few users responded that they use tagging to bookmark. Using the bookmarks bar is the favorite option by many users to store their important bookmarks. Also, few users have mentioned that the bookmarks bar is their only way of bookmarking and showed disinterest towards categorizing and tagging.

We asked the users if they open multiple related websites at the same time in the web browser. Many of the users mentioned that they do so frequently. Browsing scholarly articles, social networking websites and news websites seem to be the common scenarios. When we asked the users about their workflow in this scenario, we got a mixed response from the users. Opening the websites individually or opening the last saved browsing session is the favorite choice for many. Few users responded with other choices like using bookmark categories or RSS readers.

Downloads and Uploads

We asked the users how they manage the downloaded content on their computer. Based on the responses, we can categorize the users into ones who download to a default location and others who organize the downloaded files into appropriate folders. More than half
of the users responded that they download to the default location which the web browser provides. As a follow up question, we asked these users how often they clean up or maintain their downloads folder. Again, a majority answered that they never maintain it or often leave it as it is. We also asked the users whether they remember all their downloaded files and avoid downloading an already downloaded file. Surprisingly, many users reported that they re-download files very frequently. Others, who replied that they do not re-download, mentioned that they often search for already downloaded files to avoid re-download.

We tried to find out from users how often they upload content. Uploading photos to social networking sites seems to be a common task. Most of the users responded that they traverse through the folder hierarchy to select the target folder from where they want to upload files. A couple of users mentioned that they prepare an upload folder which is easily accessible through the web browser and put all the upload content in this folder.

**Email**

We asked the users on how they organize their email. Interestingly, half of the users responded that they do not organize their email. Search based access seems to be the most common way of accessing their email. Others, who organize their email, mostly do it either manually through folders or automatically through filters. We asked the users specifically how they deal with email attachments. Most of the users responded that they save their attachments to local disk. As a follow up question, we asked these users while working with a saved attachment how often they had to access the original email from which it had been saved. Surprisingly, many of them mentioned that they encountered such a scenario frequently. An interesting scenario in which they had to reaccess the email is when they had to edit an attachment and send it back to the original sender.

**Calendar**

We concluded the interview asking the interviewees if they use any calendar application and for what purposes they use it. All except one replied that they use a calendar application. Google calendar, Outlook and mobile based calendar applications are the various types of calendar applications used. Outlook is used as the calendar application by more than half of the users. All of them used calendar to keep track of their appointments and reminders. As a follow up question for the Outlook users, we asked if they knew that they can create hyperlinks to personal resources from calendar appointments. Most of them replied that they do not know that such a feature is available but felt that it was a very useful feature.

### 4.2 Inference of Use Cases

The user interviews helped in designing use cases which encompass two main tasks. Firstly, the use case should identify the possible place in the user’s workflow where link creation functionality can be placed. Secondly, the use case should identify key factors that provide an incentive for users to actually spend the additional effort of creating a link. Each of the use case is described through an example user workflow, link creation procedure and the user benefit.
4.2.1 Document Links

As we saw from the user interview results, storing or retrieving the documents from the folder hierarchy is the favorite choice of many users for managing documents. Working with multiple related documents seems to be a common task. However, considering the fact that many store their documents across the folder hierarchy, a semi-automatic way to associate the documents and access the documents through these links would definitely ease the user experience. This motivates us to present the use case where the users can implicitly or explicitly create links across documents and use these links to access the linked documents easily and efficiently.

- Example user workflow: Let us say, a user is planning his travel and puts his travel related details in a document. He also maintains two separated documents for budgetary decisions and schedule. All of these documents are stored at independent locations in the folder hierarchy, as each of the documents has its own purpose. But it is very common for the user to access the budget document or the schedule document whenever he accesses his travel document and vice versa.

- Link creation: As the user opens all these documents at the same time, an option to link all these open documents should be provided.

- User Benefit: When the user opens one of the documents, he should be able to easily open the linked documents at ease. For example, in the scenario considered, whenever the user opens the travel document, the user should be able to access the budget document or the schedule document with minimal user action.

4.2.2 Associations between Web Pages

Though most of the users use bookmarks, storing and accessing the bookmarks through the bookmark bar seems to be the preferred option rather than categorizing the bookmarks. Also, opening multiple tabs to open related websites is a common scenario while browsing. Most of the users in the interviews mentioned that they do so only by accessing each of the web pages individually. One may use web browser bookmarks to bookmark all of the related webpages into a single category. However, even with bookmarks, a typical user workflow would be to check out the current webpage’s category in the bookmarks, navigate to the category and open the related webpages. This involves some user action which can be minimized by creating associations between the related web pages. This motivates us to design the use case of context aware bookmarking, which can help the users to access the related bookmarks from the web page itself.

- Example user workflow: A simple example scenario is a user opens a group of news websites or a group of review websites together. For a scholarly person, opening set of scholarly web pages together is a common scenario. You would like to group or associate them, so that whenever you access one of them, you have a chance to access the other associated web pages.

- Link creation: In general, related web pages are opened at the same time across different tabs in the web browser. The user can be allowed to create associations by selecting the web pages in the opened tabs in his browsing session.
4.2. INFERENCE OF USE CASES

• User benefit: Once the links are created, whenever the user accesses one of the related webpages, the user can be shown the addresses of the associated web pages in the web browser immediately. This allows the user to easily navigate to the other pages, without any need to access the bookmarks section.

4.2.3 Associations between Messages and Saved files

We save email attachments to the local disk for later use. Also sometimes during Instant Messenger (IM) conversations, we receive files and store them to the local disk. But when we access the local files, we may want to access the original email or the original IM message from which we saved the file. Even during our discussions in user interviews, many users mentioned that they save their email attachments locally and had to reaccess the email sometimes while working with the attachment. This provides an incentive to create links between the saved attachment and the email or IM conversation, through which the users will be able to access the original email or IM conversation from the saved attachment. This motivates us to design the use case of bi-directional links between emails/IM conversations and saved files, which we discuss below.

• Example user workflow: A user receives an email with an attachment and saves the attached document to his local disk. Later, the user edits the document and in order to understand the context, he wants to open the email in which he received the file. Also the user may want to send the email back to the sender after editing the document.

• Link creation: Whenever the user saves an attachment from an email, a bi-directional link can be created between the local file and the email.

• User benefit: Whenever the user accesses the local file, he can immediately access the original email from which he had saved the attachment through links. This will help to reduce a lot of steps that the user would have done in order to access the email in the normal case.

4.2.4 Temporal Associations

In some scenarios, we want to associate date/time information to our resources. Even from the feedback of user interviews, associating chronological information to personal resources seems to be implicitly or explicitly done by the users to store or access the resources, especially for images and documents. As the existing operating systems do not provide this feature by default, providing the users a way to link the resources to chronological information would definitely help in improving the user experience. Though existing applications like Microsoft Outlook provide the concept of hyperlinks to associate calendar events to resources, the links are unidirectional in nature. For example, you can only create a link from an appointment to the resources but not vice versa. But as our links are bi-directional in nature, we allow the user to navigate across the resources in both the directions.

• Example user workflow: A simple example scenario is a user that attaches a file to his upcoming calendar event and later wants to access the calendar event when he accesses the file. Other examples are when a user wants to associate date and time information to his photos or would like to link his electronic flight tickets to his travel date.
• Link creation: Whenever the user attaches a file to a calendar event, an association should be implicitly created between the file and the calendar event. In other cases, the user should be able to manually link resources to a particular date.

• User benefit: A provision to bi-directionally link resources directly to calendar events would benefit users a lot. Using links, the user can use the calendar event to access the associated resources. Also by accessing the resource, the user can have a provision to open the associated calendar event directly.

4.2.5 Associations between Downloaded files and Web pages

In the user interviews, most of the users mentioned that they download directly to the default location chosen by the browser. Forgetting whether a resource was previously downloaded and redownloading the file again seems to be quite common. To avoid this case, there seems to be no alternative rather than to browse or search to find out if the file had been downloaded previously. Thus, providing an option to allow the users to link a webpage with the files downloaded from it would definitely help the users to avoid searching or re-downloading files. Also it allows the users to easily access the associated local files from the web page in the web browser. Similarly while using a local file, there exists no alternative to access the original webpage from which it has been downloaded. A link between the local file and its source web page would allow easy navigation to the web page while using the local resource.

• Example user workflow: We download a lot of files and store them in our local disks. But sometimes we may want to access the page from which we had originally downloaded the file. Also we tend to download files multiple times, unaware that we have downloaded that file previously. In the latter case, we would like to know if we had already downloaded the file, and the location to which we had downloaded the file.

