DESIGN OF A SERVICE-ORIENTED DASHBOARD

by

GAYATHRI SUNDAR

MURAT M. TANIK, COMMITTEE CHAIR
DAVID G. GREEN
GARY J. GRIMES
JOHN L. HARTMAN IV

A THESIS

Submitted to the graduate faculty of The University of Alabama at Birmingham,
in partial fulfillment of the requirements for the degree of
Master of Science

BIRMINGHAM, ALABAMA

2007

DESIGN OF A SERVICE-ORIENTED DASHBOARD

GAYATHRI SUNDAR

ELECTRICAL ENGINEERING

ABSTRACT

Service-Oriented Enterprises are currently focused on offering enhanced user experiences to customers, employees, and enterprise partners who use the enterprise's software applications. A better user experience is delivered through the presentation of services and business processes to the user in a composite manner, and by facilitating a better user-application interaction for the user. Our experience with the bioinformatics domain helped us to see that although existing models of enterprise portals offer an integrated environment to present applications and enterprise resources, they are not sufficient to address certain complex user-application interactions. In this thesis, we present a design for a Service-Oriented Dashboard to address this deficiency.

The service-oriented dashboard leverages Service-Oriented Architecture to present an integrated environment to access and interact with the diverse and isolated software tools, applications, and resources of an enterprise. We address the design of the dashboard at three levels: the presentation layer, the task composition layer, and the services layer. We introduce in the thesis the notion of describing the user's job function as a composition of tasks. Users, through this approach, will be able to compose applications according to the user's tasks. This thesis also presents the design of the service-oriented dashboard, focusing on two notions – the tools-as-services and service-enabled tools approach. The tools-as-services approach involves making the tools in an application available to the user as services. The service-enabled tools approach allows applica-

tions to consume services and thereby provides the functionality of the consumed service inside that application.

We also present two case studies from the bioinformatics domain. We use the case studies to demonstrate the need for the dashboard, and also to derive the requirements for the design of the dashboard. We also illustrate how the dashboard can enhance the workflow in the case studies described in the thesis.

DEDICATION

To my beloved father and mother for teaching me some of the most precious lessons and values in life, and for providing me with great opportunities to learn and grow.

ACKNOWLEDGMENTS

My foremost gratitude goes to my advisor and committee chair, Prof. Murat M. Tanik. His assistance and guidance over the last three years has been invaluable to my thesis. I also extend my appreciation to my committee member, Dr. John Hartman, and the University of Alabama at Birmingham (UAB), UAB Department of Genetics. I thank him for his support during my employment with the Department of Genetics. Dr. Hartman's direction and expertise were also vital during the case study conducted under his supervision. Heartfelt thanks also go out to Dr. Donna Arnett, Dr. Laura Vaughan, and the Department of Epidemiology. Dr. Arnett and Dr. Vaughan have been very cooperative in guiding us during the course of our case study in the Department of Epidemiology. I also thank Dr. David Allison for his astute insights on some of our research approaches while we became accustomed to the bioinformatics domain.

My sincere appreciation is extended to my committee members, Professors Gary Grimes and David Green from the Department of Electrical and Computer Engineering and Professor Tanju from the School of Business, for their guidance and suggestions on improving the thesis. Additionally, I thank Professor Leon Jololian for his advice regarding certain essential parts of the thesis. The Division of Continuing Medical Education (CME) has been extremely supportive during my thesis development. I extend my heartfelt thanks to Ms. Katie Crenshaw and CME for their valuable assistance and patience during my research. I also thank Najaf Shah and Dr. Wei Li for their help during the case study in the Hartman lab. I thank Dr. Thomas Jannett of the Department of Electrical and

Computer Engineering for his kind assistance during the preparation of the document. I also thank Ms. Maria Whitmire and Ms. Sandra Muhammad of the Department of Electrical and Computer Engineering for their continuous support and timely assistance. I am also grateful to Jesse Chambers for his editing services. I thank my family for their unrelenting support and motivation, especially my uncle Dr. Mohan Srinivasan, for being a great source of inspiration and for constantly driving me to reach higher goals. Finally, I thank my dear husband Rajani Sadasivam for all his valuable and never-ending help, encouragement, and patience.

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

LIST OF FIGURES

Figure *Page*

LIST OF ABBREVIATIONS

Ajax	Asynchronous JavaScript and XML
API	Application Programming Interface
AUGC	Area Under the Growth Curve
BIRN	Biomedical Informatics Research Network
BPEL	Business Process Execution Language
BPEL4WS	Business Process Execution Language for Web Services
BPM	Business Process Management
CBD	Component-Based Design
cM	Centimorgans
CORBA	Common Object Request Broker Architecture
CPP	Cell Proliferation Phenotypes
CSS	Cascading Style Sheets
DCOM	Distributed Component Object Model
DHTML	Dynamic HTML
DNA	Deoxyribonucleic Acid
DOM	Document Object Model
EAI	Enterprise Application Integration
EIS	Executive Information Systems
ESB	Enterprise Service Bus
FPI	Final Population Intensity

FTP	File Transfer Protocol
GO	Gene Ontology
GUI	Graphical User Interface
HCI	Human Computer Interaction
HTCP	High Throughput Cellular Phenotyping
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
IDE	Integrated Development Environment
iLIMS	Intelligent Lab Information Management System
J2EE	Java 2 Enterprise Edition
JCP	Java Community Process
JSR-168	Java Specification Requests (Portlet Specification)
KPI	Key Performance Indicator
LIMS	Lab Information Management System
MSR	Maximum Specific (growth) Rate
NCBI	National Center for Biotechnology Information
OASIS	Organization for the Advancement of Structured Information Standards
ODBC	Open Database Connectivity
OMIM	Online Mendelian Inheritance in Man
OOD	Object-Oriented Development
RDF	Resource Description Framework
REST	Representational State Transfer

RIA	Rich Internet Applications
ROI	Return on Investment
RPC	Remote Procedure Call
RSS	Really Simple Syndication
SET	Service-Enabled Tools
SGD	Saccharomyces Genome Database
SMTP	Simple Mail Transfer Protocol
SOA	Service-Oriented Architecture
SOD	Service-Oriented Dashboard
SOE	Service-Oriented Enterprise
SOSC	Service-Oriented Smart Clients
TAS	Tools-as-Services
UDDI	Universal Description Discovery and Integration
URI	Uniform Resource Identifier
W3C	World Wide Web Consortium
WfMC	Workflow Management Coalition
WPF	Windows Presentation Foundation
WPF/E	Windows Presentation Foundation/Everywhere
WSCI	Web Service Choreography Interface
WSDL	Web Services Description Language
WSFL	Web Services Flow Language
WSIA	Web Services for Interactive Applications
WSRP	Web Services for Remote Portlets

XAML	Extensible Application Markup Language
XHTML	Extensible HyperText Markup Language
XLANG	Web Services for Business Process Design
XML	Extensible Markup Language
XSLT	Extensible Stylesheet Language Transformations

I. INTRODUCTION

A. Background

Enterprise Application Integration (EAI) is the process of creating an integrated infrastructure for linking disparate systems, applications, and data sources within the corporate enterprise and across enterprises [1]. EAI is defined as the unrestricted sharing of data and business processes among connected applications or data sources in an enterprise [2]. Enterprises rely heavily on their information systems and applications for their success [3]. Therefore, enterprises have been using EAI as an integration strategy for over a decade [4], [5]. The EAI approach can be decided by many factors, including the size of the enterprise, the complexity of the software used, integration or project complexity, and budget. EAI solutions can exist at many levels. Typically, there are four types of middleware solutions to EAI: user-interface integration, data integration, process integration, and method or function integration [1].

Currently, Web services have been gaining prominence as an EAI approach because of their use of open standards. Web services based on the Service-Oriented Architecture (SOA) provide a technical foundation for business processes and application integration methodologies.

B. Motivation

The advent of SOA and Business Process Management (BPM) technologies have changed the way enterprise software applications are designed [6]-[8]. SOA promotes the use and reuse of functional components as services. SOA also enables composition of business processes with specific tasks at any time to form larger business applications. Hence, businesses can build and use applications focused on interoperability and location transparency. The maturing SOA standards and tools are enabling enterprises to retool their systems to establish as service-oriented enterprises (SOE) [9]. As enterprises continue to adopt SOA, their focus has shifted on two important concepts:

- Presentation of services and business processes to the user in a composite manner.
- Presentation of a better experience for the user.

1) Current Trends in Presenting Information in a Composite Manner

Portals have traditionally aggregated content from different Web sites and enterprise resources and presenting it in a unified interface. Currently, many businesses have been using portals for their SOA applications for the purpose of offering an integrated environment for the employees to access the distributed resources of the enterprise. Hence, a portal is an integral part of the enterprise integration process. The software enterprise has projected enterprise portals as the user-interface model for businesses for quite a few years [10]. A portal could be considered to be a Web site that has multiple applications and integrates data from different sources [11]. Portals offer a variety of fea-

tures and functionalities as portlets [12]. Portlets are pluggable components and self-contained applications in the user-interface of a Web portal.

More recently, portal developers have been focusing on integrating portals with business processes. Portals that manage processes are often called process portals. These process portals provide users with sophisticated functionalities. These functions allow users to accomplish specific tasks embedded in a streamlined process [13].

2) Current Trends in Improving User Experience in Enterprise Applications

Enterprises are increasingly using corporate portals to present information to their users. Corporate portals are designed and customized to serve a variety of users, such as employees, customers, or even users from a partner enterprise. Such a gamut of target audiences demands enhanced connectivity. The Internet has been invaluable in connecting people and resources globally; therefore, the presentation medium of choice for enterprise portals is the Web. The Web interface for the portals is most commonly a browser. Web browsers lack the richness and responsiveness of the desktop applications the portals users are accustomed to. However, recent advances in the field of presentation of rich applications over the Internet have made the difference between desktop and browser-delivered applications inconspicuous. Currently, supporting the collaborative and interactive nature of Web 2.0, improved user experiences are being provided using Rich Internet Applications (RIAs) in commercial Web applications and portals [14],[15]. RIAs are made predominantly using asynchronous JavaScript and XML (eXtensible Markup Language) or Ajax, Flash, and occasionally some JavaScript and Dynamic HyperText Markup Language (DHTML) [14]-[16]. Ajax is used to build interactive Web

applications. Developers of applications are already investigating Web-based delivery of existing desktop applications, such as spreadsheet applications, word processing programs, or feature-rich email applications. Some companies have succeeded in developing such applications. Examples of RIA usage for spreadsheet or word processing can be seen in such applications as Google Docs. Flash is also used extensively to develop highly user-friendly, visually pleasing, and responsive applications.

C. Thesis Approach

Portals make the transition for the user from one data source or an application to another unnoticeable. RIA technologies are used to present the unified content in an interactive manner and thereby, provide a better experience for the user. Combined, portals and RIA provide a powerful approach for presenting information in an SOA application.

However, after analyzing the user requirements on highly data-intensive and interactive biological SOA applications in our case studies, we contend that portals and RIA are not sufficient to meet all user needs. This deficiency of portals and RIA to support user interactions in certain scenarios is highlighted in the problem scenario described in Section D. In this thesis, we present the design of a Service-Oriented Dashboard (SOD). Traditionally, enterprise dashboards are tools that permit the users of an enterprise to monitor and analyze certain key metrics in selected processes as visually captivating charts or graphs [17], [18]. Our definition of a dashboard is different. The Merriam Webster dictionary defines dashboard as:

> *"A panel extending across the interior of a vehicle (as an automobile) below the windshield and usually containing instruments (as a speedometer) and controls [19]."*

Extending the definition, the SOD allows users to perform their work on their dashboard and also presents the necessary metrics for monitoring related processes. The SOD could be a standalone system or a part of an enterprise portal. Although this thesis describes the Web-based version of the SOD, the dashboard system can be implemented as a desktop application with modifications [20]. In order to better explain the SOD, we present an example problem scenario. We present the design of the SOD in Chapter III. In Chapter IV, we use two case studies to depict how the SOD can enhance the workflow described in the case studies.

D. Problem Scenario

Consider an example of a portal enhanced with RIA, such as the Yahoo finance portal [21]. The Yahoo finance portal offers a wide variety of services, such as market summary, personalized portfolios, recent quotes, top movers, and basic charts to show the day's standings. There is also updated news and many other Web-based tools that supplement the primary content on the site.

In order to consider a business process, we examine the hypothetical situation of an employee in a firm that deals with stock quotes. Assume that one of the specific tasks of the employee is to obtain some data from the Yahoo finance portal; perform a few transformations on a spreadsheet, such as sorting and drawing customized charts with the data; and, finally, generate personalized reports. He uses Microsoft Excel to do the trans-

formations such as sorting, and uses the charting feature of Microsoft Excel to draw the charts. He also uses Microsoft Word to generate the reports. As depicted in Fig. 1, it is apparent that this process is highly cumbersome, requiring the user to switch back and forth between four to five different applications that are both Web- and desktop-based. This particular case highlights three issues that the user confronts:

- The user switches between one Web application to another, for example, from the market summary to recent quotes.
- The user switches between one desktop application to another, for example, from Microsoft Excel to Microsoft Word.
- The user switches from Web-based application to another desktop application or vice versa, for example, from the market summary to Microsoft Excel.

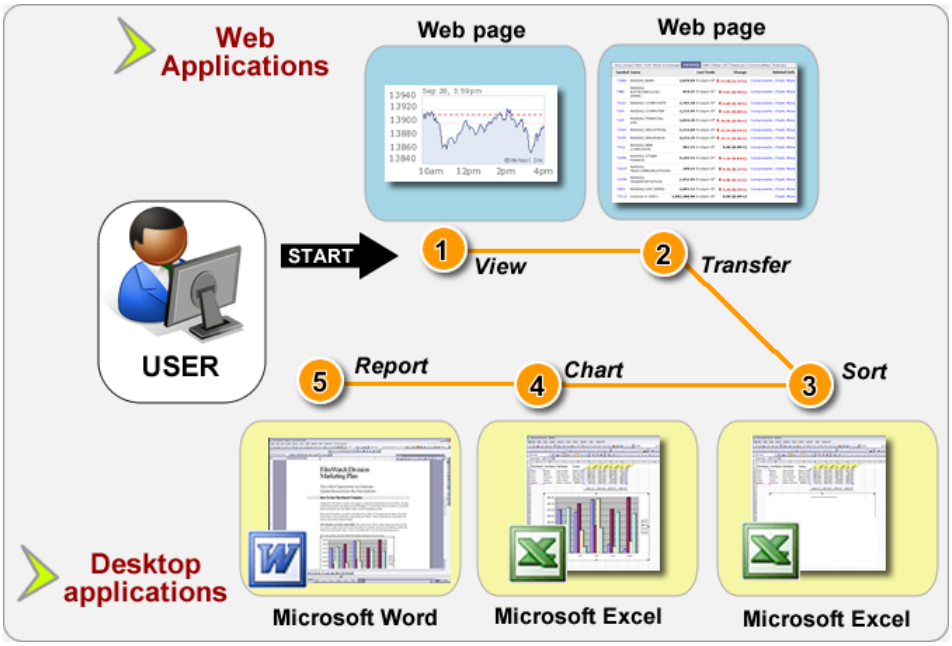


Fig. 1. Sample workflow described in the problem scenario, where the user uses several Web-based and desktop applications.

During all of the above transitions, the tasks performed by the user are error-prone. Hence, extra caution is required to transfer data accurately between applications. This process results in a poor user experience and poor performance [22]. There are ways around the copying and pasting routine. For example, Web services can be used to transfer quotes for the required symbols to a Microsoft Excel spreadsheet [23]. However, even this method would not solve the problem of switching from one application to another. To avoid the transition from the Web to the desktop and back, one solution would be if we could have Web-based tools replace desktop tools. There are already many ventures of this type. For example, Google has its own Web-based document applications providing solutions to word processing and spreadsheets. The Web-based document applications essentially try to provide the same functionality of desktop-based tools. However, this is not an optimal solution in most cases. In-house, custom re-development of desktop applications to use on the Web is a very cumbersome, and in some cases, a cost prohibitive task.

E. Thesis Outline

Fig. 2 shows the organization of the thesis chapters.

The SOD is primarily based on the SOA. In Chapter II of the thesis we introduce SOA concepts. We also describe the Enterprise Service Bus (ESB) as an approach to deploy SOA applications. We also present a review of RIA concepts and technologies that are currently in use. In addition, we present a brief discussion on portals and mashups.

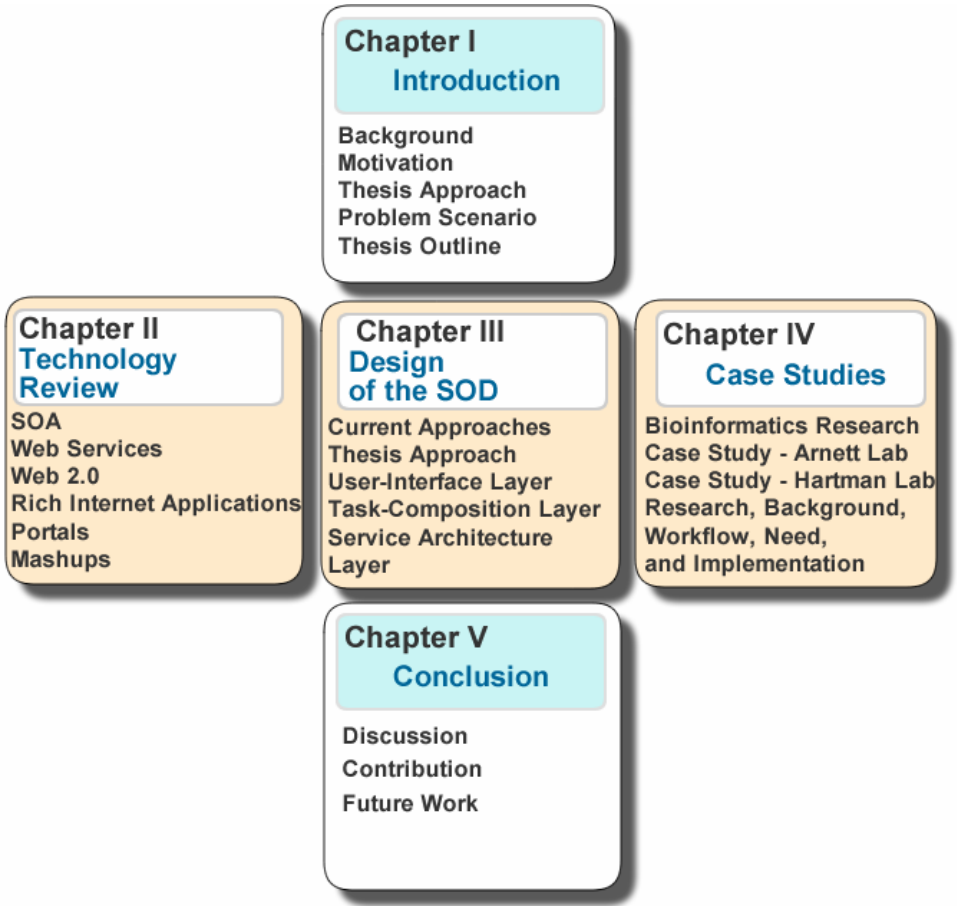


Fig. 2. Overview of chapters and topics addressed in this thesis.

Chapter III commences with a review of the literature concerning some current dashboard approaches. We then present our design of the dashboard at three levels: the services layer, task composition layer, and user-interface layer. We also introduce the tools-as-services (TAS) and service-enabled tools (SET) notions as design approaches.

In Chapter IV we present two cases of applications of the SOD in the bioinformatics domain. A bioinformatics application is a good comparison to an enterprise application. We illustrate the similarities between research laboratories and business enterprises by providing a comparison in this chapter. We use the two case studies to explain how

the dashboard can be an effective solution to the needs of the laboratories with sophisticated workflows.

In Chapter V, we provide the concluding remarks of the thesis and present an overview of projected future work.

II. TECHNOLOGY REVIEW

A. Introduction

This chapter provides an overview of the technologies that are incorporated in the design and development of the SOD. We also include a background study on existing and current approaches towards application development. Beginning with the evolution of a computer application, this chapter also covers such concepts as RIA, portals, and SOA. We also include a discussion on Web 2.0. As a growing number of enterprises begin to adopt SOA and Web 2.0 technologies, the major priorities for enterprises are the presentation of reusable components, services and business processes in a composite manner, and the presentation of the composed content as a rich, personalized user experience. As evident by the trends in the Web 2.0 era, the most important aspect of Web 2.0 is the user [14], [24]. Regardless of the user's role as a customer, an employee, or a partner enterprise, application developers are aiming to develop highly user-centric, personalized applications. This chapter examines a few of the current approaches towards the development of user-centric applications.

B. Software Architectures

Software architectures can be defined as the structural organization and integration of the system components and their interaction and relationship patterns [25]. In-

creasing number of enterprises use software applications to enhance their processes and increase their return on investment (ROI). Application software has evolved from being a single-user, stand-alone application to large multi-user systems over networks. Applications represent the “business-aware” functionality of a system [26]. As the needs of an enterprise change, applications should evolve with them [27]. Consequently, software architectures have to be flexible.

In this thesis, we categorize the architecture of software applications from these two viewpoints.

- Architecture based on the application logic: In this category, the classification is based on the logical architecture of the application. Classification is based on the location of the business logic, presentation logic, and the data access logic.
- Architecture based on software abstractions: This type of classification is based on the software paradigm such as objects, components, or services.

Sections 1 and 2, together offer a brief discussion on the evolution of SOA.

1) Architecture Based on the Application Logic

Initially software architectures were comparatively simple, when they were confined to stand-alone systems or mainframes. However, with the onset of networking and distributed computing, the existing software architecture had to be revamped to accommodate modularity and accessibility. As applications became more complex, they were separated into parts to be processed on different computers. This type of modularity led to a tiered architecture [28]. The client-server model is the simplest tiered architecture

[29]. In the client-server architecture, the computer that requests a service is referred to as the client and the computer that processes the requested service is referred to as the server. Based on the software configuration, a computer can function as a client or a server. For instance, in order to process a service request, the user first requests a service through the client. The client sends the request to the server, where it gets processed. The client receives the processed service from the server and displays it back to the user. Fig. 3(a) shows a client-server model in which each of the four clients interacts with the server to process some information requested by the users. The two-tier client-server approach developed into the 3-tier and finally the n-tier models to accommodate scalability and enhanced accessibility and modularity. The n-tier architecture is also referred to as the multi-tier architecture.

In a three-tier architecture, the three tiers or layers are for presentation logic, business logic, and data-access logic. The presentation tier is responsible for the interaction between the user and the application. This includes controlling the behavior of the user-interface, receiving instructions from the user, and presenting the data back to the user. The business logic or the core functionality of the application is present in the business logic layer. The data-access layer or the persistence layer is for the storage of the data and for interaction with legacy systems. Fig. 3(b) shows a three-tier architecture.

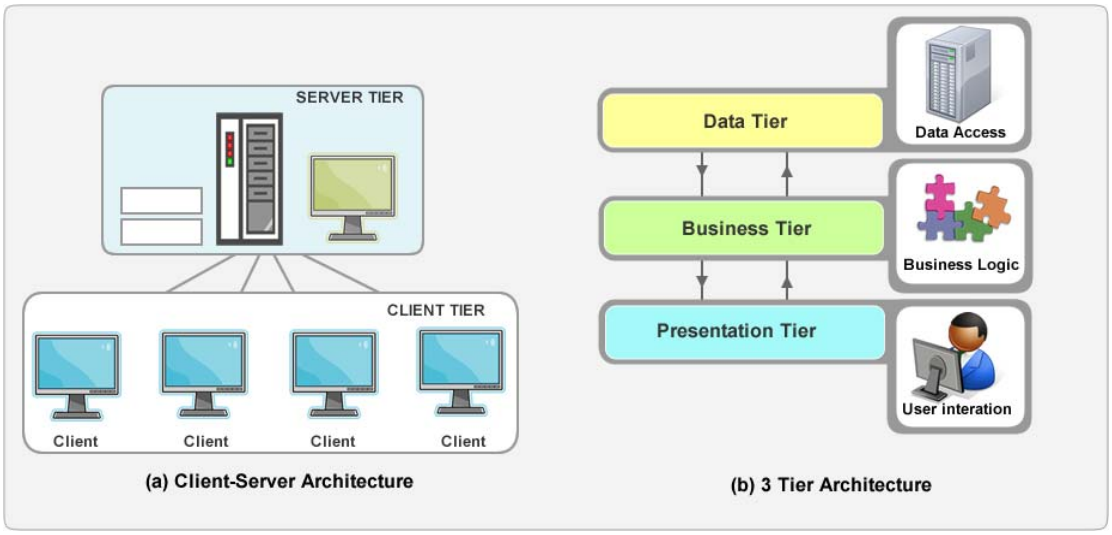


Fig. 3. (a) Client server architecture and (b) three-tier architecture.

