



# **Towards a Utility Framework for Enterprise Business Intelligence Mashups**

by

Sabbir Ahmed

Thesis submitted to the Faculty of the Graduate and Postdoctoral Studies

in partial fulfillment of the requirements for the degree of

Master of Science

in

Electronic Business Technologies

Supervisor: Dr. Umar Ruhi

University of Ottawa

Ottawa, Ontario, Canada

*In the name of Allah  
the Compassionate, the Merciful*

# Abstract

---

Organizations today are adopting business intelligence (BI) systems at a fast pace with the expectation that these systems will help them make better business decisions and improve their performance. But with the ever changing industry dynamics and variable business needs of end-users, the BI requirements of organizations are also getting unpredictable and increasingly harder to deliver by the service providers. Additionally, the advancement of web 2.0 & enterprise 2.0 technologies has opened up more possibilities for the development of user-centric innovative business applications. Enterprise BI Mashups are a specific breed of such technologies that have the potential to empower end-users with self-service capabilities and facilitate problem-solving in ad-hoc situational BI scenarios. This research project attempts to explore the current landscape of Business Intelligence (BI) Mashups and to identify gaps in technology with respect to user requirements and corporate objectives. Through an empirical investigation of BI mashups use cases, specific issues and challenges associated with the use of mashups in BI have been ascertained. Working in collaboration with IBM Cognos, we have formulated a taxonomy and utility framework for Enterprise BI Mashups. The formulated taxonomy provides a basic framework for understanding the domain of BI mashups and is aimed to aid application development initiatives for creating BI mashups toolkits. The utility framework draws upon real-world use cases for BI Mashups as well as pertinent software design patterns that can facilitate the development of BI mashup tools and services. These frameworks are expected to advance an understanding of business process requirements that can be satisfied through the use of Enterprise BI Mashups, and also aid in the development of mashup toolkits targeted at BI end-users.

# Acknowledgment

---

I would like to take this opportunity to express my appreciation to all the wonderful and kind people around me whose help and support have abled me to write this thesis.

It is with immense gratitude that I acknowledge my supervisor, Dr. Umar Ruhi, for his guidance and help throughout this thesis project. It has been an immense learning experience working under his supervision. I thank him for believing in me and for giving me the opportunity to work with all the great people at IBM Cognos.

I would also like to thank my manager at IBM, Mrs. Emilia Klein and my mentor Hassan Bassam for their support during my internship.

I am thankful to the IBM Center for Business Analytics & Performance and Mitacs Accelerate program for their financial support towards my thesis.

I am grateful to my dear friend Sara Mohammadi for her close friendship during the difficult times of my graduate studies.

Finally, I would like to dedicate this thesis to my parents and my brother who have been the source of inspiration and hope for me. My family has always encouraged me throughout my studies and more importantly my life.

# Publications

---

Ahmed, S., & Ruhi, U. (2013). Towards a Functional Taxonomy of Enterprise Business Intelligence Mashups. In *The Second International Conference on Informatics and Applications (ICIA2013)* (pp. 98–103)

# Table of Contents

---

<b>Chapter 1 Introduction</b> .....	<b>1</b>
1.1 Research Motivation .....	2
1.2 Problem Definition.....	6
1.3 Objectives .....	8
1.4 Contribution of the Thesis and Key Benefits.....	9
1.5 Structure of the Thesis .....	10
<b>Chapter 2 Background and Literature Review</b> .....	<b>11</b>
2.1 The Mashup Concept .....	11
2.1.1 Web 2.0.....	12
2.1.2 Enterprise 2.0.....	13
2.1.3 The Birth of Mashups .....	15
2.2 Mashup Categories: .....	16
2.2.1 Mashup Categories based on Functional Range.....	17
2.2.2 Mashup Categories based on Target/User Groups .....	18
2.2.3 Mashup Genres .....	20
2.3 Benefits of Enterprise Mashups .....	22
2.4 Mashup Patterns:.....	26
2.4.1 Activities Related to Mashup Patterns.....	27
2.4.2 Pattern Descriptions.....	29
2.4.2.1 Harvest Patterns .....	29
2.4.2.2 Enhance Patterns .....	31
2.4.2.3 Assemble Patterns .....	34
2.4.2.4 Manage Patterns.....	36
2.4.2.5 Testing Patterns.....	37
2.4.3 Mashup Design and Architecture .....	38
2.4.4 Mashup Development Classifications .....	41
2.4.4.1 Manual and Tool-Assisted Mashup Development .....	41
2.5 Business Intelligence .....	52

2.5.1	Business Intelligence Tool: IBM Cognos BI.....	56
2.5.2	Situational Business Intelligence.....	56
2.5.3	Business Intelligence & Self-Service .....	61
2.6	Enterprise BI Mashups and their current landscape .....	64
2.6.1	BI Mashup Development Tools.....	68
2.6.1.1	IBM Cognos Software Development Kit (SDK) .....	68
2.6.1.2	IBM Cognos Mashup Service (CMS) .....	69
<b>Chapter 3 Research Methodology .....</b>		<b>70</b>
3.1	Design Science Research methodology for Information Systems.....	70
3.2	Research Methods and Steps.....	73
3.3	Methodology of Taxonomy Construction.....	74
3.4.	Taxonomy Construction for Our Research .....	76
3.5	Research Evaluation and Validation .....	76
<b>Chapter 4 Results and Findings .....</b>		<b>80</b>
4.1	Proposed Functional Taxonomy for Enterprise BI Mashups .....	80
4.1.1	Taxonomy Construction.....	81
4.1.2	Taxonomy Analysis .....	84
4.1.2.1	Enterprise BI Mashup Enablers .....	85
4.1.2.2	Enterprise Users .....	87
4.1.2.3	Development Method.....	88
4.1.2.4	Functional Range .....	89
4.1.2.5	Data Sources .....	93
4.1.3	Mashup Pattern Mapping .....	95
i.	Patterns for Mashup Presentation.....	96
ii.	Patterns for Data Utilization .....	97
iii.	Patterns for mashup functionality.....	98
iv.	Patterns for Processes .....	98
v.	Patterns for Supporting required Data Sources .....	99
4.2	Proposed Utility Framework for Enterprise BI Mashups .....	99
4.2.1	Underlying Infrastructure.....	101
i.	Content Sources.....	101

ii.	BI Mashup and BI Enabler Components .....	103
4.2.2.	User Types .....	105
4.2.3	User Tasks .....	105
4.2.4	Mashup Platform Client Composition.....	108
4.2.5	Required Functionalities.....	109
<b>Chapter 5</b>	<b>Prototypes and Evaluation .....</b>	<b>114</b>
5.1	Prototype 1: Airplane Seat Information.....	114
5.1.1	Overview .....	114
5.1.2	Application Requirements .....	115
5.1.3	Prototype Mashup Implementation .....	115
5.1.3.1	Cognos Report Creation and Access to the Report.....	115
5.1.3.2	Mashup Composition .....	116
5.2	Prototype 2: Organizational Hierarchy Mashup .....	119
5.2.1	Overview .....	119
5.2.2	Application Requirements .....	119
5.2.3	Prototype Mashup Implementation .....	119
5.2.3.1	Cognos Report Access through CMS and Direct URL.....	120
5.2.3.2	Data Modeling and Transformation .....	122
5.2.3.3	Mashup Presentation Composition.....	125
5.3	Evaluation of the Prototype Mashup Applications .....	127
<b>Chapter 6</b>	<b>Conclusions .....</b>	<b>131</b>
6.1	Summary of the Research .....	131
6.2	Contributions of the Thesis .....	132
6.3	Limitations .....	134
6.4	Future Work and Research Directions .....	136
References	.....	138



# List of Figures

---

<i>Figure 1: Enterprise Mashups &amp; Long Tail Applications .....</i>	<i>6</i>
<i>Figure 2:Typcial Organizational Hierarchy .....</i>	<i>14</i>
<i>Figure 3: Enterprise Mashups data sources.....</i>	<i>20</i>
<i>Figure 4: Benefit Model for Enterprise Mashups.....</i>	<i>23</i>
<i>Figure 5: Enterprise Mashup Environment.....</i>	<i>39</i>
<i>Figure 6:Comparison summarization of SOA and Mashups.....</i>	<i>51</i>
<i>Figure 7:Information Provision Process.....</i>	<i>58</i>
<i>Figure 8: How BI Mashups Work.....</i>	<i>65</i>
<i>Figure 9: Integration of the mashup platform into a DWH architecture .....</i>	<i>66</i>
<i>Figure 10: BI mashup maturity model.....</i>	<i>67</i>
<i>Figure 11: Research Framework.....</i>	<i>72</i>
<i>Figure 12: Preliminary Taxonomy Framework for Enterprise BI Mashups.....</i>	<i>84</i>
<i>Figure 13: Mashup Pattern Mapping with Taxanomy Elements.....</i>	<i>96</i>
<i>Figure 14: Proposed Utility Framework.....</i>	<i>101</i>
<i>Figure 15: Underlying infrastructure for a Enterprise BI Mashup Platform .....</i>	<i>102</i>
<i>Figure 16: Report Prompt Selection and Report Generation.....</i>	<i>116</i>
<i>Figure 17: Report URL for Mashup Center Usage .....</i>	<i>116</i>
<i>Figure 18: Configuring the URL customizer widget .....</i>	<i>117</i>
<i>Figure 19: Wiring congfiguration of the widgets .....</i>	<i>118</i>
<i>Figure 20: Airplane Seat Information Mashup .....</i>	<i>118</i>
<i>Figure 21: Organization Hierarchy Report.....</i>	<i>120</i>
<i>Figure 22: Accessing the Report data through Web Service.....</i>	<i>121</i>
<i>Figure 23: Employee Posistion and Pay-Scale Information Report .....</i>	<i>121</i>
<i>Figure 24: Organization Hierarchical Data Level Mashup.....</i>	<i>122</i>
<i>Figure 25: Source Operator Configuration.....</i>	<i>123</i>
<i>Figure 26: Data Modeling and Transformation.....</i>	<i>124</i>
<i>Figure 27: Organizational Hierarchy Data Mashup in XML format.....</i>	<i>125</i>
<i>Figure 28: URL Customizer Widget Configuration.....</i>	<i>125</i>
<i>Figure 29: Organization Hierarchical Mashup Presentation.....</i>	<i>126</i>

# List of Tables

---

<i>Table 1: Framing the Research Problem.....</i>	<i>7</i>
<i>Table 2: Bitzer and Schumann (2009) classification model .....</i>	<i>17</i>
<i>Table 3: Summarization of Harvest Patterns.....</i>	<i>29</i>
<i>Table 4: Summarization of Enhance Patterns .....</i>	<i>32</i>
<i>Table 5: Summarization of Assemble Patterns .....</i>	<i>34</i>
<i>Table 6: Summarization of Manage Patterns .....</i>	<i>37</i>
<i>Table 7: Summarization of Testing Patterns .....</i>	<i>38</i>
<i>Table 8: Mashup component and composition model summary.....</i>	<i>41</i>
<i>Table 9: Functionality classification of mashup tools.....</i>	<i>42</i>
<i>Table 10: Information Workers' Decision Support Nirvana .....</i>	<i>63</i>
<i>Table 11: Comparison of mashup prototype implementations with available IT developed solutions .....</i>	<i>129</i>

# List of Acronyms

---

Acronym	Definition
CRM	Customer Relationship Management
DWH	Data Warehouse
Ajax	Asynchronous Javascript & XML
AOP	Aspect Oriented Mashup
API	Application Programming Interface
B2b	Business to Business
CMS	Cognos Mashup Service
DSR	Design Science Research
DTD	Document Definition Type
EMML	Enterprise Mashup Markup Language
ERP	Enterprise Resource Planning
ETL	Extract, Transform and Load
GUI	Graphical User Interface
IS	Information System
JSON	Javascript Object Notation
LDX	Layout Data XML
ODS	Operational Data Stores
RDF	Resource Description Framework
REST	Representational State Transfer
RSS	Rich Site Summary
SCM	Supply Chain Management
SLA	Service Layer Agreement
SOAP	Simple Object Access Protocol
UI	User Interface
XML	Extensible Markup Language

# Chapter 1 Introduction

---

*“If I have seen further it is only by standing on the shoulders of giants.”*

- Isaac Newton

In this wonderful statement, Newton not only acknowledges and gives credit to the work of others, but also confirms that what his mind has seen is a result of contributions from many other great minds. Those minds provided him with a stream of thoughts, which were put together to create new insights. The mind is indeed the ultimate knowledge *mashup*. What if technology tools and applications were able to mirror such behavior and facilitate the discovery of new insights by their end-users? *Mashups* provide a means to achieve this objective. *Mashups* are composite applications that combine content, presentation, and application functionality from multiple disparate sources and create useful new services and applications (Makki & Sangtani, 2008; J. Yu, Benatallah, Casati, & Daniel, 2008).

Mashup technologies are especially well-suited to the context of enterprise *Business Intelligence (BI)* solutions by virtue of their primary purpose, which is to empower end-users to create and adapt individual information centric and situational applications (Volker Hoyer & Fischer, 2008). BI systems are increasingly becoming critical to the daily operation of organizations. But with ever changing business needs of the users, their BI requirements are getting unpredictable and increasingly harder to deliver by the service providers. *BI Mashups* are a new breed of web based user-centric enterprise applications that unify disparate data and services to respond to situational requirements i.e. ad-hoc and on-going analytical needs at personal and enterprise scales(Grimes, 2010; Hassanzadeh et al., 2011). *Situational applications* pertain to these unpredictable and transient organizational requirements and follow an end-user centric approach (Volker Hoyer & Fischer, 2008; Watt, 2007). Such applications are extremely

common in today's dynamic and complex business environments – there is a constant demand for information, and more importantly, for insights obtained through the information to assist in decision making practices. More than ever before, knowledge workers are faced with many unpredictable events which result in novel situations that require just-in-time management of information (Watt, 2007). In use contexts such as these, traditional methodologies for BI development and deployment can prove to be insufficient and the economics of formal development processes may no longer make sense (Mohammadi, Khalili, & Ashoori, 2009; J. Yu et al., 2008).

In this project, we aim to investigate the scope and application of enterprise mashup technologies within the realm of business intelligence (BI) solutions. By deliberating how enterprise mashups can support the business intelligence needs and requirements of organizations and their end-users, this project aims to formulate a *utility framework* for Enterprise BI Mashups. Such a framework is expected to advance an understanding of business process requirements that can be satisfied through the use of Enterprise BI Mashups, and also aid in the development of mashup toolkits targeted at BI end-users.