• Link creation: Whenever the user downloads the file, the user can be provided an option to link the downloaded file to the source web page from where it is downloaded. Following options may be considered:

  1. The user can configure the option to create the links automatically whenever he downloads a file.

  2. Whenever the user downloads a file, the user can be notified and prompted whether he would like to link the file being downloaded to the webpage. If the user agrees, a bi-directional link gets created from the webpage to the location of the downloaded file.

• User benefit: Whenever the user accesses the local file, the user can immediately access the web page, which has been linked to this file while downloading. Also when the user wants to download a file and would like to know if he has downloaded the file or access the already downloaded file location, he can immediately follow the link previously created. Also whenever, the user tries to download a file multiple times, the user can be prompted that the file has been already downloaded pointing to the link.
4.2.6 Associations between Uploaded files and Web pages

In some situations, we would like to associate the local files in the computer to their corresponding online versions. For example, we upload our photographs to social networking websites like Facebook/ Flickr. But we may want to access the local file location on the computer from which the photograph has been uploaded. There can be many reasons for it, for example to edit the picture, or to access a better version of the photograph stored on the computer or to access the entire folder on the computer. Vice versa when you visit the local file on the computer, you may wish to visit all the web pages which are associated with this file.

- Example user workflow: A user browses through his image collection online (say Facebook/ Flickr). The user wishes to access the local version of it.

- Link creation: Whenever the user uploads a file, a link can be implicitly created from the local file to the web page to which the file is getting uploaded.

- User benefit: Whenever the user accesses the local file, the user can immediately access the web page, which has been linked to this file. Similarly, the user can access the local file from the associated web page without any extra effort.

4.2.7 Linking Embedded Files and Documents

While working with Office suites like Microsoft Office, we embed images or other media files into our documents, presentations etc. In general, Office suites support this feature through a simple drag and drop operation of the file into the document. But when we use a file, we may want to know if the file has been embedded into other documents as any changes to the file may affect the documents into which it is embedded. Also we may want to access an embedded file from the document in order to make changes to the file. Though Office suites create associations between the documents and the embedded files, the associations are unidirectional. One can access the embedded documents from the document they are embedded in but not the other way. Through a bi-directional link, we could allow the users to navigate in both the directions.

- Example user workflow: A user embedded an image into multiple documents and presentations. As the user wants to edit the image, he wants to know all the documents into which he has embedded the image file.

- Link creation: As the user embeds the file into a document, a link should get created automatically between embedded file and the document into which the file is embedded.

- User benefit: Using links, the user can readily access the embedded files from the document. Similarly while using a file, the user can know to what all documents the file has been embedded into and readily access them. This helps especially in avoiding deleting or editing the file unknowingly which may affect the documents into which the file is embedded.
In the previous chapter, we have designed several use cases which describe where exactly end-users could create and use links in their workflow. In this chapter, we will discuss the feasibility of implementation of software prototypes that realize the designed use cases using existing technologies. We shall discuss the various types of technologies we considered and present our analysis in choosing some over others for the implementation of the prototypes. Firstly, we will discuss the different kinds of application choices we had for implementing the prototypes. Then we discuss about the selection of relevant ones among them for the implementation of the use cases. Also we will consider other important factors in the design of our system like the choice of programming languages, choice of database, choice of utility applications etc.

We select the technologies and design the user interfaces mainly in the context of our system design goals as discussed in the chapter 3. The primary design goals are to have a lightweight mechanism for creation and access of links. Also the links should be created as a part of user’s workflow. Finally, the links created should be bi-directional in nature allowing access from both ends of the links.

5.1 Range of Technologies

Various technologies were considered as possible choices for developing the prototypes for the use cases. We considered different applications a user would use in his normal desktop computing routine. Figure 5.1 illustrates the wide variety of technologies that we have considered for developing our prototypes. We decided to develop only for the Microsoft Windows platform but our approach is also applicable for other platforms that provide similar extension features.

- Desktop applications
  Desktop applications provide a simple and easy option for developing prototypes. They
run standalone and provide a user interface to interact with. Considering the various technology choices and the rich resources we have for implementing desktop applications, they seem to be a good choice for developing our prototypes. But the primary disadvantage with desktop applications is that they have to be invoked separately from the user’s workflow. So an important consideration to be made while choosing a desktop application to implement our prototypes is that the application has to be lightweight and it should be very well integrated with the user’s workflow. The user should perform minimal user action to launch and use the application.

- **Explorer Integration**
  Windows Explorer is a file manager application that is included with the Microsoft Windows operating system and provides a graphical user interface for accessing the file system. It is responsible for many user interface components of the operating system such as the taskbar and desktop. Shell Extensions are in-process COM objects which extend the functionalities of Windows Explorer. Through shell extensions, the Windows explorer can be customized for different features. For example, we can have a customized context menu or display a customized overlay icon for a shell object (folder or file).

- **Desktop gadgets**
  Windows Gadgets\(^1\) are mini applications for Microsoft Windows 7, which are generally designed to do simple tasks, such as clocks, calendars, RSS notifiers or search tools. They are customizable and can be resized and placed anywhere on the desktop. They can also be used to control external applications such as Windows Media Center.

- **Office Suite extensions**
  We work with our documents using Office suites. Many Office suites facilitate ex-

---

\(^1\)http://windows.microsoft.com/en-US/windows/downloads/personalize/gadgets
tensibility of their features by allowing developers to write extensions. For example, Microsoft Office add-ins allow to complement the basic functionality offered by applications like Word, PowerPoint, Excel etc.

- **Email and Instant messenger add-ons**
  Email client applications are used to access and manage user’s email. Some popular email clients like Outlook, Thunderbird provide the features to write extensions to add new functional capabilities and automate some routine operations. Also Instant messenger software like Miranda IM and Pidgin allow to complement their basic functionality and add additional features through extensions.

- **Web Browser extension**
  Web browser extensions allow developers to extend the functionality of the web browser. Most of the popular web browsers like Internet Explorer, Firefox and Google Chrome allow developers to write extensions for them. Web browser extensions also allow extending the user interface of the web browser in various ways.

### 5.2 Selection of Technologies

In this section we discuss the technologies selected by us for implementing the use cases. We have to make a selection of different aspects, from the choice of application to choice of database.

#### 5.2.1 Applications and Extensions

As elicited in the use case to create links between documents, we intend to support both automatic and manual ways of associating documents. To create links automatically, we need to provide a mechanism to monitor the currently active documents. For this we need a service/application which runs in the background and monitors the currently active documents. However to allow the user to manually create links between documents, we need to provide the user with a user interface to interact with. But the application should be readily accessible by the user any time from the user’s workflow. System tray and sidebar provide an easy way to access the applications in Windows. The System tray also provides a way to notify the user with a balloon with useful information. This feature is especially useful for us to show a notification when a document has related documents.

It is possible for the user to access the linked documents at two points in his workflow. The user may want to open all the linked documents together. For this we need to provide a way the user can open all the linked documents as he opens one of them. One way is to provide an option to open all the related documents by customizing the application window using an add-in. The disadvantage with this approach is that all the applications that operate on documents like the Office suite and Acrobat have to be customized. A more natural way is to integrate the notification and access mechanisms with the explorer. This is where shell extensions are useful.

In the use case involving email client, we need to associate the saved attachment to the original email. For this we need to extend the functionality of the email client. We chose
Outlook as the email client for our prototypes because of its popularity. Outlook add-ins provide us a way to extend the functionality of Outlook to create links. As our links are bi-directional, the users should also be able to access the email from the saved attachment. The shell extensions again provide the mechanism to access the email from the explorer just like the case of documents.

Associating chronological information to files through links is one of the interesting use cases we have elicited. A possible way to implement is to have a separate calendar application which provides the user a way to link calendar dates to files. Implementing the calendar as a desktop gadget would be a lightweight choice. The other option is to use the existing calendar application like Outlook to create associations. Using an existing calendar application is more integrated into user’s workflow. Also we need not implement a separate application just to create chronological links. Hence, we opted to implement the chronological links feature through an Outlook add-in. As with the other cases, shell extensions provide the mechanism to use the bidirectional feature of the links by providing the access to the calendar events from local files.