2) Architecture Based on the Software Abstraction

As software architectures were advancing into a tiered model, software programming was becoming increasingly complex. Once again, modularity is considered the key to writing reusable code and making programming easier. The use of subroutines and functions offered modularity and better reusability. However, there were still problems in the scalability and maintenance of software applications, and a need for a higher level of abstraction. This need resulted in the use of object-oriented application development [30], [31]. Object-oriented development (OOD) involves the modular disintegration of a software system into objects [29]. Objects interact with other objects to produce the functional characteristics of the system. However, developers wanted to reuse not just the code, but also the functions. So another layer of abstraction was developed. This model of abstraction was component-based software development. By aiming to develop systems with a modular design, OOD established the divide and conquer notion. This notion

was captured in the concept of component based development (CBD) [32]. CBD amalgamates several characteristics of development models, such as OOD, and distributed computing technologies, such as distributed component object model (DCOM) and common object request broker architecture (CORBA) [33].

CBD offered a higher level of abstraction in terms of components, and its use rose with the popularity of commercial off-the-shelf (COTS) products or components [34]. In CBD, an application is developed through component selection, evaluation, and assembly [35]. To summarize, both OOD and CBD approaches paved the way for software reuse and greater modularity. However, there were still certain complexity issues that neither development model could solve. These complexity issues could be addressed by the SOA model [36].

The Internet was becoming a popular medium for modern application development. As users of the Internet increased and businesses started making use of it, use of application servers, stateless servers, and distributed architectures grew exponentially, increasing the complexity of the applications. As Booch stated,

> *"The trend toward more complex software is driven by greater connectivity and user expectations [37]."*

Enterprises started to shift away from the paradigm of building custom-made, in-house software applications to the new prospect of reuse and composition of functionality. Enterprises were building applications that required improved connectivity to distributed systems. This increased need for system functionality, improved interoperability across platforms and devices, and extensibility demanded a seamless integration solution. "Chunks" of applications from disparate and scattered sources had to be composed to

form a complete system [36]. One of the ways to conceive such a system was through the use of interoperating services. With the introduction of XML [38] – a markup language and a standard to improve interoperability, and SOAP, the procedure of passing XML messages through hypertext transfer protocol (HTTP), the concept of Web services was born [39], [40]. The use of Web services with SOA was a great way to equip modern day applications with the growing demands of seamless integration.

3) Web Services

In order to better define SOA, it is important to first define a service. A service can be defined as a software entity or resource that performs a unit of work. A service is provided by the service provider, and the service is used by a consumer. The World Wide Web Consortium (W3C) defines a service as:

> *"An abstract resource that represents the capability of performing tasks that form a coherent functionality from the point of view of provider entities and requester entities [41]."*

Web services have interfaces that are published by providers and discoverable by requesters. This implies that a Web service can be dynamically discovered and invoked by the consumers. Web services are independent, as they preserve their own states, and are platform-independent, as the communication through their interfaces is based on Internet protocols, such as HTTP, file transfer protocol (FTP), and simple mail transfer protocol (SMTP). A software agent is a software program that represents an organization or a person. The software agent can be defined as either the provider or the consumer of a service based on its role as a provider or a consumer. These software agents communicate

through messages that usually conform to the XML schema. The W3C identifies two types of Web services: the representational state transfer-compliant (REST) Web services that control the XML representations of Web resources using a uniform set of stateless operations, and other arbitrary Web services where the service exposes a random set of operations [40]. Both classes of Web services identified by the W3C use uniform resource identifiers (URI) to discover resources and use Web protocols, such as HTTP and SOAP, and XML data formats for messaging. In some cases Web services, in general are classified as SOAP- and REST-based services. SOAP is a protocol for exchanging XML-based messages across agents, using the communication protocols HTTP/HTTPS (hypertext transfer protocol secure). In SOAP Web services, messages in XML are carried through SOAP and the services are written in Web services description language (WSDL) [42]. SOAP Web services could be either remote procedure call (RPC) Web services or document-centric Web services (part of the SOA) [40]. Authors who wish to publish their Web service do so in an XML-based registry called the universal description, discovery and integration (UDDI), where the services can be discovered [39].

4) Service-Oriented Architecture

SOA is an architectural style that aims at loose coupling among interacting software agents [43]. SOA guides all aspects of creating and using business services throughout their lifecycle [44]. An SOA allows for the definition of different applications and resources in an enterprise as services. This enables these applications to exchange data and participate in business processes regardless of the application's programming language or the software platform it is designed for.

5) Enterprise Service Bus

The enterprise service bus (ESB) is a streamlined, middleware infrastructure technology that combines XML and Web services support to provide SOA-required features for the integration needs of an enterprise [45]. The ESB architecture is composed of three layers:

- The existing enterprise infrastructure.
- The ESB layer, which includes adapters to expose existing systems and provide transport connectivity.
- The business services layer created from the existing information technology systems [45]. Via ESB, SOA systems can perform application integration once and then automatically reuse the integration services whenever necessary [46].

To build an SOA, a highly distributable communications and integration backbone is required. This functionality is provided by the ESB. An ESB uses Web services standards to support a wide variety of communications patterns over multiple transport protocols and deliver value-added capabilities for SOA applications [47].

C. Software Applications

A software application can be defined as a program or a software system that performs a specific enterprise function. In a typical software application, the user interacts with the application through the user-interface. The user-interface, usually a graphical user-interface (GUI) or Web browser, controls the flow of input data and commands delivered to the processor. It also controls the commands to direct the user. The logic or the

core functionality of the application processes the commands and stores the data. Based on the architecture of the application, the logic layer, data layer, and presentation layer could be either on the client or on one or more server machines. According to this concept of the location of the application and its components, software applications can be classified as thin or thick clients.

1) Thin Clients

Thin clients are applications whose functional logic resides on the network server [48]. Web applications that are browser-based, including Web-based email, are great examples of thin clients. With the arrival of the Internet, enterprises started relying heavily on the Internet for marketing needs. Internet applications were a great step towards increased connectivity and global reach. Browsers were used by consumers to access applications that resided completely on the enterprise's server. However, users were not completely satisfied by the experience offered by the browser [48]. The Web-based user-interfaces that replaced the feature-rich GUIs that the users were accustomed to using were simple and dull. They lacked the myriad controls and flexibility that were offered by the desktop GUIs. This was because the Web-based interfaces were in simple hypertext markup language (HTML). There are several advantages to using thin clients. They require no client-side versioning/updates or installation, are platform independent, and have great accessibility. On the downside, they have limited accessibility to the client's resources, require Internet connectivity, require browsers, and must be designed to be used on the lowest possible browser version. The most important shortcoming of thin clients is their lack of richness [48].

2) Thick Clients

Thick clients are applications whose logic is processed on the client and which occasionally connect to the server for data storage and retrieval [48]. Thick clients are more secure and responsive than thin clients because most of their logic resides on the client, thereby cutting down on the trips made to the server. Thick clients are programs or applications that are installed on the client machine by executing from a file that is either downloaded from the Internet or loaded onto the machine using an external storage device. Rich screen functionality, responsiveness, and superior interaction are the prominent features of thick applications. On the contrary, updating and versioning of the system is complicated, and the system cannot be accessed from anywhere except on the clients that have the application installed [48].

3) Smart Clients

Smart clients are applications that capture the advantages of the thin and thick client systems [48]. They are delivered on the Web and hence do not require any installation. They have simplified updating systems. Rich interfaces, improved responsiveness, high accessibility, and platform independency are some of their impressive qualities. They offer additional capabilities, such as the ability to automate recurring processes, centralize common information resources to avoid expensive replication, and decrease production time through enhanced deployment techniques. By using both local and remote resources, smart clients produce finer user experiences than the more traditional form of a Web application.

Smart clients based on SOA are termed Service-Oriented Smart Clients (SOSC) [48]. An SOSC uniquely extends the capability of the smart client by incorporating intelligence and agility into its SOA [48].

4) Web 2.0

Originally designed for the presentation of information to users through static Web pages, the Web has grown to offer a plethora of interactive and dynamic applications from a wide range of domains. These domain ranges from e-commerce, entertainment, communications, collaborations, and personal publishing to the simplest form of information sharing. The delivery of applications on the Web has evolved along with the Web itself. Modern applications developed for the Web usually are measured in terms of the following criteria [49]: reliability, usability, security, availability, scalability, maintainability, and time to market.

Recent trends such as RIAs, portals, and other Web 2.0 concepts are revolutionizing the development of applications. In the following sections, we illustrate these Web 2.0 concepts and review the use of some of the current technologies used to build RIAs.

The term Web 2.0 was first coined by Tim O'Reilly and Dale Dougherty at the O'Reilly Media Conference in 2004 [14]. It refers to a new approach of presenting content on the Web. As stated by Cooper, this new approach is characterized by interactive content that allows sites to combine features in new ways [50]. Traditionally, the Web acted as a static data repository. Web 2.0, on the contrary, is a platform where a novice can contribute easily without the need to master special skills. Simplicity is a significant feature of the Web 2.0 generation, enabling not just the experts but also the technically

non-savvy users to build applications, publish content, and navigate the Web with ease. Web 2.0 is very user-centric, enabling personal publishing in which users can create and own their data [24]. According to O'Reilly, the key concept of the Web 2.0 era is that users add value [14]. As indicated by Gibson, the Web 2.0 generation is heralded as an arena for participation and dynamic interaction [51].

5) The Differences Between Web 2.0 and Web 1.0

There has been a great deal of debate as to the soundness of the term Web 2.0 and, occasionally, what special events marked the transition of the Web to Web 2.0 [52]. In their column in an online magazine, MacManus and Porter list a few key concepts that clearly distinguish the two [53]. Web 1.0 was the version of the Web when the World Wide Web was simply a collection of documents on specific Web sites. When users needed information, they would go to these specific sites. However, Web 2.0 has changed that. Now users employ really simple syndication (RSS) aggregators, portals, application programming interfaces (APIs), and Web services to collect information from various sites and to present the collected information back to the users on their choice of interface.

Web 2.0 interfaces deliver rich user experiences not just at the presentation level but also at the business-logic level. This was not true during the Web 1.0 age, when it was difficult to render applications on the Web with the quality and functionality of desktop applications. Quoting O'Reilly,

> *"Web 2.0 could yield many new Web applications, both truly original applications and rich Web reimplementations of PC applications [14]."*

6) Web 2.0 Technologies

Web 2.0 technologies are characterized by notable transformations at the presentation, data access and architecture levels. As categorized by Andy Budd, a few of the Web 2.0 characteristics are represented by Table 1 [54].

TABLE 1
CHARACTERISTICS OF WEB 2.0

Data	Architecture of participation	User experience
Open data formats, data access across devices, user-created and -owned data	Service instead of a product, reuse, online user participation, collective intelligence	Ease of use, social networks, rich user-interface, aggregated, customized information

Based on these characteristics, some of the most prominent concepts of the Web 2.0 era are SOA, RIAs, Portals, open source software, and collaboration.

An August 2007 press release from Gartner, a leading information technology research and advisory company, declares that Web 2.0 innovations are one of the key technologies that could make a huge impact on business organizations [55]. According to their Hype Cycle report and Priority Matrix for emerging technologies, 2007, the Web 2.0 phenomenon and the Web 2.0 workplace technologies have been declared as being transformational. Important members of the Web 2.0 suite, including mashups and RSS enterprises, have also been featured on the matrix as adoptable in 2 to 5 years.

Although predictions have been made that the Web 2.0 trend is about to transform Web development, the pace is hindered by concerns about security, inaccessibility threats from unmoderated user content, use of third-party content, ethical concerns, and breaches

of copyright [50], [51], [56]. The concerns about Web 2.0 have been highlighted by a press release that followed the Gartner report for emerging technologies. Gartner Fellow Joseph Feiman adds that in spite of all the vulnerabilities present in Web 2.0 adoption, a few basic preventive and best practices can shield an enterprise against these threats [55].

This chapter discusses some of the Web 2.0 technologies such as RIAs, mashups, and portals.

D. Rich Internet Applications

RIAs are interactive and highly responsive Web applications that are built to provide great experiences for users. RIAs are browser-based Web applications that can be classified as Web 2.0 applications. Desktop applications have always ranked higher in terms of user experiences. The better standing of desktop applications is due to faster performance, better interactivity, and steady connectivity. RIAs focus on delivering these three traits in Web-based applications. RIAs are much easier to use and more functional than traditional Web applications [16]. The most widely used technologies to build RIAs are Ajax (Asynchronous JavaScript and XML) or Adobe Flash. Table 2 presents a list of approaches broken down by technology platform.

TABLE 2
OVERVIEW OF RIA TECHNOLOGIES PRESENTED IN THE THESIS

Ajax	Adobe	JAVA	Microsoft
XML	Flash	JAVA FX	.NET Framework
XHTML	Flex	NexaWeb	WPF
CSS	MXML		Silverlight
XSLT	ActionScript		XAML
JavaScript			ASP .NET AJAX
DOM			
DHTML			
XMLHTTPRequest			
Ajax frameworks			

1) Ajax-Based Rich Internet Applications

The word Ajax stands for Asynchronous JavaScript and XML, and was coined by Jesse James Garrett, the President and the Founder of Adaptive Path [57]. According to Garrett, Ajax is not a technology but rather a collection of existing technologies or concepts such as Extensible HyperText Markup Language (XHTML), Cascading Style Sheets (CSS), Document Object Model (DOM), Extensible Markup Language (XML), Extensible Stylesheet Language Transformation (XSLT), XMLHttpRequest (XHR), and JavaScript. Most of these technologies were standardized in the late 1990s [16].

a) Ajax technologies

HTML, short for HyperText Markup Language, is used for the markup of Web pages. A markup language has information on the content of the Web page and also some information regarding the presentation of that content. This “labeling” of content uses tags placed throughout the content as appropriate. The majority of today’s Web sites still

use basic HTML or XHTML to deliver content along with CSS and JavaScript to enhance their presentation, dynamic nature, and some functionality. According to a recent study in August, 2007, 59% of sites use JavaScript and 54% use CSS. This is up from 37% and 13% respectively in 2001 [58].

XML is a markup language that was designed to handle and describe various types of data [38]. It is used in structuring data handled by an application. XHTML is defined by W3C as a family of current and future document types that are an extension of HTML. XHTML documents would imply that they are also are XML-based [59].

CSS, also a W3C standard, is a style sheet language that is used to describe the presentation of Web pages in HTML, XHTML, or even XML. The Web pages or documents contain special markups to denote different styles. The style information is placed in a separate CSS document. A Web page or a document could refer to more than one style sheet at a time. CSS is used in the separation of presentation from content, and developers can better control the way browsers display the Web pages using CSS.

XSLT is a language used to transform XML documents to a format that can be read by humans or in to another XML document. Extensible style sheet language or XSL was developed by the W3C to provide an XML-based style sheet language. XSLT can be used to convert an XML document into another XML document, or into other browser-recognizable formats, such as HTML. During the transformation process, XSLT uses XPath to define parts of the source document that should match one or more predefined templates [60]. XPath is a major element in the W3C's XSLT standard and is a language for locating information in XML documents. XPath is used to navigate through elements and attributes in an XML document [61].

JavaScript is a scripting or a programming language that is used to tie together objects and resources on clients and servers. It is usually used on the client-side for executing a minor program on logic and validation. JavaScript was released by Netscape and Sun in 1995 [62]. JavaScript is based on Java and is an object-oriented language. Ajax uses asynchronous JavaScript. A page using asynchronous JavaScript makes asynchronous calls to the server to fetch XML documents, assisting in the update of partial rather than whole pages. So the content of the application is loaded inconspicuously while the user interacts with the program.

Initially, dynamic page behavior in Web pages was achieved through the use of Dynamic HTML (DHTML). DHTML is a combination of HTML, CSS, and JavaScript along with DOM, which is another W3C standard, and is defined as a platform-independent, language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of documents. In simpler terms, DOM allows developers to treat the elements of a Web page as program objects. This facilitates the Web designer in the design and manipulation of elements on the page. DHTML and DOM can help dynamically change the look of Web pages [16].

DHTML was not standardized by the W3C, so there are many incompatibilities across browsers, browser versions, and even operating systems. Ajax is very similar to DHTML, as it uses a similar combination of technologies to generate the presentation and responsiveness of an application [16]. The major difference is that Ajax uses server calls using the JavaScript-based XMLHTTPRequest [63].

The XMLHttpRequest forms the spine of an Ajax application. It is an object and used to be called remote scripting. Traditional Web applications use HTTP to trigger a

request to the server, and the server does some processing and returns an HTML page to the client. The inherent holdup with this model was the delay caused every time the user interacted: the page places a request, waits for the server to respond while the server takes time to process the request, and then responds with the next page. Ajax follows a different model that reduces the holdup.

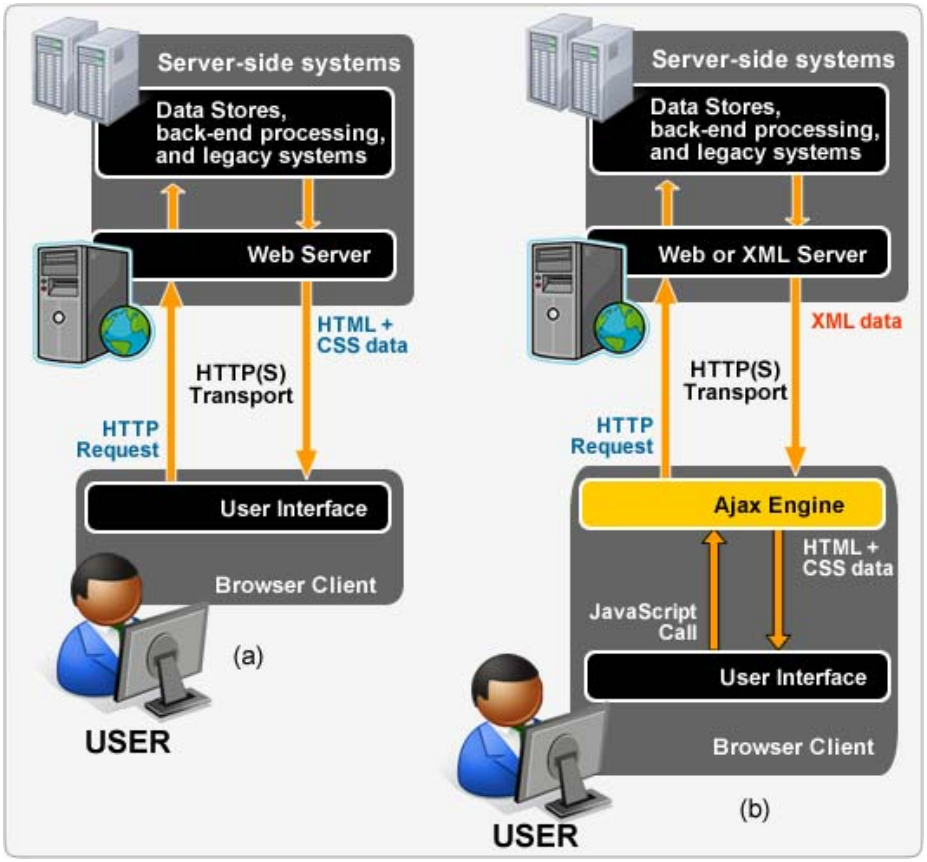
b) Ajax approach

When a page that has an Ajax application is loaded for the first time, a JavaScript-based Ajax engine is also loaded. This engine runs on the browser and acts as an intermediate between the server and the application on the client. Fig. 4 compares the traditional approach using HTTP and the Ajax approach. Fig. 4(a) depicts the traditional HTTP request from the client and response from the server. As opposed to that, Fig. 4(b) shows the application communicating with the Ajax engine using JavaScript calls. This engine in turn communicates with the server through HTTP. In this case, if a part of the page needs to be updated, the whole page need not be refreshed; hence, the server response times are notably reduced, resulting in a better user experience [64].

c) Example of an Ajax application

Let us consider an application that shows the user the first result from a Google search for a particular key word. Also present in the page are navigation menus and other applications and images. Traditionally the search application would need to reload the pages, thus causing a significant delay. In an Ajax application, the result appears in the page without the complete page having to reload. A more common example of Ajax at

work would be Google Map [65]. When the user uses the mouse to drag the map inside the interface, the map shows the latest position without a delay, as the whole page does not reload. The area that displays the map is the only part of the page that reloads. This results in greater user experiences because of better responsiveness.



Note: Adapted from "Ajax: A New Approach to Web Applications" by J. J. Garrett 2005, *http://www.adaptivepath.com/ideas/essays/archives/000385.php*. Copyright 2005 by Adaptive Path. Adapted with permission.

Fig. 4. (a) Classic Web application model (b) Ajax Web application model.

d) Frameworks and toolkits

There are many Ajax frameworks available to aid developers in easily developing Ajax applications. The frameworks are collections of snippets of code that can be used to create a complete application. There are many frameworks on the Internet available for free under the GNU General Public License. Some of the frameworks are specific to languages used for server-side scripting. Some are independent of the language used. The popularity usually depends on a few factors such as browser compatibility, language independence, and back button and forward button issues. Roodt, in his thesis, describes some of these factors in detail [66]. In addition, a good comparison can be found in reference [67].

Some of these frameworks are available with an integrated development environment (IDE). These are easy-to-use interfaces that one can use to create Ajax applications. Some examples for such frameworks are TIBCO General Interface, Google Web Toolkit, Open Rico, Prototype, Yahoo User-interface Library, and Dojo.

2) Flash-Based RIAs

Several popular RIAs are developed for delivery on Flash players that are loaded on Web browsers. The following sections describe RIA development technologies aimed at RIA development for delivering applications on Flash players.

a) Adobe Flash

Formerly known as Macromedia Flash, it is an environment that facilitates the creation of RIAs. Flash has a two-part infrastructure: the programming component for developers, and the player component for the end-users. The Flash player is a runtime component that resides on the client's machine and requires a download. The player is mandatory to play content developed in Flash. The programming model and runtime work exactly the same for developers, regardless of the device platform [68].

b) Adobe Flex

Adobe Flex, formerly called Macromedia Flex is a presentation server for the development of RIAs. The key difference between a traditional Web application and a Flex application is that Flex applications conduct most of the processing on the client using the Flash component on the Web browser. However, a traditional HTML application uses a server. Tasks such as sorting, validation of fields in the application, and certain visual effects can save considerable time if executed on the client, as the time for the transfer of data between the client and the server is avoided. The Flex presentation server is composed of the Flex application framework and the Flex Runtime services. The Flex presentation server currently deploys only on a Java application server, although a .NET release is expected soon. RIAs developed using Flex use MXML and ActionScript, along with the class library [69]. Flex applications are written using both ActionScript and MXML.

c) MXML

MXML stood formerly for Macromedia XML and currently, since Adobe took over Macromedia, it has been called Flex XML or, as some suggest, Multimedia XML. MXML is a markup language used to describe the elements and define the content and functionality of user-interfaces.

d) ActionScript

Action Script 2.0 is an object-oriented programming language used in Flex development and is based on the JavaScript Standard (ECMAScript). As in any programming language, ActionScript is used to define the client-side logic of Flex applications.

e) OpenLaszlo

OpenLaszlo is an open-source platform for the development of RIAs. OpenLaszlo applications are written in LZX, a standards-driven language that combines XML and JavaScript. The final compiled format of the applications is supported by Adobe Flash and can be delivered on the Flash player present in most end users' browsers. In addition, the latest version of OpenLaszlo, version 4, the applications can be directly compiled to DHTML ruling out the need for a Flash player.