## 1.1 Research Motivation

Industry experts agree that mashups will play an increasingly important and major role in future enterprise BI strategies. Previously *Gartner Research* had predicted that one-third of analytics applications applied to business processes will be delivered through coarse-grained application mashups (Gartner, 2009). The *Forrester Group* predicted that the enterprise mashups market would reach nearly \$700 million by 2013 (Young, Gualtieri, Daley, Shey, & Ashour, 2008). According to the *Forrester Group* mashup based tools will also play a significant role in

bringing BI to end-users across the organization (Kobielus, 2009). Following are some drivers for the adoption and use of Enterprise BI Mashups:

***BI Mashups assist in increasing BI end-user adoption***

BI Mashups, by their very nature, complement traditional BI offerings. They provide building blocks (components) such as widgets, feeds, and interfaces for knowledge users to assemble their own information views and analytics dashboards, hence acting as an enabler for the *self-service BI* paradigm (Hassanzadeh et al., 2011; Watt, 2007). *Self-service BI* capabilities go well beyond user friendly or intuitive interfaces. The potential of mashup technology in enabling situational BI applications, providing self-service, and facilitating real-time business intelligence are some of the benefits purported by industry leaders who have adopted these tools to complement and enhance their BI technology base (Grimes, 2010; Kobielus, 2009; Zou & Pavlovski, 2007).

***BI Mashups enhance decision-making by leveraging contextual internal and external resources***

With the huge growth of data available through the Web, many BI use contexts comprise leveraging information available on the Internet. BI Mashups have the potential to incorporate information from these external sources with data and views from internal sources to aid the enterprise decision-making process (Hassanzadeh et al., 2011). The growth of information on the Internet has also been complemented by an evolution of various technologies, applications and standards that facilitate the assimilation and integration of resources on the web. These include

the maturity of integration standards such as *simple object access protocol* (SOAP), *representational state transfer* (REST), the advancement of structure and context specifications through initiatives such as the *resource description framework* (RDF), and the availability of easy-to-use *graphical user interface* (GUI) toolkits with visual drag-and-drop tools for composition of elements and workflows. Together, these advancements have paved the way for enterprise mashups to be included in an organization's portfolio of technology applications (Volker Hoyer & Fischer, 2008; Volker Hoyer, Stanoesvka-Slabeva, Janner, & Schroth, 2008; Merrill, 2009).

In addition to standards that help leverage external data and application resources, many organizations are revamping their internal application environments through the deployment of service oriented architecture. *Service-oriented architecture* (SOA) is the foundation platform for building agile business applications in which software is packaged into reusable, self-managed services using a web services interface. In the context of BI infrastructures, an SOA based implementation has the potential to enable more seamless integration of disparate technology elements into a coherent business intelligence environment (Wu, Barash, & Bartolini, 2007). Mashups offer a good fit within the context of SOA (Luo, Xu, Song, & Song, 2008) – they can help organizations grasp new business opportunities and improve resource usage by letting users assemble internal and external data sources in an opportunistic manner (Hassanzadeh et al., 2011).

***Mashups align with demands of current BI paradigms such as real-time & self-service BI and agile development:***

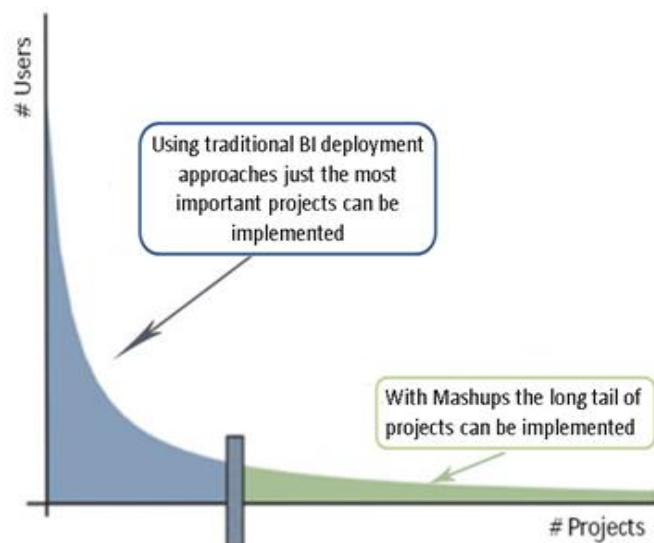
Traditional approaches in BI solutions often draw upon various assumptions such as predetermined end-user requirements, resolved & concrete business requirements or knowledge workers willing to adapt their working habits in order to accommodate the BI systems. However, these assumptions do not always hold true. In fact, the varying requirements and needs of businesses and end-users have compelled many organizations to focus on making their BI initiatives more agile. Agile development methodology calls for incrementally delivering products versus a big-bang approach, for rapid prototypes versus specifications, for reacting versus planning and for personal interaction with business users versus documentation (Evelson, 2010). Agile processes have been identified as a central best practice in contemporary BI initiatives (Evelson, 2011).

Mashup technologies offer a means to advance business agility through cross-platform applications, flexible configurability of components, and reusability of workflows (Luo et al., 2008; Xie, Xu, & de Vrieze, 2010). More specifically, BI Mashups have been purported to play an increasingly important role in situational BI applications as mentioned before, BI projects with real-time requirements (T. Yu et al., 2009), and use contexts with highly personalized BI needs (Scientifique, 2003). In addition to realizing these personal needs, mashup features in BI environments, lets non-technical users build context-rich, role-tailored, ad hoc views of disparate data and explore information in greater depth (Kobielus, 2009).



## 1.2 Problem Definition

Despite years of investing in BI, many organizations have had great difficulty connecting BI with their core decision making tasks and getting business users involved in the effective utilization of BI tools and applications. According to industry reports, more than 60% of BI implementations fail due to inadequate end-user adoption or perceived gaps in BI solution offerings (Carney, 2012). This high failure rate is an indication of underlying problems with traditional BI offerings. Some of the problems arise due to rigid data structures that are relatively hard to modify, and in many cases, small modifications at the end-user level require a lot of changes at the back-end and a considerable investment of time, effort and the involvement of IT staff and database administrators.



*Figure 1: Enterprise Mashups & Long Tail Applications, based on Pahlke et al. (2010)*

Many industry experts believe that BI Mashups can help in decreasing the BI projects failure rate and in facilitating the end-user adoption (Evelson, 2011; Hassanzadeh et al., 2011; Zou & Pavlovski, 2007) . For example, as depicted in figure 1, traditional BI offerings address the most common project requirements through major technology initiatives within the enterprise

(represented by the short-tail). However, there are many niche, short life-spanned, situational projects for a small number of targeted users that require just-in-time delivery of good enough solutions that are not addressed by traditional offerings. With the help of mashup technologies, these long tail projects can be implemented with greater ease (Evelson, 2011; Hassanzadeh et al., 2011; Volker Hoyer & Fischer, 2008; Nelson, Edia, & Ntertainment, 2010).

Additional user-adoption barriers include difficulties in unifying access to heterogeneous data sources within the enterprise and the constant demand for BI developers and data professionals who have limited time to cater to the needs of specific groups. These barriers affect the overall quality and effectiveness of decision-making, present challenges for data integration, and consequently affect end-user adoption (Popovic, Coelho, & Jaklič, 2009). With the availability of mashup tools & supporting resources, these barriers can be overcome.

*Table 1: Framing the Research Problem*

Steps	Description
Observation	<p>End-users situational BI requirements are increasing with the growing amount of unstructured data in the web (Löser, Hueske, &amp; Markl, 2009).</p> <p>The business users require these situational applications delivered to them in a rapid manner so that they can make sense of their data right away right now</p> <p>IT departments cannot cater to the long-tail of user requests due to resource constraints</p>
Thesis	An Enterprise BI Mashup platform will increase BI systems usage success rate by enabling tech savvy end-users with self-service capabilities to meet their own situational demands.

Enthymeme	An enterprise mashup platform providing self-service capabilities will offer rapid situational application development capability by the BI end-users themselves without requiring any assistance of the IT service providers.
Problem Statement	There is a lack of formalized frameworks for the development of Enterprise BI Mashup platforms which will enable end-users with self service capabilities in order to meet their situational BI needs.
Objective	Formulate a <i>utility framework</i> for Enterprise BI Mashups. Such a framework will draw upon real-world use cases for BI Mashups as well as pertinent software design patterns that can facilitate the development of BI mashup tools and services
Research Questions	<p>What are the most common end-user requested situational BI applications that go un-served or partially served by the service providers?</p> <p>What characteristics within a mashup platform enable end-users with self-service capabilities and create situational BI applications?</p> <p>What are the functional and architectural requirements of an Enterprise BI Mashup Platform?</p>

### 1.3 Objectives

In this project, we aim to explore the current landscape of Enterprise BI Mashups and identify the gaps in existing technology offerings with respect end-user requirements in terms of their situational data needs. By explaining how enterprise mashups can support the business

intelligence needs and requirements of organizations and their end-users, this project aims to formulate a *utility framework* for Enterprise BI Mashups. Such a framework will draw upon real-world use cases for BI Mashups as well as pertinent software design patterns that can facilitate the development of BI mashup tools and services. The framework is expected to advance an understanding of end-user requirements that can be satisfied through the use of Enterprise BI Mashups, and also aid in the development of mashup toolkits targeted at BI end-users.

## **1.4 Contribution of the Thesis and Key Benefits:**

The goal of this study is to advance the knowledge base for Enterprise BI Mashups by offering strategic insights and actionable guidelines that can improve the adoption of business intelligence technologies across the enterprise and enhance the end-user experience with utilizing mashups for their daily business analytics needs. Some of the key contributions of this thesis are as follows:

1. Identifying and understanding the situational BI requirements of the clients as well as the bottlenecks of the service delivery process of these requirements by the service providers.
2. A taxonomy of Enterprise BI Mashups through ontological modeling
3. Formulating a utility framework that identifies and describes the requirements for suitable component and composition models for Enterprise BI Mashups
4. Integration of BI and Mashup products: Cognos BI and IBM Mashup Centre
5. Product enhancement and novel use cases of Mashup Enabler tools (Cognos Mashup Service)

## 1.5 Structure of the Thesis

The thesis is organized as follows: in chapter 1 we have presented an introduction, context and the scope of our research. Chapter 2 provides background information of general concepts related to this thesis. In chapter 3 we discuss the methodology we followed to perform our research work. In chapter 4 we present the results and finding of our research. The following chapter consists of the mashup implementations that we have done and their evaluation process. The final chapter provides a conclusion of our work as well the limitation and the planned future work of this research.

# Chapter 2: Background & Literature Review

---

## 2.1 The Mashup Concept

Mashups are one of the hallmarks of second generation web applications or Web 2.0. This exciting breed of interactive web applications sprouting all across internet is generated by combining content, presentation or application functionality from disparate sources. Their popularity stems from the emphasis on interactive user participation in which they aggregate and stitch together third party data (Merrill, 2009). Drawing upon content and functionality from within and outside the organizational boundaries, mashups spread their roots across the web. The etymology of the term mashup leads us back to the pop music scene where it was borrowed from; where mashup is a new song mixed from two different source songs of different genres. And at their best these mashups strive for musical epiphanies that add up to considerably more than the sum of their parts (cited in Makki & Sangtani, 2008). Mashups in Web 2.0 similarly enhance the independent and isolated nature of standalone web applications by deriving relevant data and parameters from various sources - fusing the properties and thus forming the mashed up application. But many popular definitions of mashup would indicate that they are limited to only web based artifacts e.g.: published APIs, RSS/Atom feeds or HTML “screen scraping”. But mashups are not just composite applications created from these sources, they can also be used to mine and manipulate data, migrate content or even present data in a different way making themselves open to a much broader world of data including databases, binary formats (Excel and

PDFs), XML, delimited text files and so on. Mashups effectively simplify the discovery and presentation of complex information through making connections between pieces of information. But mashups do not confine themselves in providing simplicity only, through leveraging creative composition of existing functionalities, data and interfaces to achieve a more complex goal (Bozzon, Brambilla, Facca, & Carughu, 2009). The goal of creating mashups is to solve problems and create new opportunities through the process of finding and using data, functionality and services.

Mashups are regarded as a product of the Web 2.0 (Liu, Liang, Xu, Staples, & Zhu, 2011) and subsequently adopted in the Enterprise 2.0 arena due to the user demands of Web 2.0 technologies within the existing corporate infrastructure (Mcafee, 2006). In the following sections we will discuss about Web 2.0 & Enterprise 2.0.

### **2.1.1 Web 2.0**

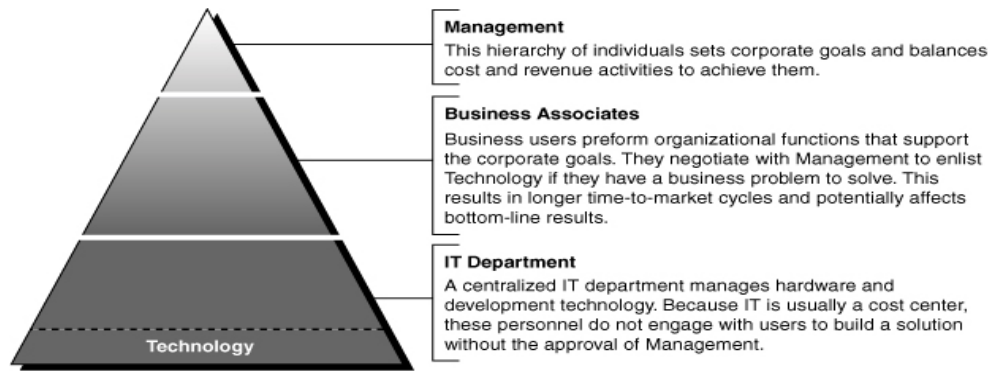
The advent of Web 2.0 came with the shift from transaction based Web pages to interaction-based ones. Tim O'Reilly, credited with popularizing the term, defined it as *“the business revolution in the computer industry caused by the move to the internet as platform, and an attempt to understand the rules for success on that new platform”* (O'Reilly, 2006). Users can easily share their opinion and resources with the set of services the Web 2.0 provides. The power of individual web users is mashed, mixed and multiplied to create value. The components that make Web 2.0 possible consists of social networking sites and image sharing services /sites, folksonomies (collaborative tagging, social bookmarking), wikis, blogs , RSS, podcasts and of course mashups (Ogrinz, 2009; Yakovlev, 2007). But the one trait of Web 2.0 technologies which stands out and makes them unintentionally malleable is their flexibility, which is also the underlying driver of Web 2.0. Components of Web 2.0 have the capability to accept and adjust to

new uses undertaken by the users, when they are used outside the scope of their original intention, rather than breaking down (Ogrinz, 2009). The core of Web 2.0 principles are defined as “simple, low barrier and fast” and “every user himself is the center on internet” (Mohammadi et al., 2009).