For associations between webpages and links between webpages and downloads, web browser extensions are our favorite choices for prototypes. To allow the creation of web page associations manually, we need to provide special options in the web browser’s user interface which allow the user to select web pages to associate them together. Similarly, to create associations between web pages and downloaded files implicitly, we need to monitor for download activity in the browser. Web browser extensions allow us to extend the web browser functionality with such features.

<table>
<thead>
<tr>
<th>Use case</th>
<th>Technologies</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Document links</td>
<td>Desktop application, Shell extensions</td>
</tr>
<tr>
<td>2. Associations between web pages</td>
<td>Web browser extensions</td>
</tr>
<tr>
<td>3. Associations between messages and saved files</td>
<td>Office extensions (Outlook), Shell extensions</td>
</tr>
<tr>
<td>4. Temporal associations</td>
<td>Office extensions (Outlook), Shell extensions</td>
</tr>
<tr>
<td>5. Associations between downloaded files and web pages</td>
<td>Web browser extensions, Shell extensions</td>
</tr>
<tr>
<td>6. Associations between uploaded files and web pages</td>
<td>Web browser extensions, Shell extensions</td>
</tr>
<tr>
<td>7. Associations between embedded files and documents</td>
<td>Office extensions (Microsoft Word, PowerPoint etc.), Shell extensions</td>
</tr>
</tbody>
</table>

Table 5.1: Usecases and chosen technologies to implement

We selected to implement five of the seven use cases that we have designed in Chapter 4. We chose the most promising ones that will help us to demonstrate our approach. Also we considered the feasibility of implementation using the technologies that we have chosen. For example, to implement the use case of embedding local files to documents, we need to monitor for the events when files are dragged and dropped into documents. But in our study, we could not find a way to get notified of these events in our Office add-ins. We chose Firefox as the web browser for reasons discussed later. For monitoring uploads, Firefox does not provide an XPCOM object by default. We would need to write a complete XPCOM object on our own to have an extension to detect uploaded files. Considering the time factor and the complexity of implementation, we have not opted to implement this use case as well. Below is the final set of use cases which we have decided to implement:

- Document links
- Associations between web pages
- Associations between messages and saved files
- Temporal associations
- Associations between downloaded files and web pages

5.2.2 Programming Languages

A choice of programming languages has to be made for developing the prototypes. In some of the cases like web browser extensions, it is mandatory to develop in a particular programming language as required by the platform. But in other cases like desktop applications, we have an opportunity to choose programming languages which more suit to our needs.

As discussed in previous section, we implement a desktop application to create associations between documents. We considered the choices of Java, C# and C++. Java applications are very advantageous as they are platform independent and easy to develop. However C++ and C# provide an easier way to interact with Windows native functions and WINAPI, which we need for document monitoring purposes. C# provides a relatively simpler way to call existing code in DLLs. A C# program can call any function in any DLL using a combination of the extern keyword and the DllImport attribute on the method declaration. A major advantage of this is that neither the method being called has to be specifically written to be called from C# nor is any ”wrapper” necessary.

Visual Studio Tools for Office (VSTO)\(^3\) is a Microsoft Visual Studio add-on which provides a set of development tools for add-in development of Microsoft Office applications. VSTO allows to develop add-ins in either Visual C# or Visual Basic languages. We chose to write our add-in in C# as our other components were implemented in C# and also because of the availability of an existing driver for MongoDB. To implement the Firefox extensions, we did not have any options to choose. JavaScript has to be used for programming the application logic and Mozilla’s XML User Interface Language (XUL) for programming the user interface.

\(^3\)http://msdn.microsoft.com/en-us/office/hh133430
5.2.3 Database

The links created by our prototypes should be persisted in a data store. For this, we need a database which our software prototypes interact with and store the links. When considering a database, we have a lot of choices such as a relational database like MySQL or a document oriented database like MongoDB or CouchDB.

In our system, we do not find a highly dynamic range of queries full of outer and inner joins, unions and complex calculations over large tables. Links can be perceived in the form of documents where each document represents all the outgoing links from a single node. Also from a development perspective, as we have our prototypes in different programming languages, it would be beneficial to us if there are preexisting drivers to interact with the database. Considering all these factors, MongoDB is our favorite choice for the database.

MongoDB\(^4\) is an open source document-oriented NoSQL database system. It stores everything in BSON, which is the binary format of JSON [2]. MongoDB has official drivers for various programming languages like C, C++, C#, Java etc.

5.2.4 Web Browser

As web browser extensions are one of the main components in developing the software prototypes, we had to select a web browser to develop extensions. From the statistics of the user study, Firefox seems to be the favorite web browser among users. From the usage share statistics by Wikimedia, Firefox stands in the second position after Microsoft Internet Explorer with a usage share of 33.3\(^5\). Because of its popularity and the high degree of extensibility Mozilla platform offers, we chose Firefox as the web browser to develop extensions. Also there are a large number of resources and documentation available for developing Firefox extensions.

\(^4\)http://www.mongodb.org/
In this chapter we discuss the design and implementation of software prototypes which allow us to realize the use cases designed in chapter 4. We present the full system architecture and implementation of the system. We will also discuss the user interfaces that have been designed and implemented for interacting with our software prototypes. While designing these user interfaces, we have taken into account the principles of usability and human computer interaction (HCI).

6.1 System Architecture

Our system has a set of software prototypes which run independently along with a centralized database. We have developed the prototypes for Microsoft Windows 7 operating system. The latest versions of Microsoft Office (version 10) and Firefox (version 11) have been chosen as the Office suite and web browser respectively.

The main components in our system are the shell extensions, document linker, Outlook add-in and the Firefox extensions. The shell extension handlers are in-process Component Object Model (COM) objects and are implemented as DLLs. The document linker is an application implemented in C# which runs in the background and also provides a user interface that can be easily accessed through the system tray. Outlook add-ins are also Microsoft Component Object Model (COM) objects that enhance Microsoft Outlook application. The database is MongoDB, which is a document oriented database.

Each of the software components interacts with the MongoDB to either store the links or access the links. The interaction takes place through appropriate interfaces. All the components except the Firefox extensions interact with the database through programming language specific drivers. As the JavaScript code in the Firefox extensions runs in a sandboxed mode, the database cannot be accessed directly from the extensions using a driver. So we created a web service which provides a REST interface to interact with the database.
6.2 Data Model

We chose a simple data model to internally represent the associations in our system. MongoDB stores data in a structured format in documents using BSON (Binary JSON). A BSON document contains zero or more key/value pairs. In our system, a BSON document represents all the outgoing links from a single resource. This type of representation allows a direct and fast retrieval of associations for a particular document. The snippet below shows a typical document stored by our system.

```json
{
    "id" : ObjectId("4f7c367859c7b6099c000002"),
    "Source" : "C:\ Projects\Spring\MidTerm\Report.docx",
    "http://sites.google.com/semilinks" : 1,
    "D:\ Presentations\midterm.pptx" : 1,
    "email:\\3456789045612342" : 1
}
```
The key "id" is used to uniquely identify the document. The value corresponding to the key "Source" identifies the source resource from which the links are originating. The other keys indicate the destination resources of the links from the source resource. In this case, the BSON document indicates links from document "Report.docx" to a web page, a presentation document and an email. Emails and appointments are identified by a unique ID, which we will discuss later in this chapter. The values of the keys which identify the destination resources are used by some of our prototypes to store data related to the links. For example, the document linker application uses them to store the frequency of opening documents together.

6.3 Document Linker

The document linker application implements the document links use case, which we have discussed in chapter 4. The document linker application allows to associate documents by creating links between them. The documents can be any of Microsoft Word, Excel, PowerPoint or PDF files. The application allows documents to be linked in two different ways - automatic linking and manual linking. The application can be chosen to run in either of the modes or both. In the automatic linking mode, the application monitors the documents which are frequently opened together and associates them together by creating links between them. In manual linking mode, the user explicitly invokes the user interface of the application to create links. The application always runs in the background and can be accessed from the system tray. By providing the system tray option to access the application, we provide the users a readily accessible way to open the application any time. By clicking
on the system tray icon, the application window is launched as shown in figure 6.2. The left pane in the application window shows a tree view with all the existing links between documents. Each tree node indicates links from the parent node to the children nodes. The users can manage the existing links by using the context menu of the tree view. The right pane shows the list of currently active documents. The user can create links between documents by selecting the documents using the checkboxes and pressing the link button. Using the 'Select Local Files' button the user can also select documents that are currently not active.