3) Java-Based Rich Internet Applications

There are several Java frameworks and development environments for RIA development. IDEs such as NetBeans and Eclipse are the preferred development environ-

ments. The following sections explain about JavaFX and NexaWeb – a java-based thin client development tool.

a) JavaFX

Java FX is an extension to the JAVA platform and is used in the creation of RIAs. An important capability of RIAs produced by Java FX is disconnected use, as opposed to some rich applications that can be accessed only on the Internet, hence requiring connectivity. Java FX uses the runtime libraries that are installed on the local client with the help of JAVA for the desktop integration of over-the-wire applications. This is done using the JAVA SE files. It draws on JAVA FX Script for content authoring of the Web applications. According to reference [70], Java FX Script is oriented around the interface rather than the Web page.

b) NexaWeb

NexaWeb is a software development kit used to develop SOA-based RIAs for enterprises towards building applications for the Enterprise Web 2.0. It provides a toolkit to built Java 2 enterprise edition (J2EE) and XML applications that are browser-based.

4) Microsoft RIA

a) .Net framework

The .NET framework is a development framework introduced by Microsoft in 2002 as .NET Framework 1.0. Currently .NET framework 3.0 consists of four major components:

- Windows Presentation foundation.
- Windows Communication foundation.
- Windows workflow foundation.
- Windows CardsSpace.

The .NET framework can be considered a component of Windows and offers numerous pre-coded solutions for certain programming needs. These needs cover a wide range from user-interface and Web applications to database connectivity.

b) Windows Presentation Foundation

The Windows Presentation Foundation (WPF) is a graphical subsystem of the .NET Framework 3.0. Formerly codenamed Avalon, the WPF assists in the clear separation of the user-interface and the business logic. A WPF application can be either desktop- or Web-based. Microsoft Silverlight is the Web-based element of WPF.

c) Silverlight

Silverlight is the Microsoft contender in the RIA player environment on browsers (the other major competitor is Adobe Flash). It is a cross-browser, cross-platform plug-in

that can be compared to the Adobe Flash player on browsers. Silverlight supports the delivery of RIAs, especially Microsoft's .NET-based interactive applications. Apart from being a runtime for media or interactive applications on the Web, Microsoft Silverlight's complementary service Windows Live offers free hosting and streaming solutions to developers and Web designers. Microsoft's Silverlight was earlier codenamed Windows Presentation Foundation/Everywhere (WPF/E) and is a direct offshoot technology of the Windows Presentation Foundation (WPF) [71].

d) XAML

Extensible application markup language (XAML) is an XML-based markup language and is object-oriented [72]. It was introduced by Microsoft as a language to define application user-interfaces. XAML allows developers to control the .NET user-interface elements, though it relies on XML to establish the meaning of the elements. XAML can be compared to other markup-based application languages, such as XUL (eXtensible User-interface Language), HTML, and FLEX. Markup-based interfaces are quick to build and modifications can be done easily. In addition, the applications require less code than traditional structured programming.

e) ASP.NET AJAX

ASP.NET AJAX is an Ajax framework developed by Microsoft. Formerly called Atlas, developers use it primarily with ASP .NET 2.0 to create RIAs. It is basically a set of control based on basic Ajax principles can be achieved using ASP.NET AJAX.

E. Mashups

Mashups are a new genre of applications being used for interactive richness in many fields, as in delivering the news, for trendy entertainment or, more lately, even for e-commerce purposes. Mashups are amalgamations of applications. They started as innovative experiments, done by hackers, but are now distinctive and unique facets of Web 2.0. A mashup can be defined as an application that is created as a result of ad hoc composition and integration of content from dispersed applications or services or both. This composition is enabled by providing simple Web APIs for the services that can be composed. Mashups enable anyone to compose services or form complete applications, very similar to the way in which blogs allow anyone to publish without sophisticated skills.

The architecture of a mashup application is composed of three different constituents. These parts are logically and physically separated by network and organizational boundaries. According to Duane Merrill, the parts of a mashup architecture can be typically classified as follows [73]:

- The API provider or content provider: Providers of content usually use existing Web protocols, such as REST, SOAP, and RSS feeds. Some mashup applications draw content from Web sites through screen scraping methods.
- The Web site or application that consumes API or content or hosts the mashup: Although the logic for the application resides on this Web site/application, it could be executed either with traditional Web applications, on the Web site through server-side technologies, or on the client-side using client-side scripting. Most often mashups combine client-side and server-side scripting for their execution.

- The Web browser of the client viewing the mashup: The browser used by the end-user of the mashup applications renders the presentation layer of the mashup. This layer of the architecture displays the user-interface for the mashup and brings the user interaction. Some mashup applications have their logic implemented on the client's browser.

Mashups often use Ajax, especially in the event of client-side scripting, to deliver richness and more user interactivity for the application. Mashups, as mentioned before use such protocols as REST and SOAP to communicate with remote services. These protocols are most commonly used when it comes to content aggregation. Also used are syndication technologies, such as RSS and Atom. They are XML-based and are most common when the content aggregation is in the news and blog spheres. Although quite common, techniques such as screen scraping are considered inelegant due to the need to constantly synchronize the content format of the provider and the content consumer. Despite that, Semantic Web efforts aim at establishing formal protocols for screen scraping. The Semantic Web uses resource description framework (RDF) efforts at augmenting data with metadata to provide it a meaning and establish syntactic structures for the data, thereby allowing the use of screen scraping in a more formal way. Lasilla and Hendler talk about a probable Web 3.0 that is actually the semantic Web [74]. Ingbert, *et al*, argue that mashup techniques are not completely new but are based on both formal traditions of software reuse and component-based programming [75]. The quality of service in mashups depends on the composite-services [76]. As mashups integrate content from different services, the quality of the mashup is adversely affected by bad quality of services used in integration.

F. Portals

EAI involves the integration of numerous individual applications within an enterprise and across several enterprises [1]. These individual applications are occasionally interconnected and process different kinds of data from different sources. EAI is the process of creating an integrated infrastructure for linking disparate systems, applications, and data sources across the corporate enterprise. EAI solutions can take on many forms and exist at many levels but the common methods of integration are user-interface integration, data integration, process integration, and method or function integration.

User-interface integration sometimes referred to as refacing involves redefining the graphical interfaces of different kinds of systems, such as legacy systems, PCs etc. Web-based user-interfaces have recently been replacing the more traditional GUIs. Portal frameworks are one such Web-based solution. A portal is a becoming an integral component of an enterprise integration strategy [11].

Portals are in general defined as a center point to access multiple resources, and it follows the desktop metaphor. There are several definitions available to define a portal, some of which are as follows:

A portal is an online service that provides a personalized, single point of access to resources that support the end-user in one or more tasks (resource discovery, learning, research, buying plane tickets, booking hotel rooms, etc.). The resources made available via a portal are typically brought together from more than one source [77].

A portal is a single, integrated point of access to information, applications, and people. Portals integrate diverse interaction channels at a central point, providing a comprehensive context and an aggregated view across all information [12].

Portals provide integration at the user-interface level, whereas other integration technologies support business process, functional or data integration [11].

According to [78], some of the key features of a portal are as follows:

- Content aggregation – The ability of portals to aggregate content from multiple, disparate sources and provide one unified view of the same.
- Personalization – This refers to the ability of the portal to tailor the content/services that it offers according to the user of the portal.
- Search – Most if not all portals, offer some form of searching of its content and sometimes even content from external sources.
- Single Sign-On – This feature allows users to authenticate once to the portal and then gain access to many other systems without the need for re-authentication.

1) Portals and Web Sites

A Web site is described as a resource on the Internet that consists of several Web pages with hyperlinks to each other. The Web site is made available online by an individual or an organization that would like to publish some information on the Internet.

Portals, on the other hand, are more personalized or customized. In general, portals and Web sites may be distinct entities, but in most cases they overlap and complement each other. Portals and Web sites can never replace one another. One can say that a Web site presents information regarding an organization or an individual to the outside world, whereas a portal provides multiple user roles with a common access point to broad resources [79].

Web sites usually target a wider range of audience with the same role, and provide targeted content from specific resources. They have public interfaces and may or may not have authentication. Portals can have public and private interfaces. It is a common access point to multiple user roles. Portals provide users with content from distributed resources. Portals usually require authentication for the user to make use of the customization. Other major differences include that in a Web site the user has to search for something he or she is looking for, whereas in a portal, the information is presented to the user according to his or her needs.

2) Examples of Portals

Example of portals are the Google and Yahoo homepages [80], [81].These portals allow users to add or delete features from the page. Users can customize the way the page looks in several ways, by adding tabbed pages to the portal or by dragging and dropping of portlets (features) into different positions.

Users can specify the different types of content that appear on the page by selecting from a list of available content. Apart from layout customization, users who are logged in can personalize the content. For example, a user can get the weather forecast for a city or request the stock quote for his or her favorite company. There are two levels of personalization available in these types of portals – one to modify the look and feel of the portal, and another to personalize the content.

3) Portlets

Portlets are pluggable components in the user-interface of a Web portal. Typically a portal would have more than one portlet. Each portlet has its own specific functionality. For example, a portal could have four different portlets, one each for a forum, news, "word of the day," and weather alerts. Portlet standards are evolving to allow users to create portlets that can be simply plugged to a portal that supports the portal standards. Each portal can be compared to a self-contained application. Web applications can be wrapped as portlets so that they can be plugged into third-party portals. Differences between Web applications and portlets basically stem from the distinct running settings and the targeted end-users of each type of software [11].

Portlets act as content aggregators and can display information gathered from different resources on distributed sources. The portlet standards control the way the look on a portal and also the way they behave.

The customization features on a portal allow registered users to add portlets to their portals or to edit or delete them. They can also customize individual portlets, personalizing not only the way the portal looks but also the content it delivers.

The standard for remote portlets, called the Web Services for Remote Portlets (WSRP) protocol, is to provide a Web services standard that allows the rendering of remote running portlets from disparate sources on a portal [82].

The Java Portlet Specification, Java Specification Request (JSR-168), enables interoperability for portlets between different Web portals. This specification defines a set of APIs for interaction between the portlet container and the portlet addressing the areas of personalization, presentation, and security [83].

4) Web Services for Remote Portlets

WSRP is a standard for Web portals currently under the management of Organization for the Advancement of Structured Information Standards (OASIS), a not-for-profit, international consortium that drives the development, convergence, and adoption of e-business standards. This specification is a joint effort of two OASIS technical committees, Web services for Interactive Applications (WSIA) and Web services for Remote Portals (WSRP). The joint authoring of these interfaces by WSRP and WSIA allows maximum reuse of presentation-oriented, interactive Web services while allowing the consuming applications to access a much richer set of standardized Web services [82].

WSRP is a set of presentation-oriented Web services that controls the way content is accessed and displayed on portals from different remote sources. A portal typically would require the integration of content and application logic from disparate sources or content providers. These providers use different interfaces and protocols. Assimilating content from these remote providers requires considerable custom programming. Portal vendors usually write custom adapters. WSRP's goal is to enable an administrator or an application designer to choose from a wide range of compliant remote content and application providers and to integrate them with minimal or possibly no programming effort.

WSRP is a platform-independent specification model, as it defines Web service interfaces using WSDL for interacting with presentation-oriented Web services. This standard also enables Java-based portals to be interoperable with Microsoft's proprietary WebParts technology [84].

5) JSR-168

Java Specification Request (JSR-168) is a project of the Java community process (JCP). This is a complementary specification to WSRP focused on portal interoperability for the Java development environment. JSR-168 is a standard Java interface for portlets that builds on the J2EE programming servlet model and is Java only [85].

Prior to JSR-168, almost all portal platforms offered their own proprietary approach to create pluggable portal components. For example, IBM had IBM portlets, Sun (iPlanet) had Providers, SAP had iViews, and Plumtree had Gadgets. JSR-168 aims to standardize these pluggable portal components so that they are independent of the actual portal server that they are written to [78].

The JSR-168 specification defines a set of APIs for portlets and deals with standardization for preferences, user information, portlet requests and responses, deployment packaging, and security [86].

6) WSRP and JSR-168

WSRP is a standard messaging interface for interacting with compliant UI components. JSR-168 is a standard Java interface for portlets that builds on the J2EE programming servlet model. It is an interface between a particular Java type of UI component and its hosting container. Some other differences are that JSR-168 is Java only, whilst WSRP is platform independent, and that JSR-168 is generally local, while WSRP is remote [85].

The main differences lie in WSRP's focus on XML and Web services deployment and on WSRP's inclusion of interchanges with remote portlets as part of its specification.

The WSRP specification anticipates the interoperability of portlet services developed either in the Java environment or in the Microsoft .NET environment. JSR-168 is directed at the immediate needs of the Java developer community and the ability to share JSR-168 portlets between any of the J2EE portal environments, such as IBM Websphere Portal and BEA WebLogic.

7) Portal Server

A portal server is a package that provides a basic infrastructure with the common services required to rapidly deploy a portal such as application connectivity, integration, administration, and presentation. It is an out-of-the-box solution that encompasses all portal needs. Portals can be built even without a portal server by using any Web-based technology: J2EE, .Net, or PHP.

G. Conclusion

We have reviewed some of the common architectures used in the design of computer applications. This chapter also introduces SOA and ESB as the approach we choose to apply to the design of the SOD described in Chapter III. We also assess the different RIA techniques offered on diverse platforms. The study helped us evaluate the possible interactions and features that applications can offer when they are designed as RIAs. Portals are becoming the user-interface of choice for enterprises focused on integrating their applications and resources. This chapter also provided a summary of current portals and the technology behind their delivery.

III. DESIGN OF THE SERVICE-ORIENTED DASHBOARD

A. Introduction

This chapter gives a comprehensive account of our approach to designing the SOD. We introduce current approaches and a brief literature review, and then concentrate on the design and organization of the SOD at three levels:

- The task-composition level: This section addresses the design from the process perspective.
- The user-interface level: This section addresses the design of the some of the dashboard's user-interface features.
- The services level: This section's objective is to be able to understand the SOD model at the system level.

B. Current Dashboard Approaches

The volume and the complexity of data used and produced during the business processes of an enterprise result in complex analysis. It is also expensive and time-consuming to use the data produced to enhance project goals. Project managers are often in need of tools to efficiently monitor the evolution of a project in real time [87]. In order to resolve this shortage, in the 1980s and 1990s, Executive Information Systems (EIS) were used [88]. EIS aided analysts and executives in decision-making through graphical

displays and accessible interfaces [89]. More recently, EIS has been replaced by enterprise dashboards and scorecards. Enterprise digital dashboards are recognized as one of the most effective tools for presenting and disseminating real-time business information in a concise, personalized, and easily comprehensible format, enabling managers to make rapid and informed decisions [18], [90]. Dashboards are vital elements in the day-to-day operation of modern enterprises, as they provide to analysts, or dashboard users, an overview of the essential metrics that reflect the performance of the business [90]. These metrics are called as key performance indicators (KPI).

Several definitions of dashboards exist in the literature [91]-[93], and address the dashboard as an executive dashboard , digital dashboard, or enterprise dashboard. However, most of these definitions depict the dashboard as a display or an integrated view of the KPIs of the business. In a report by Wayne W. Eckerson, dashboards are defined as multilayered performance management systems, built on a business intelligence and data integration infrastructure [94]. This type of infrastructure enables organizations to measure, monitor, and manage business activities using both financial and non-financial measures. Some dashboard developers also add intelligent collaboration features. These features include but are not limited to instant messaging, document sharing, and video conferencing. For instance, Microsoft defines a digital dashboard as:

> *"A customized solution for knowledge workers that consolidates personal, team, corporate, and external information and provides single-click access to analytical and collaborative tools [95]."*

Microsoft was one of the pioneers in dashboard development. Their digital dashboard uses chunks of applications or content with controls called Web Parts. Web

Parts can be used in Web sites to allow flexibility for its users in the area of customization of the user-interface. However, in general, dashboards are considered to be a single-screen display of critical business metrics and analytics to enable faster and efficient decision making. In his book *Information Dashboard Design: The Effective Visual Communication Of Data*, Stephen Few has compared different dashboards and defines a dashboard as a visual display of the most important information needed to achieve one or more objectives [17]. According to him, the dashboard fits entirely on a single computer screen so it can be monitored at a glance.

In references [17], [94], the authors list the three types of dashboards based on different problems or functional areas: strategic dashboards, analytical dashboards, operational dashboards. These classifications are based on visual design, and more importantly, the roles the dashboards play. These roles are decided based on what the KPIs displayed on the dashboard are used for. Bose *et al.* discuss several features a dashboard should have for enhanced personalization and user experience [96]. Dashboards should be optimized, context-sensitive, customizable, and workflow-integrated [97]. A key benefit of dashboards is that they equip users with sufficient information to help them to quickly respond to business dynamics. They provide an amalgamated point of view for financial reports and operational trends. Some dashboards are accessible even without Internet connectivity [93]. Not only do organizations use dashboards and scorecards to link budgets, forecasts, strategies, and reports, these tools are useful in complementing business intelligence implementations.

Chowdhary *et al.*, describe a model-driven approach towards the construction of a dashboard that was devised to minimize development times and costs [90]. Ganesh *et al.*

describe a Web services approach to building a service-based enterprise digital dashboard [18]. This approach is very similar to our approach in building the SOD but differs in several user-oriented design aspects. Dashboard models can also be designed to show the status and progress of various processes. Leymann *et al.* elaborates on a federated dashboard based on Web services, OWL and BPM [7].

Today, many different dashboard solutions exist on the market. Most of them are proprietary software; however, some open source solutions exist [87]. Dashboard technology can also be used in workflow consolidation and distribution, urgency evaluation, and process monitoring [97]. Aaorn says that it is not sufficient to simply display static data [88]. The importance of a dashboard is to display significant patterns of data, or information. Even better would be to display significant patterns of information with action plans, or knowledge. On that note, taking the dashboard a step beyond a simple display of KPIs would allow its users to interact with the processes.

In this thesis we present a design approach that allows the user to interact with processes while being able to study the related KPIs and make informed decisions to control the processes. The design presented also has a user-centric element. The proposed development and implementation is executed using SOA principles.

C. SOD Design Approach

We propose a three-tier model for the SOD design. We build this design on existing enterprise portal architectures. An enterprise portal's logical architecture can be depicted as a three-tier model, as shown in Fig. 5. The services layer includes all of the enterprise resources, including data sources and applications that are required by the users.

The business process layer controls the orchestration of the services to compose processes. The presentation layer controls the way enterprise information is presented to users. Our approach to the design of the SOD builds on this existing architecture by introducing two notions. One is the TAS-SET concept, and the other is the notion of exercising users' tasks as the objective for composing services or applications. To design the SOD, we focus on three layers of design. The first level is the user-interface tier. This layer is the part of the SOD that interacts with the user. We discuss the information and interaction design concepts of the SOD in Section 2. The user-interface layer, however, is layered on the task-composition layer, which is very similar to the business process layer of a traditional SOA. However, in the SOD design, we consider an individual user's process to personalize the dashboard. We use SOA to enable this approach. Further, we discuss the concept of delivering an application's functions as services (TAS) and enabling an application to consume a service (SET). Fig. 5 shows how our design approach builds on an existing enterprise architecture.

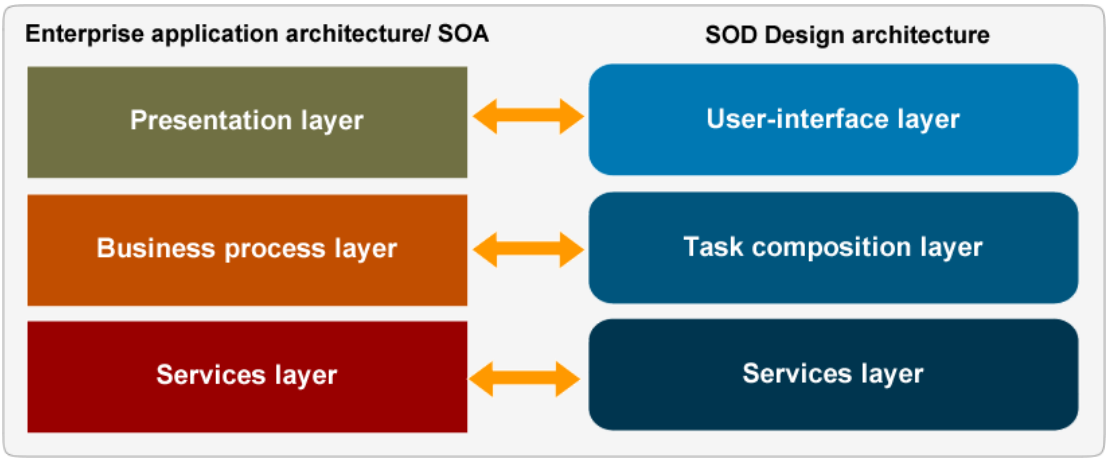


Fig. 5. Logical architecture of the SOD and relationship with traditional enterprise application architecture.

This chapter addresses three sections of the SOD design.

- Design of the SOD at the task-composition level: This section addresses the design from the process perspective. This phase design draws the requirements for the different features of the SOD design on the process abstraction of an enterprise and to frame the user-computer interaction more efficiently.
- Design of the SOD at the user-interface level: This section addresses the design of some of the SOD's user-interface features.
- Design of the SOD at the services level: This section's objective is to be able to present the SOD model at the system level and to demonstrate how the use of established technologies and novel approaches can be used to build the dashboard system.

1) SOD Design at the Task Composition Level

Enterprises are defined by its business processes. Business processes are a set of procedures that collectively realize the business objectives of an enterprise [98]. Business processes often require varying degrees of automation. This type of automated business processes within an enterprise can be called a workflow. The WorkFlow Management Coalition (WFMC) defines a workflow as:

> *"The automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules [98]."*

However, a great deal of ambiguity revolves around the term workflow [27]. In this thesis, we wish to use the definition provided by Georgakopoulos *et al.* [27]: a work-

flow as a collection of tasks (activities) organized to accomplish some business process. The WFMC defines an activity or a task as a piece of work that forms one logical step within a process that could be executed by a human participant or a machine. Workflows are designed to accomplish fractional objectives such as processing some information or performing a service to contribute to the overall goal of the enterprise. Fig. 6(a) shows a simple workflow in an enterprise where human and computer entities interact with each other.

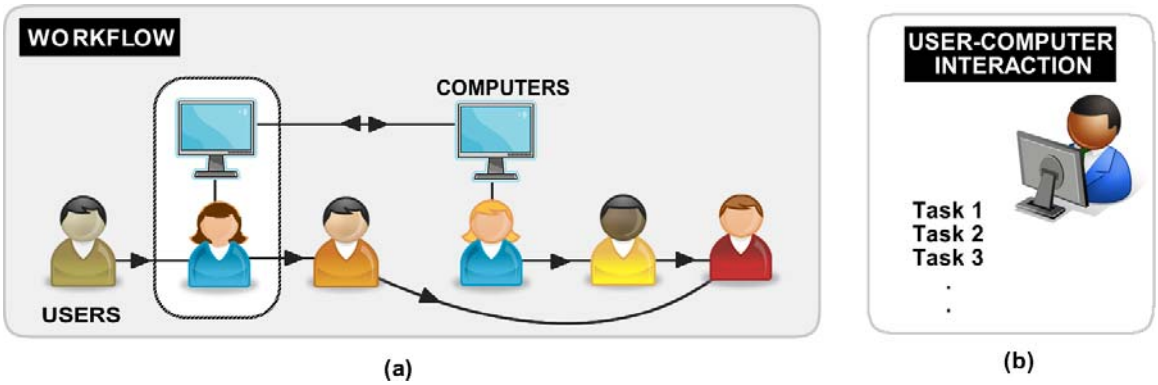


Fig. 6. An illustration of an enterprise workflow involving human and computer entities.

According to our definition, a user in a workflow, interacting with a computer, could have several “tasks”, as in Fig. 6(b). Some of these tasks may require the processing power of a computer, and some of the tasks performed by a computer could require human involvement. Composing a series of tasks performed by a person or a program is a common way to model high-level business processes in workflow systems [99]. This kind of interlaying of tasks performed by both machines and humans causes a great deal of communication between computers and the human personnel. This kind of communi-

cation manifests a need for human-computer-interaction (HCI) studies [100], [101]. One of the facets of HCI is the design of user interactions. Interaction design deals with the theory and practice of designing interactive and usable products [102]. There are also several resources that address interaction design patterns in improving usability of applications [103]-[105]. This section presents a drill-down approach to the construction of an SOD from the users' process perspective.

a) Dashboard layer

Beginning from the user, he or she faces multiple tasks as a part of his/her responsibility in the enterprise workflow. We call this the user's work function. The SOD is introduced to the user at this level, so any human entity interacting with a computer would use the SOD as the user-interface. The primary role of the user is to execute certain tasks, mostly in a specific order. Therefore, the first layer in the SOD model is the "Task" subset. As illustrated in Fig. 7, the SOD encompasses one or more tasks. In addition, in the context of the SOD, we considered "Tasks" as activities executed by human entities.