### ***2.1.2 Enterprise 2.0***

Enterprise 2.0 breaks down the traditional divisional barriers within organizations and encourages building bridges. With the notion of “bring your own device (BYOD)” we have seen employees opting for bringing their own devices to their workplaces to make it easier to manage the applications they use. This comes from the inevitability of people discovering useful tools and personalizing them outside the workplace and then wanting to use it at their offices too. In the same manner in which the propagation of personal devices at workplace is accelerating, the demand for self-service application development available in the consumer internet space for non-technical users is growing in the more tightly governed enterprise scenario. As we have mentioned before user demand for Web 2.0 technologies within corporate walls is the driver for Enterprise 2.0. To take the advantage of “economies of collaboration” the challenge for organizations is to integrate peer-based collaborations models of Web 2.0 with legacy technologies and the mindsets. In the traditional organizational hierarchies, the established control mechanisms of how solutions are being delivered are typically isolated.





*Figure 2:Typcial Organizational Hierarchy (Ogrinz, 2009)*

Figure 2 depicts the traditional divisional barriers in organizations. Each particular tier, rather than having a collaborative work process, address different data security and information protection concerns. IT department's actions on security are purely mechanical, done through the use of secure protocols, authentication, encryption, and so on. While the business users actions depend on training and education on how to maintain standards while utilizing the services provided to them. And from the top tier where the management lies, attention is given to whether the organization as a whole conforms to the regulatory or industry specific tools. In the Enterprise 2.0 concept these rigid structures are removed through collaborative solution delivery. Equal access to technology and flexible yet powerful tools enable every segment of the organization to build its own solutions. Especially business or non-technical users are empowered to create their own application solutions without management or IT's engagement. The power of crowdsourcing and folksonomies is leveraged to create an open interaction model through which teams can discover how other lines of business operate and this in turn leads to changes in strengthening relationships across departments.

### 2.1.3 The Birth of Mashups

The concept of reuse in software engineering plays an important role towards delivering a quick, easy and affordable application development process. Instead of starting from scratch, reusing something that has already been built, tested and paid for through the use of external libraries, web services, subroutines, object orientation, templates etc aids in achieving this goal. But these milestones were created by software development professionals with the purpose of sole use by other developers. But an Enterprise 2.0 environment necessitates business users participating more directly in the development processes, either by themselves or in partnership with IT department. But a certain level of knowledge about development best practices is a prerequisite for a successful involvement of the business team. On the other hand, it is imperative for the IT department and personnel to learn more about the business users' work processes & goals and creating an environment which facilitates rapid service delivery and product construction which the users require (Ogrinz, 2009).

In addition to that, the ability to utilize the reuse concept in the Web 2.0 could pave the way to having a “semantic web” that Tim Berners-Lee envisioned of (Berners-lee, Hendler, & Lassila, 2002). Mashups, using the web as a platform, are making this vision a reality. This empowering technology provides the opportunity to reuse resources which are not designed to be re-used. The traditional approach of creating APIs, compiling packages and writing documentation has left applications developers who reused resources at the mercy of the original designers. Mashups liberate the reusers in that sense. With mashups we are not confined to only using the existing APIs; or we even can impose our own if none exists. Mashup's ability to provide programmatic access to unlimited data together with the fact that the tools for constructing mashups are reaching a usability level where the line of business users can utilize them to build their own

solutions, provides immense opportunity to bridge the existing gap between IT and business (Ogrinz, 2009)

## 2.2 Mashup Categories:

Before delving in to discussing mashup types, we will look into a classification model of mashups which will help us better understand mashup categorization. Mashups have been subject to different specifications in terms of their classification. The classification model that we have chosen is based on Bitzer and Schumann (2009) and Pahlke et al. (2009). Table 2 summarizes the classification and characteristics model.

Type	Classification	Characteristics
Functional Range	Presentation Level Mashups	Provide layout and information in various ways as Web services
	Data Level Mashups	Concentrate on the extraction and combination of data from different sources and integrate them into the user's own internet site
	Functionality	Combine and integrate information and application functionality services proved by different sources
	Process	Integrate and orchestrate information services, business functionality services according to a business/work processes sequence

Target/User Groups	Consumer Mashups	Mainly used for private applications intended for the general masses
	Enterprise Mashups	Individualization of software within organizations

*Table 2: Bitzer and Schumann (2009) classification model*

### 2.2.1 Mashup Categories based on Functional Range

Presentation level mashups provide layout and information in various ways as web services resembling customized portals comparable to iGoogle. A presentation mashup's focus is on retrieving information and layout from different sources without integrating data and functionality (Daniel & Matera, n.d.). The users have the ability to simply drag and drop pre-built components into a common user interface and subsequently re-use and share the results. Data level mashups provide the capability of extracting and combining data from different sources and integrate them in the user's own site. One of their well-known examples is Healthmap.org where relevant data is combined with online maps (Bitzer & Schumann, 2009). Data mashup meanwhile do integrate data and information services from different sources and presents the results in a unified view. They are mostly intended for adhoc business analysis from combining internal data with publicly available information. IBM Mashup Center and Jackbe Presto are commercial examples of this type of enterprise mashup tools. Different kinds of components like information and business service applications can be combined and integrated in via generic interfaces in Functionality oriented enterprise mashups. Lastly Process oriented mashups focus on facilitating the orchestration of information and business application services according to a process sequence for complex enterprise mashups (Pahlke et al., 2010). They

enable user interface integration (Daniel & Matera, n.d.) by combining process orientation with elements from end user driven application development as well as end user participation. The next categorization of mashups is based on user or target groups. This is by far the most commonly associated categorization type of mashups. These user or target groups are differentiated into consumer and enterprise mashups. We will discuss about these two categorizations in details in the following section.

### **2.2.2 Mashup Categories based on Target/User Groups:**

#### ***1. Consumer Mashups***

Consumer mashups, generally associated with Web 2.0, are usually created for personal use for situational problem solving, but could be shared among peers (Zhao, Bhattarai, Liu, & Crespi, 2011). They require a lesser amount of programming expertise. One of the classic examples is showing Craigslist listings on a Goggle Map (Ogrinz, 2009).

Consumer mashups only rely on public web sites that expose well-defined APIs and feeds. The limitation factor here is that the resources used in constructing consumer mashups have to be “mashup ready”. However, most of the consumer mashups tend to be built around a small number of sites with public API. Most consumer mashups tend to be built using the popular Google Maps API. Yu and Woodard (2009) noted that the task of characterizing the mashup ecosystem is made more complicated by how the web of relationships among mashups and APIs has evolved along with the populations of each. The dynamics of these relationships depend upon the process of choosing certain APIs to build on by mashups creators. This in turn depends on decisions of API providers and the expectation of mashups creators’ target audiences. The

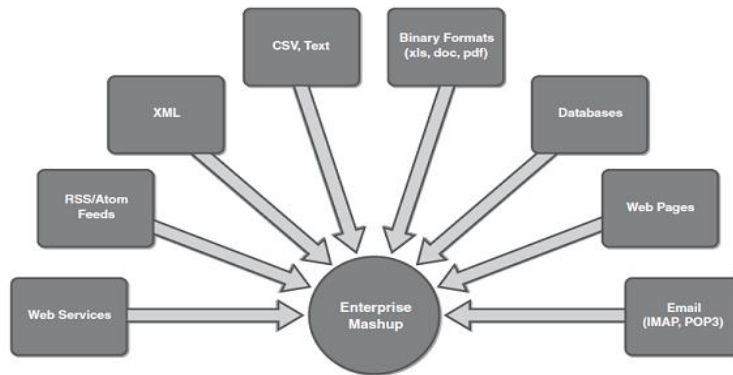
interlinked decisions put mashups and APIs as a single evolving network rather than as independent populations of discrete entities (S. Yu & Woodard, 2009).

## **2. Enterprise Mashups :**

Enterprise mashups have a more complex structure than consumer mashups. Hoyer and Fischer's (2008) definition of mashups is based on 16 definitions from literature of different perspectives: technical, business, application, consulting, software vendor and community. According to them:

*“An enterprise Mashup is a Web-based resource that combines existing resources, be it content, data or application functionality, from more than one resource in enterprise environments by empowering the actual end-users to create and adapt individual information centric and situational applications”*

In addition to the traditional API and RSS feed integration, enterprise mashups' data sources are much more extended in terms of variety, treating the whole World Wide Web as first class data source. While consumer mashup builders depend upon content providers to expose their API, enterprise mashups on the other hand use web harvesting (section 2.3.1 ) to grab whatever data they want (Ogrinz, 2009). Figure 3 depicts the different types of data sources that enterprise mashups can utilize.



*Figure 3: Enterprise Mashups data sources (Ogrinz, 2009)*

### 2.2.3 Mashup Genres

We end our review of mashup classification by listing different mashup genres both in the consumer and enterprise space. The following genre of mashups mostly applies to the consumer mashups category, but none the less also applies to the enterprise space:

**Mapping/Geocoded Mashups:** The prodigious amount of data annotated with location information can be graphically presented through mapping or geocoded mashups. The advent of Google Maps and other mapping APIs have enabled the developers and even the hobbyists and tinkerers to mash all sort of data on to maps.

**Video and Photo Mashups:** The emergence of video & photo hosting sites like YouTube and Flickr API that expose video and photo sharing led to the creating of interesting mashups. With the images and videos having metadata associated with them mashup designers can easily mash these contents with other relevant data.

**Search and Shopping Mashups:** Search and shopping mashups existed long before the term mashup was coined when they utilized business-to-business technologies or screen-scraping to

aggregate price comparison data. Currently with the popularity of mashups growing, consumer marketplaces such as eBay and Amazon have released APIs to programmatically access their content.

**News Mashups:** News sources have used syndication technologies like RSS and Atom to disseminate news feeds related to various topics. These feed mashups can be utilized to create personalized newspaper that caters to the reader's particular interests by aggregating a user's feeds and presenting them over the web (Merrill, 2009).

The following genre of mashups is more suited towards the enterprise usage:

**Client Presentation Mashups:** Client presentation mashups are simple syndication of content from various internal and external providers. Targeted towards non-technical users, these mashups aid in building portals by allowing users to compose individual portlets.

**Client Service Mashups:** These genre of mashups require "build from scratch" as opposed to using packaged widgets and adding them to pages. Through client service mashups end-users or developers can fetch, process, display, and dynamically modify information in a manner the targeted desires.

**Web Services based Mashups:** Based on service-oriented architecture, web service based mashups are built utilizing web service APIs allowing developers to create applications catering to the needs of end-users.



## 2.3 Benefits of Enterprise Mashups

Enterprise mashups have the potential to utilize the creative energy of a large number of people to react in a flexible manner on continuous dynamic changes of the business environment. This technology has the potential to empower knowledge workers engaged in information sensitive applications through the process of transforming individual working environments according to situational needs in an intuitive manner. Through enterprise mashups, knowledge workers have the liberty to enjoy increased flexibility and can react on the ever changing business environment without soliciting help from the IT department (V Hoyer, Stanoevska-Slabeva, Kramer, & Giessmann, 2011). Hoyer et al. (2011) have developed an enterprise mashup benefit model in alignment with the generic four balanced scorecard perspectives. The framework is based on four perspectives: user orientation, operational excellence, future orientation, and financials.

At the core of the enterprise mashup paradigm, which is also the lowest level, is user orientation and the user value proposition. This layer is the starting point for the identification of cause-and-effect relationships driving and supporting the upper perspectives. Increased user flexibility through enterprise mashups enhances user satisfaction and also leads to a faster and better decision making. This benefit then improves competitiveness of the organization in the future orientation perspective, which in turn improves productivity in the financial perspective (V Hoyer et al., 2011). This benefit model indicates that enterprise mashup technology is a key, if not the most important enabler of self-service in organizations. The overarching benefits based on this model are discussed in the following section:

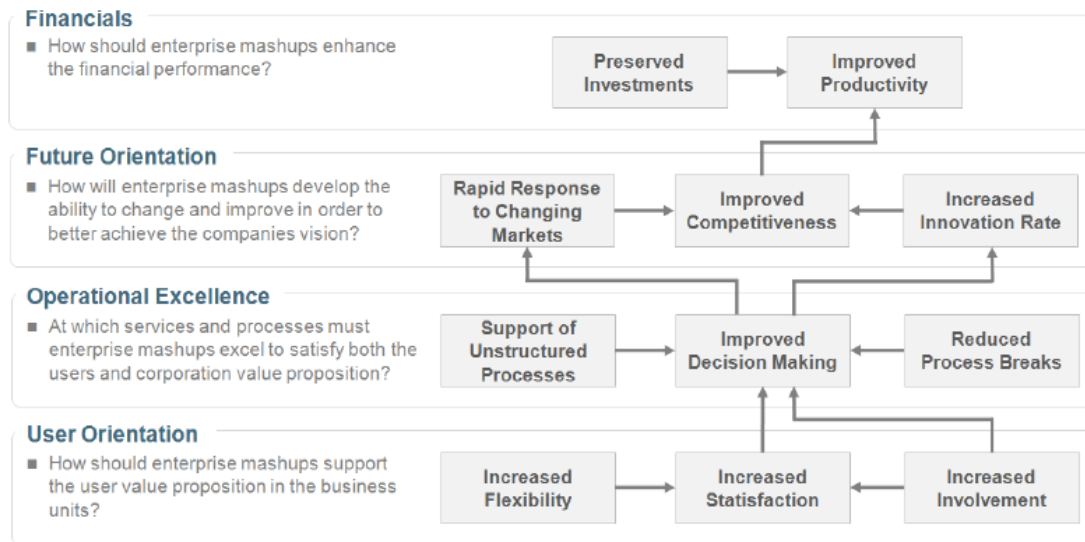


Figure 4: Benefit Model for Enterprise Mashups(V Hoyer et al., 2011)

### ***Increased Satisfaction:***

Through a couple of value propositions enterprise mashups can increase the satisfaction level of knowledge workers. Increased flexibility means the users' ability to design their own working environment by efficiently accessing available internal and external data sources and by creating ad hoc applications with the aid of a flexible mashup platform. The flexibility dimensions provided through this perspective are: information search, process integration and user customizability. Increased involvement refers to the core concept of mashup paradigm relating to the empowerment and involvement of actual end-users as they can create situational applications with little or no programming skills. Users are actively involved in typical community and collaboration features providing valuable feedback to the mashup creator and directly contributing to mashups adoption and improvement (V Hoyer et al., 2011). Two aspects of increased involvement are: networking and communicating personal experiences (Carrier, Deutsch, Gruber, Heid, & Jarrett, 2008). Increased satisfaction is the result of both increased

flexibility and involvement. In addition to these benefit items increased satisfaction also refers to the reliability of information and ease of use.