The document linker application also provides a notification mechanism for alerting the user about existing links of a particular document. Whenever the user opens a document that already has been associated with other documents, a system tray balloon is shown indicating the user that the document has links to other documents. This is illustrated in Figure 6.3. The user can click on the balloon to open another window where he can select some or all of the associated documents to open. This technique conveys the appropriate information to the user without distracting him. Other notification mechanisms like a popup alert on document launch would be more noticeable but would also disrupt and annoy the user. Also the system tray balloon appears only at the first time a document is brought to foreground after it is launched. This is to avoid displaying the system tray balloon too frequently which can distract the user.

Document Linker Architecture

The application creates three helper threads on startup - active document monitor, active window monitor and association inference daemon. The document monitor thread continuously monitors all the documents currently open and maintains a collection of them. For this it uses the Microsoft Office Primary Interop Assemblies (PIA) to obtain the list of currently active
documents. The Microsoft Office PIAs contain the official description of commonly used Microsoft Office type libraries for applications such as Microsoft Word, Microsoft Excel, and Microsoft PowerPoint etc. These PIAs make it easier for managed code to interoperate with Office COM type libraries. The code script below shows how to get the list of currently open Microsoft Word documents using Microsoft Office PIAs from managed code in C#.

```csharp
public void GetActiveWordDocsList(HashSet<String> cur_set)
{
    try
    {

        {
            cur_set.Add(doc.FullName);
        }
    }
    catch (System.Runtime.InteropServices.COMException exc)
    {
        System.Diagnostics.Debug.WriteLine("WordList Exception");
    }
}
```

The active window monitor thread continuously monitors for the current foreground window of the user. This is done by a call to the `GetForegroundWindow` method in the user32 Windows DLL which returns the handle of the current foreground window. By using the handle, the active window monitor thread retrieves the window title. The code script below shows how to retrieve the title of the current foreground window.

```csharp
private string GetActiveWindowTitle()
{
    const int nChars = 256;
    IntPtr handle = IntPtr.Zero;
    StringBuilder Buff = new StringBuilder(nChars);
    handle = GetForegroundWindow();
    GetActiveWindowsList();
    if (GetWindowText(handle, Buff, nChars) > 0)
    {
        return Buff.ToString();
    }
    return null;
}
```

Using the current window title, it is checked if the current window is a document window or any other application window. If the current foreground window is a document, it is verified if the current document already exists in the collection of currently active documents. If the document does not exist in the currently active document list, it indicates a new document opened by the user. So the active window monitor queries MongoDB to find out if the current document has any existing links. If the document has already some links to other documents, the system tray pops up a balloon suggesting that there are some associated documents.

The association inference daemon thread is started by the application if the 'Automa-
tic Linking’ option is enabled by the user. This thread creates links between each pair of documents that are open together based on a condition. The condition can be the frequency of togetherness, where two documents are linked if they are opened together for more than a specific number of times defined by a threshold value. Or the condition can be based on how long the documents were open together. In our software prototype, we create links across documents based on frequency matching criterion.

6.4 Web Browser Extensions

The links between web pages and local files as well as links between webpages are realized through the web browser extensions. We have implemented two web browser extensions for the Firefox browser - Webpage linker and Download monitor. Webpage linker allows the user to create links across web pages. Unlike the document linker, the webpage linker allows the user to associate webpages only manually. The download monitor implicitly associates a downloaded file with the webpage from which it is downloaded by creating a link.

6.4.1 Webpage Linker

As discussed in the chapter 4, creating links across related pages is a very useful use case. By creating links across web pages, one can easily navigate from the opened webpage to its linked webpages. The web page linker helps the user to achieve this exactly. This Firefox extension allows the user to create links across web pages as well as use the links to access the associated web pages.

The user interface of the webpage linker involves an associations bar which displays all the associated webpages of the webpage in the current web browser tab. A context menu option to create and manage links is provided. To create links across related tabs, the user can launch the link creation window by using the context menu option. In the link creation window, the webpages that are to be associated with the current webpage can be selected from the list of all currently opened web pages across tabs. On confirmation, the Firefox extension communicates with the database to create links across the selected webpages.

When a new link is created, the extension sends a request to the Web service with the link details. The web service which is a C# application stores the link in the database. Also the associations bar has to be updated with the associated webpages every time a page is loaded or a tab is switched. For this appropriate listeners are registered in the JavaScript code to observe these events. Whenever the events occur, the JavaScript code calls the web service to get all the associated web pages of the current foreground web page. The web service queries the database and sends the result back to the Firefox extension. The addresses of the associated web pages are added as buttons in the associations bar. The figure 6.4 shows the associations bar being populated with associated web page addresses when a web page is visited. The user can easily access the associated web pages by clicking the buttons in the associations bar.
6.4.2 Download Monitor

The download monitor associates a downloaded file with the webpage from which it is downloaded by creating a link. The link creation process is done implicitly without any intervention from the user. Like the previous case the associations bar is used to provide a notification mechanism to show the existence of links. Whenever the user visits any webpage which has links to local files, the drop-down option in the associations bar named 'Local files' gets highlighted. When the drop-down menu is expanded, all the associated local files are displayed in the list along with their locations. Also by selecting an item in the drop-down menu, the user can directly launch the local file with its associated default application.

When the extension loads, it gets an instance of the Download Manager’s nsIDownloadManager interface. The nsIDownloadManager is an interface provided in Mozilla’s cross-platform component object model (XPCOM) to get the status of downloads in the web browser. The Download Manager instance is cached in a member variable and its addListener() method is called to start listening for download status updates. The onDownloadStateChanged() method in our JavaScript gets called whenever a download’s state changes. Once the download state changes to completed, we call the Web Service to create a link. In this call, we send the source web page and the local location of the downloaded file as parameters. The web service interacts with the database to create a bidirectional link between the source webpage and the downloaded file. We call the web service only after the completion of the download in order to avoid creating links for aborted downloads.

Also the extension registers a listener for the page load and tab switch events. When these events occur, the web service is called with the current web page address to find any associated links. The web service queries the database and sends the result back to the Firefox extension. The extension updates the results in the drop-down list. Also when one of the items in the drop-down list is selected, we need to directly launch the file with its associated default application. For this we use the nSIProcess interface provided in the XPCOM. The code snippet below shows how to invoke a system process from a Firefox extension.
By implementing an Outlook add-in, we realize two use cases - Temporal associations and associations between emails and saved files. Whenever an attachment from an email is saved to the computer, a bidirectional link gets created between the local file and the email. Similarly when an appointment is saved with a hyperlink to a local file, a bidirectional link gets created between the local file and the appointment.

For developing the add-in, we used Visual Studio Tools for Office (VSTO) which is a Microsoft Visual Studio add-on. We extended the basic template provided by Microsoft Office to develop an add-in for Outlook 2010 using Microsoft Visual Studio .NET Framework 4. The template is available in both Visual Basic and Visual C# languages. We chose Visual C# to implement the add-in for the reasons discussed in chapter 5. The provided template includes the OutlookItem class, which enables developers to work with generic Outlook items through late-binding, without determining the item type first.

To develop add-ins for Microsoft Office Outlook, we need to interact with the objects provided by the Outlook object model. The Outlook object model provides classes that represent items in the interface. For example, the Microsoft.Office.Interop.Outlook.Application class represents the entire application, the Microsoft.Office.Interop.Outlook.MAPIFolder class represents a folder that contains e-mail messages or other items, and the Microsoft.Office.Interop.Outlook.MailItem class represents an e-mail message. Outlook provides a wide variety of events, which an add-in program can get notified whenever an event occurs. For example, Outlook events can notify the add-in program when a new email has arrived or an appointment is closed.

### 6.5.1 Associating a saved attachment with an email

We need to create a link when an attachment is saved from an email to the local disk. An Outlook attachment can be saved from two different kinds of windows: Explorer and Inspector. The Explorer window represents what one would recognize as the Microsoft...
Outlook user interface, when the application is launched\(^1\). For example, when we open Outlook, we are working in the Outlook Explorer object. A window that contains a specific Outlook item, such as a mail message or a contact, is an Outlook Inspector object. We need to support the functionality to create links when an attachment is saved from both these types of windows.

For this we need to get notified with the event when an attachment is saved from the email. However the current event interface does not support the notification of an event when an email attachment is saved to local disk. So we need to provide an option to the user to create a link when he saves his attachment. Outlook allows us to programmatically customize the user interface from the add-in code. Explorer ribbons, inspector ribbons, context menus, contextual tabs etc. are the various UI components that can be programmatically customized. As the existing option to save an attachment from an email is through the context menu, we would like to extend it by providing an additional menu option to save an attachment with links as illustrated in figure 6.5.