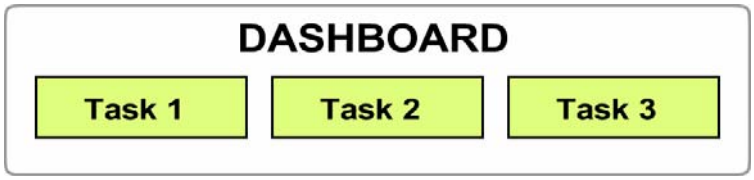


Fig. 7. Composition of tasks as a dashboard.

b) Task layer

Frequently, when a user interacts with a computer to execute a task, the user makes use of an application in the form of a Web page, a tool e.g., a word-processing application, or a program that was built to run a custom process. Hence, the user interacts with several applications to complete a task. Consequently, as illustrated in Fig. 8, we define the term "application" in the context of the dashboard as a subset of "task."

Fig. 8. Composition of applications to form a task.

c) Application layer

In the proposed SOD model, we assume the word application to refer to an ensemble of composed Web services. Let us consider an example of an application with respect to service composition. During a trip, in order to navigate to the closest Italian restaurant, User X would need to use three different services: a GPS service to determine his or her exact location of the user, a second service that would provide Italian dining options for a location, and finally a map service. When these three services are composed, we obtain an application that can guide the user to the nearest Italian restaurant during a road trip. As depicted in Fig. 9, an application can be composed from several composite services.

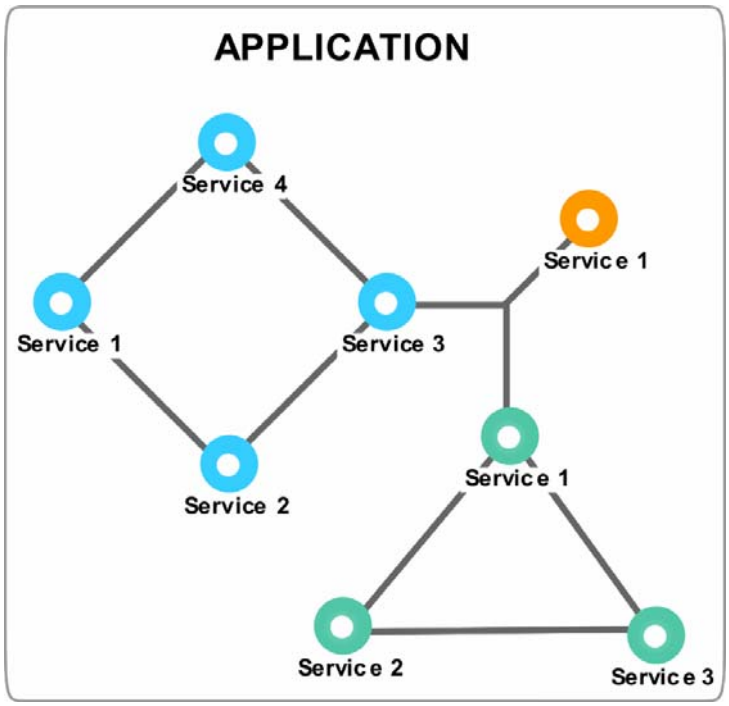


Fig. 9. Composite services as an application.

d) Composite services layer

Composite services are two or more Web services put together to deliver a service. Although, the term "service" usually represents the end result of the composition of two or more services, in this thesis we use the term to indicate a Web service. An example would be a service that calculates the current weather for a location by its ZIP code. A Web service is a service that is identified by a URI, has its features exposed programmatically over the Internet, and can be implemented via a self-describing interface. As services combine to form composite services or applications, it is important to pay attention to the design used in composition. Compositional design involves the interconnection of a set of services. According to Manolescu and Lublinsky [106], and Khalaf *et al.*

[99] there are two major compositional design approaches, hierarchical and conversational composition.

The SOD design is based on components – in the form or services, applications, or tasks, as depicted in Fig. 10. During the composition of services to form applications, there exists the possibility of type mismatches, architectural errors, and coupling. We use axiomatic design to detect these aberrations [107].

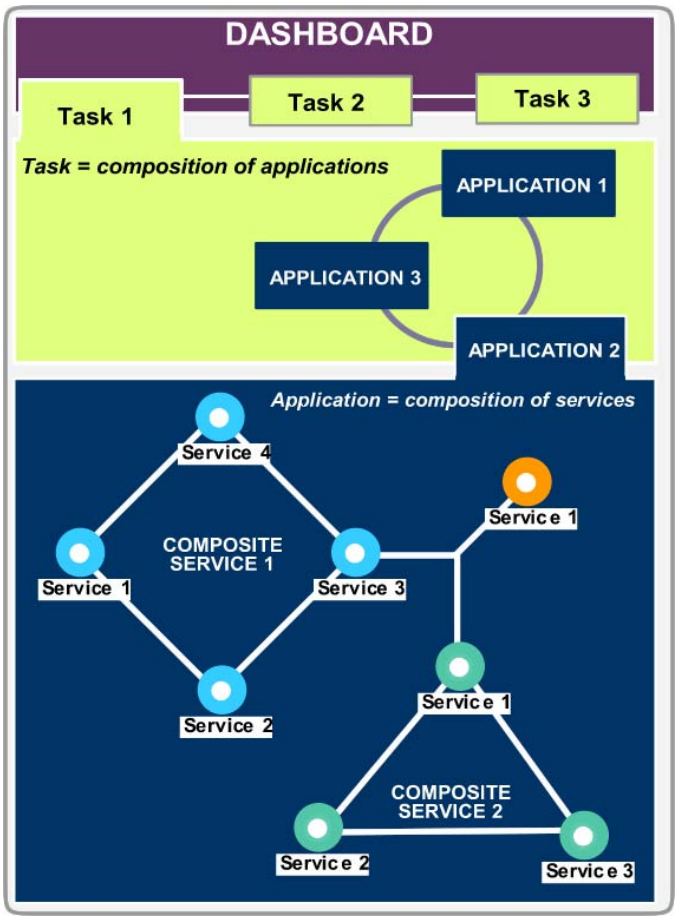


Fig. 10. SOD design from composite services.

2) Design of the SOD at the User-Interface Level

SOD design at the user level involves three main layers of development.

- Visual design.
- Information design.
- Interaction design.

Suggested by Jesse James Garrett, the above layers, along with the layers for functional specifications and user needs, form the elements of user experience [64]. Although each layer is laid on top of the other, these phases in design are not distinctly separated by rigid timelines. In this section, we focus on interaction design and information design. Visual design is not discussed in the thesis. However, visual design is the aspect of design that controls the look and feel of the dashboard in terms of such details as the color palette, font color, and typography etc [64].

a) Information Design

Information design is broadly defined as the art and science of preparing information so that it can be used efficiently by human beings. It is the process of effectively structuring the information that is delivered to the user. An element of information design is user-interface design. This concept for the design of the dashboard interface facilitates user input and system output to and from the dashboard system. Information design also involves the processes of communicating the relationships between the different elements of the interface. Although some of the literature cites interaction design as a part of information design, in this thesis we choose to differentiate the two [108].

The SOD is structurally split into blocks or containers based on the information architecture and connectors or navigation components. These components allow for the modularity of the SOD system. Fig. 11 shows a schematic representation of the SOD with respect to the information design elements discussed below. It also shows a hierarchical view of the SOD elements. The dashboard components are as follows:

- Dashlet – A dashlet is the smallest element of information that is displayed on the dashboard. Sometimes features such as dashlets are referred to as tiles or portlets in a portal. A dashlet is usually an application and provides a systematic approach of integrating the tools in a dashboard. Dashlets are similar in concept to content aggregators in portals and therefore, a dashlet can be compared to a portlet. In certain scenarios, as in the integration of a Web-based tool, dashlets can be defined using the Web services for remote portlet specification (WSRP) [109].
- Task Tabs – This feature of the dashboard is a tabbed view corresponding to each task that the user has. “Tasks” can be added by the user as tasks. Users can use the dashboard for multiple tasks at the same time. These tasks may require different tools and information from different sources. To avoid a cluttered dashboard with all of the information on the same page, we use a tab for each task. Tasks or tabs can be defined and dashlets can be added to a task or removed from a task using the personalization section or by using the edit option on the dashlet. Task tabs can be named as per users’ preferences.

- Tools – These are common enterprise utility components such as email, calendar, collaboration features such as instant messaging, and video conferencing.
- Alerts – The Alerts feature notifies the user as to the status of processes that are running while the user works on other tasks. For example, different colors can be used for different levels of attention, and clicking on those alerts takes the user directly to the process that requires the user's attention. Naturally, alerts can be disabled at the users' discretion.
- Sections – These are the different pages of the dashboard. Each section of the dashboard offers different controls for the continuous development or reconfiguration of the dashboard.
    - Assignment: The most important section of the dashboard is the Assignments section which displays the user's tasks. Several tasks make up an assignment.
    - Task Organizer: The task organizer section allows the users to frame the tasks using specific applications from the applications library and connecting them with rules.
    - Personalization: The Personalization section allows users to change general settings, such as the way the user receives alerts, and the visual presentation, including the color scheme of the dashboard. This feature also enables the user to select what content he or she would wish to see on the dashboard. Tools that are not a part of tasks, such as email or calendars, can be added to the dashboard.

- Applications Library: The Tools Library is a collection of available tools that are TAS enabled. The TAS approach is described in detail later in Section 3(a). Users can select from the list of tools and applications they would prefer to use on the SOD and eventually use in their tasks. In the hypothetical example used in the problem scenario section, Microsoft Excel (or a spreadsheet application) and Word (or a word-processing) could be two tools in the Tools Library if they are service-enabled. That is, they should be pluggable to the SOD. Other tools could be visualization components, such as KPIs and supporting analytics. In either case, it is important to choose the visualization that best meets the end user need in relation to the information he or she is monitoring or analyzing. Microsoft Excel offers great capabilities for charting KPIs on a dashboard [110]. Based on the information infrastructure of an enterprise, the application and tool libraries could be a repository of service-oriented components that can be coalesced to form dashlets. Dashlets could also support syndication feeds. This repository could also contain related feeds. Applications, tools, and feeds can be added to the library using a search mechanism for published components.
- File Repository: The File Repository is used to store all of the documents and files that are produced by any of the applications in the SOD. These documents can be recalled for further edits or can be batch processed for such jobs as printing or making charts.

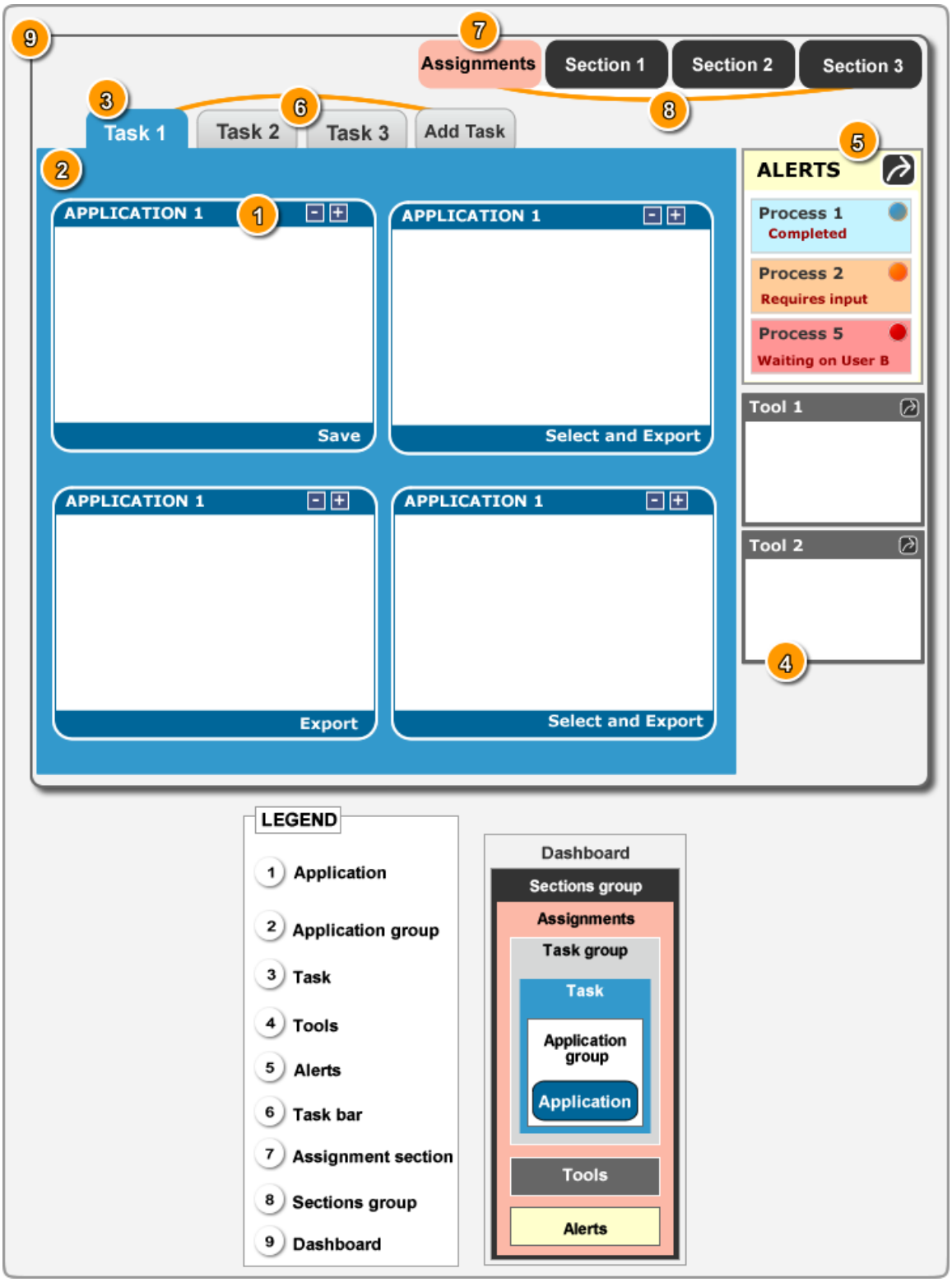


Fig. 11. Illustration of the components of the SOD interface.

b) Interaction Design

Interaction design involves the cognitive process of designing interactive systems [102]. The term “Interaction Design” can be defined as:

> *“The selection of behavior, functions, information and their presentation to the user [111].”*

Yamamoto and Nakakoji describe interaction design as the process of determining the representations and operations of an application system by considering what representations the user needs to interact with, through what operations [112]. Interaction design basically controls the movement and actions of the user when interacting with the dashboard and also the way in which the SOD reacts to those interactions.

A digital dashboard has traditionally been used to display KPIs or aid in analytics. The SOD adds a layer of interaction to the other services. The SOD serves as an integration solution offering seamless execution of users’ routine tasks that could require interaction with several applications and tools. The SOD is designed to simulate an enterprise’s member’s job functions. The dashboard is modeled with the user in perspective, and the user-interface components represent the components of the user’s work function. Every human entity in an enterprise has a set of responsibilities, usually referred to as his or her assignment. The user executes the assignment by interacting with the assignment section of the dashboard. An assignment is a set of tasks represented by the tabbed task views of the dashboard. The user interacts with several applications during the course of a task. These applications are represented by an equal or fewer count of dashlets. Fig. 12 represents this relationship between the user’s work functions and a dashboard’s design element.

Composition of a task is achieved by composing applications. A task can be described as the interaction between the user and a set of dashlets. The user has maximum interaction with the dashlets. As applications are made available through the dashlet, they should accommodate minimization, maximization, and docking and undocking features to support enhanced flexibility. Dashlets can be added or removed to compose a task in the Tasks organizer section. The Applications library is a repository of tools and applications that can be added to the dashboard as dashlets; however, this kind of composition to create dashlets may in some cases require custom design and system integration efforts.

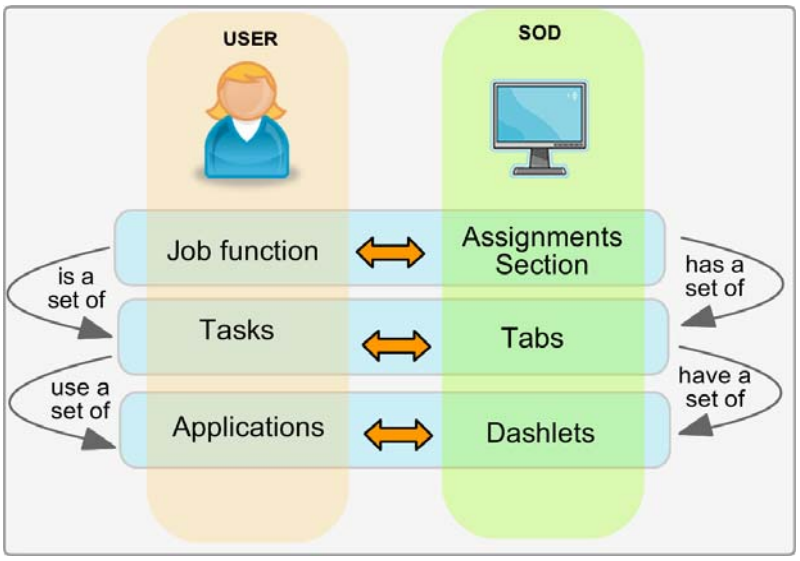


Fig. 12. Mapping user function elements with SOD interface elements.

In order to execute a task, users simply have to use and interact with the dashlets in a Task tab. Each dashlet can have specific data transfer operations, such as saving a document from a word processing dashlet or exporting data from a Web page into another dashlet. The process of completing a task is demonstrated using the hypothetical

scenario described in the Problem Scenario section in Chapter I. Fig. 13 depicts an SOD instance for the hypothetical scenario. The user logs onto the Web-based SOD. Task 1 is the tab he focuses on as it models his first task. There are two dashlets in this task. The first one displays the trend and quotes for a list of symbols that are preset by the user. After analyzing the quotes, the user transfers the data by exporting it to Microsoft Word as a table (SET), and then the user performs some transformations to the data using a service from Microsoft Excel (TAS). The user uses the charting tool of Microsoft Excel to chart the data into different charts, and then creates a complete report including the charts produced by the charting tool – all inside the Microsoft Word dashlet. The documents can be stored in the file repository for later retrieval.

The Personalization component allows the users to change general settings, such as the way the user receives alerts and the color scheme of the SOD. This feature also enables the user to select the content he or she would prefer to see on the SOD, apart from the tasks.

The SOD interactive features should be spontaneous and the data transfer seamless. RIAs are examples of applications that have great interactivity and connectivity; hence, a Web-based dashboard can be implemented as an RIA. In addition, pluggable components or applications called mashups can be used as dashlets. The dashboard can be a pluggable module in a portal, or the dashboard design can be adopted in a portal. The design will allow portals to append a relationship between portlets to frame tasks and workflows.

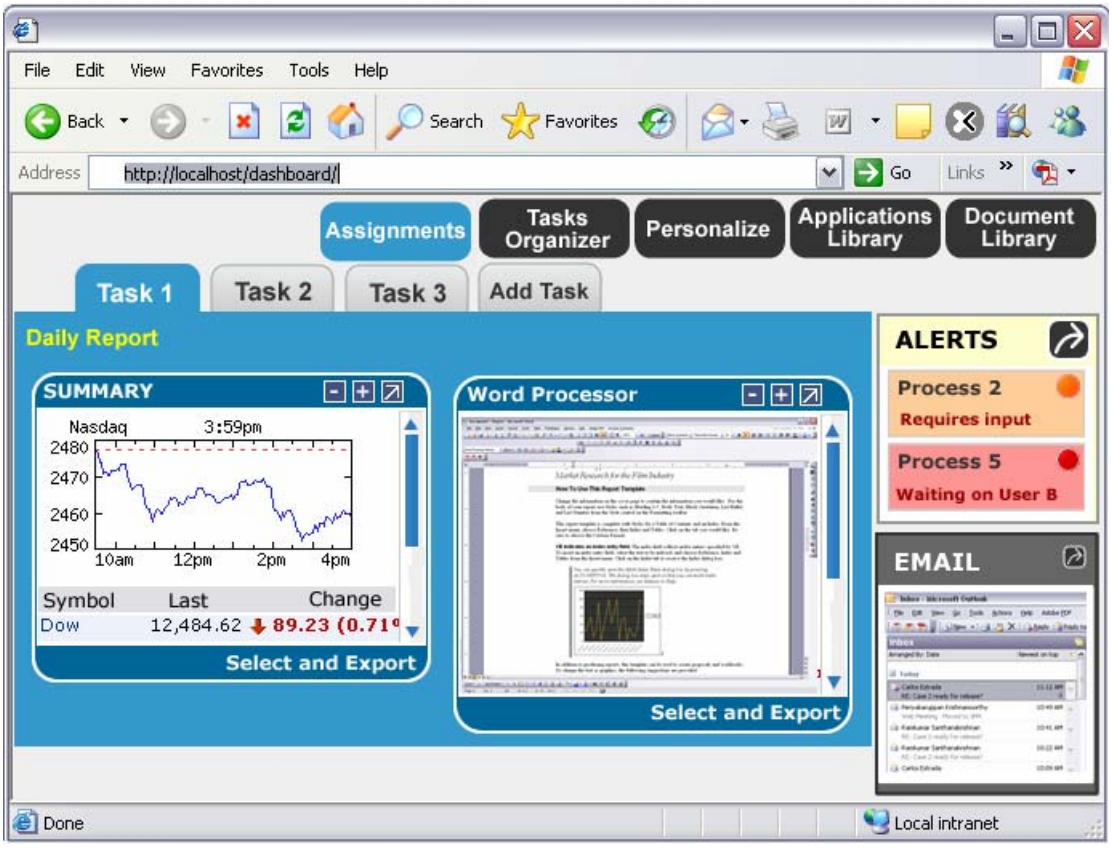


Fig. 13. A simple representation of a Web-based SOD for the problem scenario described in the Chapter I.

3) SOD Design at the Services Level

The SOD design at this level is based on SOA. There are two subsections in this section. Subsection (a) describes an application of TAS and SET in SOD design to improve interaction. Subsection (b) describes the SOA approach that is followed in this thesis.

- TAS-SET model [113], [114].
- SOA and related concepts.

The TAS-SET approach is a bipartite notion in which the first concept is the Tools-as-Services (TAS) approach and the second, complementing concept is the Service-Enabled Tools (SET) approach. These notions involve the introduction of applications and their functions as services and the perspective of applications consuming Web services. SOA offers us the capability of integrating content from multiple sources through Web services, similar to the approach used in portals and other content aggregators. Along with such concepts of Web services, ESB, and adapters etc, the SOD design includes features that accommodate various popular notions of Web 2.0, such as syndication feeds and mashup features, etc.

a) Tools-as-Services and Service-enabled Tools approach

The Encyclopedia Britannica and the Merriam Webster online dictionary refer to the word “Tool”, in a computer science context, as:

> *“An element of a computer program or application that activates and controls a particular function.”*

An application in this section simply refers to a software application or a computer program. A Web site that delivers some kind of a service could be an application. Microsoft Excel is a spreadsheet application. The chart wizard feature of Microsoft Excel is a tool. We use these concepts of tool and application in this section. Fig. 14 illustrates a tool in the context of an application.

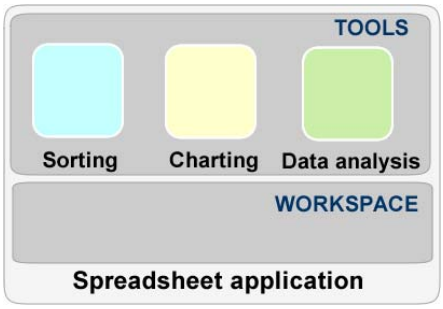


Fig. 14. Multiple functions or tools in a spreadsheet application.

In the TAS approach, an application and its functions (tools) could be used the way Web services are used [113], [114]. An example of TAS is provided in [115], [116], which shows the spell-check feature of Word exposed as a service. This concept is a three-step process. In the first step, the desired tools would be componentized using service descriptions and placed in a repository, such as the UDDI. This step is analogous to publishing a Web service. The second step involves the use of an application that searches for tools in the registry based on their service descriptions. This process is similar to service discovery. As a third step, an aggregator application such as the SOD or a portal can then use the invoked tool, thereby delivering the invoked tool's functionality in a completely different application interface.