### ***Improved Decision Making:***

In competitive markets where the window of opportunity is small, the capability to rapidly respond to changing market needs through the operational excellence that mashup platforms provide, accelerates the provision of actionable knowledge and allows immediate implementation. Through support of unstructured processes the seamless integration of different data sources according to the requirements of ad-hoc decision processes reduces the processing time and improves decision making of knowledge workers. Improved decision making refers to the fact that enterprise mashups can also improve the quality of undertaken decisions.

### ***Addressing the ‘Long – Tail’ of Situational Applications:***

The deployment of enterprise mashup platforms seems to adequately fulfill business users’ individual and heterogeneous needs. Based on lightweight composition and orchestration principles as well as easy UI integration, enterprise mashups are the answer to required new development approaches. They incorporate the group of non-technical business users into the development process. The enterprise mashups paradigm represent a promising solution to adequately fulfill address the “long-tail” of requirements. In particular their benefits are listed as:

- Increased business agility, flexibility, and innovation to meet changing business demands
- Problem mitigation between the IT department and business units with regard to poor quality of support, low reaction time, and high cost of adequate IT governance
- Cost reduction by means of higher resource utilization and reusability, as well as lower IT operating and development costs (Pahlke et al., 2010).

### ***Rapid Response to Changing Market Needs:***

Organizations can achieve their visions by sustaining innovation and change through enterprise mashups, as they enable continuous improvement and preparation for future challenges. In competitive markets where the window of opportunity is small, the capability to rapidly respond to changing market needs through the operational excellence perspective accelerates the provision of actionable knowledge and allows immediate implementation. Enterprise mashups also provide improved competitiveness and thus enables faster and better decision making. By rapidly creating mashup applications in order to meet immediate goal or requirement, enterprise mashups delivers are able to respond to these dynamic needs. While increased innovation rate means the enterprises can take advantage of the creativity of a large number of users (V Hoyer et al., 2011). And thus “an ubiquitous laboratory for innovation throughout an organization” is created and competitive advantage is gained (Hinchcliffe, 2007).

### ***Reusability:***

One significant benefit of enterprise mashups that enable rapid response to changing market needs is their reusability. Mashups offer three different aspects to this reusability concept:

With the aid of mashups, reuse is no longer an ivory tower concept limited to only application architects as end users hand in hand with developers will be creating solutions and thus being practiced by everyone.

Mashups impose reusability in a “after the fact” manner where the creators build their own APIs with minimum functionality rather than the traditional method of undertaking additional planning and coding together with front-loads development efforts to create open APIs and extra documentation which may never be used.

Mashups are implicitly reusable, which creates a never-ending cycle of potential associations and recombination, thus eliminating the “dead ends” dilemma resulting from the traditional practices not requiring a system that leverages existing code or libraries itself be reusable (Ogrinz, 2009).

### ***Improved Productivity:***

The benefits experienced through increased satisfaction, collaboration, improved decision making and competitiveness provided by enterprise mashups can have a positive impact on the productivity of an organization and its knowledge workers. As the business users are the best shepherd of the business requirements that are traditionally passed on to IT for development, enterprise mashups can increase the end-results of these applications by empowering the business users to cater to their own needs. This is complemented with the fact that having a mashup platform cuts processing lead time to get any information requests fulfilled by the IT. While enterprise mashups helps in preserving the IT investments on Service Oriented Architecture of an organization which are built on modular components enabling reuse and integration with new systems. Enterprise mashups put a face on SOA leveraging the existing capabilities. Thus this reusability concept stemming from SOA coupled with transferring the long-tail user-requests to the users themselves enables an organization to utilize its valuable IT resources in long term and more complex projects. This can potentially increase the productivity level of both the IT users and knowledge workers. The increased productivity eventually has an impact on the financial perspectives of an organization in a positive manner.

## **2.4 Mashup Patterns:**

A pattern, more specifically a design pattern, attempts to incorporate best practices to achieve a resolution to a commonly occurring problem in software design. Once a problem and its general

reusable solution has been paired and named, it becomes a way of identifying other problems and providing template of how it should be handled. Mashup patterns provide templates for solving different problems and challenges while constructing mashups. These patterns possess the characteristics to be applied outside the context for which they were initially meant to be applied. Mashup patterns, unlike traditional patterns, do not come with an accompanying implementation or sample code which has the tendency to unintentionally sabotage a good solution(Ogrinz, 2009). Mashup patterns typically use one or more of the following core activities as described by Ogrinz (2009), which describe the general capabilities that underline most enterprise mashups:

#### **2.4.1 Activities Related to Mashup Patterns:**

**Data Extraction:** Data extraction refers to a mashup platforms ability to obtain information from closed sources where content is not exposed for programmatic access.

**Data Entry:** Data entry provides the key capability for chaining multiple sites together often going hand in hand with data extraction operation. This activity supplies data to mimic the steps that a user would do in order to navigate to a desirable point while using a web application.

**Data visualization:** Data visualization encompasses the ability to create user-friendly results based on the data collected by a mashup.

**Scheduling and Surveillance:** Scheduling, through automation, determines when a specific task occurs in addition to identifying which resources are engaged for a mashup. While surveillance is referred to the data extraction operation that executes a periodic schedule which records baseline statistics regarding changes in a web resource.

Clipping: Clipping refers to the practice of grabbing a “chunk” of a website as opposed to mining discrete sections of a web resource in the data extraction activity. Clipping allows repurposing of web content without requiring any changes to the underlying code base.

Transformation: Transformation is an essential part of turning extracted data into a useful format, e.g. explicit data casting, changing to uppercase/lowercase, database and table lookups, formatting masks, applying mathematical operations, transforming HTML and binary data.

Action: Action refers to the activity of assigning a specific event to trigger a mashup. This activity provides an alternative where typical data extraction, scheduling, and surveillance tasks are inefficient.

Publication: Publication enables the mashup developers to leverage the economies of collaboration across an organization, by making mashups easily available for usage while ensuring they don’t proliferate to an unmanageable level.

Assembly Canvas: Closely related to data visualization and clipping activities, an assembly canvas provides an environment that supports intercommunication among mashups providing more effectiveness than manually combining mashups.

Ogrinze (2009) has categorized mashup design pattern into five types, which utilize these above mentioned core activities. These patterns are also interconnected to each other. The patterns are further elaborated in the following sections. The sub-patterns of each overarching pattern are described in terms of the problem they address, their solution approach, core activities used in the solving the problem, the related patterns and fragility. Each sub-pattern is given a fragility score ranging from 0 to 5. Any mashup, however thoroughly tested has some degree of fragility, so no pattern is given a score of 0.

## 2.4.2 Pattern Descriptions:

### 2.4.2.1 Harvest Patterns

The main goal of harvest patterns is to mine existing assets for unique data. They describe the method for extracting information previously viewed as closed. Harvest patterns can circumvent specific requirements of a particular product or interface and can circumvent these layers by superimposing a consistent method for access on top of the underlying implementation. The most important characteristic of harvesting is its ability to retrieve data both from structured and unstructured source in addition to harvest the outcome of certain requests and interactions.

Following is a summarization of some harvest patterns:

*Table 3: Summarization of Harvest Patterns (Ogrinz, 2009)*

Pattern	Addressed Problem	Solution	Core Activities	Related Patterns	Fragility
Alerter	Range of data to be monitored is typically enormous	Intelligent Agents are configured to automatically monitor various conditions and trigger alerts. Monitored resources: Web Pages (HTML), email (IMAP,POP3), binary formats (XLS,PDF,DOC), XML, RSS,ATOM, CSV/Text and Databases	Data Extraction, Surveillance	API Enabler, Time Series	1



API Enabler	Valuable content is locked away in closed or proprietary formats.	Create a custom API for static resources (e.g. web pages) so that they can be utilized as a dynamic data source. Data Sources include: Web Pages (HTML), email (IMAP,POP3), binary formats (XLS, PDF, DOC), XML, RSS,ATOM, CSV/Text	Data Entry, Data extraction, Transformation	Infinite Monkeys, Feed Factory	2
Competitive Analysis	Finding out a competitor's product and price offerings	Extracts pricing and product information or advertising trends from competing firms to compare against your own offerings	Data Visualization, Surveillance	API Enabler, Sentiment Analysis, Leading Indicator, Timer Series	2
Infinite Monkeys	Finding the right information consists of a series of dull, recurring tasks that yield value only in a small fraction of cases	Automates a repetitive task to a scale unachievable by normal human agents	Data entry, Data extraction	API Enabler	2
Leading Indicator	Detecting how different series affect one another as leading or coincident indicators	Enables mashups to regularly monitor information that may indirectly serve as a leading indicator	Data Extraction, Data Entry, Transformation	Infinite Monkeys, Time Series	3

Reality Mining	Identifying the undercurrent of activity that forms the patterns of an organization's workforce's daily work-routine	Incorporate environmental and behavioral data to better understand human interaction	Data Extraction, Data Entry, Transformation, Surveillance	Time Series	2
Reputation Management/ Sentiment Analysis	Identify how a company is perceived in the web	Use mashups along with Sentiment Analysis techniques to be scan for words that connote emotion and then rank how a document "feels"	Data Extraction, Data Entry, Transformation	Infinite Monkeys, Time Series	2
Time Series	Tracking internal and external data to improving an organization's decision making process	Use a mashup to extract and store information at regular intervals in hopes of observing trends in the data	Data Extraction, Data Entry, Transformation	Infinite Monkeys, Leading Indicator	2

#### 2.4.2.2 Enhance Patterns:

Enhance patterns provide a template to extend the capabilities of existing resources to get the most value. The goal of enhance patterns is to enable mashups with the capability of making the process simpler and less expensive as opposed to the traditional way of enhancing existing software systems (Ogrinz, 2009). The objectives of enhance patterns listed by Ogrinz (2009) also include

1. Extending applications to a wider audience

2. Fixing bugs without touching the underlying code
3. Making software more user-friendly
4. Improving the “findability” of data
5. Incorporating changing business rules

The following table summarizes the typical enhance patterns:

*Table 4: Summarization of Enhance Patterns (Ogrinz, 2009)*

Pattern	Addressed Problem	Solution	Core Activities	Related Patterns	Fragility
Accessibility	Maintaining unique interface environments for multiple users and devices to ensure existing resources remain compatible with advances in other fields	Construct an alternative application interface with no impact on the original code base	Data Extraction, Data Entry	Field Medic, Usability Enhancer, Widget Enabler	3
Feed Factory / RSS Enabler	Legacy systems and even current may not support RSS	Create an RSS/Atom Feed for a site that doesn't expose a feed, and create new feeds by remixing existing ones	Data Extraction, Data Entry, Transformation	Filter, API Enabler, Widget Enabler	3
Field Medic	When a critical bug is discovered in an essential system, a patch might not be possible to be implemented due to reasons which are out the developers control	Provide a temporary patch to a system when you are unable to correct the problem directly	Data Extraction, Data Entry, Clipping, Data Transformation, Action	Accessibility, Usability enhancer	5
Folksonomy Enabler	Most enterprise class applications are not designed with social tagging concepts	Add community-driven tagging or rating features to existing applications	Clipping	Field Medic, Usability Enhancer	5

Fragility Reducer/ Load Balancer	Mashups are at risk of breaking down when unanticipated changes are made to the systems they connect	Add redundancy to mashups by leveraging multiple sources	Data Entry, Data Extraction, Transformation		2
Smart Suggestions	Most applications rely on training and documentation to aid the user and overburden them numerous required actions to accomplish a single goal	Enhance productivity by using mashups to suggest material relevant to users' tasks	Data Entry, Data Extraction,  Data Visualization, Transformation, Action	Usability Enhancer	2
Super Search	Adding search capabilities to an that doesn't have any, enhance search applications to increase the input parameters and output results	Apply business specific knowledge to enhance user search activity so that results are obtained from multiple sites relevant to the problem domain	Data Entry, Data Extraction,  Transformation	API Enabler, Infinite Monkeys, Smart Suggestions	4
Translation / Language Converter	Receiving information only from biased local sources can lead to unintentional undesired consequences	Pass content through a service to add clarifications or convert it to a different language	Data Extraction,  Data Visualization, Transformation	Accessibility, Field Medic, Usability Enhancer	5
Usability Enhancer	Usability is less prioritized compared to delivery of a working solution	Construct a mashup "wrapper" (or façade) which exposes only the functionality necessary to use the system	Data Entry, Data Extraction,  Transformation	Accessibility, Field Medic	4
Workflow	Adding workflow capabilities to an application that is not originally built to support workflow requirements	Add workflow capabilities to a system or chain of systems	Data Extraction, Data Entry, Action	API Enabler, Usability, Enhancer, Field Medic	5

### 2.4.2.3 Assemble Patterns

Assemble patterns provides a template for best practices for remixing data and interfaces to serve new purposes and thus create something entirely new. Assembly mashups can be delivered ‘right now’ rather than the traditional way of going through formal design and specification phases of application development. Assembly mashups empowers users – both technical & non-technical and thus the organization to quickly create ad hoc systems and data streams that can be used in new solutions to existing problems (Ogrinz, 2009). Below is a summarized table of assemble patterns:

*Table 5: Summarization of Assemble Patterns (Ogrinz, 2009)*

Pattern	Addressed Problem	Solution	Core Activities	Related Patterns	Fragility
Communication and Collaboration	The communication channels that link people together have little or no way to discern the intrinsic value of the information they broadcast creating an interruption overload for the workforce	Combine internal communication products to solve problems related to Interruption Overload	Clipping, Data Entry, Data Extraction	Super Search	3
Content Aggregation	Creating a “single task” from disjointed operations to make up for inefficiency in multitasking	Multiple resources are combined to remove inefficiencies caused by frequent task-switching between applications	Action, Clipping, Data Extraction, Data Entry	Dashboard, Emergency Response, Usability Enhancer	1