Outlook provides the \textit{IRibbonExtensibility} interface through which the Ribbon user interface (UI) communicates with a COM add-in to customize the UI. The \textit{IRibbonExtensibility} interface has a single method, \textit{GetCustomUI}. To customize a UI component, we must provide XML markup to the \textit{GetCustomUI} method which supplies callbacks that we can use to respond to a button click, or to control the visibility of our ribbon controls. To customize the attachment context menu and add a new option, we provide a XML markup as shown below. The \textit{OnMyAttachmentSaveClick} is the callback method that is called when the option is selected.

\(^{1}\text{http://msdn.microsoft.com/en-us/library/aa141351(v=office.10).aspx}\)
In the callback method, we need to create a link between the locally saved file and our email. But for that, we should be able to uniquely identify the email. MAPI (Messaging Application Programming Interface) store providers are dynamic-link libraries (DLLs) that present the services of an underlying storage mechanism to handle the storage and retrieval of messages for the users of client applications like desktop applications. A MAPI store provider assigns a unique ID string when an item is created in its store. The `EntryId` property of a `MailItem` object in the Outlook Object model corresponds to this unique ID string. The `EntryID` does not change unless an item is moved into another store, for example, from Inbox to a Microsoft Exchange Server public folder or from one Personal Folders (.pst) file to another .pst file. Hence we can use the `entryId` value to uniquely identify an email. The add-in uses the `entryID` of the email and the full file path location of the local file as the source and destination parameters to send the request to create a link to the database. As our add-in is in C#, it uses the C# database driver to interact with MongoDB.

### 6.5.2 Associating Calendar appointments to local files

Our Outlook add-in should also support the functionality to allow the users to link their local files to Calendar appointments. Though Outlook allows to create a hyperlink from an appointment to a local file, the link is unidirectional i.e. the link is only from the calendar appointment to the local file but not in the other direction. We want the users to be able to create bidirectional links between the calendar items and the local files.

Unlike the email attachment case, the Outlook event model provides way to notify when a calendar event is saved. So by registering a listener to this event, we can create a link in the database whenever an appointment is saved. The code script below shows how to register a delegate in the add-in code to receive notifications when an appointment is saved.

```csharp
Outlook.AppointmentItem apptItem = Inspector.CurrentItem as Outlook.AppointmentItem;
if (apptItem != null)
{
    apptItem.AfterWrite +=
    new Microsoft.Office.Interop.Outlook.ItemEvents_10_AfterWriteEventHandler(←
        ItemEvents_10_AfterWriteEventHandler);
}
```

This allows us to create a link without any intervention from the user. As the link creation process takes place implicitly, we do not need to provide any user interface. The appointment
body is to be parsed first to find any hyperlinks to local files. If any hyperlinks to local files are found, the add-in calls the database to create bidirectional links between each of these files and the appointment. Similar to emails, an appointment identity is given by the EntryID of the Appointment Object in the Outlook Object model.

6.6 Shell Extensions

Shell Extensions are in-process COM objects which extend the functionalities of Windows Explorer. The shell extension handlers can be implemented for a variety of actions that the Shell can perform. When the Windows Shell performs any action, it queries the corresponding extension handlers and thereby gives them a chance to extend the functionality. In our system, we use two types of shell extensions - icon overlay and context menu. We use the icon overlay shell extension to notify the user when associations exist for a resource. The context menu is used to allow the users to easily access the associated resources.

6.6.1 Icon Overlay

Icon overlays are tiny images that are placed at the lower left corner of the icon of Shell objects such as folders, files and shortcuts. They can be laid over an object’s icon to provide some extra information. In our system, we use the icon overlays as the notification to the users to inform that associations exist. Whenever a file has links to other resources, we will use the Icon overlay shell extension handler to mark that file with an overlay icon.

As Shell extension handlers\(^2\) are in-process Component Object Model (COM) objects, they are implemented as DLLs. Microsoft’s Active Template Library (ATL)\(^3\) is a set of template based C++ classes which are intended to simplify the programming of COM objects. Before the shell can use a shell extension handler, it must be registered in the Windows registry. The Windows Registry is a hierarchical database that stores configuration settings and options on Windows.\(^4\) To register, we need to create a GUID for the handler. A Globally Unique Identifier (GUID) is used by the Windows registry to identify COM objects and DLLs. Whenever the Shell does an action that involves an extension handler, it checks the appropriate registry key. The key under which an extension handler is registered thus controls when it will be called. We create a sub key for our Icon Overlay handler under the ShellIconOverlayIdentifiers key (HKEY_LOCAL_MACHINE/Software/Microsoft/Windows/CurrentVersion/Explorer/ShellIconOverlayIdentifiers) in the registry.

The IShellIconOverlayIdentifier interface exposes methods that handle all communication between icon overlay handlers and the Windows Shell. When the Windows Shell starts up, it initializes all the icon overlay handlers that are registered by calling the IShellIconOverlayIdentifier methods. One of the methods is GetOverlayInfo, which the Shell calls to request the location of the handler’s icon overlay. The icon overlay handler returns the name of the file containing the overlay image. The Shell then adds the icon overlay to the system image list. From then, the system only uses this image which is cached in its system image

\(^3\)http://msdn.microsoft.com/en-us/library/3ax346b7(v=vs.71).aspx
\(^4\)http://en.wikipedia.org/wiki/Windows_Registry
Before the Shell draws an overlay on an icon, it passes the shell object’s name to each icon overlay handler’s `IsMemberOf` method. The icon overlay is displayed based on the return value of this method. So in this method, we query the database with the object’s name to check if the object has any links. To interact with MongoDB from the C++ code in our shell extension handler, we use the C++ driver which is provided in the MongoDB server source repository. The code below shows how the shell extension handler interacts with MongoDB to find if a file has any links. If we get a response from the database that the object has links, we return `S_OK`. The Shell then calls the handler’s `GetOverlayInfo` method to determine which icon to display and displays the icon. Figure 6.6 shows an explorer window with overlay icons on resources that have associations.

```cpp
bool hasDBLinks(string source)
{
    try {
        mongo::DBClientConnection c;
        c.connect("localhost");
        auto_ptr<mongo::DBClientCursor> cursor = c.query("LinksDB.Links", QUERY("Source" << source));
        if (cursor->itcount() > 0) return true;
    }
    catch(mongo::DBException& e) {
        cout << "caught DBException " << e.toString() << endl;
    }
    return false;
}
```

### 6.6.2 Context Menu

In the Windows file explorer, a context menu is displayed when you right-click a Shell object like a file or folder. A context menu handler is a shell extension handler that adds commands
to an existing context menu. In our system, when the user right-clicks on a Shell object that has links to other resources, the corresponding access shortcuts should be displayed in the context menu. For example, as shown in the figure 6.7, the document has links to appointment(s) and email(s), and hence displays the options to open them in the context menu. Depending on the type of associations a file has, we display the appropriate options in the context menu as shown in the table below.

<table>
<thead>
<tr>
<th>Context menu entry</th>
<th>Association</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open associated documents</td>
<td>Object associated to other documents</td>
</tr>
<tr>
<td>Open associated web page</td>
<td>Object downloaded from a web page</td>
</tr>
<tr>
<td>Open associated email</td>
<td>Object saved from an email</td>
</tr>
<tr>
<td>Open associated appointments</td>
<td>Object linked calendar appointments</td>
</tr>
</tbody>
</table>