The TAS approach can be demonstrated using the following illustration. Let us consider a process where a user must use two applications, to complete her task. Microsoft Excel and Word, both these tools function independently. As a part of her task, the user employs Excel's charting function for a report that she types in Word. This involves using two applications, switching between them and copying and pasting or embedding content from one to another. In this simple case, the applications considered are easy-to-use and there are no obvious circumstances for errors, but in a real-life situation, com-

puter users use several applications and tools to perform their tasks. In this case, there could be significant loss of time and accuracy in switching between applications or transferring data manually.

Second, most of the applications have excessive toolbars. Most of these “features” are not used by the average person. For example, the user in our scenario may need only the Excel charting tool and no other functions, as a part of the workflow. Hence, it is sufficient for the user when the dashboard displays just the necessary part of Excel (application) and the chart plotting function or tool as a dashlet. This aspect of personalization focuses attention on the tool used and the task to be performed by avoiding clutter in the user’s workspace. Fig. 15 shows the toolbars of Microsoft Excel. For a person who uses Excel only for its charting function, these tool bars are needless.

Fig. 15. Chart wizard tool in Microsoft Excel interface.

In the scenario under discussion, the tool is invoked in the parent application itself (that is, the charting tool is invoked by Excel). However, in SOA, it should be possible to invoke a tool of an application into a totally different but structurally similar application. For example, if the user has a table in Word, instead of copying the table data from word to Excel, plotting the chart in Excel, and copying it back in to Word, the user could simply use the charting function of Excel in Word. As evident from Fig. 16, Excel’s charting

function and the Word applications are exposed as services that are consumed by the dashboard in one single dashlet application.

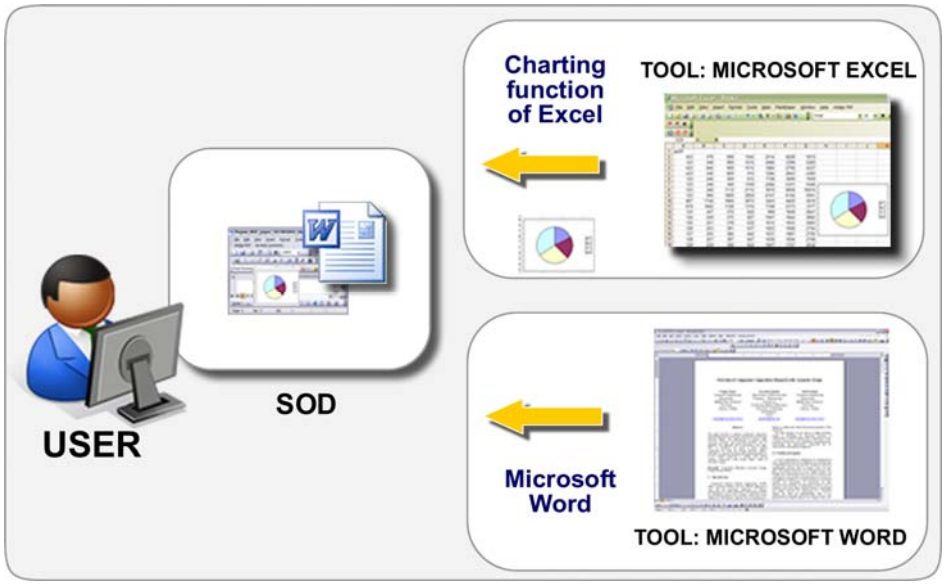


Fig. 16. A manifestation of the SOD using Microsoft Excel's function as a service inside Word.

In order to realize this architecture, it is important for application developers to follow SOA. It is important to expose the tools interface as Web services. In addition, applications have to be built to invoke and apply Web services or tools as services. This concept is the SET approach. Fig. 17 illustrates the concept of tools-as-services and service enable tools approach.

The TAS approach has a complementary method in the form of the SET approach. In TAS, the functions in an application are published as services. On the contrary, service-enabled tools are applications that can consume a service and allow the application user to access the invoked service in the application's interface. An example of this

concept is the ability of Microsoft Excel to display the results of Web services. For instance, the delayed stock quote Web service provided by Xmethods, along with the Web service references tool can be used to check stock quotes over the Internet. Microsoft Excel can now be used to display the results of the Web service's operation in its interface [23]. An application can consume and operate a Web service within itself.

Recognizing the TAS and SET approaches provides significant value-added features to the dashboard in several ways. Use of multiple tools on a single interface (dashboard): First, the TAS model enables the user to complete his or her tasks using a single application (dashboard) or in an application, with which he is most familiar. Second, although the user-interface of the tool is preserved, data connectivity is seamless when used on the dashboard. There is no need for the user to switch between applications. Third, we reduce the manual movement of data between tools by the user. This potentially reduces significant data entry errors.

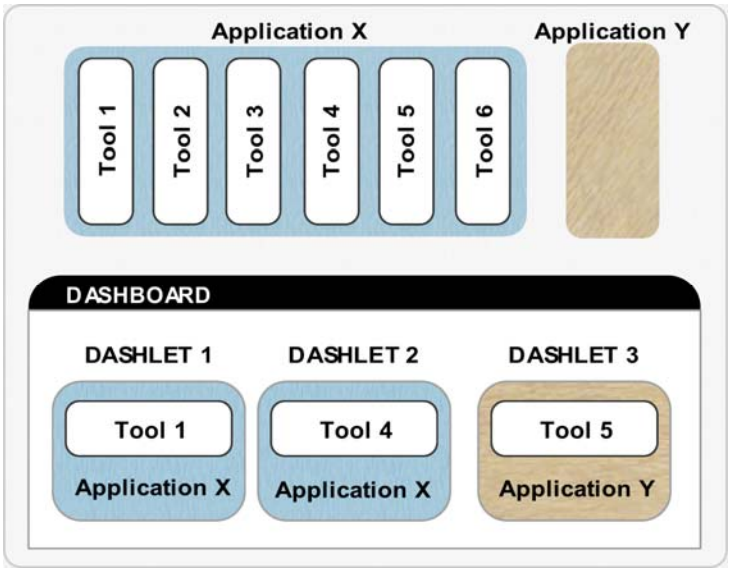


Fig. 17. Tools-as-services and service-enabled tools concept.

Some of the notable benefits of the TAS-SET model are as follows:

- Reduction of clutter in the application: Several applications have numerous functions in the form of toolbars. When only a single function is required in the application, the dashlet can display just the working interface and the tool required for that task. This approach avoids clutter on the user's workspace, thereby improving efficiency.
- Reusability of tools: A tool is designed for a specific purpose or niche. By replacing a tool with Web pages or another GUI, we are replacing years of engineering effort that makes a tool effective for being used in a particular activity.
- Choice of user-interface: The applications in the TAS-SET model can act as both TAS and as clients. Consider a scenario where a user uses Microsoft Excel and MATLAB for a task. MATLAB is a very specialized program and requires a learning curve for the efficient use of its functionalities and interface. The user uses MATLAB only for a single function. With the TAS-SET model, the user may prefer to work with Excel as his choice of application and choose not to use MATLAB. In this case, MS-Excel can invoke the required, exposed MATLAB function. Microsoft Excel acts as a client (SET), while MATLAB's function acts as a service (TAS). If the user is more familiar with MATLAB and not so much with Excel, a different scenario will be that MATLAB invokes Microsoft Excel functions; thus MATLAB acts as a client while Microsoft Excel acts as a TAS. Fig. 18 illustrates this scenario.

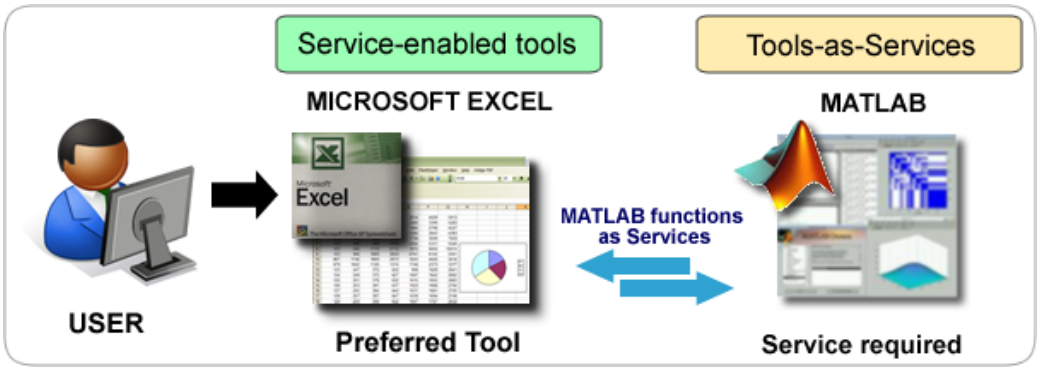


Fig. 18. TAS example with MATLAB's functions as a service in Excel.

b) SOA for integration

We envision the dashboard as a personalized, service-enabled, RIA. Our dashboard model is essentially a composite application of desktop and Web-based tools interoperating seamlessly with each other. We have considered several approaches that could be used to develop the dashboard.

The different components of the dashboard are integrated using an SOA-based approach. The implementation will use Web services standards. There are two types of Web services, as described in Section B3 in Chapter II. The document-style exchanges data as XML documents, while the RPC-style Web service describes the interface in the format of a method-signature and takes input and output in a programming-language-specific data type [117]. The document-style approach offers significant advantages over RPC call approaches. It allows the full use of XML, which makes the handling of the complex biological documents easier. It can handle asynchronous processing better than RPC systems, which helps in improving the reliability, scalability, and performance of systems [118]. Importantly, the document-style approach can divide the system concep-

tually into two distinct layers, the infrastructure layer and the application layer, by programming to an interface rather than an implementation [119]. By using this divide and conquer model, we can exclude application logic during integration of the data sources and software tools. This ensures that the resultant services are easily exportable and reusable by different applications, rather than tied to a single implementation. During the development of an application, the different interfaces can be composed together, either programmatically or by using compositional approaches such as the business process execution language (BPEL) based on an analysis workflow. The document-style approach can be implemented using an ESB, one of the approaches that can be used to construct the dashboard's integration system.

c) ESB-based integration

The ESB is a software architecture construct and is described as a highly distributed approach for enterprise integration that provides capabilities for building integrated systems in incremental, digestible chunks, maintaining their own local control and autonomy, while still being able to connect together each integration project into a larger, more global integration fabric [120]. The use of ESB offers distinct advantages [45], [121], [122]. ESB can connect to different components such as data sources and computational tools using different transport protocols but expose them as services using a common protocol [45]. The initial steps involved in integration with data sources or computation tools involve writing adapters to connect with the information sources and generating the XML document. Most ESB provide adapters to connect to data sources; however, we can extend them by using custom adapters if an appropriate adapter is not available. Adapters

can be written to any component using standard or proprietary protocols. For example, adapters can be written to connect to databases using such standards as the open database connectivity (ODBC) [123]. Fig. 19 shows the adapter in the ESB that connects the dashboard to the Web services.

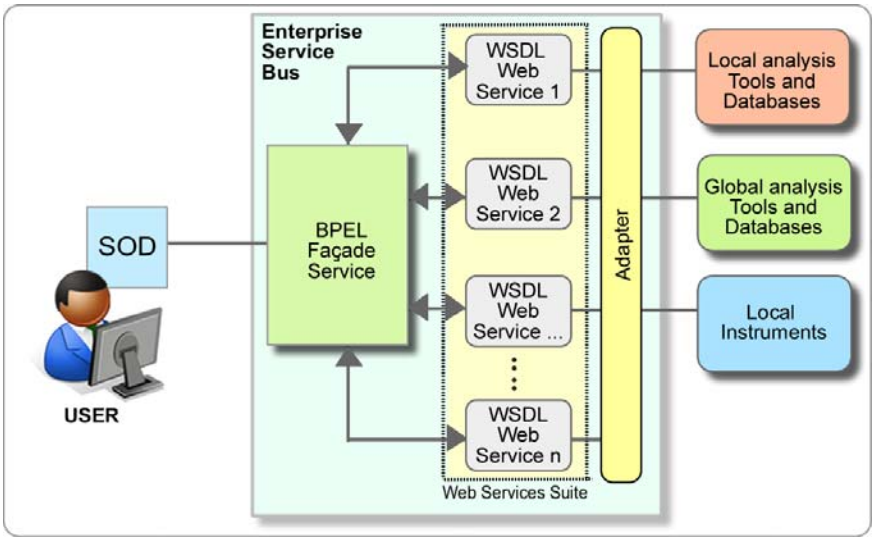


Fig. 19. Application of ESB in the design of SOD.

The two important aspects of the dashboard services layer are as follows:

- Workflow composition and process management: Many competing composition languages exist in the Web services model to execute workflows. The most popular orchestration language seems to be business process execution language for Web services (BPEL4WS), a combination of IBM's Web service flow language (WSFL) and Microsoft's Web services for business process design (XLANG). Another language is the Web service Choreography Initiative (WSCI) approach proposed as a W3C standard by Sun, BEA Weblogic, and

other partner companies. Stabb *et al.* compares BPEL4WS, WSCI, and many of the other orchestration approaches that currently exist [124]. Process management techniques can be developed on top of the BPEL4WS layer for monitoring the workflows [125].

- Process personalization: The dashboard is built on a framework as illustrated in Fig. 20. The services and tools are discovered in service and tool registries, respectively. The ESB layer provides the system's integration with Web services, tools, and applications through the mediator. The mediator service is responsible for the communication of data with the tools and applications through the wrappers or adapters. Business processes and workflows are controlled by BPM and orchestration layers. Personalization is offered at a higher level at which users can have customized information presented to them based on their user profiles. On top of the personalization layer is the dashboard layer. The dashboard has a tool-service adapter and its user-interface.

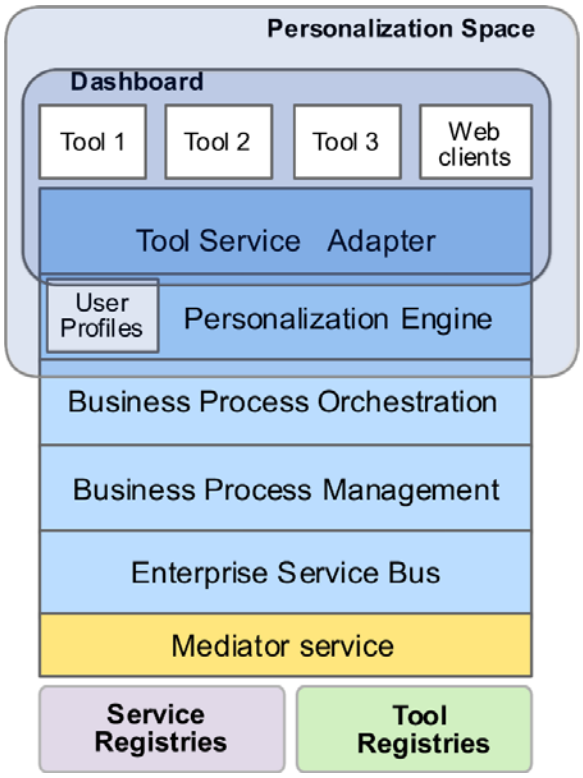


Fig. 20. SOD framework.

D. Conclusion

While there are several issues to address in the development of the SOD such as data mismatches, mediation between different tools, unified login experience, and personalization capabilities, this thesis focuses on the different components of the SOD at the design level. The next chapter describes an example deployment of the Dashboard model to improve user experience in a biological research application.

IV. CASE STUDY

This chapter introduces two case studies from the bioinformatics domain. The case studies involve laboratories that deal with extensive data collection/analysis and complex workflows. The bioinformatics research lab can be a good example of an enterprise model because a bioinformatics laboratory, like an enterprise, faces a constant need for integration of such resources as databases. This chapter starts by comparing the bioinformatics domain to a business enterprise model. For each case study, we briefly discuss the background of the research in the laboratory, the specific workflow considered during the study, the need for integration methods, and finally how the SOD can enhance the workflow of the lab.

A. The Bioinformatics Domain

The term bioinformatics was coined by Hwa Lim in the late 1980s but gathered momentum only after the human genome project, one of the most remarkable milestones in life sciences research over the past few decades [126]-[128]. The availability of genome sequences for hundreds of different biological species has created a foundation for the development of a broad array of high-throughput experimentation tools and methods for automated generation of biological data [123], [129]. Because of the use of technology, automation, and other evolving techniques in the laboratory, unprecedented volumes of analytical data were produced [130], [131]. There was an emergent need for the me-

thodical collection, storage, and analysis of biological information and bioinformatics refers to the application of information technology to the computational systems used in this effort. The bioinformatics domain and the business domain have several similarities. For example, the integration of bioinformatics resources, such as databases, is a massive challenge. This problem is analogous to the enterprise integration problem in the business domain. Likewise, both domains face the necessity of developing tools and software to solve certain analytical problems with efficiency and to aid in operational processes. For this reason, we consider that a bioinformatics research lab to be a special case of a business enterprise model.

1) Bioinformatics Data

Although research organizations and business enterprises have some common objectives, life sciences data differs from the data used in business application domains [132]. The characteristics of bioinformatics data are as follows:

- More complex and has constantly developing data structures when compared to business data.
- More distributed in source and heterogeneous in nature.
- High-volume, noisy, formatted differently, occasionally incomplete, and incompatible.
- Lags behind business data in ontology development.

Although the characteristics of bioinformatics data might be different from enterprise data, the technology used for integration is similar. For instance, SOA techniques are being used for integration in the bioinformatics domain. This is because the services

model offers an abstraction model for the data. This implies that only the schema of the data is different and that however, the integration steps remain the same.

2) Need for Integration

> *Biological databases have been invaluable for managing these data and for making them accessible. Depending on the data that they contain, the databases fulfill different functions. But, although they are architecturally similar, so far their integration has proved problematic[133].*

The integration challenge faced by the bioinformatics domain is appropriately described by researcher and author Lincoln Stein. New and developing techniques in research have resulted in massive, incredibly heterogeneous, and discrete data. Resourceful integration is almost becoming a necessity for more effective research. The lack of intelligent integration of data, tools, and research methods is proving to be a critical bottleneck in scientific research. Fig. 21 depicts this need for integration.

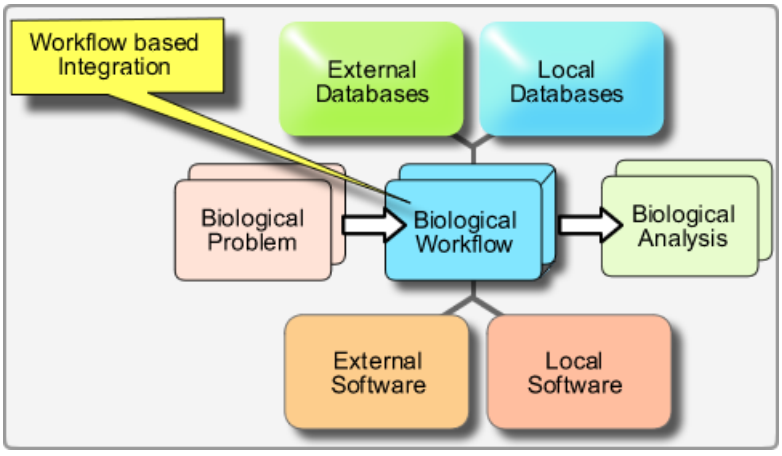


Fig. 21. Need for integration in bioinformatics.

Fig. 21 depicts the need to connect databases, software tools, and applications within the lab with the databases, software tools, and applications outside the lab. These resources are written by different people using different data formats and different technologies. For instance, the integration of such diverse and discrete databases has been challenging.

In the life sciences domain, integration issues encompass four aspects:

1. Integration of external databases.
2. Integration of analysis and computational tools.
3. Integration of the tools and instruments used within the lab.
4. Integration of results published.

Although there are numerous approaches to the integration of biological resources available [134]-[137], we consider only the service-oriented approach in this thesis.

3) Current Infrastructures for Integration

SOA is used in several bioinformatics integration approaches [117], [138], [139]. SOA can be defined as a specific software platform based on a set of decoupled, well-defined software components or modules that have well-described interfaces and message exchange protocols [140]. One of the essential features of the SOA is the use of Web services. Many bioinformatics data sources are becoming available on the Internet and, more recently, as Web services [136], [141], [142]. These Web services are available as REST Web services or as SOAP-based Web services.

In order to better define the purpose of the service during the data integration process, developers use semantics. The semantic Web has been developed to provide

common meaning between concepts used in Web pages and services, and to improve the integration process. Fully developed ontology languages have been defined more recently. Ontology is a group of conceptual definitions that describe an application domain. An example of an ontology language is the Web Ontology Language (OWL).

Large capacities of computing resources or grids are connected at a national or even worldwide scale to build virtual supercomputers. A few examples of such computing infrastructures are MyGrid, Biomedical Informatics Research Network (BIRN), and BioMoby [22], [143] .

The use of software agents in the integration of disparate biological data sources is another concept that is growing popular [130], [136], [144]-[148]. The software agent metaphor constitutes a flexible approach towards the integration of bioinformatics tools and the dynamic composition-execution of data analysis workflows [145]. The bioinformatics domain is characterized by rapid, substantial changes over time [146]. Although the volume of data poses a large threat, the change in the resources available to the bioscientist is another problem. Any system intended for application to the bioinformatics domain should be able to cope with this dynamism and openness, and nothing addresses these concerns as comprehensively as the agent approach [146]. Using an agent-based framework for Web services is efficient, especially in the field of bioinformatics integration [148]-[150].

B. Case Studies

In order to depict and study the problem scenario more effectively, we conducted two separate studies of two research laboratories and their workflows. This chapter de-

scribes in detail some of the labs' case-specific workflows and processes. The case studies represent a problem domain that in part follows a business model in certain aspects, such as integration and the development of tools and information management systems. Our focus for the case study was to derive the general requirements for a generic dashboard system. The need for the SOD is to provide an effective interface to facilitate interactions with the researchers and their different data sources, processes, computational programs for analysis, and instrumentation used in experimentation.

These studies were designed, conducted, and analyzed with the following specific objectives:

- To first document the laboratory's workflow under study,
- To design a service-oriented approach for a part of the workflow,
- To record the requirements for a dashboard to monitor the workflows, and
- To illustrate how the service-oriented approach might improve the workflow.

The sources of data for the case studies were laboratory documentation, interaction with laboratory members, direct observation, and demonstrations of laboratory processes.

These studies enabled us to formulate the needs and the design for an effective SOD. The methods in the laboratories, their workflows, and the results of the case studies are discussed under the respective case sections.

1) Case Study – Arnett Lab

The first case study presented in this chapter was conducted in collaboration with the Department of Epidemiology at UAB. This case study focuses on a specific part of

the laboratory's research workflow that requires the researcher to use several Web pages and Web-based tools. It also requires the researcher to search several large documents manually. The case description includes the design of a dashboard-based solution and the implementation of a prototype that creates a part of the solution.

a) Research background

HyperGEN is the Hypertension Genetic Epidemiology Network. Now in its ninth year, this study seeks to identify the genes contributing to hypertensive heart disease in participating families [151]. One of their studies involves left ventricular hypertrophy, the heart condition in which the myocardium of the left ventricle thickens.

HyperGEN has identified four genomic regions showing evidence of linkage, has fine mapped these regions, and has successfully identified variations within positional candidate genes contributing to interindividual variation in left ventricular mass [151]. They have recently completed a genome-wide association study to identify genetic variants that play a modest role in the hypertrophy phenotype. Genetic association studies are performed to determine whether a genetic variant is associated with a disease or trait. Genetic linkage studies are done by linkage mapping and are critical for identifying the location of genes that cause genetic diseases.

Linkage and association studies typically identify regions of association that cover a region of ten to thirty centimorgans (cM), which can contain up to 300 genes. There are two ways to identify the genes in a particular region.

1. Method 1: Conventionally, these regions are identified by the linkage and association studies, and are then fine-mapped. During the process of fine-

mapping, the area under each region is saturated with additional molecular markers and these markers are then tested for association with the trait in question. This technique greatly reduces the area under each region and results in a handful of recognized candidate genes for each region. However, it can be costly and time-consuming.

2. Method 2: The other method is to identify the genes that lie within the region of interest and to select genes from this list for further analysis based on some criteria, such as previously demonstrated association with the phenotype. The large number of genes in a linkage region, and the difficulty involved in identifying and cataloging those genes, makes this process complicated.