Content integration	Managing transparent flow of data to and from applications	Extend a system that accepts an incoming feed by mashing together multiple sources into a new feed that conforms to the original standard. (Provides data exchange techniques for relational databases, web services, XML, RSS,JSON)	Data Extraction, Data Entry, Transformation	Portal Enabler	2
Distributed Drill Down	Drill-down operations typically occur within the same application	Provide Master/Detail functionality across multiple systems	Clipping, Data Extraction, Data Integration	Content Aggregation, Usability Enhancer	1
Emergency Response	Rapid application development in emergencies of drastic nature – where the system and resources of first responders are not designed to interoperate	Create an ad hoc solution in situations where response time is crucial	Clipping, Data Extraction, Data Integration	Smart Suggestions, Super Search, Quick Proof-of-concept	2
Filter	Increased connectivity and complexity of communication has created information overload	Remove unnecessary or unneeded data from a system or data feed	Data Entry, Data Extraction, Transformation, Action	Smart Suggestions, Super Search	2
Location Mapping	Verify accuracy of location data	Geocode data for location mapping or address verification	Data Entry, Data Extraction, Data Visualization, Transformation	Emergency Response, Reality Mining, Super Search, Widget	1

				Enabler	
Splinter	Efficiently distributing aggregated data	Separate a unified data source into smaller, specialized streams of focused information	Data Extraction, Transformation	Content Integration, Filter	3

#### 2.4.2.4 Manage Patterns

Manage patterns deals with the challenges of data management and its relations to mashups by leveraging the investment in existing assets more effectively. More specifically, this pattern category tackles with issues regarding transmission and storage of IT and thus inclined more towards usage by the IT department focusing on tasks like: transitioning between systems, condensing data or processes to align with specific goals, securing access to valuable knowledge (Ogrinz, 2009). The following table summarized the manage patterns:

Pattern	Addressed Problem	Solution	Core Activities	Related Patterns	Fragility
Content Migration / Broadcast / Propagation	Difficulties in moving to alternative platform due switching costs	Migrate information from one or more applications to a new environment	Data Extraction, Data Entry, Transformation	Infinite Monkeys	2
Dashboard	Exposing summaries of internal data (of applications which were not designed to do so) for inclusion within a unified real-time monitoring console	Acquire and display summary status information from multiple systems on a single-page	Clipping, Data Extraction, Data Entry, Data Visualization	Alertter, Content Aggregation, Infinite Monkeys, Time Series, Portal Enabler, Widget Enabler	2

Portal Enabler	Creating compliant portlets can splinter development resources and require continuous maintenance of multiple code bases	Move existing content onto enterprise Portals without requiring custom coding	Clipping, Data Entry, Data Extraction, Transformation	Content Aggregation, Widget Enabler	2
Quick Proof-of-Concept / Prototype	Maximizing available resources to determine which solutions are worthy of initial exploration and subsequent funding.	Use mashups to validate a business or product idea that will entail a significant investment	Data Entry, Data Extraction, Data Visualization, Scheduling, Clipping, Transformation	Content Aggregation, Content Integration, Emergency Response	2
Single Sign-on	Absence of integration an internal “password vault” with SSO solutions will create gaps for the security to breached	Allow a user to supply credentials one time for authentication across multiple internal and external systems	Clipping, Data Entry, Data Extraction, Scheduling	Field Medic, Usability Enhancer	2
Widget Enabler	Portal platforms have limited ability to adapt to the diverse needs of the users and requires maintaining a huge infrastructure	Repackage existing systems for viral distribution via popular Widget platforms	Data Entry, Data Extraction, Transformation	Portal Enabler	2

*Table 6: Summarization of Manage Patterns(Ogrinz, 2009)*

#### 2.4.2.5 Testing Patterns

Testing patterns enables us to see how mashups can be used to perform basic testing functions across a wide range of platforms and technologies otherwise limited to products costing enormous sums of money.



Pattern	Addressed Problem	Solution	Core Activities	Related Patterns	Fragility
Audit / Aspect Oriented Mashup (AOP)	Lack of existing multisystem audit capabilities can result in misuse of application	Use mashups to create an aspect-oriented view of application usage	Data Entry, Data Extraction, Transformation	Field Medic, Usability Enhancer	2
Load Testing	Mashups can create applications or services outside the capabilities of test products contained within an enterprise	Multiple mashups run simultaneously can simulate the activity of hundreds of users and assist in load and stress-testing	Data Entry, Data Extraction	Audit, Infinite Monkeys	2
Regression Testing	Changes to a third-party product can detrimentally affect a mashup application	By employing a predefined collection of data, ensure that input/output results across versions are as expected	Data Entry, Data Extraction	Content Migration, Infinite Monkeys	1

*Table 7: Summarization of Testing Patterns(Ogrinz, 2009)*

Overall these patterns can be adapted to an organization's needs and implemented with the products they use within their boundaries.

### **2.4.3 Mashup Design and Architecture:**

There are several mashup architectures mentioned in the literature, but a lack of a single, commonly accepted conceptualization of mashup architecture is evident. For our purpose we chose the design and architectural model of Pahlke et al. (2010). The architectural elements upon which their model was based one are described below:

1. Resource: Content, data, and functionality resources which are accessible through established but specific APIs. Resources are also termed as assets.

2. Component: Also referred to as Mashlet, Widgets or Gadgets, these are virtualized components that can be easily “mashed” through generic APIs or UIs
3. EM (Enterprise Mashup) Application: A lightweight application combining components from different sources.
4. EM Platform: Also referred to as EM system, this is overall technology that provides functionality to create, deploy, modify, and share EM applications.
5. EM environment: Consists of technical platform as well as the organizational structures and actors.

Based on these elements, figure 5 summarizes the conceptualization with regard to the involved actors and their roles in the development and allocation process.

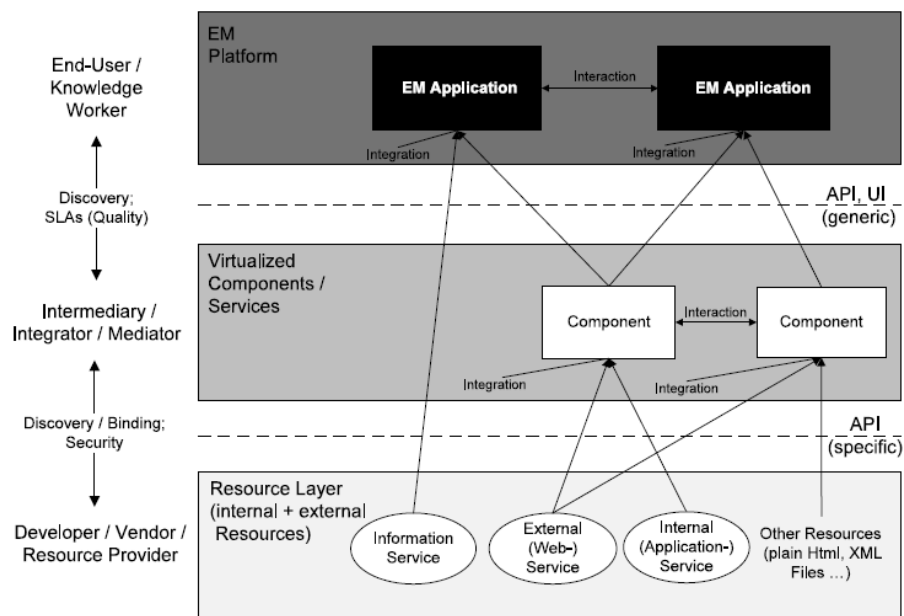


Figure 5: Enterprise Mashup Environment (Pahlke et al., 2010)

Internal and external resources are located in the lowest layer and are abstracted by standardized interfaces to facilitate loose coupling. On the immediate upper level, the mediator through APIs

or UIs virtualizes the resources and integrates different resources into usable and shareable components. The goal here is to provide additional graphical and simple user interaction mechanism abstracting from underlying resources and the corresponding technical interfaces. Built upon these two layers, the highest level of abstraction enables knowledge workers to create, adopt, use, and share EM applications. To further illustrate the integration of different component models we have identified the J. Yu, Benatallah, Casati, & Daniel(2008) mashup characterization approach. Here the mashup paradigm is categorized from two perspectives 1) Components, objects of integration and 2) Composition, how objects are glued together. Table 11 summarized these models:

Model	Category	Properties
Component Model	Type	Can be Data, Application Logic, User Interface
	Interface	Exposes: CRUD interface, APIs or GUI elements
	Extensibility	Ability to create or extend new components
Composition Model	Output type	Data, Application Logic, User Interface
	Orchestration	Flow Based Event Based Layout Based
	Data-passing style	Dataflow and Blackboard Approach

	Others	Instance Based or Continuous
		Exceptions and Transactions

*Table 8: Mashup component and composition model summary (J. Yu et al., 2008)*

#### **2.4.4 Mashup Development Classifications:**

Mashup development has been classified into different models in the literature. We have chosen two models which are described in the following sections.

##### **2.4.4.1 Manual and Tool-Assisted Mashup Development**

J. Yu et al. (2008) had classified the mashup development process into manual and tool-assisted development approaches.

Manual mashup development requires programming skills and intimate knowledge about the schemes and semantics of data sources or the business protocol conventions for integrating data and applications into a coherent and value-adding application. Even though new technologies have simplified mashup development, manual mashup development is still a prerogative of skilled developers (J. Yu et al., 2008).

Various mashup-specific development tools and frameworks have recently emerged in order to accelerate the mashup development process and to enable business and inexperienced users to mash up their own applications. The basic functionality of these tools can be further explicated with the Volker Hoyer & Fischer (2008) classification model:

Catalogue	Adapter
	Repository
Editor	Transformation/Aggregation
	Presentation Layer

*Table 9: Functionality classification of mashup tools*(Volker Hoyer & Fischer, 2008)

A catalogue consists of libraries of existing resources and widgets. The adapter within the catalogue integrates existing resources types both on syntactic and semantic level, while a repository organizes number of resources and widgets in the internet of services. The editor meanwhile allows creating and modifying and aggregating of individual software applications by connecting resources retrievable from the catalogue. The transformation and aggregation capability of the editor allows the users to combine data and content according to lightweight resource composition style by reusing building blocks in different contexts; meanwhile the presentation layer presents content from disparate sources together in a unified view and runs the composition.

#### **2.4.4.2 Mashup Builders and Enablers:**

Liu et al. (2011) categorized mashup development resources into two groups: mashup builders and mashup enablers. Mashup builders are tools that produce user interface for Mashups. They enable non-developers to compose Mashups by connecting widgets to create composite applications. Mashup enablers serve functionalities to Mashup builders by accessing unstructured data and making internal and external resources available. They have also noted that some tools capable of playing both roles.

### ***1. Mashup Builders:***

Mashup builders are tools that facilitate and enable mashup creation providing user interfaces, widgets and other building blocks. These tools enable non-developers to compose Mashups by connecting the provided widgets to create composite applications. According to Liu, Liang, Xu, Staples, & Zhu (2009) mashup builders define the workflow to connect data and create composite applications. For our study purpose we have chosen two mashup builders: IBM Mashup Centre and JackBe Presto. These two products are mashup building platforms aimed at non-technical users as well as users with programming skills. In general they both provide mashup building capabilities using web information. But they are not specifically targeted for using in a BI context, but rather for situational application development as whole. Below we have given a brief overview of the products.

#### **a) IBM Mashup Centre:**

IBM Mashup Center is a product of the information management category of the IBM product family. It provides quick application building capabilities by utilizing, remixing, and transforming various information feeds. Through rapid assembly and sharing of mashups, without the requirement of coding, this mashup tool enables situational application development. Mashup Center is composed mainly of two loosely coupled components through a common catalog : 1) InfoSphere MashupHub and 2) Lotus Mashups (Kasman & Roder, 2011a).

##### ***i. InfoSphere MashupHub:***

The InfoSphere MashupHub component is used for creating the feeds and feed mashups for consumption by the Lotus Mashups. Through a common catalog which is used by both the components, the created feeds, feed mashups, and other resources stored, published and shared.

In terms of technical infrastructure the InfoSphere MashupHub is a browser based application development tool. It is built using Ajax and uses web services, specifically the REST protocol, to communicate with the application server. The different components of the InfoSphere MashupHub client are targeted towards users with different expertise level. It consists of three main sub-components : 1) Feed Creator 2) Mashup Builder 3) Catalog (Singh, 2008a).

#### *ii. Lotus Mashups:*

The Lotus Mashups is the mashup presentation builder tool which uses the mashups that are created in the MashupHub. It is used to create web pages to display the data feeds using pre-built widgets. The pages can also consist of external information sources. The presentation composition is done through a graphical user interface which doesn't require any programming tasks. The interaction of different widgets within a mashup page is done through wiring which allows the passing of different events to different widgets (Kasman & Roder, 2011b; Singh, 2008a).

#### **b) JackBe Presto:**

The Presto mashup tool is a product of the enterprise application vendor – Jackbe. The Presto mashup tool allows a different array of users to combine data from various sources. Through this mashup tool both historical and real-time can be utilized to create the mashups. The overarching functionalities of Presto are very similar to the IBM Mashup Center. Having said that, Presto provides extra features to create mashups targeted for various devices and takes up “app development” approach. Being a vendor which focuses on creating mashup tools, JackBe's

Presto mashup platform enables mashup creation for enterprise users in a Web 2.0 style. Below is a brief overview of the core components of Jackbe Presto (Jackbe, 2012).

***i. Presto Hub:***

Presto Hub is the collaborative workplace for undertaking the tasks for mashup development. The workplace consists of features that enable power users, developers, and administrators to create mashables and mashups, create and publish apps, and manage all the created artifacts. The Presto Hub is constructed using the following sub-components (Jackbe, 2012).

- a) *Mashboard*: A visual drag and drop development environment allowing users to assimilate or integrate multiple apps or views in a dashboard in order to solve challenging business problems or to cater to an ad-hoc need.
- b) *Wires*: By providing easy access to all the mashables and mashups available within Presto, the Presto Wires tool provides a graphical interface embedded with drag and drop features for creating mashups. This interface is targeted for business users allowing them to create mashups without the need for coding.
- c) *Mashup Editor*: Mashup Editor is a web-based authoring tool for developers for constructing the mashups through the Enterprise Mashup Markup Language (EMML). This tool is aimed at users with programming knowledge.
- d) *App Maker*: Through App Maker in Presto Hub, users can create basic apps utilizing the information sources or created mashups with the help of visual wizard.