Just like any other shell extension handler implementation we need to export the IUnknown interface. Also in particular to a context menu handler, we need to export two other interfaces: IShellExtInit and IContextMenu. The Shell initializes the handler through the IShellExtInit interface. The Shell calls IContextMenu::QueryContextMenu and passes a HMENU handle. This handle can be used to add new items to the context menu. In this method, we query the database to find out what types of other resources, the current object is associated with. Depending on the response from the database, we add those options to context menu. We use the InsertMenu method to add menu items and return a HRESULT value with the severity set to SUCCESS. The code snippet in the next page shows how we add the “Open Associated documents” option to the context menu, if an object has associated documents. When a context menu option is clicked, the Shell calls IContextMenu::InvokeCommand. The handler can then execute the appropriate command to launch the corresponding resource.
```cpp
IFACEMETHODIMP CFileContextMenuExt::QueryContextMenu(HMENU hMenu, UINT indexMenu, UINT idCmdFirst, UINT idCmdLast, UINT uFlags)
{
    if (CMF_DEFAULTONLY & uFlags)
    {
        return MAKE_HRESULT(SEVERITY_SUCCESS, 0, USHORT(0));
    }

    // SelectedLinks is a class with boolean variables indicating the types of associations
    SelectedLinks *sLinks = getAssociatedLinks();
    if (sLinks->bDocs == true)
    {
        InsertMenu(hMenu, indexMenu, MF_STRING | MF_BYPOSITION, idCmdFirst + IDM_OPENDOCUMENT, _T("&Open Associated Documents"));
    }
    if (sLinks->bEmails == true)
    {
        InsertMenu(hMenu, indexMenu, MF_STRING | MF_BYPOSITION, idCmdFirst + IDM_OPENEMAIL, _T("&Open Associated Emails"));
    }
    if (sLinks->bWebPages == true)
    {
        InsertMenu(hMenu, indexMenu, MF_STRING | MF_BYPOSITION, idCmdFirst + IDM_OPENWEBPAGE, _T("&Open Associated Webpages"));
    }
    if (sLinks->bAppointments == true)
    {
        InsertMenu(hMenu, indexMenu, MF_STRING | MF_BYPOSITION, idCmdFirst + IDM_OPENAPPOINTMENT, _T("&Open Associated Appointments"));
    }
    return MAKE_HRESULT(SEVERITY_SUCCESS, 0, USHORT(IDM_OPENAPPOINTMENT + 1));
}
```

Figure 6.7: Explorer window with context menu options to open associated resources
To goal of the evaluation phase is to assess the usability of our system. For this, we conducted a user study through a controlled laboratory experiment where we asked users to do some routine tasks involving personal information resources. We compared the user experience in doing these tasks with and without our tools.

7.1 Method

The user study was carried out as a controlled laboratory experiment with 15 participants. The participants were provided with a computer running Windows 7 Operating system. We chose Windows as the operating system as some of our tools currently run only on Windows. The participants had to perform sets of tasks in two phases. In one phase, the users performed a set of tasks without using associations. In the other phase, the users performed the same set of tasks using associations with our system. The order of the phases was randomized and counterbalanced to avoid any learning effects. In the phase involving our system, if any of our tools supported multiple ways of using the links, we asked the users to try all of them. Again, we randomized the order of trying these options to avoid any ordering effects.

The set of tasks included some routine tasks the users would do on a computer like browsing web pages, downloading files, opening and editing documents, viewing emails, saving attachments, creating calendar events etc. The whole set of tasks are part of a broader scenario which encompasses some of tasks which users do routinely on their workstation. The users had to assume that they were working in a project. As an introduction to the environment, we explained the users about the set of documents and web pages the users will use in the user study. The documents included a project report, project presentation and an excel sheet which had a collection of useful references. All these documents were stored at different locations. In choosing the locations to store these documents, we used the feedback of the users in the user interview phase as discussed in chapter 4. For example, many of the users mentioned that they store their presentation documents in a chronological
folder hierarchy. The sample web pages included a project webpage which we created for this user study and two other web pages, which the users had to assume that they generally refer while working on this project. Firefox and Microsoft Outlook were chosen as the web browser and email client respectively as we have developed our tools for these applications. The latest versions of these applications (Firefox 11 and Microsoft Outlook 2010) were provided to the users in the experiment. Also Outlook was configured with an email account which already existed and contained a lot of email to emulate a real scenario.

The time needed to perform some of the tasks was recorded as a measure regarding efficiency. Before using our system, the users were trained on using our tools. Below we list the set of tasks the users performed in our user study in each of the phases. The complete sequence of tasks can be found in the appendix B.

1. **Opening documents**
   The users had to open all the related documents which were stored at different locations. In the phase with our tools, the users had the option to use the document associations to open the documents.

2. **Opening email**
   The users had to save an attachment from an email. After working on the attachment, they had to resend the file back to the sender of the email. In the phase with our tools, the users had the option to use the associations between the email and the saved attachment to retrieve the email from the saved attachment.

3. **Opening web pages from an associated file**
   In one of the tasks, the user had to download a file from a web page. In another task, the user had to get some information from the same web page from which he downloaded the file. In the phase with our tools, the users had the option to use the associations between the web page and the downloaded file to reaccess the web page.

4. **Opening multiple related webpages**
   The users had to launch some related web pages together. In the phase with our tools, the users had the option to use the associations between web pages to launch the web pages together.

5. **Opening calendar appointment**
   In one of the tasks, the user had to create a calendar appointment and add a hyperlink to a local file from appointment. Later in another task, the user had to access the calendar appointment while he was working on the local file. In the phase with our tools, the users had the option to use the bidirectional link between the local file and the appointment to access the appointment.

At the end of each phase the participants had to fill out a post-task questionnaire in which they rated statements about the perceived user experience, based on 5-point Likert scales\(^1\). In particular the users had to rate how easy and efficient it was to do the tasks. Furthermore, in the post-task questionnaire of the phase where associations were used, the users had to rate the usefulness of different kinds of associations on a scale of 1 to 10 (1 - strongly disagree, 10

---

\(^1\)http://en.wikipedia.org/wiki/Likert_scale
- strongly agree). In the end, the participants also had to fill out a questionnaire that asked for demographic information as well as background information. The complete questionnaires can be found in Appendix B.

### 7.2 Results

Of the 15 participants, 12 were male and 3 were female at an average age of 28 years. 10 participants were from the Computer Science department. All the participants used Microsoft Windows as the operating system.

In Figure 7.1, we present the average times needed by the users for completing different tasks. We found a significant reduction in the time taken to complete the tasks using associations in general. In particular the time required to access the documents required less than half of the time that was required when done without links. This was also the case with reaccessing the emails from saved attachments. We observed that participants had to spend more time in searching for the email than opening the email client. For the web pages, the improvement is less because in our particular case the users did not interact with a large number of web pages and also we allowed them to access the previously visited web pages through history. But in general, we expect when the user interacts with a lot of web pages and has different related groups, using associations can show reduction in time. Also for the case of reaccessing the outlook appointment, we saw a significant reduction in time using links.

![Figure 7.1: Average times(in seconds) needed for completing the tasks](image)

Based on the post-task questionnaires filled out by the participants after both the phases,
Figure 7.2 shows the perceived user experience in terms of easiness and efficiency of tasks. The perceived easiness and efficiency of opening documents and reaccessing emails improved significantly using links. Also for the web pages, the users could launch web pages more efficiently with links. There was no significant improvement in easiness or efficiency of reaccessing appointments which might be due to the fact that the appointment created by the users in our task was due the very next week and was readily viewable by the user when Outlook was launched. However this difference could increase if there had been significant number of other appointments in the calendar and also if the user had to spend more time in the calendar view to locate the appointment.

Also in the post-task questionnaire after the evaluation phase with links, we asked the users to rate the usefulness of the various linking options we have provided in our systems. As shown in the figure 7.3, we got very good ratings for the document associations, web page associations and associations between emails and saved attachments. In case of web page associations, we got a better rating for the feature of associating web pages to local files than the case of associations across web pages. This may be because the users are already familiar with other options like bookmarks and tagging for accessing related web pages. From the ratings, we can see that users felt that linking appointments is a less strongly useful feature. This could be explained by the fact that very few users knew about the option of linking appointments to local files before.

Specifically for the document linking case, the users were asked to rate the two different options we have provided for accessing documents - context menu and system tray. The users rated the context menu better but the difference was not too large. This might
be due to the fact that the user interface provided by the context menu is more integrated into the desktop environment. Though the users have not evaluated the automatic linking of documents in their tasks, we explained them about the feature to know their feedback. All of them mentioned they would like to have both manual linking and automatic linking features to be included with the document linker tool.
In this thesis, we presented an approach to enable end-users to create associations between selected personal resources and use these associations for the retrieval of these resources, in a lightweight way and part of their workflow. We have designed and implemented a system which realized these goals. Our system is integrated with the desktop environment of the users and enables bi-directional link creation. In addition, our system supports an unobtrusive notification mechanism to notify the end-users about any existing associations.

We found out that many of the existing PIM tools implicitly create links based on analyzing content or metadata of resources or by analyzing user behavior. In contrast, we detected a lack of lightweight support for enabling users to explicitly associate resources. Also these systems were standalone and require the users to explicitly invoke them either to create associations or to get benefit out of them. In addition, there was little support for any notification mechanism to make the user aware of any existing associations. The aim of this thesis was to fill these gaps.