The case study we have described in this chapter focuses on method two. Genetic researchers often resort to a manual approach when following this method. This case study examines the various manual steps that the researcher has to execute to find the necessary information to identify genes in a region.

In order to identify genes in a particular region of interest, the second method involves searching for each individual gene and retrieving the corresponding PubMed citations and Online Mendelian Inheritance in Mankind (OMIM) information. This information is again subjected to a manual search with key words of interest. This process is ineffective, cumbersome, and highly time-consuming. Once selective genes are chosen from the list of genes in the region, investigators then try to use comparative genomic approaches, where those chosen human genes are compared to conserved genes found in model organisms, such as rats and mice. Mouse and humans share regions that have the same gene order and content, aligning these regions can help to identify a smaller region

of interest than the initial linkage analysis. This technique has been successfully employed in only a few circumstances as in Vitt *et al.*, in which a kidney disease quantitative trait locus was narrowed from 10 to 11.5 Mb by identifying syntenous regions in humans and rats [152].

b) Workflow

This section describes the step-by-step workflow employed in method 2 described in the previous section. The National Center for Biotechnology Information (NCBI) database is used extensively in this workflow for the purpose of obtaining the following information:

- The set of genes in a region of interest.
- Information on the OMIM.
- PubMed information.

The workflow can be split into four non sequential tasks. For example, the list of genes obtained may number have about 300 genes. Starting on the first gene, the OMIM information must be retrieved, searched, and recorded (if the search is successful) or discarded (if a keyword search results no hits). Each gene may have more than one OMIM reference and usually more than one PubMed citations. Each of the records has to be searched for key words of interest. Finally the decision is made by the investigator as to whether to select the gene. This sequence of tasks is carried out for every gene in the sequence. This process could literally take days if executed by a single individual. Fig. 22 depicts the workflow that is made up of four tasks discussed in the following sections.

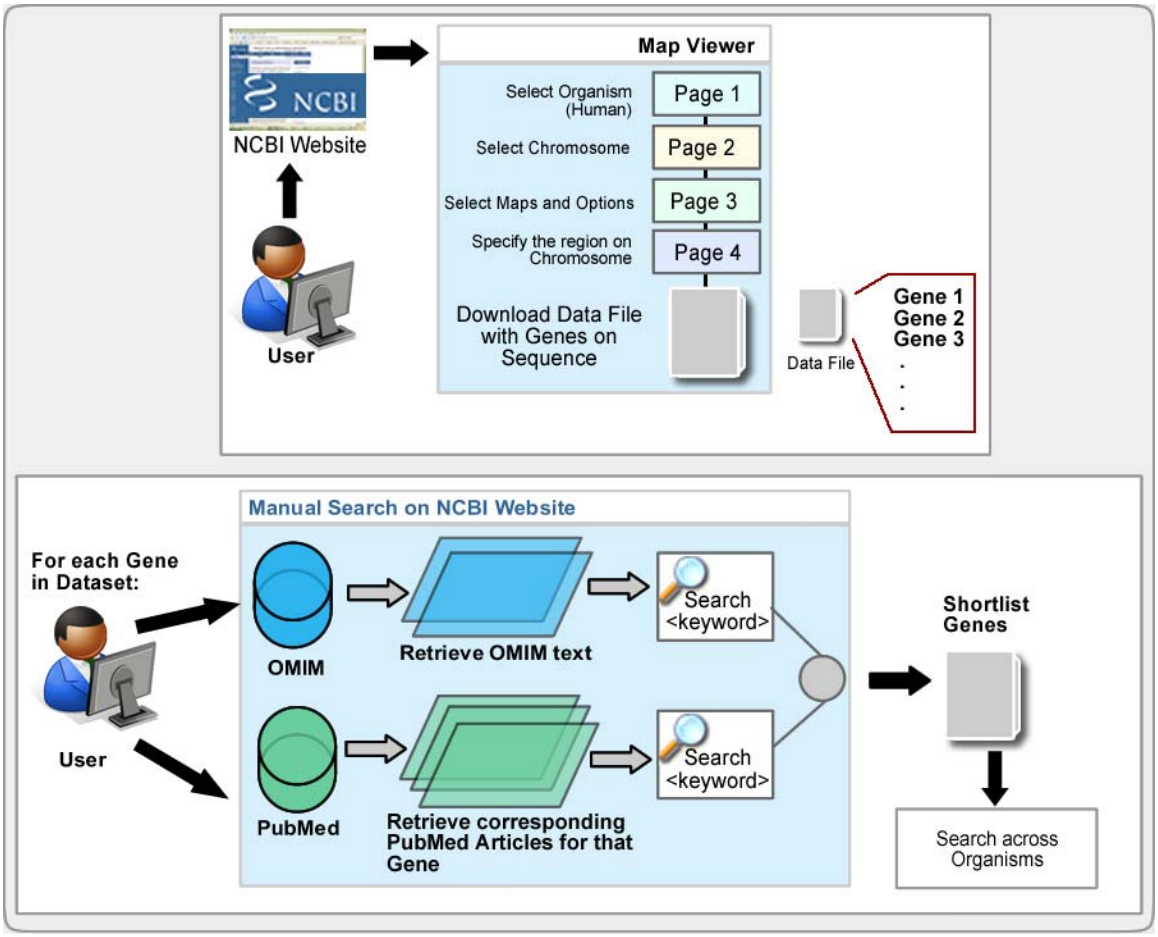


Fig. 22. Workflow of case study – Arnett lab.

The following modules elaborate in detail the tasks involved in the workflow of the case study – Arnett Lab:

- Module 1: Obtaining the set of genes in a region of interest. The first task in this process is to manually download a file from the NCBI Web site. This file is a list of genes in the specified region of a chosen chromosome. Downloading the file requires that the user must browse through several different pages, select specific options, and specify the chromosome region. Fig. 23 shows the various steps involved, with each step marked by a page transition (server request). The

researcher visits the NCBI Web site and uses a tool called Map Viewer. This tool shows integrated views of chromosome maps for many organisms, including human and numerous other vertebrates, invertebrates, fungi, protozoa, and plants [153]. The researcher uses this tool to obtain the chromosome maps and then to study the genes in a specific region of the chromosome. The user has to traverse eight steps before downloading the list of genes. However often this sequence is used, the researcher has to go through all of the steps, as there are no customization features. The appendix describes each step with a screenshot. The steps are as follows:

- Step 1: The user goes to the NCBI Web page, scrolls down, and clicks on the Map Viewer link.
- Step 2: On the next page, the user selects the organism that is under study, in this case Homo sapiens (build 36). A new page is loaded in the browser.
- Step 3: On this new page, the user selects the chromosome that is being investigated. The genes to be studied would be on a specific region on this chromosome.
- Step 4: The chromosome map is displayed on the next page. The user clicks on the “Maps and Options” button on this page.
- Step 5: A popup opens. The user makes the selection for the appropriate map options used to display the chromosome map.
- Step 6: Back in the chromosome map page, the user provides the region of choice in the fields provided.

  - Step 7: The chromosome map refreshes to display the map of the chosen region. On the same page, a link points to the page where the sequence can be downloaded. This link results in another page.
  - Step 8: On this page the link to the actual file is present. The user clicks on the file link to download the tab delimited file. This downloaded document is a flat file with a list of genes with corresponding columns of specific information. This document is a tab-delimited file and is most readable in spreadsheet applications, such as Microsoft Excel.

- Module 2: Obtaining the OMIM information. On studying this document downloaded from Map Viewer, the researcher identifies the genes that have associated OMIM information. OMIM is an NCBI database that lists all the known diseases and related genes in the human genome [154]. Each gene may have one or more OMIM IDs. Each OMIM entry is given a unique six-digit number called the MIM number, or in OMIM database terms, the OMIM ID. The OMIM ID uniquely identifies each OMIM entry. The researcher manually goes to the NCBI OMIM search interface and looks up the OMIM information for each gene in the flat file. The OMIM information is text-based and, in order to perform a keyword search, the browser's search feature must be used. Or, the complete text has to be copied into a file corresponding to the gene. The keyword search is then executed using the spreadsheet's search/find options. Either ways, it is extremely cumbersome and error prone.

- Module 3: Obtaining the PubMed Information. Similar to OMIM information, each gene in the list can be associated with one or more documents from PubMed, an NCBI database that stores citations for biomedical articles [153]. PubMed can be accessed via PubMed Central or Entrez. Entrez is a more sophisticated Web portal/search engine than NCBI and can be used to query many of the databases offered by NCBI. When the researcher types a gene into the search interface of PubMed, it results in a list of all of the documents from the database that cites that particular gene. Each document will have summaries or abstracts of the article. Once again when the researcher has to determine the significance of that gene in the study by using a key word search of the PubMed articles, it is done using the browser or by employing the copy-paste routine.
- Module 4: Searching the information. The OMIM and the PubMed information, associated with each gene, offer vital information for the researcher to identify candidate genes for further research. This search process is done by carrying out a keyword search on the retrieved PubMed and OMIM information. The genes that yield “hits” during the search are those genes that are identified as candidate genes. Other genes in the flat file are discarded.

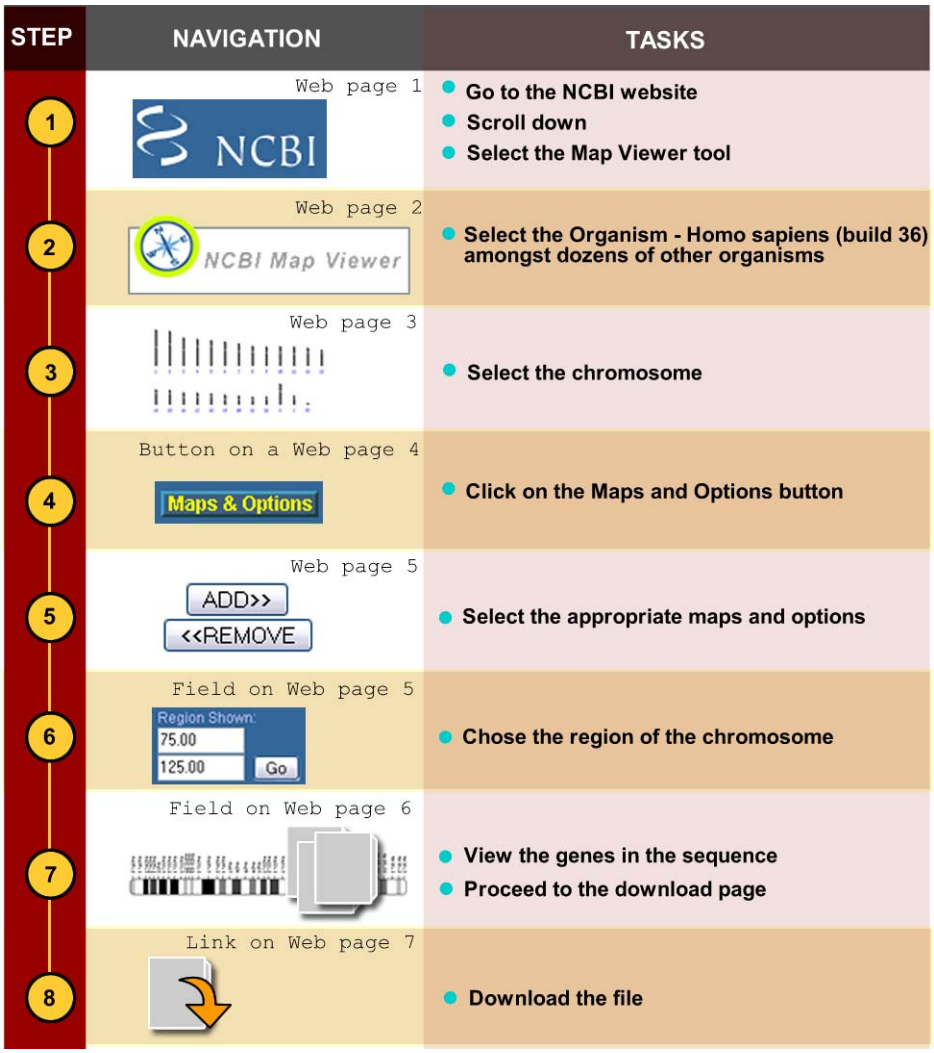


Fig. 23. Steps in a single task that requires the researcher to use the Web-based tool, MapViewer.

Although the techniques in method 2 (identifying genes in a region based on a key word search) are promising, the technical difficulties in implementing them have prevented them from becoming popular. To overcome these issues, adaptable, easy-to-use software that automatically identifies and retrieves genes located within a region of association or linkage, retrieves the PubMed citations for those genes for humans and other

species, and allows for a key-word-based search, would be invaluable to genetic researchers.

c) Need for automation/integration

The workflow described above uses data from multiple databases hosted by the NCBI. The databases are accessed through Web interfaces (Web pages). Although several APIs are available to the users of these databases, very few researchers make use of these APIs for their workflows. The workflow described in Section 1(b) of this chapter is just a part of the major workflow objective in which those chosen human genes are compared to conserved genes found in model organisms, such as rats and mice. The mice and rats databases are hosted independently and are not a part of NCBI. Therefore, there are no available APIs to integrate with the rat and mouse systems. This lack of proper interfaces present a great need for an integration models to exist that can connect independent databases to the users workflow. In addition, as evident from the workflow described in the previous section, the researcher has to peruse several different pages before eventually arriving at the final result. A tool that could personalize the information of the researcher and also allows the researcher to conserve time and resources would be of great use.

d) The Service-Oriented Dashboard approach

Fig. 24 illustrates the design of the SOD for the research workflow described. This workflow would eventually replace workflow steps 1-4. Fig. 24 does not depict the

other features of the SOD such as “personalization” as explained in the components of the SOD section.

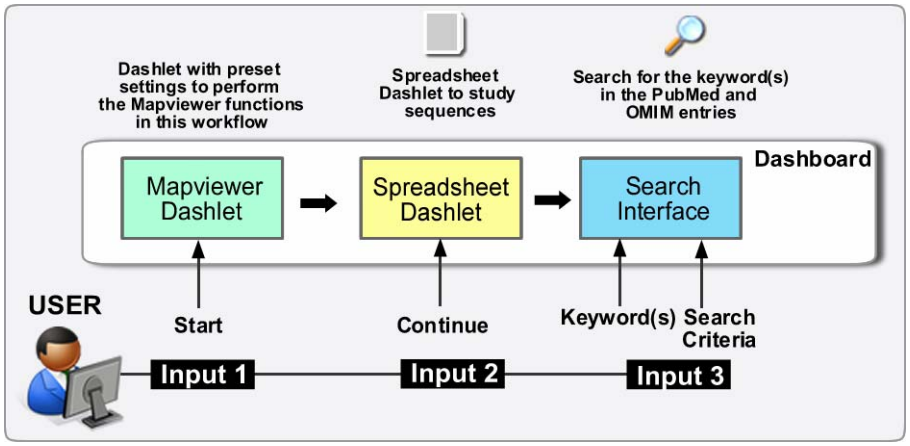


Fig. 24. Dashlets for the tasks described in the Arnett case study workflow.

The SOD used in this case has three dashlets. The first dashlet can be customized to hold preset values for the various parameters that the user would normally enter when using the Map Viewer. These parameters include the type of organism, the chromosome in the organism under study, the region in the chromosome and also the map or viewing preferences. Presetting values will save time for a researcher. At present, the dashlet just produces the genes on the sequence requested in a tab-delimited file format. This file can be exported to the spreadsheet dashlet. This allows the user to study the genes on the list. In future versions, the maps displayed by the Map Viewer could be displayed by the dashlet.

The next step involves searching the PubMed and the OMIM information corresponding to each gene in the list and storing the information. Using an intelligent keyword search on the stored information, the candidate genes are identified corresponding

to whether the PubMed or OMIM information yielded hits. The search dashlet is used for this purpose. This dashlet provides an interface, which has multiple search criteria that can be used to formulate advanced queries. Different queries based on assorted sets of keywords can produce several files. All of these documents can be retrieved from the file repository of the Dashboard. Besides providing intuitive user-interfaces, the SOD development for this case study provides the user an easy way of transparently transferring data from dashlet to another transparently. Using the SOD will be highly efficient in several ways. The following are a few of the advantages:

Using the Map Viewer on the Web site would make the user browse several Web pages with superfluous information. The dashlet could retrieve the same information in a fraction of the time as it is highly personalized.

The user does not have to go to the Web site to search the PubMed and OMIM information for each gene. This process is very challenging and could yield erroneous information. The Search dashlet uses the Web services offered by NCBI support for querying the PubMed and the OMIM databases and employs sophisticated queries to search the information [153]. Using a Dashboard for this process is not only time-saving but results in an improved user experience.

2) Case Study – Hartman Lab

Dr. John Hartman is an Assistant Professor in the Department of Genetics at UAB. His research focuses on genetic buffering of DNA replication. Budding yeast is the model organism used in his research. The Yeast Genetics Lab served as an ideal envi-

ronment for a case study because it could benefit greatly connectivity to external databases and improved integration within the laboratory.

a) Research

The objective of the lab is to systematically infer gene function from time series of data generated by observing the growth of different yeast knockout strains perturbed under different conditions [155]. The application of systems analysis to understanding the complex interactions between genetic and environmental variations on cellular phenotypes is called cellular phenomics. However, studying localized qualitative effects of single genes on phenotypes is not adequate to completely understand the complex interworking of pathways. In order to do so, one should be able to look at the quantitative effects of combinations of genes with other genes and environmental factors. Empowering such an extensive study for this experimental paradigm would require high-throughput capabilities. The Hartman lab is in the process of obtaining high throughput cellular phenotyping (HTCP) capabilities that can facilitate a more insightful, astute, and improved discernment of the genetic interaction networks. One of the ways to understand the gene interaction networks is through cellular array analysis, where individual cell cultures are under observation as opposed to genomic DNA. Currently, cellular array phenotyping are non-automated or semi-automated.

To fully utilize the potential of HTCP, enhanced instrumentation and improved data collection and analysis are very important. Additionally, HTCP results in large amounts of data. This would mean the data would require publishing and could be compared with data from other researchers or even need data from external resources such as

databases for further analysis. This system would therefore greatly benefit from integration within the laboratory and also with external databases, before adopting high-throughput capabilities.

b) Workflow

In general, the aim of the experimentation process is to know the degree to which the cell, as a system, depends on each genetic component for stability. This is determined by the extent of genetic interaction. Interactions are inferred from proliferation of the corresponding deletion strain (knock-out strain). Proliferation is the system output that is measured and describes the growth of the cell cultures. To determine the relevant (physiologic) range of effects for a given perturbation (e.g., a drug), growth of the reference strain (strain without deletion) is tested. The threshold of buffering capacity for the system is the minimum inhibitory concentration of the drug, and thus one can expect to begin to see many genetic interactions (i.e., non-additive effects of gene deletion on growth/proliferation) above this concentration. The research workflow in the Hartman lab is presented in this section. The workflow is described in the context of essential processes, data generated, applications and instruments used, and connections to external databases. The workflow is composed of the following steps:

- Development of a hypothesis: A hypothesis is a logical and intelligent conjecture as to how a process or research experiment would react to the relationship between two or more variables that influence the process. A hypothesis is tested by the process of experimentation. In the genetics lab, the researcher

develops the hypothesis based on studying related data or the data from a similar experimental process or simply by cognitive thinking.

- Design the experiment: Experimental design follows the development of a hypothesis. Design is biologically motivated. The researcher in the genetics lab, at this stage, selects the strains he or she would like to research further, and the perturbations (drugs) and environment variables that are to be used in the experiment. During this stage the researcher uses data from external resources such as OpenBiosystems and Saccharomyces Genome Database (SGD) to make more prudent design decisions. This data needs to be retrieved only when a new knock-out strain set is used; however, the external, retrieved data that is stored locally should be updated frequently.
- Experimentation: There are several sub-processes that take place during experimentation. The strains chosen during the design stage are transferred to solid agar plates after using an instrument called the plate reader. The agar plates contain the perturbations as well as the media that promotes the growth of the yeast strains. This process is termed “printing.” The printed strains on the plates are called cellular arrays. The printing of the cellular arrays is carried out by the liquid handling system by depositing droplets of dilute cell suspensions on the agar plates. Also used is the automated plate handling system that stores and handles the plates used in printing. Once the printing process is complete, these agar plates are stored in an incubator at about 30 degree Celsius for about 15 hours. During this period the yeast cell cultures (spot) on the cellular array are rapidly multiplying, meaning that their colonies would

become increasingly visible over time. This proliferation is the phenotype that is measured in the data collection process. There are three instruments used during this part of the workflow. A software application is used to interact with the system that integrates the instruments and also controls and records the data that is used by the system. Microsoft Excel is another application that is used during the experimentation process, primarily to store data from the plate reader.

- Data collection: During data collection, the plates are removed from the incubator and imaged in sets of ten. A scanner is used for the imaging, and a shadow rather than the reflection of the colonies are obtained. The measured quantity is intensity of color. Darker areas indicate higher proliferation. The plates are returned to the rack, maintaining the same order, and the next batch of ten plates is processed similarly. The systematic arrangement of the plates for each scan, together with the ordering of the scan sequence, is used for tracking all of the information. This process is repeated about twenty times at an interval of one to two hours to get a series of scans or images. The effect of perturbation over the knockout strains is evident by the varying intensities of the spots on a plate and the varying rate of intensities of the spots over time. The scans (a single scan captures 10 plates) are stored in folders according to the time of scanning. Fig. 25 is an image of one single scan of ten plates taken at three different time points to demonstrate the cell proliferation over time. An image scanning application is used to interface with the scanner. This application also names these scan images sequentially. Microsoft Excel is used

in this phase to create a template file that is used in the data analysis stage. This file basically relates the numerical intensity values to the strain information corresponding to each spot in a plate. An image viewer application is used to view the images for abnormalities, and any aberrations are noted in the Microsoft Excel template.

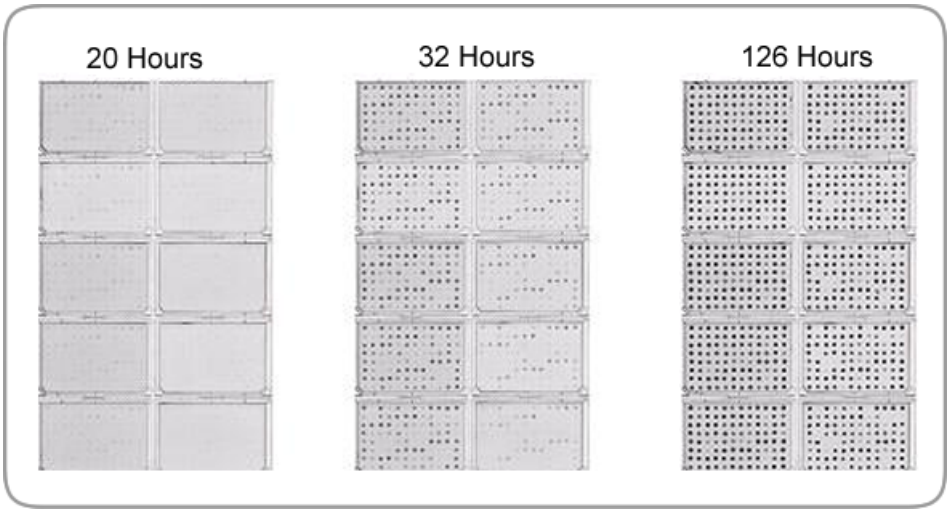


Fig. 25. Images of scans of cellular arrays capturing the growth of yeast colonies over time.

- Data analysis: After the imaging process, the scans of the plates that substantiate cell proliferation have to be analyzed to obtain numerical data. The scans require a bit of preprocessing before they are fed to an image analysis program. Along with the images, the program also needs information that correlates the position of the strains on scans to the actual knockout strains. The program then calculates the intensity values of each spot over time. The program produces a text document containing the intensity values over time for

each knockout strain. Currently the image analysis programs are in JAVA and MATLAB. The output of the image analysis programs is printed in the Excel template that was created in the image collection phase.