***ii. Presto Repository & AppDepot:***

Presto has two main resource catalogs which are used for different purposes. The Presto Repository is used for meta-data for all mashables, mashups, views and apps, while the



AppDepot stores all the created apps which makes it easier for users to find the app which will suit their need (Jackbe, 2012).

## ***2. Mashup enablers:***

Mashup enablers are defined as mechanisms by which mashup builders access required data sources, functionalities making both internal and external resources available. Mashup enablers can be specific tools just like mashup builders or could be specific technologies which enable builders to perform their intended tasks. Two examples of mashup enabler technologies are Web Services and SOA which are discussed in the following sections:

### ***1. Web Services***

According to W3C “A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format”. Web services are based on two opposing types of standards SOAP and REST (zur Muehlen, Nickerson, & Swenson, 2005). Both utilize the transport layer for creating communication between client and servers (Hildebrand, Shankland, & Baya, 2012). The standards differ in terms of the construction mechanism. SOAP is more formal, bureaucratic, and rigid in nature and is able to create complex and proprietary mechanisms to connect to components. While REST is more simple, language & platform agnostic, and requires less complex skill set to play around with.

#### ***i. SOAP***

The SOAP based web services framework is divided into three areas – communication protocol: SOAP, service description: WSDL, and service discovery: UDDI. A SOAP message has a very

simple structure: an XML element with two child elements, one of which contains the header and the other the body. A WSDL or a Web Service Description Language Document describes a web service's interface and provides users with a point of contact (Curbera, Duftler, Khalaf, & Nagy, 2002). The Universal Description, Discovery, and Integration (UDDI) specifications offer users a unified and systematic way to find service providers through a centralized registry of services that is roughly equivalent to an automated online “phone directory” of Web services (Curbera et al., 2002).

## *ii. RESTful Web Services*

Representational State Transfer protocol or RESTful web services enables the permeable enterprise, in which capabilities and assets inside the enterprise are easily combined with assets and capabilities outside the enterprise. In this standard a client communicates with a server—not directly with the source of information on that server. REST uses simple HTTP and therefore standard commands—such as GET, PUT, POST, and DELETE—to coordinate communication between clients and servers. In RESTful designs, the client does not need to know about the implementation on the server. The server is free to store data as it likes, and the client can store the same information differently. This loose coupling means that as long as the interface is stable, the implementation on the client or the server can independently change. This independence creates flexibility in distributed software systems (Hildebrand et al., 2012).

Hildebrand et al. (2012) also listed six constraints which are required to be met in the REST architectural style:

1. Client/server loose coupling: A clean separation of duty exists between client and server.

The type of data storage does not matter to the client, and the client interface or client

state does not matter to the server. With a stable interface, the client and server may be developed and replaced independently of each other.

2. Stateless: The interface that dictates how the client and server interact does not allow client states to be stored on the server. Information about client states is embedded in the messages the clients send to servers.
3. Cacheable: Clients can have the ability (and must let the server know whether they do or not) to temporarily store data received from the server.
4. Layering: Servers do not know whether there are layers of abstraction between themselves and the end client; for example, whether they are passed through multiple security policies, APIs, and so forth.
5. Code on demand: Servers are able to temporarily send custom functions as executable code to clients for them to execute.
6. Uniforms interface: Servers and clients can interact, change, and be modified independently as long as the interface that binds them remains the same (Hildebrand et al., 2012).

For this project we will be using the RESTful web services. In the IBM CMS section (2.9.1.2) we will delve into more details of the RESTful API of Cognos Mashup Service with specific examples.

## ***2. Service-Oriented Architecture***

In the Service-Oriented Architecture paradigm, data and logic functionality are encapsulated with only their input and output being exposed to others for usage (Hirschheim, Welke, & Schwarz, 2010). The “service” in SOA is referred to a business task rather than a specific technology. Loose coupling with other services facilitates the implementation of these tasks. This

in turn, fosters an atmosphere where developers have the capability to create new applications that can reuse and recombine existing functionality. Their underlying open standards enable them to be equally consumed across independent development platforms (Ogrinz, 2009).

SOA, in essence is more than just an IT architecture as it caters to the services being defined by IT in response to ongoing operational needs and business functional requirements. The need for a broader business architecture from which higher level applications, be it coarse-grained or composite, can be readily applied to the development of lower level services is driven by SOA. Web service interoperability standards provide a consistent and interoperable basis for building SOA-conforming capabilities (Hirschheim et al., 2010). The SOAP based web services have become the industry preferred method for implementing SOA (Ogrinz, 2009). Having said that, the RESTful web services are also utilized in the SOA paradigm(Hirschheim et al., 2010)

The desired vision of SOA is that of a strategic initiative that involves both business and IT. Hirschheim et al. (2010) mentioned the following benefits associated with the above viewpoint:

1. Increased flexibility and agility
2. Inter-organizational relationship and value-stream improvement
3. Common view on key entity information and
4. Improved business processes and customer touch points

### ***Mashups and Service-Oriented Architecture***

Mashups are both a precursor and a beneficiary of the SOA paradigm. With both SOA and “web services” being used interchangeably, implementing a successful SOA will require the service-enablement of their existing applications. Enterprise mashups are a means of accomplishing this requirement as we have seen from the API enabler pattern (section 2.3.1). Enterprise mashups

are capable of both producing services as well as consuming them with the same level of agility and thus leverage the SOA centric organizations' web service resources (Ogrinz, 2009). However the desired outcome of enabling easy access to the functionalities of services is still to reach a level for paving the way towards a vision of "internet of services" and "web service ecosystem". Mashups have the potential to fill this gap through their user-centric and lightweight approach of creating simple services or tools from any web resource. The adoption of mashups at the enterprise level, in the context of SOA, can be used to empower the consumers of pre-defined SOA solutions to become producers of their own applications which suit their actual requirements and specific needs. The creativity and productivity of an individual user, who is an expert in his or her own domain, can be leveraged to improve service-to-user interaction. Moreover, easy access to services is not only beneficial for large organizations, but also helpful in increasing the SOA acceptance among new target groups in small and medium sized enterprises both in non-profit and private sector. An enterprise mashup platform supporting simplified service integration, composition and application design concepts can be a crucial point in a user-centric SOA (Nestler, 2008). The following table shows that, as mentioned by Bitzer and Schumann (2009), that the different intentions of SOA and mashup clearly show that both architectures are not mutually exclusive, but complement one another.

Feature	Mashup	SOA
Design of Architecture	Use of loosely linked and encapsulated services for the connection of dispersed applications	
	Well defined, standardized interfaces	
	Creation of assembled services	
	High degree of agility	
	User as the essential element of the value proposition	Connection of separate business functions
	Freedom of structure and content	Ex-ante restriction via control and governance processes
	Internal and external services	Mostly internal services
	For situational applications	For complex and standardized business processes
Design of Technology	Independence of system software and programming language	
	Integration through standardized interfaces	
	Reuse of components, little programming work	
	Web services as mainly used services	
	User (functional department) integrated directly into the developing process	Speciality department only integrated into the developing process in cooperation with the IT-department
	Orchestration of data and services by Mashup Building Platforms	Orchestration of data and services by BPMN and BPEL
	Semantic interpretation by the user	Semantic interoperability has to be guaranteed
	Trend towards REST Web services	SOAP Web services as de facto standard
	Connection of services and data in widgets	Connection of services as complex services

*Figure 6: Comparison summarization of SOA and Mashups (Bitzer & Schumann, 2009)*

Having said that, while SOA generally focuses on server-side architecture and internal corporate resources, mashups have a certain “gung-ho” approach. Mashup design patterns and standards are still at a nascent stage while SOA’s maturity has reached greater clarity in terms of its capabilities, protocols, implementation and use. So mashups should focus on practical examples which would drive broader adoption leading to consolidation and standardization similar to what SOA has achieved (Ogrinz, 2009).

## 2.5 Business Intelligence

*Business Intelligence* (BI) is an IT framework that helps organizations in managing, developing and communicating their intangible assets such as information and knowledge. BI architectures include data warehousing, business analytics, business performance management and data mining and mostly deal with structured data. BI is an umbrella term that combines architectures, tools data bases, applications, practices and methodologies. Wiess et al. (2003) defined BI as the the combination of data mining, data warehousing, knowledge management, and traditional decision support systems. A more structured definition was given by Alnoukari et al. (2012) :

*“Business intelligence is the use of all the organization’s resources: data, applications, people, and processes in order to increase its knowledge, implement and achieve its strategy, and adapt to the environment’s dynamism”*

### ***Business Intelligence vs Business Analytics:***

When it comes to differentiating between business intelligence and business analytics (BA), there is no established common academic or industry standard. Within the industry they are both defined separately and used interchangeably in some cases. When comes to drawing a line between these two terms, BI as mentioned above, deals with taking the information sources and converting them into knowledge aiding the decision making process. Based on the existing business data that an organization has, BI systems use a consistent, repeating set of metrics to steer future business strategy and setting benchmarks for the future; while BA focuses on using data to set new insights through statistical or predictive analytics. Dealing with static and historical data, traditional BI systems often fail to make predictive decisions and predict the future market. Some vendors thus use BA as an umbrella term which includes data warehousing,

BI, enterprise information and performance management, analytic applications along with governance, risk and compliance (DellaPorta, 2012; Elliott, 2011).

Having said that, the goal of BI solutions is to turn data into information and subsequently to knowledge after accessing it from multiple sources and transforming it. These processes work towards improving the organization's decision making capabilities. The measure of any business intelligence solution is its ability to derive knowledge from data. The challenge is to meet the ability of identifying patterns, trends, rules, and relationships from large amount of information which is too large to be processed by human analysis alone (Alnoukari, Alhawasli, Alnafea, & Zamreek, 2012).

Business intelligence applications can be divided into the following three layers:

1. Data layer: Responsible for storing structured and unstructured data for decision support purposes. Data are extracted from various data sources, i.e. structured data from operational data stores (ODS), data warehouses, and data marts while unstructured data from SCM, ERP, CRM or from external data sources. Once extracted, data is transformed and loaded into data warehouse by ETL tools
2. Analyze layer: Provides analyzing functionality of data and rendering as knowledge. This consists of OLAP, data mining, aggregations etc.
3. Visualization layer: realized by some sort of BI application interface or portals (Alnoukari et al., 2012).

BI success of an organization is related to the positive value an organization obtains from its BI investment. Implementation of BI is targeted towards achieving a variety of organizational benefits such as improved profitability and efficiency, reduced costs etc. But specific BI success



measures vary across organizations and even across particular BI implementations within an organization. A lack of fit between an organization's BI implementation and its goals & characteristics is one reason for a lack of BI success (Işık, Jones, & Sidorova, 2013).

The extent through which an organization can leverage business intelligence is related to the capabilities of its BI system. Işık, Jones, & Sidorova (2013) have identified 5 capabilities from an organizational BI perspective:

1. **Data Quality:** Traditionally, BI has largely relied on structured and/or numerical data, which can be measured on a numerical scale and analyzed with statistical methods and computing equipment. However, in an increasing number of BI application areas, the collection and analysis of qualitative and/or unstructured data especially from external sources are critical. As companies incorporate data from a wider variety of sources, they will continue to face new and ever-increasing issues surrounding the quality of the data on which they rely.
2. **Integrating with other applications:** The integration between BI and other systems in an organization is another critical factor for BI success. For organizations that use data from multiple sources and feed the data into multiple information systems, the quality of the communication between these systems directly affects the overall performance. The growing number and variety of data sources for BI in many organizations place increasing pressure on the integration between the systems from which the data are sourced. The higher the quality of integration of BI with other systems in an organization, the greater the BI success.

3. **User Access:** One single BI implementation does not cater to the need of every user. Building, supporting, and managing multiple vehicles for a variety of user access methods and to support a variety of analyses is a critical BI capability. It is critical that organizations achieve the necessary balance to allow the way BI users access information to fit the types of decisions they make using BI. The higher the quality of user access to BI in an organization, the greater its BI success.
4. **Flexibility:** Flexibility is the organizational capability of BI to provide decision support when variations exist in business processes, technology or the business environment in general. To achieve the competitive advantages provided by BI, organizations must select the underlying technology to support the BI operations carefully; flexibility is one of the most important factors to consider. Ideally, the system must be compatible with the existing tools and applications to minimize cost and complexity. The level of BI flexibility positively influences BI success.
5. **Risk management support:** Risk management support refers to the organizational BI ability to support decisions under conditions of uncertainty when not all the facts are known. For example, innovative organizations, which are typically considered risk-tolerant, rely on BI to make entrepreneurial decisions motivated by the exploration and discovery of new opportunities and new risks. BI may be more successful if it has the ability to address risk in the decision making environment (Işık et al., 2013).

### 2.5.1 Business Intelligence Tool: IBM Cognos BI:

*IBM Cognos Business Intelligence* tool (Browne et al., 2010) is one of the popular tools in the marketplace in the field of BI. This BI tool has a three-tier architecture. Each tier is comprised of several components which communicate with each other using web-services.

#### 1. Presentation Tier:

The presentation provides a web user interface for accepting user commands. This layer is also the outlet for rendering report and layouts.

#### 2. Middleware Tier:

The middleware tier is responsible for routing each request to the proper service as well as for querying and outputting the data requested by the web client.

#### 3. Data Tier:

The data storage and maintenance of different data stores within the Cognos content store the responsible tier is the data store. The *content store* is the internal data store used by the Cognos server and stores information on all the artifacts related to business intelligence such as folders, reports and report specifications (Browne et al., 2010).

For our research purpose we have used a couple of the services of the middleware tier: 1) Cognos Software Development Kit (SDK) 2) Cognos Mashup Service (CMS). These two services are further elaborated in the BI Mashup Development Tools section.

### 2.5.2 Situational Business Intelligence

As we have mentioned in the previous sections business critical systems and applications requiring high availability, scalability which are typically requested by a large number of users. But besides these critical applications, the “long tail” of situational applications exists because of

the growing amount of unstructured data on the web and conventional BI tools do not address them. However situational BI applications tackle this problem as they tap into this wealth of unstructured information in order to determine new trends and give enterprises the competitive edge(Löser et al., 2009).

Increasing release of government & enterprise data and emerging web services in the context of Web 2.0 developments, which offer valuable opportunities for computer-assisted-decision support processes, are driving the development of situational BI (Thiele & Lehner, 2012).