We started with a scenario analysis where we analyzed how end-users use and manage their personal resources. In this regard, we conducted user interviews. Based on the findings from the interviews, we designed more specific use cases which identify the possible places in the user’s workflow where links can be created. In addition, these use cases identified the benefit the end user gets by going through this whole additional process of creation of links. We did a feasibility study considering various choices of technologies to implement these use cases. After different considerations, we decided to use Shell extensions, Firefox extensions, Outlook add-ins and a C# application as the software components to implement the use cases. Finally, we conducted a user study through a controlled lab experiment with the software prototypes we have implemented. The results indicated that users could complete tasks involving personal resources easily and effectively using our system.
8.1 Contributions

In this section, we present a summary of the contributions that were developed as a part of this Master’s thesis.

- **Semi-automatic links**  
  We presented the concept of semi-automatic links, where personal resources can be associated implicitly or manually by users depending on the scenario. The primary design goal of our system is to allow the users to associate personal information resources in a lightweight way and as part of user’s workflow. All the links created by our system are bi-directional in nature. Because our links are bi-directional, we allow navigation using the links in both the directions which is a very useful feature.

- **Identification of use cases**  
  Based on scenario analysis, we designed a set of use cases which identify the possible places in the user’s workflow where link creation functionality can be placed. In addition, the use cases also identify key factors that provide an incentive for users to actually spend the additional effort of creating a link.

- **Desktop integrated linking environment**  
  We implemented software prototypes which are well integrated with the desktop environment of the users. The software prototypes included a desktop application, browser extensions, Office suite add-ins and Shell extensions. All our software prototypes run individually but persist the links in a unified storage space provided by the database. The software prototypes with their integration components to the database and the database together define our system architecture.

- **Integration with applications**  
  Our tools are integrated with some of the common applications which are used by end-users while working with resources. Browser extensions extend the functionality of the web browser to enable end-users to create and use associations from the web browser. Similarly, our Outlook add-in facilitated users to create associations between resources from the email and calendar applications.

- **User experience evaluation**  
  We evaluated our system using a user study through a controlled lab experiment. This evaluation revealed that users found our system easy to use and that semi-automatic links between resources improved the productivity of the users with tasks involving personal resources.

8.2 Future Work

- **Integration with a Personal Information Management (PIM) tool**  
  All the prototypes implemented in the system can be integrated with a more sophisticated Personal Information Management tool (PIM). By doing so, we get the benefits the tool provides and also provide the user a more integrated environment where personal resource management is extended with the advantages of links. For example, we can
provide advanced link management facilities. Also the users can be allowed to have better visualization of the links.

- **Extended user evaluation**  
  In our user study, we evaluated the user experience in a controlled lab experiment. But a better analysis of the perceived user experience can be obtained if the users are allowed to use the tools over a period of time. By doing so, we provide the users a more natural environment. Also different categories of users should be evaluated to analyze how well links fit for each of them. In a future study, we could evaluate our system’s effectiveness over extended use.

- **Support for other operating systems**  
  All our prototypes are developed for the Microsoft Windows Operating system. But to provide a generic interface, the system should be extended to other popular operating systems like Linux, Mac OS. It is easy to integrate the Firefox Extensions as they do not rely on the underlying operating system. But the shell extensions in our system are entirely a concept of Microsoft Windows. So, the alternatives provided by other operating systems, which are similar to shell extensions should be studied. Also a detailed analysis should be made on how well the concept of links integrates with their environment.

- **Support for other types of desktop software**  
  We have made some application choices for implementing the prototypes. For example, Mozilla Firefox is chosen as the browser for implementing extensions and Microsoft Outlook is chosen as the email and calendar application. As discussed in the feasibility study section, we have based our decisions for the selection of browser based on the application popularity and the ease of development. But to make our system generic, our software prototypes should be extended to support all other popular application choices.

- **Multiple computer environment**  
  Nowadays, the personal resources of the end-users are not confined to a single computer. We have our resources spread across multiple computers. The system should be extended to support link creation across multiple computers. The system architecture will be much complex in that case, as the software prototypes are spread over devices and communication takes place through the network.

- **Extending to mobile devices**  
  Increasingly, more number of mobile devices like smart phones and tablets are being adapted by users for their computing requirements. The idea of creating and using links should be extended to these devices too. For this a complete user study has to be done, where the typical user workflow in each of these devices should be studied to analyze how using and creating links would benefit the end users.

- **Sophisticated link visualization model**  
  Our system provided a simple view of links to edit and manage them. However more sophisticated models to visualize links can be developed which allow users to have a better overview of the existing associations and also allow extending them.
8.2. FUTURE WORK
The following are the questions we asked the users in the user interviews of the scenario analysis phase.

1. How do you manage your documents in your computer? How do you store and retrieve them?

2. How do you manage your media resources (images, videos) in your computer? How do you store and access them?

3. How often do you have to open related documents simultaneously? If so, how do you try to do the same?

4. How do you categorize your files in the folder hierarchy?

5. How do you organize downloaded content in your computer? Where do you store them and how do you access them?

6. How often do you have to use your local files with the browser (ex. to upload/view)? How do you do it?

7. Do you maintain bookmarks? How do you organize and use them?

8. Do you open multiple related tabs in your browser (ex: for news, sports, reviews etc.)? How do you do it?

9. How do you organize your email? Do you categorize your email?

10. Do you download/save your email attachments? How do you access/reuse the saved attachments?

11. Do you use any calendar application? For what purposes, do you use it for?
In this user study, we try to mimic some of the common computing tasks in your routine. You need to imagine you are working on a project with a team. You will have to perform some routine tasks like visiting web pages, editing documents, checking email etc. as listed in the task sheet. You will perform these tasks in two phases. At the end of each phase, you will need to fill a questionnaire.

As the first step, you will need to familiarize with the files, applications and documents you will be working with in this study.

• **Documents**
  The table below lists the documents you will be using in this user study. You can open them now and have a look at their content. The table below lists their locations:

<table>
<thead>
<tr>
<th>File name</th>
<th>Location</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>SemiAutomaticLinks-Report.docx</td>
<td>C:\Spring\Projects\Linker-Project</td>
<td>Project Report</td>
</tr>
<tr>
<td>SemiAutomaticLinks-Presentation.pptx</td>
<td>F:\Presentations\2012\March\</td>
<td>Project Presentation</td>
</tr>
<tr>
<td>UsefulReferences.xls</td>
<td>M:\Personal\Academic\</td>
<td>Your personal collection of useful references</td>
</tr>
</tbody>
</table>

• **Web pages**
  The table below lists the web pages you will be working with. The browser is already open with these webpages in separate tabs. Please have a look at the content of each of them.

• **Applications**
  You will use "Firefox" as the Browser and "Outlook" as the email client and calendar
<table>
<thead>
<tr>
<th>Website</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="https://sites.google.com/site/semilinks/">https://sites.google.com/site/semilinks/</a></td>
<td>Project page</td>
</tr>
<tr>
<td><a href="http://becktench.com/hci/">http://becktench.com/hci/</a></td>
<td>Commonly used</td>
</tr>
<tr>
<td><a href="http://www.mendeley.com/computer-and-information-science/human-computer-interaction/">http://www.mendeley.com/computer-and-information-science/human-computer-interaction/</a></td>
<td>Commonly used</td>
</tr>
</tbody>
</table>

application. Both of these applications can be accessed using shortcuts from desktop. Please close all the documents, Outlook and Firefox now.
B.1 Tasks done without using links

1. Launch Firefox and open your project website (http://sites.google.com/site/semilinks).

   As you can see from the updates, you need to attend a meeting on 30th March. Also there is a file named "Meeting Agenda", which you would like to download to your computer.

2. Download the file "Meeting-Agenda.pdf" to any location in your computer you wish.

3. Close the Browser window.

   Next, you need to create an appointment in your Outlook Calendar for the meeting scheduled. Also you would like to attach the "Meeting Agenda" document to the appointment.

4. Open Outlook and select "Calendar".

5. Create new appointment on Friday 30th March, 2012 at 9:00 AM with subject "Monthly meeting".

6. Hyperlink the "Meeting Agenda" document to this appointment. To create a hyperlink using Outlook, please follow the following steps:

   • Right click on the body of the appointment to show the context menu.
   • Select the "Hyperlink" option.
   • Navigate to the file you would like to link and select it.
   • Press "Save and Close" button.