- Data Modeling: This step involves deriving results after analyzing the numerical data produced during the data analysis procedure. The numerical data corresponds to the intensity of the image of the culture, which directly signifies cell proliferation. This phase is crucial towards interpreting results. MATLAB programs are used to model the numerical data into comprehensible information in the form of graphs using certain special functions. Growth curves are graphs that show how the cell cultures proliferate over time. Growth curves are plotted with the intensities of the cell culture spot over time. Growth curves can be produced and analyzed in several different modes based on different characteristics. We refer to these growth characteristics as cell proliferation phenotypes (CPP), because they are all related to the growth curves, but each has distinct biological implications. Some of the CPPs are the area under the growth curve (AUGC), the maximum specific growth rate (MSR), final population intensity (FPI), and lag. AUGC represents an overall measure of growth. MSR is represented as the rate of proliferation divided by the population size. The efficiency is the FPI, which is an asymptotic value. The lag refers to the time that it takes for cells to achieve their maximum growth rate. We use MATLAB and Microsoft Excel during this period.
- Evaluation of results: The researcher conducting the experiment evaluates the validity of the data (images) and also the results produced as the output of the

experiment. The graphs obtained at the modeling stage also aid in the substantiation of results. This phase also helps in rejecting, enhancing, or developing a new hypothesis based on the significance of the findings.

Any data that leads to a discovery or data of some significance is published. This allows the researcher to establish the discovery and allows other researchers to use the data for further research or even in the creation of newer hypotheses. Currently numerous online journals are available for that purpose. Some online resources are akin to a central archive of many journals and magazines. PubMed is one such central resource. These repositories are also used to validate the data of a researcher and can also be used in the comparison stage. Some research labs prefer to compare results from other similar research. This particular step toward the process of discovery is up to the researcher's discretion. However, in special areas of research such as genetics, it is highly desired. Several laboratories dispersed around the world have created repositories of their data.

Fig. 26 illustrates the workflow in terms of the various instruments and software applications used. It is evident that the workflow that the user employs operates on several applications on the computer. This set of applications includes Web sites, Web interfaces that interact with external databases, instrument control software, Microsoft Excel, MATLAB and JAVA programs, and applications that facilitate the viewing of images.

c) Need for the proposed architecture/integration

The workflow of many laboratories such as the Hartman lab typically requires the manual passing of data through several programs in sequence at each phase. Fig. 26 depicts the various tools and instruments used in each phase of the workflow used in the

Hartman lab. Every time data is delivered from one application to another, it needs to be formatted to conform to application-dependent file formats. Often, manual transfer of results between the different applications is done by a copy-and-paste routine. Although problematic and error-prone, this approach has facilitated scientific exploration through experimentation with different hypotheses using different services [156].

To visualize the real scope for integration, we have to study the different conduits of data in the genetics laboratory. Based on our study, we identified three principal areas that could benefit from advances in integrations. These principle areas are as follows:

- Integration of the assorted instruments in the lab: The laboratory uses various instruments during the course of experimentation, such as the liquid handling system, the plate handling system, the microplate reader, and the scanner. Each instrument has various automated and manual steps for data collection. Generally, instrument vendors utilize their custom-built software to control the instruments and to handle data flow between instruments in research laboratories. These custom integration applications are usually stand-alone systems. In order to extend the integration capabilities to encompass other laboratory systems, it is essential for the user/integrator to study and understand the vendor software before writing extensive programs and methods to achieve the desired level of integration. In the Hartman lab, a single software application is used for controlling the microplate reader and the liquid and plate handling systems. The software also manages the data used and produced by the instruments, but does not evaluate or model data.

- Integration of software applications used in the workflow: The software that is used for image analysis is different from the modeling program, both in the software language used in development and in the architecture, so it is not easy to integrate these programs to work seamlessly without human interaction. Currently the output data from the image analysis software requires some preprocessing before it can be transmitted as input to the modeling software. One solution could be to write another program that would do the image analysis, the preprocessing for the input for the modeling program, and the modeling, but the process of completely rewriting an application to replace three existing applications would only prolong the automation efforts. This type of integration predicament can exist in many laboratories when several stand-alone applications are used. Data needs to be converted from one format to another between applications, and human interaction may be required during the process, so the rewriting of existing applications is overkill.
- Integration of data handled during the workflow: The user manages information at all stages of the workflow. At a given instant, the user performs at least one of the following data transactions:
    - Seeks data from external sources.
    - Stores data produced within the lab.
    - Publishes the data produced in the lab.

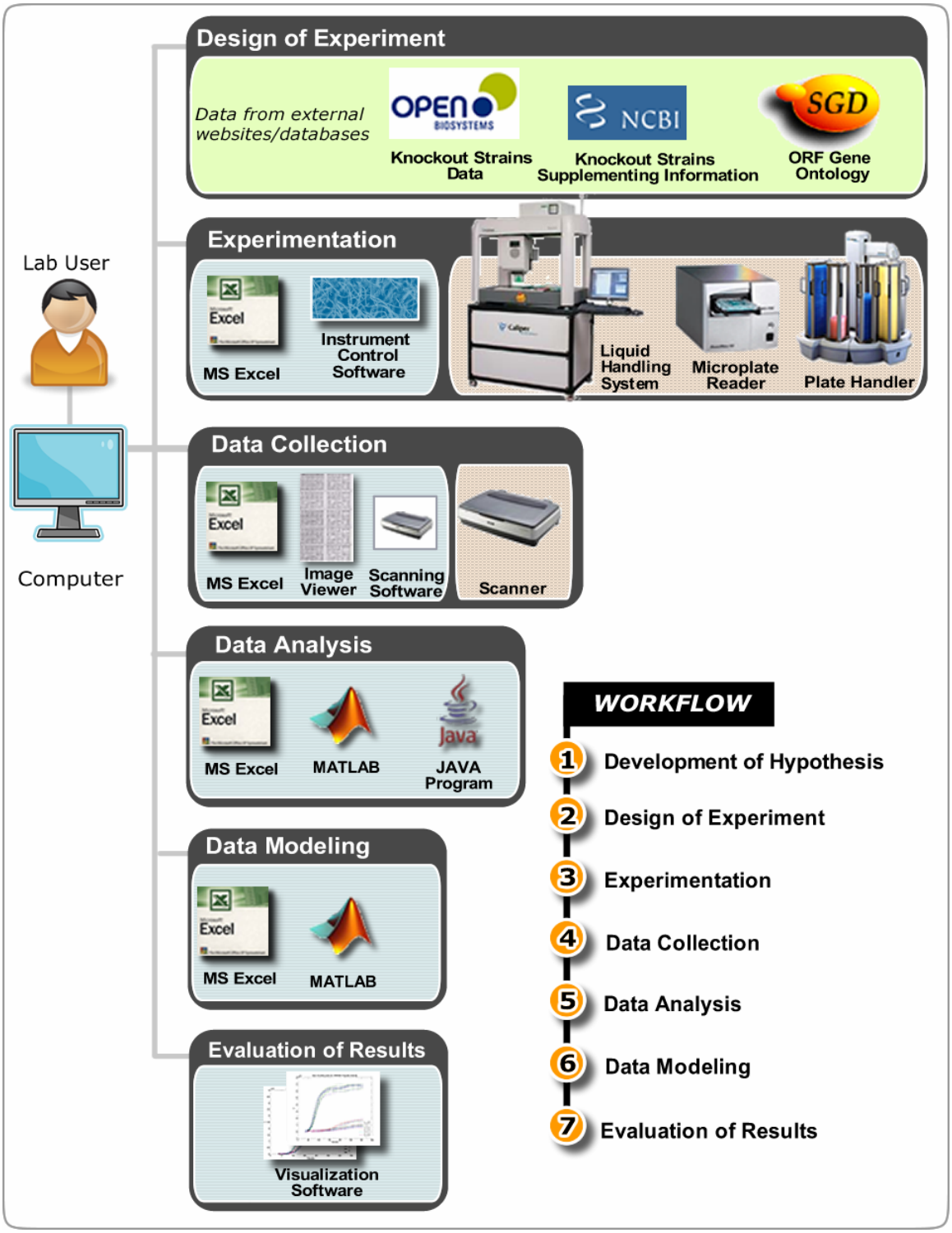


Fig. 26. The various software applications and instruments used in the genetics laboratory.

The researcher in the Hartman laboratory would wish to connect to an external data sources several times. The data from these sources may be obtained either through a Web interface, by direct download from a Web site, or by FTP. Each time, it is possible that the interfaces are different, and the researcher is required to know the process of obtaining the data from a source. This could be the process of traversing across multiple Web pages or operating an FTP. External resources can be databases such as the Saccharomyces Genome Database (SGD), Gene Ontology (GO), or Ensembl, and tools such as BLAST to augment analysis of the primary data. Data integration with external resources will present opportunities for innovation that will be synergistic for new breakthroughs in understanding complex biological processes.

Research laboratories produce a great deal of data during experimentation. The data from the genetics lab are in diverse formats and are produced at different stages. For example, the scans or images of the arrays are one type of data, while the intensity values produced as the result of image analysis can be another, and the resultant modeling graphs that are used during the evaluation phase are a totally different kind of output. However, data or results in all of the formats must be recorded efficiently. Once an experiment is complete, the data is usually published. This stage could be an extension of the storage phase. Data and results can be stored effectively in such a way that they can be retrieved by personnel within the lab or even an external user.

d) The Dashboard Approach

Several laboratories use lab information management systems (LIMS) to manage and process the lab's data. We envision the dashboard as a personalized, service-enabled,

RIA for interacting with the LIMS system to provide a user-friendly and comprehensive integration solution. In Sundar *et al.*, we describe the shortfalls of a traditional LIMS and introduce the intelligent LIMS (iLIMS) [114]. iLIMS is based on SOA, and the paper presents a Web services approach to integration.

In the previous section, we categorized various areas of the genetics lab that can be enhanced by integration. In this section we demonstrate how a dashboard system could provide a solution to various aspects of integration, such as integration of external data sources, instruments and tools in the lab, and software applications used in workflows.

- Integration of external and internal databases: Integration of data from multiple databases, especially in the field of bioinformatics, has been an area of active research recently. These biological databases use different data formats and different approaches to store the data. Therefore, the issue of integrating them remains an important challenge [157]. Examples of such databases are Ensembl, PubMed, UCSC Genome Browser, FlyBase, WormBase, and GeneOntology. There are numerous possibilities for integration of biological resources [134]-[137]. Here we consider a Web services approach to the creation of iLIMS [117], [139], [141], [158]-[160]. The use of Web services is particularly advantageous because many bioinformatics data sources already provide Web services compatible interfaces, such as MyGrid and BioMoby [136], [141], [142].
- Integration of the robotic instruments in the lab: Integrating the iLIMS with analytical instruments and other applications in the laboratory is another important component of LIMS development [160]-[162]. There is a fundamental

issue regarding the development of iLIMS. As a result of third-party instrument integration products, these products have traditionally been designed for single PC operation and consequently struggle with today's networked operations [162]. This puts constraints on both the flexibility and scalability of hardware and software solutions. Modern instrument integration solutions need advanced data mapping and parsing capabilities to process data from disparate instrument types and multiple vendors, thus eliminating the need for separate interfaces for each instrument system. The literature shows evidence of existing Web services infrastructure for instrument integration [160]-[162]. They also describe an XML-based format for handling data from instruments. This metadata can be used in the SOA that underlies the LIMS. The iLIMS can leverage these approaches to establish the component for instrument integration.

- Workflow Composition and Process Management: Many competing composition languages exist in the Web services model to execute workflows. The most popular orchestration language is BPEL4WS. Another language is the WSCI approach proposed as a W3C standard by Sun, BEA Weblogic and other partner companies. Stabb *et al.* compare BPEL4WS, WSCI, and many of the other orchestration approaches that currently exist [124]. Process management techniques can be developed on top of the BPEL4WS layer for monitoring the workflows [125].
- Tools-as-Services (TAS): Recognizing TAS and SET offers a promising approach to developing a dashboard model for the iLIMS or as a stand-alone

dashboard system. The significance of TAS is illustrated with the use of the workflow in the genetics lab. Based on the tools used, the Hartman lab workflow can be described as a combination that is inclusive of the four processes – experimental design, image analysis, modeling, and biological analysis/evaluation – that are associated with the tools Microsoft Excel, MATLAB-based image analysis, MATLAB-based modeling, and Java applications. In order to perform a complete experiment, the experimenter moves data back and forth between these multiple tools. This process could be highly error-prone and inefficient.

An ideal user-environment would allow the experimenter to access all lab resources from a single interface. To complete a task such as modeling, the researcher uses Microsoft Excel and MATLAB for processing. Microsoft Excel has the data that MATLAB uses to produce curves. The curves are generated using special functions from MATLAB. This process requires the researcher to be acquainted with MATLAB. Instead of moving from one tool to another, if the functions of MATLAB could be used in Excel, the process in much simplified and the users don't have to know special tools or applications. Fig. 27 illustrates this concept of TAS and SET. This model would provide significant advantages over traditional Web pages approach. MATLAB has a component called MATLAB® Builder for Excel®. This component enables one to make use of MATLAB's sophisticated functions from the familiar environment of Microsoft Excel [163]. The MATLAB functions are deployed as Excel add-ins that can be executed when using Excel. Excel® Link 3.0 from MATLAB is another component of MATLAB that has similar features [164].

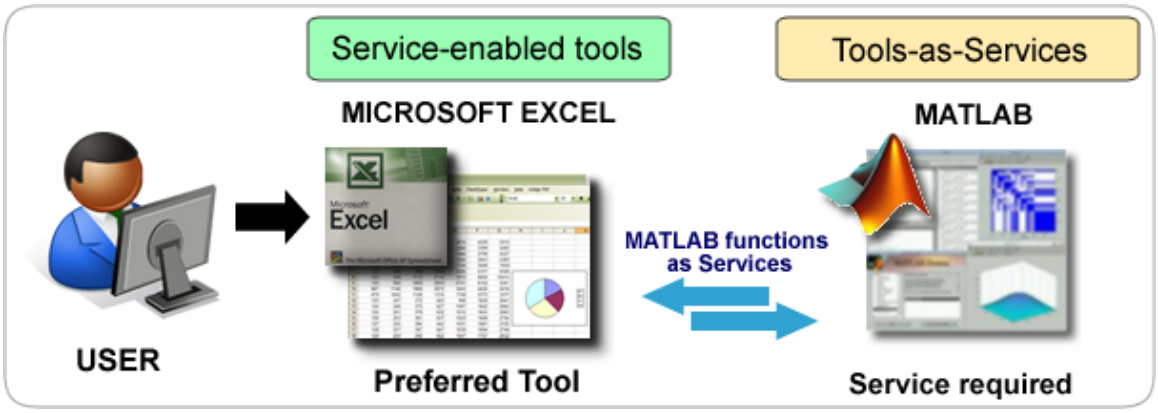


Fig. 27. MATLAB function as a service in Microsoft Excel.

C. Conclusion

In Chapter III, we presented the design of the SOD. In this chapter, we used two cases from the bioinformatics domain as case studies. The case studies were conducted in two laboratories. We described the laboratory's research background, workflow, and the need for integration techniques. In this chapter, we also demonstrated how the workflow of the laboratories could be enhanced using the SOD.

V. CONCLUSION

A. Discussion

EAI involves the integration of numerous individual applications within an enterprise and across enterprises [1]. EAI involves integration at three logical layers of application architecture [165]. These layers are the data logic layer, business logic layer, and presentation logic layer. Currently, portals are the choice for providing an integrated user-interface for enterprise applications [166]-[168]. Enterprise portals also function as an integrated environment at the presentation layer. In order to provide enhanced user experiences, RIAs are used as applications on the portal. However, as we indicated in Chapters III and IV, it is not possible to address certain complex interactions on the portal. In this thesis, we presented the architecture and design of the SOD to show how the service-based approach enables us to design interactive, composite environments. The SOD enhances the current interactive application development and extends an existing SOA and related technologies.

B. Contribution

This thesis presents the design of the SOD. In Chapter II, we discussed the current technologies in use to present and develop service-oriented and RIA. We presented the design of the SOD in the third chapter. In Chapter III, we introduced two notions towards

the design of the SOD. The TAS and SET model offers an approach to publish tools and functions of applications as Web services. This technique also enables applications to consume the published services. We also presented several existing examples that depict this approach.

In the design of the SOD, we also highlighted the process of using task-based composition for composing services to introduce composite services as applications or dashlets on the SOD. In addition to the task composition aspect of the SOD, we addressed some of the information and interaction design features of the dashboard.

In order to indicate how the SOD can enhance enterprise workflows, we used two case studies in the bioinformatics domain. The case studies were conducted in laboratories that deal with extensive data collection/analysis and complex workflows. Therefore, the case studies helped in drawing requirements for the design of the SOD and served as an environment to demonstrate the effects of the SOD.

C. Future Work

In the implementation of the SOD model, it is necessary to address several issues including authentication, personalization, and semantic mediation. We believe that the following directions of development offer opportunities of further research:

- Single-Sign on – Single sign-on refers to the ability of the users to authenticate once at one tool and then gain access to the other tools in the environment [169], [170].
- Tool registry – The SOD design involves the use of an application library. This library is composed of composite applications that have tools published

as services. The SOD design should enable users to discover the different TAS applications with ease.

- Data transparency - Users must be able to operate on the data transparently while moving from one tool to another. The SOD could host several tools to compose a task. In order to make the transition from one application to another seamless, it is important to consider data mediation. Data mediation could be achieved using application ontology. The ontology can offer seamless mapping between the services of one application and another application.

The SOD system developers should also address issues such as security, reliability, and integrity.

LIST OF REFERENCES

[1] G. Samtani and D. Sadhwani. (2001). EAI and Web services. Easier enterprise application integration? [Online]. Available: http://www.webservicesarchitect.com/content/articles/samtani01print.asp

[2] D.S. Linthicum, *Enterprise Application Integration.* Reading, MA: Addison-Wesley, 2000.

[3] P. Herzum and O. Sims, *Business Component Factory: A Comprehensive Overview of Component-Based Development for the Enterprise.* New York, NY: John Wiley, 2000.

[4] F. Losavio, D. Ortega, and M. Perez, "Modeling EAI," in *Proc. 22nd Int. Conf. of the Chilean Comput. Sci. Soc., SCCC,* 2002, pp. 195-203.

[5] A.A. Mosawi, L. Zhao, and L. Macaulay, "A model driven architecture for enterprise application integration," in *Proc. 39th Annu. Hawaii Int. Conf. on Syst. Sci., HICSS '06,* 2006, pp. 181c.

[6] T. Erl, *Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services.* Prentice Hall PTR, 2004.

[7] F. Leymann, D. Roller, and M.-T. Schmidt, "Web services and business process management," *IBM Syst. J.*, vol. 41, no. 2, pp.198-211, 2002.

[8] A. Stone, "Demanding Internet enterprise," *IEEE Internet Comput.*, vol. 8, no. 3, pp. 13-14, 2004.

[9] W.T. Tsai, "Service-oriented system engineering: a new paradigm," in *Proc. IEEE Int. Workshop on Service-Oriented Syst. Eng., SOSE'05,* 2005, pp. 3-6.

[10] E. Anuff. (2004). WSRP and the enterprise portal - A strong, and important, start [Online]. Available: http://soa.sys-con.com/read/44674.htm

[11] O. Diaz and I. Paz, "Turning Web applications into portlets: raising the issues," in *Proc. Symp. on Applicat. and the Internet,* 2005, pp. 31-37.

[12] C. Wege, "Portal server technology," *IEEE Internet Comput.*, vol. 6, no. 3, pp. 73-77, 2002.

[13] P. Lunt. (2002). Portals open doors to processes [Online]. Available: http://www.providersedge.com/docs/km_articles/Portals_Open_Doors_to_Processes.pdf

[14] T. O' Reilly. (2005). What is Web 2.0: design patterns and business models for the next generation of software [Online]. Available: http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html

[15] T. Noda and S. Helwig. (2005). Rich Internet Applications - technical comparison and case studies of AJAX, Flash, and Java based RIA [Online]. Available: http://www.uwebc.org/opinionpapers/archives/docs/RIA.pdf

[16] L.D. Paulson, "Building rich Web applications with AJAX," *IEEE Computer*, vol. 38, no. 10, pp. 14-17, 2005.

[17] S. Few, *Information Dashboard Design: The Effective Visual Communication of Data.* Cambridge, MA: O'Reilly, 2006.

[18] J. Ganesh and S. Anand, "Web services, enterprise digital dashboards and shared data services: a proposed framework," in *Proc. 3rd European Conf. on Web Services, ECOWS'05*, 2005, pp. 8.

[19] Merriam-Webster, Inc., *Merriam-Webster's Collegiate Dictionary*, 11th ed. Springfield, MA: Merriam-Webster, Inc., 2003.

[20] R. Sadasivam, M. Tanik, and L. Jololian, "Drag-and-Drop communication of data via a computer network," U.S. Patent, 2005.

[21] Yahoo, Inc. (2007). Finance portal [Online]. Available: http://finance.yahoo.com/

[22] K.H. Buetow, "Cyberinfrastructure: empowering a 'third way' in biomedical research," *Science*, vol. 308, no. 5723, pp. 821-824, May 6, 2005.

[23] P. Cornell. (2002). Checking stock quotes over the Web using the Web service references tool and Microsoft Excel [Online]. Available: http://msdn2.microsoft.com/en-us/library/aa140265(office.10).aspx

[24] K.J. Lin, "Building Web 2.0," *IEEE Computer*, vol. 40, no. 5, pp. 101-102, 2007.

[25] J. McGovern, *A Practical Guide to Enterprise Architecture.* Upper Saddle River, NJ: Prentice Hall PTR, 2004.

[26] A. Umar, *Object-Oriented Client/Server Internet Environments.* Upper Saddle River, NJ: Prentice Hall PTR, 1997.

[27] D. Georgakopoulos, M. Hornick, and A. Sheth, "An overview of workflow management - from process modeling to workflow automation infrastructure," *Distributed and Parallel Databases*, vol. 3, no. 2, pp. 119-153, Apr. 1995.

[28] S. Hashimi. (2003). Service-oriented architecture explained [Online]. Available: http://ondotnet.com/pub/a/dotnet/2003/08/18/soa_explained.html

[29] C. Nelson and J.S. Kim, "Integration of software engineering techniques through the use of architecture, process, and people management: An experience report," *Rapid Integration of Software Eng. Technique*, vol. 3475, pp. 1-10, 2005.

[30] I. Jacobson, *Object-Oriented Software Engineering: A Use Case Driven Approach.* Reading, MA: ACM Press, Addison-Wesley, 1992.

[31] S.R. Schach, *Object-Oriented Software Engineering,* 1st ed. Dubuque, IA: McGraw-Hill, 2007.

[32] M.M. Tanik, P. Ng, and R.T. Yeh, “Electronic enterprise engineering – A 21st century discipline for a new generation of engineers,” *NJIT Research*, pp. 6-11, 1997.

[33] W. Kozaczynski and G. Booch, “Component-based software engineering,” *IEEE Softw.*, vol. 15, no. 5, pp. 34-36, 1998.

[34] E.J. Weyuker, “Testing component-based software: a cautionary tale,” *IEEE Softw.*, vol. 15, no. 5, pp. 54-59, 1998.

[35] A.W. Brown and K. Short, “On components and objects: the foundations of component-based development,” in *Proc. 5th Int. Symp. on Assessment of Software Tools and Technologies*, 1997, pp. 112-121.

[36] A. Brown, S. Johnston, and K. Kelly, "Using service-oriented architecture and component-based development to build Web service application," Rational Software Corp., White paper, 2002, Available: http://www-106.ibm.com/developerworks/rational/library/content/03July/2000/2169/2169.pdf

[37] G. Booch, “Coming of age in an object-oriented world,” *IEEE Softw.*, vol. 11, no. 6, pp. 33-41, 1994.

[38] E.R. Harold and W.S. Means, *XML in a Nutshell*, 3rd ed. Sebastopol, CA: O'Reilly, 2004.

[39] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana, “Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI,” *IEEE Internet Comput.*, vol. 6, no. 2, pp. 86-93, 2002.

[40] G. Alonso, *Web Services: Concepts, Architectures, and Applications.* New York, NY: Springer, 2004.

[41] W3C. (2007). Web Services [Online]. Available: http://www.w3.org/TR/2007/WD-ws-policy-primer-20070928/

[42] W3C. (2001). Web Services Description Language [Online]. Available: http://www.w3.org/TR/wsdl

[43] A. Harrison and I.J. Taylor, “WSPeer – An interface to Web service hosting and invocation,” in *Proc. 19th IEEE Int. Parallel and Distributed Process. Symp.*, 2005, pp. 175a.

[44] E. Newcomer and G. Lomow, *Understanding SOA with Web Services.* Upper Saddle River, NJ: Addison-Wesley, 2005.