### ***End-user requirements and Traditional Approaches in Delivering Situational BI services***

Situational BI needs of an organization have been traditionally catered to by interaction between the IT unit and other departments. On a technical level, “spreadmart” solutions are used. The successful operation of underlying infrastructures upon which BI systems are built requires the integration of expert knowledge. When it comes to formulating their contextual operational requirements for IT to implement, there is no better entity than the department themselves. However co-operation between IT and other departments often creates complications in practical scenarios due to demand for situational needs.

#### ***1. Delivering Situational BI services: Organizational Approach***

From an organizational point of view through interacting with the IT, the service delivery of situational BI applications can be categorized in to a three step process Information provision, Resource Requirements and Distribution of Competencies (Thiele & Lehner, 2012).

#### ***Step 1: Information Provision Process:***

Distinguished departments state their requirements to the IT unit in the form of projects to be implemented within a certain timeframe. The information provision process is illustrated in figure 7. It is evident from the process description that the information provision process may occasionally be subject to significant complications and resulting delays. If the data acquisition processes or development cycles are too slow, solo-development activities are initiated by individual departments. Such initiatives create data silos within an organization introducing unnecessary redundancy regarding the data storage and the loading or extraction processes (Thiele & Lehner, 2012).

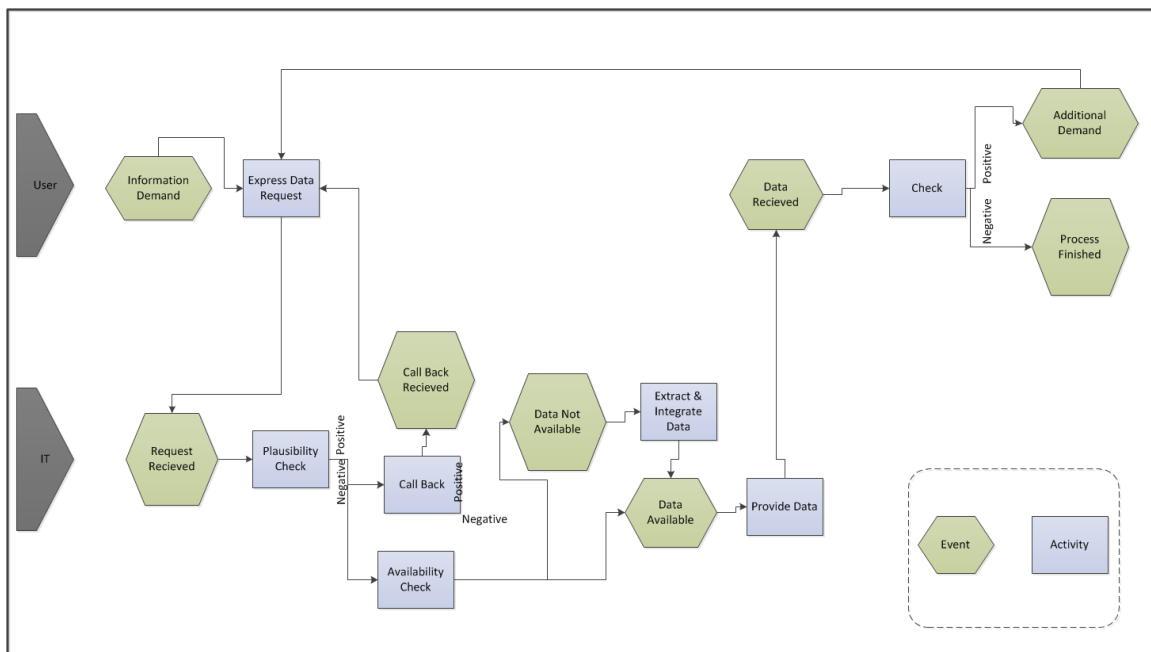


Figure 7: Information Provision Process (Thiele & Lehner, 2012)

### ***Step 2: Gathering Resource Requirements:***

Most requirements of an individual department are driven by an extensive specification of the departmental logics to be implemented. This is a prerequisite for the functional solutions that will be provided by IT. Additional iteration steps causing more overhead appear when the

specification is incomplete or inconsistent. Guaranteeing the technical accuracy of the services by IT and frequent changes to the departments' requirement in these situational analysis scenarios would increase the overall cost of development. Furthermore, rigid SLAs act as obstacles in delivering situational requests. This means any changes to the requests have to follow systematic and standardized procedural model and prohibits the change the existing data. Data production processes already in place must not be delayed beyond the period specified in the SLAs. In addition, as the IT unit aims to maintain their resources in a cost efficient manner, there are a lot of automated processes employed which conflict with situational data analyses (Thiele & Lehner, 2012).

### ***Step 3: Distribution of Competence:***

Once the requirements are gathered, the responsibilities are assigned to the subunits within IT and even to the departments in some cases.

#### **2.6.4.2 Enhancing the Organizational Approach through Competence Centers**

BI Competence centers act as link between the IT unit and all the departments formulating the BI enterprise strategy and consist of members from all across the organization. The information retrieval process is accelerated through the creation of a competence center as communication structures are improved and the information requirements can be planned in a strategic and predictive fashion. A competence center is the first point of contact for the departments when retrieving information from IT (Thiele & Lehner, 2012).

However, the concept of competence center is meant for improving the communication between the IT unit and other departments when it comes to service provision. IT is still in charge of developing and providing solutions to the situational needs of their clients, following the traditional development methods, the business users are still dependent on the IT unit for getting them what they need. The absence of self-service would still hinder the effective usage and adoption of BI systems.

## ***2. Technical Solution: Spreadmarts***

If the information provision process is too tedious or too expensive when it comes to support situational application development, manual individual solutions termed “spreadmarts” come in place. According to Eckerson & Shermann (2008):

“A spreadmart is a reporting or analysis system running on a desktop database (e.g. spreadsheet, Access database, or dashboard) that is created and maintained by an individual or group that performs all the tasks normally done by a data mart or data warehouse, such as extracting, transforming and formatting data as well as defining metrics, submitting queries, and formatting and publishing reports to others.”

However, the use of spreadmart comes with high risks. The spreadmart’s lack of compliance with IT means that the data generated with them is less reliable in terms of quality and consistency. Different departments use their own calendar definitions, naming conventions, and practices for analyses. Furthermore, increased costs are resulted due to the data integration and refinement process in this method not being a part of business analysis unit’s tasks. Despite the mentioned risks, 90% of all organizations use spreadmarts and devote 40% of their time in

creating them (Eckerson & Shermann, 2008). The drivers for organizations still using them are the delay in service delivery, higher degree in autonomy, the desire to protect interests, lower costs, and the absence of situational applications in the first place. In addition to the availability of spreadmart solutions in widely used tools such as Microsoft Excel, Powerpoint and Access plays their part in the alarming number of these type of solutions being used (Eckerson & Shermann, 2008).

### **2.5.3 Business Intelligence & Self-Service**

With the wide adoption of Web 2.0 in enterprise application architectures, the demand for self-service and do it yourself approach is also increasing. In terms of an IT perspective Oliver, Livermore, & Farag (2009) have described self-service as a process in which “aspects of customer service experience that used to be provided by the company’s employees are now provided by through the interaction of customers with the company’s website ”. In essence this process turns customers or service consumers of IT services into employees. From an organizational perspective, web-based self-service is seen as a cost-effective way of managing client interactions and inquiries than are channels that require especially IT assistance (Oliver et al., 2009). In the world of business intelligence, the notion of self-service for the clients of BI tools is an important one. In fact it is at the heart of new age BI. Requirements of BI are changing faster than what the typical IT centric support models are designed to keep up with and will carry on challenging even the most up-to-date BI deployments. The never-ending stream of requests for access to new sources, data, models, reports, dashboard, queries and applications, added with the unpredictable possibilities of organizational change in terms of sudden mergers and acquisitions, new competitive threats, new management structure or even changes in the regulatory reporting requirements can make BI applications out-dated rapidly. The business users



desire and necessity to address the requirements to serve their customer better comes in conflict here with IT's effort to maintain law and order BI application development by sticking to standard BI tools and following approved software development and project methodologies. Evelson (2010) has suggested that in an ideal BI environment 80% of all the requirements should be carried out only by the business users. IT resource constraints within an organization mean the BI service request backlogs remaining stubbornly long and increasing. The characteristics of self-service requirements in BI can be summarized by the following table:

What Users Want	Delivery Method
Speedy answers	Fast access, loading, mediation, and virtualization of canonical views and information objects
	Shorten time to answer business questions
Single view of everything	Structured and unstructured, internal and external
	Unified access, delivery, and presentation
Single version of truth	Authoritative reference information conforming to standard dimensions and hierarchies
	Matched, merged, cleansed, transformed, and enriched
Self-service information exploration	User-defined mashup of reusable, personalized, context-rich, role-tailored view of information
	Collaborative sharing of reusable user mashups

	Interactive, deeply dimensional drilldown
--	---

*Table 10: Information Workers' Decision Support Nirvana (Kobielus, 2009)*

The growing user demands which the IT staff is finding harder day by day to meet, is driving companies towards a leaner self-service BI environment. Through self-service users can quickly assemble a single view of their required reference data in addition to manipulating and exploring it in order to support their decision making task at hand. The major benefits of self-service BI listed below.

**Eliminate BI bottleneck:** By reducing the required assistance from IT for service provisioning in terms of ad-hoc queries, reports, dashboards, consolidated view of data, situational application and other typical user requests, self-service BI minimizes the time to deliver these functionalities to users and alleviate IT backlogs associated with designing, deploying and maintaining these services. Thus application developers and data modelers can focus on higher-value, more complex and long-term BI projects.

**Reduce BI costs:** One of the important if not the main goal of self-service notion is to provide cost-effective mechanisms in service provisioning for organizations. By transferring development and provision of typical BI requests to the users themselves, organizations can introduce cost-effective BI implementations by reducing the need for expensive BI application developers and data modelers. Subsequently organizations can evolve and expand their BI implementations without the need of adding more specialized BI professionals.

**Enhance BI decision support:** As businesses move faster than even the most agile IT departments, it is imperative for the business users to have the required BI resources to and make sense of their data right away, right now. By enabling the business users with the capability

retrieve the data they require and create their own personalized views, manipulations, and calculations on the fly, self-service BI contributes to superior decision support (Kobielus, 2009).

Having said that, in order for the self-service notion to be widely adopted Evelson (2011) has described the presence of following features and capabilities as an essential for self-service :

1. Data virtualization: The capability to virtually link multiple data sources.
2. Exploration and Discovery: Analyzing information based on a new hierarchy not already built into a data model.
3. Collaboration: User-to-user and user-t-developer collaboration functionality.
4. Search-like GUI: Similar GUIs that the users are used to using when it comes to consumer applications.

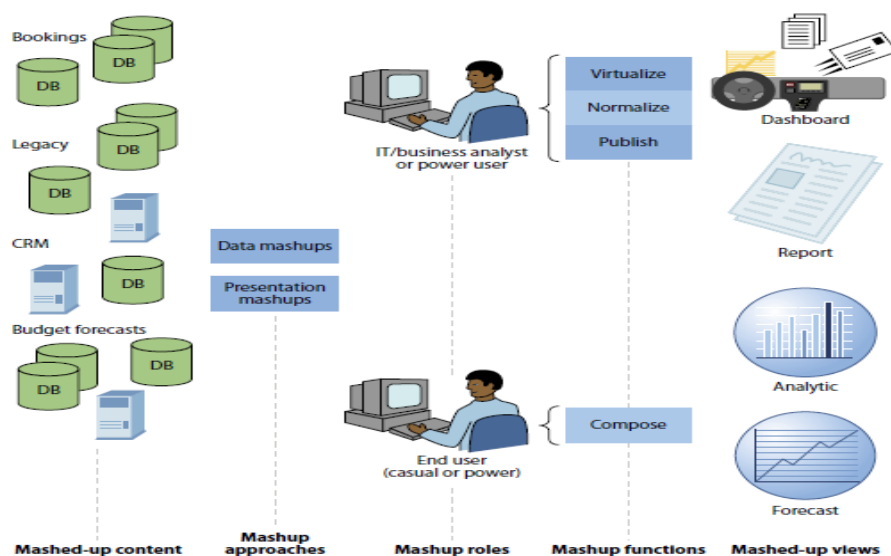
With the advancement of Web 2.0, rich internet applications integrated with point-and-click and drag-and-drop graphical user interface (GUI) are the current face of the internet for service consumers giving the users an array of mashing capability to create their personalized application views. In the BI domain, mashups have the potential to take self-service to the next level of sophistication and flexibility. BI mashups can enable both the IT professionals and business users with capabilities that will create a successful BI implementation across the organization.

## **2.6 Enterprise BI Mashups and their current landscape:**

In the web 2.0 paradigm the principal approach for self-service has necessitated the need for mashups to be incorporated with the BI environment. While data analysis applications has been developed and operated by IT so far, BI mashups take this development approach to the next level of sophistication and flexibility instilling self-service. The infrastructure components within

an Enterprise BI Mashup platform isolate users from the complexity of heterogeneous data models in a federated data environment (Kobielus, 2009).

Users are able to access reusable views and compose their own mashups leveraging maintenance features of their browser oriented BI environment along with visual, code-free, application development capabilities. It enables them to personalize reports, dashboards, and other BI views and mashed-up application. All together a mashup platform enables the users to visually compose analytics from reusable components and data. Figure 8 shows processes and different components of mashup development within a BI environment:



*Figure 8: How BI Mashups Work (Kobielus, 2009)*

Thiele & Lehner (2012) took a broader approach to show a mashup platform can be integrated within a data warehouse (DWH) architecture, where the IT unit's responsibility is provisioned around the full infrastructure of the DWH and mashup platform. The application specific departmental data marts are defined by a competence center consisting of the IT unit and

departmental staff. The IT unit still provides the departments with their regular BI needs. In case of a required need of situational data analyses, the mashup development platform comes in place. As the operator of the mashup platform, the IT unit may transfer the frequently requested mashups to the regular data production process. This concept is depicted in figure 9.