7. Select the "Mail" tab in Outlook.

8. Open the email from "Semi-Automatic Links-Team" which you received on Monday, March 19 and read the content.

   • You need to update some content in your project documents.
   • You also need to update the status in the attached document.
   • Send the attached document back to the sender.
   • Update your appointment to discuss about your references.

9. Firstly, save the attachment ("StatusReport.docx") from the email to a desired location. Now you want to update the Project report and presentation documents with the new reference in the email. Also as you find the reference interesting, you would like to add it to your personal collection of useful references.

10. Copy the reference in the email to your clipboard. (Select the text and Ctrl-C).

12. Open the following documents:
   - SemiAutomaticLinks-Report.docx
   - SemiAutomaticLinks-Report.docx
   - UsefulReferences.xls

13. As specified in the email, update each of these documents with the content mentioned in the email.

14. Close all the documents now.
    Now you need to update your status and send the document back to the sender.

15. Open the document you saved from the email. ("StatusReport.docx")

16. Update the status in the last row to "Updated" and close the file.

17. Attach the document to the email you received from your team.

18. Send the email back to the sender.

    Now you need to update your Outlook "Appointment" with your reference webpage links:

20. Launch Firefox and open each of your most used web pages.

21. Open outlook and access the appointment on Friday 30th March at 9:00 AM.

22. Copy the addresses of web pages in to the appointment.

23. Save and close the appointment.
B.2 Tasks done using links

Firstly, you need to create links between your frequently used documents.

1. Launch the "Document Linker" by clicking the system tray icon.
2. Press the "Select Local Files" button and select the documents to be linked using tree view.
3. Press the "Link" button.
5. Launch the browser.
6. Right click on the "Project" web page and select the option "Create Links" in the context menu.
7. Press the "OK" button.
8. Close the browser.
9. Launch the browser and open the project website.
10. Download the file "Meeting-Agenda.pdf" to any location in your computer you wish.
11. Close the Browser window.
12. Open Outlook and select "Calendar".
13. Create new appointment on Friday 30th March, 2012 at 9:00 AM with subject "Monthly meeting".
14. Hyperlink the previously downloaded document to this appointment. To create a hyperlink using Outlook, please follow the following steps:
   - Right click to show the context menu.
   - Select the "Hyperlink" option
   - Navigate to the file you would like to link and select it.

Check your email from your project team mates:

15. Open Outlook and select "Email".
16. Open the email from "Semi-Automatic-Links-Team" on 19 March.
17. Save the attachment from the email to your desired location using the "Save with Links" option instead of the default "Save as" option.
18. Copy the reference mentioned in the email to your clipboard (Select text ’CTRL-C)

Now to open the relevant documents ("SemiAutomaticLinksReport.docx", "SemiAutomaticLinks-Presentation.pptx","UsefulReferences.xls"), we will use the linking functionality. To open them, you can do it in two ways.

20. First we will do it using "Context Menu Option":
   
   Context menu option:
   
   • Navigate in the file explorer to open one of the documents. (For example: C:\Spring \Projects \Linker \Project \)
   • Now, right click to open the context menu and select "Open associated documents"
   • Now close all the documents and the explorer windows.

21. Now we will open the documents using the "System tray balloon" option: System tray balloon:
   
   • Open one of the documents to be opened. (You can open C:\Spring\Projects \LinkerProject \SemiAutomaticLinksReport.docx)
   • When the system tray balloon pops up, select it to open all the linked documents
   • Update each of the documents with the content mentioned in the email.
   • Close all documents and explorer windows

22. Open the document you saved from the email. ("StatusReport.docx")

23. Update the status in the last row to "Updated" and close the file.
    
    To send the email back to the sender, we will use the link to access the email.

24. Right click the document in the file explorer and select "Open associated emails". This will open the associated email.

25. Now you can attach the document with the email and reply back.

1. It was easy to open all the documents.

2. I was able to efficiently open all related documents.

3. It was easy to complete email related tasks.

4. I was able to efficiently reopen the email from the saved attachment.

5. It was easy to complete calendar related tasks.

6. I was able to efficiently open the associated appointment from the explorer.

7. It was easy to open the webpage from the downloaded file.

8. I was able to efficiently open the webpage from the downloaded file.

9. It was easy to open multiple webpages in the browser.

10. I was able to efficiently open multiple webpages in the browser.
QUESTIONNAIRE (Tasks done using Links)

1. It was easy to open all the documents. ☐ ☐ ☐ ☐ ☐
2. I was able to efficiently open all related documents. ☐ ☐ ☐ ☐ ☐
3. It was easy to create links between documents. ☐ ☐ ☐ ☐ ☐
4. It was easy to complete email related tasks. ☐ ☐ ☐ ☐ ☐
5. I was able to efficiently open the email from the saved attachment. ☐ ☐ ☐ ☐ ☐
6. It was easy to complete calendar related tasks. ☐ ☐ ☐ ☐ ☐
7. I was able to efficiently open the associated appointment from the hyperlinked file. ☐ ☐ ☐ ☐ ☐
8. It was easy to open the webpage from the downloaded file. ☐ ☐ ☐ ☐ ☐
9. I was able to efficiently open the webpage from the downloaded file. ☐ ☐ ☐ ☐ ☐
10. It was easy to open multiple webpages in the browser. ☐ ☐ ☐ ☐ ☐
11. I was able to efficiently open multiple webpages in the Browser. ☐ ☐ ☐ ☐ ☐
12. It was easy to create links between webpages. ☐ ☐ ☐ ☐ ☐
13. In the user study, you have accessed the linked documents using two options. How useful do you find each of these options? Please rate on a scale of 1 to 10 (1-Strongly disagree 10-Strongly agree)

- Context Menu
- System Tray

14. The document linking feature provides you with two options to create links between documents. Which of the two options do you think you would use more regularly?

☐ Automatic Linking  ☐ Manual Linking  ☐ Both

15. Using outlook linking, you are able to link your emails and appointments to local files. Please rate on a scale of 1-10 (1-Strongly disagree 10-Strongly agree), how useful you find each of these features:

- Linking emails
- Linking appointments

16. Using web page linking, you are able to link your webpages to webpages as well as web pages to local files in your PC. Please rate on a scale of 1-10 (1-Strongly disagree 10-Strongly agree), how useful you find each of these features

- Linking web pages to local files
- Linking across webpages
# QUESTIONNAIRE - GENERAL

## Demographics

1. Please select your age group
   - [ ] 20 & under
   - [ ] 20 – 25
   - [ ] 25 – 30
   - [ ] 30 – 35
   - [ ] 35 & above

2. You are
   - [ ] Male
   - [ ] Female

## Questionnaire

1. I use the following operating system for my daily computing tasks
   - [ ] Windows
   - [ ] Mac OS
   - [ ] Linux
   - [ ] Mobile (Tablet, PDA)

2. I think that my productivity increases by creating links between personal resources (documents, email, webpages)
   - [ ] Strongly Agree
   - [ ] Agree
   - [ ] Neutral
   - [ ] Disagree
   - [ ] Strongly Disagree

3. I am satisfied with the ease of completing the tasks in this user study.
   - [ ] Strongly Agree
   - [ ] Agree
   - [ ] Neutral
   - [ ] Disagree
   - [ ] Strongly Disagree

4. Please write any comments you have in general
List of Figures

3.1 Contextual overview of our approach ........................................ 12
3.2 Roadmap ............................................................................. 15
4.1 Various options for storing and retrieving resources plotted against the
number of participants using them .................................................. 18
5.1 Range of technologies ............................................................. 26
6.1 System Architecture ............................................................... 32
6.2 Document Linker - Application Window ..................................... 33
6.3 Document Linker - Notifications .............................................. 34
6.4 Web page and Download associations ........................................ 37
6.5 Context menu option to save an attachment with link to the email .... 39
6.6 Explorer window with overlay icons on resources that have associations 42
6.7 Explorer window with context menu options to open associated resources 44
7.1 Average times (in seconds) needed for completing the tasks ........... 47
7.2 Perceived user experience ....................................................... 48
7.3 Perceived usefulness of various linking options .......................... 49
I would like to express my deeply-felt thanks to my thesis advisor Matthias Geel for his advice, guidance and constant support over the whole course of thesis. I gratefully thank Professor Norrie for giving me a chance to work on this topic. Special thanks to Michael for helping me in designing the user study and providing valuable ideas. Also I would like to thank all the Globis group members for their patient participation in the interviews and user studies.

This Master's thesis is dedicated to my parents for their encouragement throughout my graduate studies. I am always indebted to them for their continued support and unwavering faith in me.