[45] J. Pasley, “How BPEL and SOA are changing Web services development,” *IEEE Internet Comput.*, vol. 9, no. 3, pp. 60-67, 2005.

[46] S. Ortiz Jr, “Getting on board the enterprise service bus,” *IEEE Computer*, vol. 40, no. 4, pp. 15-17, 2007.

[47] P.P. Mike and H. Willem-Jan, “Service-oriented architectures: approaches, technologies and research issues,” *The VLDB J.*, vol. 16, no. 3, pp. 389-415, 2007.

[48] S.Y. Hashimi and S.J. Steffan, *Pro Service-Oriented Smart Clients with .NET 2.0.* Apress, 2005.

[49] J. Offutt, “Quality attributes of Web software applications,” *IEEE Softw.*, vol. 19, no. 2, pp. 25-32, 2002.

[50] M. Cooper, “Accessibility of emerging rich web technologies: Web 2.0 and the semantic Web,” in *Proc. 2007 Int. Cross-Disciplinary Conf. on Web Accessibility, W4A*, Banff, Canada, 2007, pp. 93-98.

[51] B. Gibson, “Enabling an accessible Web 2.0,” in *Proc. Int. Cross-Disciplinary Conf. on Web Accessibility, W4A*, Banff, Canada, 2007, pp. 1-6.

[52] T. O' Reilly et. al. (2005). Not 2.0? [Online]. Available: http://radar.oreilly.com/archives/2005/08/not_20.html

[53] R. MacManus and J. Porter. (2005). Web 2.0 for Designers [Online]. Available: http://www.digital-web.com/articles/web_2_for_designers/

[54] A. Budd. (2005). What is Web 2.0 [Online]. Available: http://www.andybudd.com/presentations/dcontruct05/

[55] Gartner. (2007). Gartner warns Web 2.0 will force business to re-examine approach to IT security [Online]. Available: http://www.gartner.com/it/page.jsp?id=511944

[56] S. Murugesan, “Understanding Web 2.0,” *IT Professional*, vol. 9, no. 4, pp. 34-41, 2007.

[57] J.J. Garrett. (2005). Ajax: A new approach to Web applications. [Online]. Available: www.adaptivepath.com/publications/essays/archives/000385.php

[58] Security Space. (2007). Technology penetration report [Online]. Available: http://www.securityspace.com/s_survey/data/man.200705/techpen.html

[59] W3C. (2002). XHTML™ 1.0 The Extensible HyperText Markup Language [Online]. Available: http://www.w3.org/TR/xhtml1/

[60] W3C. (1999). XSLT [Online]. Available: http://www.w3.org/TR/xslt

[61] W3C. (1999). XPath [Online]. Available: http://www.w3.org/TR/xpath

[62] CNET News. (1995). Netscape and Sun unveil JavaScript [Online]. Available: http://news.com.com/Netscape+and+Sun+Unveil+JavaScript/2100-1023_3-278832.html

[63] W3C. (2007). XMLHttpRequest Object [Online]. Available: http://www.w3.org/TR/XMLHttpRequest/

[64] J.J. Garrett, *The Elements of User Experience: User-Centered Design For the Web*, 1st ed. Indianapolis, IN: New Riders, 2002.

[65] Google, Inc. (2007). Google Maps [Online]. Available: http://maps.google.com/

[66] Y. Roodt, “The effect of AJAX on performance and usability in Web environments,” M.S. thesis, Dept. Software Eng., Universiteit van Amsterdam, 2006.

[67] M. Kiesler. (2006). Round-up of 50 AJAX toolkits and frameworks [Online]. Available: http://www.maxkiesler.com/index.php/Weblog/comments/round_up_of_50_ajax_toolkits_and_frameworks/

[68] J. Allaire, "Macromedia Flash MX - A next-generation rich client," Macromedia/Adobe, White paper, 2002, Available: http://www.adobe.com/devnet/flash/whitepapers/richclient.pdf

[69] S. Webster and A. McLeod, *Developing Rich Clients with Macromedia Flex.* Berkeley, CA: Macromedia Press, Peachpit Press, 2005.

[70] A. Patrizio. (2007). Does JavaFX spell the end of AJAX? [Online]. Available: http://www.internetnews.com/dev-news/article.php/3676226

[71] C. Sells and I. Griffiths, *Programming Windows Presentation Foundation,* 1st ed. Sebastopol, CA: O'Reilly, 2005.

[72] L.A. MacVittie, *XAML in a Nutshell,* 1st ed. Sebastopol, CA: O'Reilly, 2006.

[73] D. Merrill. (2007). Mashups: The new breed of Web app [Online]. Available: http://www.masternewmedia.org/news/2007/08/09/mashups_what_are_they_mashup.htm

[74] O. Lassila and J. Hendler, "Embracing 'Web 3.0'," *IEEE Internet Comput.*, vol. 11, no. 3, pp. 90-93, 2007.

[75] R.F. Ingbert, M.C. Jones, R. Dinesh, and B.T. Michael, "Web mash-ups and patchwork prototyping: user-driven technological innovation with Web 2.0 and open source software," in *Proc. 40th Annu. Hawaii Int. Conf. on Syst. Sciences HICSS*, 2007, pp. 86-86.

[76] Y. Zhang, K.-J. Lin, and J.Y.J. Hsu, "Accountability monitoring and reasoning in service-oriented architectures," *Service-Oriented Comput. and Applicat.*, vol. 1, no. 1, 2007, pp. 35-50.

[77] UKOLN. (2007). JISC Information Environment Architecture [Online]. Available: http://www.ukoln.ac.uk/distributed-systems/jisc-ie/arch/

[78] JAVA.net. (2007). JSR 168 FAQ [Online]. Available: http://wiki.java.net/bin/view/Portlet/JSR168FAQ

[79] IatekLLC. (2006). Blog: Portal vs. Website [Online]. Available: http://www.aspapp.com/Difference-between-Portal-and-Website~

[80] Google, Inc. (2007). Google homepage [Online]. Available: http://www.google.com/ig

[81] Yahoo, Inc. (2007). Yahoo homepage [Online]. Available: http://www.yahoo.com/

[82] C. Braun, J. Broberg, M. Cassidy, M. Freedman, T.N. Jones, T. Schaeck, and G. Tayar. (2003). Web services for remote portlets specification [Online]. Available: http://www.oasis-open.org/committees/download.php/3343/oasis-200304-wsrp-specification-1.0.pdf

[83] Wikipedia. (2007). JSR 168 [Online]. Available: http://en.wikipedia.org/wiki/JSR168

[84] Microsoft. (2007). Sharepoint [Online]. Available: http://www.microsoft.com/sharepoint/server/downloads/webparts/introduction.asp

[85] OASIS. (2004). WSRP FAQ [Online]. Available: http://www.oasis-open.org/committees/download.php/11774/wsrp-faq-draft-0.30.html#jsr168

[86] SUN Developer Network. (2007). Introducing Java portlet specifications: JSR 168 and JSR 286 [Online]. Available: http://developers.sun.com/prodtech/portalserver/reference/techart/jsr168/index.html

[87] M. Pawlak, C. Bryce, and S. Lauri`ere. (2007). A project reference model for Environment for the development and distribution of open source software [Online]. Available: http://www.edos-project.org/xwiki/bin/download/Main/D5-5-3/EDOS-D5.5.3.pdf

[88] A. Marcus, "Dashboards in your future," *Interactions*, vol. 13, no. 1, pp. 48-60, 2006.

[89] H.J. Watson, R.K. Rainer, Jr., and C.E. Koh, "Executive information systems: a framework for development and a survey of current practices," *MIS Quart.*, vol. 15, no. 1, pp. 13-30, 1991.

[90] P. Chowdhary, T. Palpanas, F. Pinel, S.-K. Chen, and F.Y. Wu, "Model-driven dashboards for business performance reporting," in *Proc. 10th IEEE Int. Enterprise Distributed Object Comput. Conf., EDOC 06*, 2006, pp. 374-386.

[91] A. Granebring and P. Revay, "Service-oriented architecture is a driver for daily decision support," *Kybernetes*, vol. 36, no. 5/6, pp. 622-635, 2007.

[92] T. Palpanas, P. Chowdhary, G. Mihaila, and F. Pinel, "Integrated model-driven dashboard development," *Inform. Syst. Frontiers*, vol. 9, no. 2-3, pp. 195-208, 2007.

[93] A. Vasiliu, "Dashboards and scorecards: essential tools for managing business performance," DM Review, Direct Special Rep., 2006, Available: http://www.dmreview.com/editorial/newsletter_article.cfm?articleId=1051851

[94] W.W. Eckerson, "Deploying dashboards and scorecards," TDWI, Best Practices Report, 2006, Available: http://www.tdwi.org/Publications/WhatWorks/display.aspx?id=8206

[95] Microsoft. (2000). MS digital dashboard business process assessment guide [Online]. Available: http://www.microsoft.com/technet/archive/itsolutions/intranet/plan/bpag.mspx?mfr=true

[96] R. Bose, "Understanding management data systems for enterprise performance management," *Ind. Manage. & Data Syst.*, vol. 106, no. 1, pp. 43-59, 2006.

[97] M.B. Morgan, B.F. Branstetter, J. Mates, and P.J. Chang, "Flying blind: using a digital dashboard to navigate a complex PACS environment," *J. of Digit. Imaging*, vol. 19, no. 1, pp. 69-75, Mar. 2006.

[98] Workflow Management Coalition. (1999). Terminology & Glossary [Online]. Available: http://www.wfmc.org/standards/docs/TC-1011_term_glossary_v3.pdf

[99] R. Khalaf, N. Mukhi, and S. Weerawarana, "Service-Oriented composition in BPEL4WS," in *Proc. WWW2003*, Budapest, Hungary, 2003.

[100] A. Dix, *Human-Computer Interaction*, 3rd ed. New York, NY: Pearson/Prentice-Hall, 2003.

[101] J. Preece, *Human-Computer Interaction.* Reading, MA: Addison-Wesley, 1995.

[102] J. Preece, Y. Rogers, and H. Sharp, *Interaction Design: Beyond Human-Computer Interaction.* New York, NY: Wiley & Sons, 2002.

[103] J. Tidwell, *Designing Interfaces: Patterns for Effective Interaction Design.* O'Reilly, 2005.

[104] J.O. Borchers, "A pattern approach to interaction design," *AI & Soc.*, vol. 15, no. 4, pp. 359-376, 2001.

[105] B. Shneiderman and C. Plaisant, *Designing the User Interface : Strategies for Effective Human-Computer Interaction*, 4th ed. Boston, MA: Pearson/Addison Wesley, 2004.

[106] D. Manolescu and B. Lublinsky, "Service orchestration patterns: graduating from state of the practice to state of the art," in *Companion to the 20th Annu. ACM SIGPLAN Conf. on Object-Oriented programming, Syst., Languages, and Applicat.*, San Diego, CA, 2005, pp. 148-149.

[107] C. Togay, G. Sundar, and A.H. Dogru, "Detection of component composition mismatch with axiomatic design," in *Proc. IEEE SoutheastCon*, 2006, pp. 153-158.

[108] R.E. Jacobson, *Information Design.* Cambridge, MA: MIT Press, 1999.

[109] F. Bellas, "Standards for second-generation portals," *IEEE Internet Comput.*, vol. 8, no. 2, pp. 54-60, 2004.

[110] C. Kyd. (2005). The power of dashboard reporting with Excel [Online]. Available: http://office.microsoft.com/en-us/excel/HA012261271033.aspx

[111] A. Cooper, *The Inmates are Running the Asylum: Why High-Tech Products Drive us Crazy and How to Restore the Sanity.* Indianapolis, IN: SAMS, 1999.

[112] Y. Yamamoto and K. Nakakoji, "Interaction design of tools for fostering creativity in the early stages of information design," *Int. J. of Human-Computer Stud.*, vol. 63, no. 4-5, pp. 513-535, 2005.

[113] R.S. Sadasivam, G. Sundar, M.M. Tanik, and M.N. Tanju, "Process personalization framework for service-driven enterprises," in *Proc. IEEE SouthEastCon*, 2006, pp. 159-164.

[114] G. Sundar, R.S. Sadasivam, J.L. Hartman, and M.M. Tanik, "Intelligent lab information management systems," in *Proc. Integrated Design and Process Technology, IDPT*, San Diego, CA, 2006, pp. 651-658.

[115] L.K. Jololian, “Towards semantic integration of components using a service-based architecture,” in *Proc. Integrated Design and Process Technology, IDPT*, Beijing, China, 2003, pp. 444-451.

[116] L.K. Jololian, F.J. Kurfess, and M.M. Tanik, “Data, function, and control as elements of component integration,” in *Proc. Integrated Design and Process Technology, IDPT*, Austin, TX, 2003, pp. 194-204.

[117] R. de Knikker, Y. Guo, J.L. Li, A.K. Kwan, K.Y. Yip, D.W. Cheung, and K.H. Cheung, “A Web services choreography scenario for interoperating bioinformatics applications,” *BMC Bioinformatics*, vol. 5, pp. 25, Mar. 10, 2004.

[118] J. McCarthy. (2002). Reap the benefits of document style Web services [Online]. Available: http://www-128.ibm.com/developerworks/webservices/library/ws-docstyle.html

[119] E. Gropp. (2004). Document-centric .NET [Online]. Available: http://www.xml.com/pub/a/2004/05/12/dotnet.html

[120] D.A. Chappell, *Enterprise Service Bus*, 1st ed. Sebastopol, CA: O'Reilly, 2004.

[121] A.M.S. Filho, H.K.E. Liesenberg, and R.S.M. Barros, “Interaction pattern gathering in service-oriented applications,” in *Proc. IEEE Int. Workshop Service-Oriented Syst. Eng. SOSE*, 2005, pp. 129-134.

[122] M. Luo, B. Goldshlager, and L.-J. Zhang, “Designing and implementing enterprise service bus (ESB) and SOA solutions,” in *Proc. IEEE Int. Conf. on Services Comput.*, 2005, pp. xiv.

[123] C. Paszko and C. Pugsley, “Considerations in selecting a laboratory information management system (LIMS),” *Amer. Laboratory*, vol. 32, no. 18, pp. 38-43, 2000.

[124] S. Stabb, “Web services: been there, done that,” *IEEE Intell. Syst.*, vol. 18, no. 1, pp. 72-85, Jan.-Feb. 2003.

[125] M.B. Juric, B. Mathew, and P. Sarang, *Business Process Execution Language for Web Services: BPEL and BPEL4WS*, Packt Publishing, 2004.

[126] L. Rowen, G. Mahairas, and L. Hood, “Sequencing the human genome,” *Science*, vol. 278, no. 5338, pp. 605-607, Oct. 1997.

[127] N. Goodman, “Biological data becomes computer literate: new advances in bioinformatics,” *Current Opinion Biotechnology*, vol. 13, no. 1, pp. 68-71, Feb. 2002.

[128] E.S. Lander, *et al.*, “Initial sequencing and analysis of the human genome,” *Nature*, vol. 409, no. 6822, pp. 860-921, Feb. 15, 2001.

[129] M.D. Adams, A.R. Kerlavage, J.M. Kelley, J.D. Gocayne, C. Fields, C.M. Fraser, and J.C. Venter, “A model for high-throughput automated DNA sequencing and analysis core facilities,” *Nature*, vol. 368, no. 6470, pp. 474-475, Mar. 31, 1994.

[130] W. Wang and J. Yang, “Guest editors' introduction: special issue on mining biological data,” *IEEE Trans. on Knowledge and Data Eng.*, vol. 17, no. 8, pp. 1019-1020, 2005.

[131] H.V. Jagadish and F. Olken, "Data management for the biosciences," NSF/NLM Workshop on the Data Management for Molecular and Cell Biology, Workshop Rep. LBNL-52767, 2004.

[132] J. Chen, *Post-Genome Bio Data Management: Modeling and Applications.* Artech House, 2007.

[133] L.D. Stein, "Integrating biological databases," *Nat. Rev. Genet.*, vol. 4, no. 5, pp. 337-345, May, 2003.

[134] Y. Xia, R.E. Stinner, and P.-C. Chu, "Database integration with the Web for biologists to share data and information," *Electronic J. of Biotechnology*, vol. 5, no. 2, Aug. 15, 2002.

[135] L. Wong, "Technologies for integrating biological data," *Brief Bioinform.*, vol. 3, no. 4, pp. 389-404, Dec. 2002.

[136] K.A. Karasavvas, R. Baldock, and A. Burger, "Bioinformatics integration and agent technology," *J. Biomed. Inform.*, vol. 37, no. 3, pp. 205-219, Jun. 2004.

[137] T. Hernandez and S. Kambhampati, "Integration of biological sources: current systems and challenges ahead " *SIGMOD Rec.*, vol. 33 no. 3, pp. 51-60, 2004.

[138] L. Stein, "Creating a bioinformatics nation," *Nature*, vol. 417, no. 6885, pp. 119-120, May 09, 2002.

[139] H.T. Gao, J.H. Hayes, and H. Cai, "Integrating biological research through Web services," *IEEE Computer*, vol. 38, no. 3, pp. 26-31, 2005.

[140] C. Bussler, "The Semantic Web: research and applications," in *Proc. 1st European Semantic Web Symp., ESWS*, Crete, Greece, 2004, pp. 490.

[141] H. Sugawara and S. Miyazaki, "Biological SOAP servers and web services provided by the public sequence data bank," *Nucleic Acids Res.*, vol. 31, no. 13, pp. 3836-3839, Jul. 1, 2003.

[142] R. Stevens, A. J. Robinson, and C. Goble, "myGrid: personalized bioinformatics on the information grid," *Bioinformatics*, vol. 19, suppl. 1, pp. i302-4, 2003.

[143] W.W. Li, S. Krishnan, K. Mueller, K. Ichikawa, S. Date, S. Dallakyan, M. Sanner, C. Misleh, D. Zhaohui, W. Xiaohui, O. Tatebe, and P.W. Arzberger, "Building cyberinfrastructure for bioinformatics using service-oriented architecture," in *Proc. 6th IEEE Int. Symp. on Cluster Comput. and the Grid Workshops*, 2006, pp. 8.

[144] P. Bresciani, P. Fontana, and P. Busetta., "A knowledge-based interface for distributed biological databases," in *Proc. NETTAB, Int. Workshop on Agents in Bioinformatics*, University of Bologna, Italy, 2002.

[145] V.G. Koutkias, A. Malousi, and N. Maglaveras, "Engineering agent-mediated integration of bioinformatics analysis tools " in *Proc. 1st Int. Workshop on Multi-Agent Syst. for Medicine, Comput. Biology, and Bioinformatics*, 2005, pp. 154-161.

[146] L. Moreau, *et al.*, "On the use of agents in a bioinformatics grid," in *Proc. 3rd IEEE/ACM CCGRID*, 2003, pp. 653-660.

[147] S. Miles, "Agent-oriented data curation in bioinformatics " in *Proc. 1st Int. Workshop on Multi-Agent Syst. for Medicine, Comput. Biology, and Bioinformatics*, 2005, pp. 157-169.

[148] E.M. Maximilien and M.P. Singh, "Agent-based architecture for autonomic Web service selection," in *Proc. AAMAS 03*, Melbourne, Australia, 2003, pp. 1-6.

[149] R.M. Sreenath and M.P. Singh, "Agent-based service selection," *J. of Web Semantics*, pp. 261-279, 2003.

[150] V. Ermolayev, N. Keberle, O. Kononenko, S. Plaksin, and V. Terziyan, "Towards a framework for agent-enabled semantic Web service composition," *Int. J. Web Services Research*, vol. 1, no. 3, pp. 1-12, 2004.

[151] D.K. Arnett, R.B. Devereux, D. Kitzman, A. Oberman, P. Hopkins, L. Atwood, A. Dewan, and D.C. Rao, "Linkage of left ventricular contractility to chromosome 11 in humans: The HyperGEN Study," *Hypertension*, vol. 38, no. 4, pp. 767-772, Oct. 2001.

[152] U. Vitt, *et al.*, "Identification of candidate disease genes by EST alignments, synteny, and expression and verification of Ensembl genes on rat chromosome 1q43-54," *Genome Res.*, vol. 14, no. 4, pp. 640-650, Apr. 2004.

[153] D.L. Wheeler, D.M. Church, R. Edgar, S. Federhen, W. Helmberg, T.L. Madden, J.U. Pontius, G.D. Schuler, L.M. Schriml, E. Sequeira, T.O. Suzek, T.A. Tatusova, and L. Wagner, "Database resources of the National Center for Biotechnology Information: update," *Nucleic Acids Res.*, vol. 32, pp. D35-40, Jan. 1 2004.

[154] A. Hamosh, A.F. Scott, J. Amberger, C. Bocchini, D. Valle, and V.A. McKusick, "Online Mendelian Inheritance in Man (OMIM), a knowledgebase of human genes and genetic disorders," *Nucleic Acids Res*, vol. 30, no. 1, pp. 52-55, Jan. 1, 2002.

[155] J.L. Hartman and N.P. Tippery, "Systematic quantification of gene interactions by phenotypic array analysis," *Genome Biol.*, vol. 5, no. 7, pp. R49, 2004.

[156] C. Wroe, C. Goble, M. Greenwood, P. Lord, S. Miles, J. Papay, T. Payne, and L. Moreau, "Automating experiments using semantic data in a bioinformatics grid," *IEEE Intell. Syst. and applicat.*, vol. 19, no. 1, pp. 48-55, 2004.

[157] T.V. Venkatesh and H.B. Harlow, "Integromics: challenges in data integration," *Genome Biol.*, vol. 3, no. 8, Jul. 17, pp. 1-3, 2002.

[158] M.C. Cavalcanti, F. Baiao, S.C. Rossle, P.M. Bisch, R. Targino, P.F. Pires, M.L. Campos, and M. Mattoso, "Structural genomic workflows supported by Web services," in *Proc. 14th Int. Workshop on Database and Expert Syst. Applicat., DEXA'03*, 2003, pp. 45-49.

[159] R. Gaizauskas, N. Davis, G. Demetriou, Y. Guo, and I. Roberts, “Integrating text mining into distributed bioinformatics workflows: a Web services implementation,” in *Proc. IEEE Int. Conf. on Services Comput., SCC’04*, 2004, pp. 145-152.

[160] Y. Yan, Y. Liang, and X. Du, “Controlling remote instruments using Web services for online experiment systems,” in *Proc. IEEE Int. Conf. on Web Services, ICWS'05*, 2005, pp. 732.

[161] C.G. Thurston, “Instrument integration,” *Manufacturing Chemist*, vol. t6, no. 5, pp. 34-35, 37, Jul.-Aug. 2004.

[162] C.G. Thurston, “LIMS/Instrument integration computing architecture for improved automation and flexibility,” *Amer. Laboratory*, vol. 36 no. 18, pp. 16-21, 2004.

[163] MATLAB. (2007). MATLAB® Builder for Excel® 1.2.9 - deploy MATLAB code as Microsoft Excel add-ins [Online]. Available: http://www.mathworks.com/products/matlabxl/

[164] MATLAB. (2007). Excel® Link 3.0 - Use MATLAB® from Microsoft® Excel [Online]. Available: http://www.mathworks.com/products/excellink/

[165] K. Chari and S. Seshadri, “Demystifying integration,” *Commun. ACM*, vol. 47, no. 7, pp. 58-63, 2004.

[166] R. Weinreich and T. Ziebermayr, “Enhancing presentation level integration of remote applications and services in Web portals,” in *Proc. IEEE Int. Conf. on Services Comput.*, 2005, pp. 224-231.

[167] I. Oo, “Presentation level integration of portal personalization architecture,” in *Proc. 2nd Inform. and Commun. Technologies, ICTTA '06*, 2006, pp. 564-565.

[168] M. Chaoying, L. Bacon, M. Petridis, and G. Windall, “Towards the design of a portal framework for Web services integration,” in *Proc. Int. Conf. on Internet and Web Applicat. and Services, AICT-ICIW '06*, 2006, pp. 163-163.

[169] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke, “A security architecture for computational grids,” in *Proc. 5th ACM Conf. on Comput. and Comm. Security*, San Francisco, CA, 1998, pp. 83-92.

[170] R. Butler, V. Welch, D. Engert, I. Foster, S. Tuecke, J. Volmer, and C. Kesselman, “A national-scale authentication infrastructure,” *IEEE Comput.*, vol. 33, no. 12, pp. 60-66, 2000.