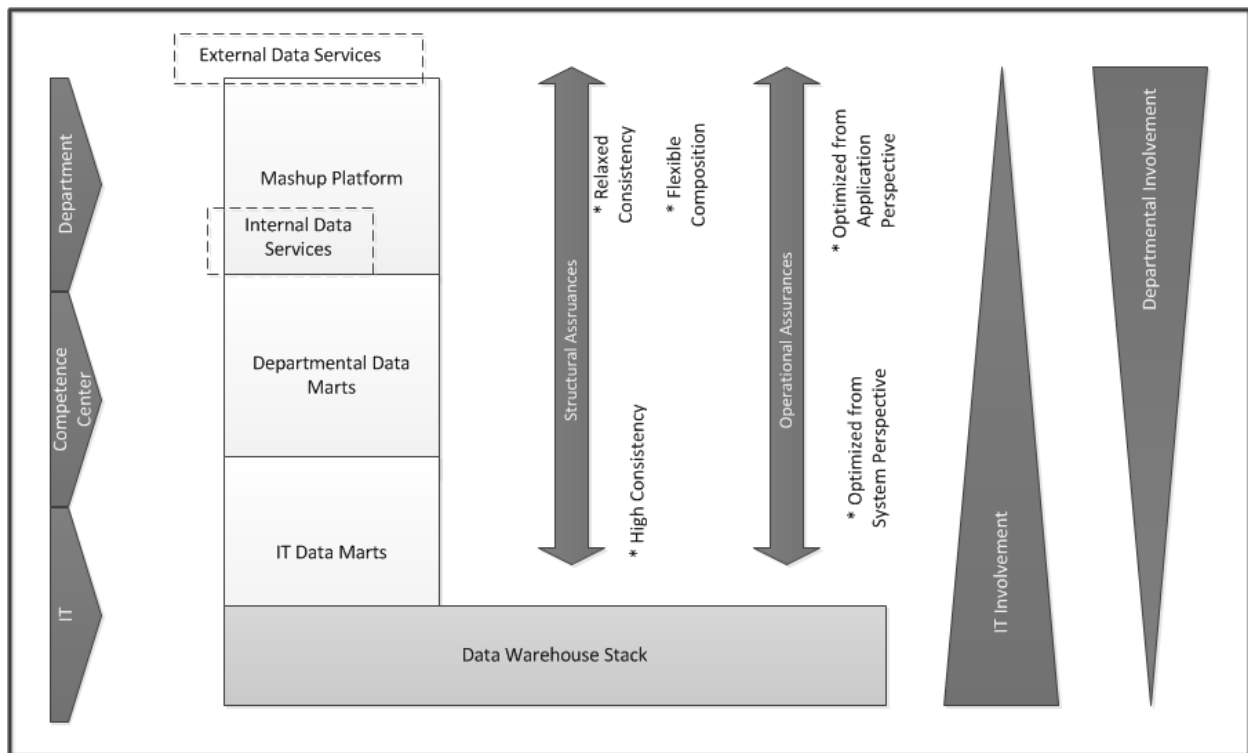


Figure 9: Integration of the mashup platform into a DWH architecture(Thiele & Lehner, 2012)

In general, mashup capabilities are offered within an enterprise as an extension or supplement to their current BI and enterprise data warehousing environment. And the typical use cases can be categorized into four types: intranet BI mashups, internet BI mashups, deep-dive BI mashups, and quick-start BI mashups. Intranet mashups deal with data manipulation tasks within the organization resources providing consolidated views of information. Internet mashups, as the name suggests, combine data within the enterprise from the outside world. Deep-dive BI

dashups are intended for power users for interactively building and visually exploring complex data models. On the other hand, quick-start BI dashups are targeted towards organizations with no prior BI platform or related support staff, providing rapid construction capabilities with no upfront data modeling or ongoing maintenance(Kobielus, 2009). In Figure 10 provides a maturity model for Enterprise BI Dashups categorizing them in four different levels stretching from lightweight presentation to a full collaborative governance.

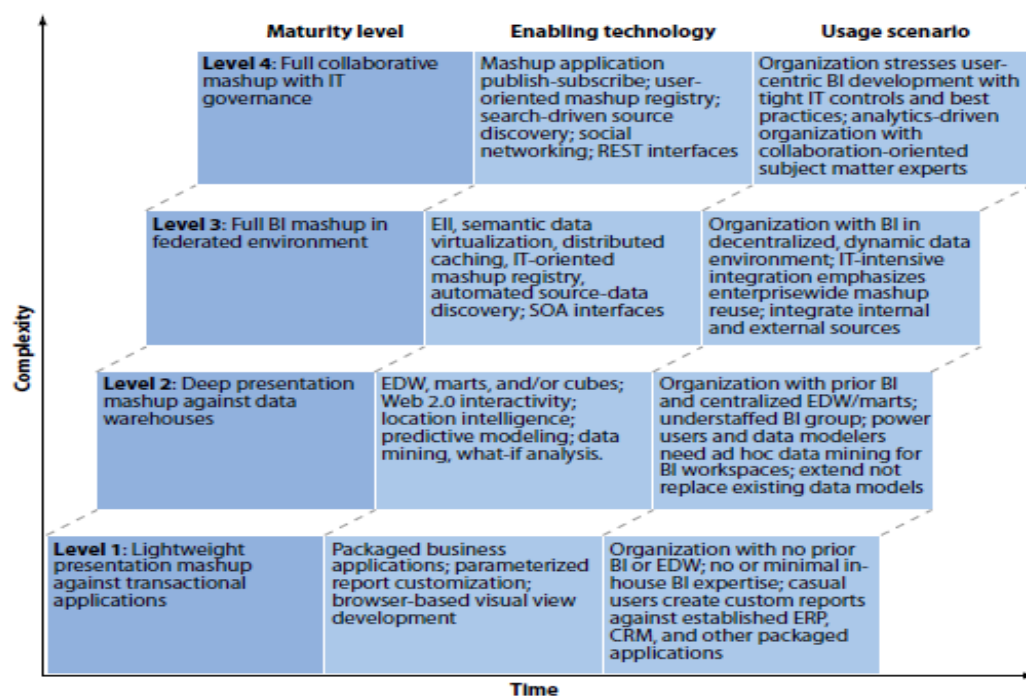


Figure 10: BI mashup maturity model(Kobielus, 2009)

However, there is an evident lack of a proper BI mashup framework that would enhance the understanding of specifically which mashup features and capabilities can cater to BI specific needs (Kobielus, 2009). The current work focuses on either on how mashup can provide the rapid and simple application development in general, not specific to any field. In BI, Pahlke,

Beck, & Wolf (2010) and Kobiellus (2009) have shown what would an enterprise mashup platform look like in BI environment, but don't illustrate how these platform relates to actual BI needs that clients develop day –to –day. A framework for Enterprise BI Mashups consisting of essential mashup utilities for BI use cases is required to provide the proper utilization guidelines of mashups in BI.

## **2.6.1 BI Mashup Development Tools:**

### **2.6.1.1 IBM Cognos Software Development Kit (SDK)**

Today's major software service providers are enabling their clients to customize or integrate their service that they are purchasing through providing them with Application Programming Interfaces on APIs. The Cognos BI tool provides a set of API that allows user to create customized applications which integrates their Cognos BI resources. In addition to that, these APIs can help users automate certain tasks that are usually done manually. This assimilation of APIs is termed as the Software Development Kit or SDK of Cognos BI application (Popescu, 2011). For our research purpose we have analyzed the applications that are created using the Cognos SDK. These applications are requested by the clients. For our implementation purpose we have taken advantage of web service technologies used by the SDK for the following capabilities:

1. *Report generation:* a specification is generated that describes the data to be returned as well as the layout information.
2. *Report storage:* a method to store the specification created as above in the Cognos server's internal database.

#### 2.6.1.2 IBM Cognos Mashup Service (CMS)

The IBM Cognos Mashup Service (CMS) is a relatively new API introduced by the Cognos Development team. With help of web services, the CMS facilitates data retrieval from reports. The retrieved reports contain information such as layout information, meta-model and report data. In the modern day enterprise architecture, where the need for data integration and sharing has become more complex, CMS provides some flexibility through BI resource integration in a specific context. Enterprises use various products concurrently since each product satisfies part of their requirements. Users may want to access all the information and tools they need for their daily activities in one place and avoid login to different systems for each task. CMS provides capabilities which let users to mashup Cognos reports with other applications. The two different interfaces that are used in this technology are the Representational State Transfer (REST) and Simple Object Access Protocol (SOAP). In CMS the report are generally accessed as LDX or layout data format which is a XML document rendering the Cognos content. LDX provides a consistent format that can be used in all other applications.



# Chapter 3      Research Methodology

---

## 3.1    Design Science Research methodology for Information Systems

In the context of information systems (IS), design science research (DSR) is set of analytical techniques and perspectives for performing research (Vaishnavi & Kuechler, 2004). DSR focuses on identifying organizational problems and subsequently designing, implementing & evaluating IT artifacts and communicating results to appropriate audiences and thus improving the performance of business organizations or creating new opportunity for businesses (March & Storey, 2008; Vaishnavi & Kuechler, 2004). Construction and evaluation of IT artifacts enables 1) the identification & description of desired information processing capabilities and their relationship to the present & desired situations and 2) the development of action and infrastructure specifications that facilitate implementation of information processing. Thus it can be construed that DSR is a problem centric approach and its either an initial research in a new problem area that focuses on constructing sufficient actions toward the ultimate goals and often involves prototype artifacts demonstrating feasibility of addressing the problem; or a subsequent research, that aims at improving effectiveness and efficiency of attaining goals or demonstrating the necessity of certain actions. A typical research undertaken in the DSR methodology has a five step lifecycle starting with the awareness of the problem through a new development of reference in the specific field of interest and resulting in an initial research proposal. Based on the information acquired in the first step, a possible solution is suggested in the next step with tentative design as output. The third step focuses on developing the solution and creating artifacts, while on the fourth step the evaluation process is undertaken against initial problem & criteria extracted from previous steps (March & Storey, 2008).

The types and levels of knowledge that can be derived from DSR can be explicated through four outputs (Vaishnavi & Kuechler, 2004) :

**Constructs:** *Constructs* are the conceptual vocabulary of a problem/solution domain. Constructs arise during the conceptualization of the problem and are refined throughout the design cycle.

**Model:** A *model* is “a set of propositions or statements expressing relationships among constructs.” March and Smith identify models with *problem and solution statements*

**Method:** A *method* is a set of steps (an algorithm or guideline) used to perform a task. "Methods are goal directed plans for manipulating constructs so that the solution statement model is realized."

**Instantiation:** The final output from a design science research effort is an *instantiation* which operationalizes constructs, models and methods.” It is the realization of the artifact in an environment.

The planned methodology for this thesis comprises an applied investigation through *design science research* techniques. Based on the outputs of DSR, the research activities are aligned accordingly which are further elaborated in figure 11. The research will start by studying traditional BI offerings and related business objectives which will be focused towards identifying the problem definition. The strengths and weaknesses of traditional BI offerings will be highlighted and used to frame and further explore the set of BI problems and business needs that may not be fully addressed by traditional BI offerings (e.g. situational analytics applications and real-time business intelligence requirements) and thus forming the constructs and model for the thesis. In addition to that, various mashup design patterns will be explored for their applicability in addressing the gaps in BI requirements. In the development and evaluation stage, BI Mashup

use cases will be mapped to software design patterns through consultation with the IBM software development team and exploration of various IBM tools and products mentioned previously creating the required methods for the research. Finally, in light of the fit between various mashup design patterns and BI use cases, functional requirements for successful BI Mashup offerings will be explicated through the *utility framework* resulting in the instantiation of the research work. The different stages of our research framework involve iterative processes. For instance, while mapping Enterprise BI Mashup use cases to specific software design patterns, we might require going back to the previous stage and fine tune our domain vocabulary and relationships as well iterate our literature review and requirements study phase. This iteration is also integrated while creating our utility framework and mashup prototypes.

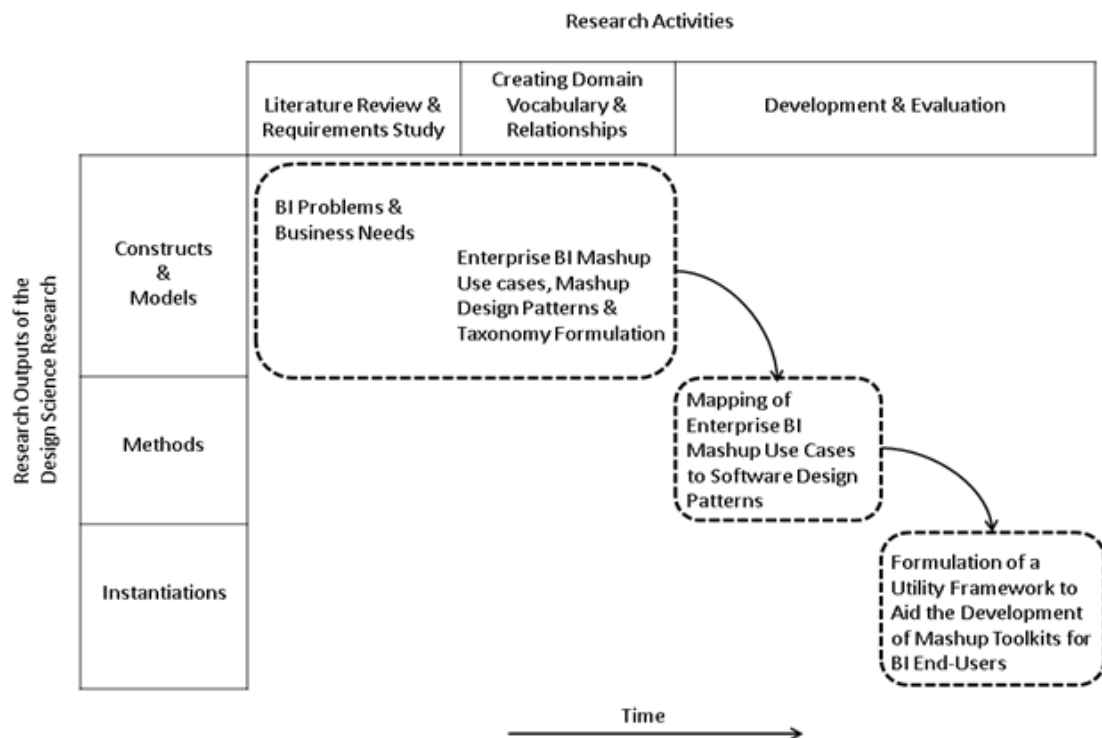


Figure 11: Research Framework

## 3.2 Research Methods and Steps

In alignment with the DSR methodology the detailed description of the research activities undertaken are described below:

The first phase of the project comprises a literature review of the current landscape of BI and an industry survey of market & vendor offerings, their weaknesses & the challenges faced in the end user adoption. The literature review will also consist of an in-depth review of enterprise mashups, their design and development patterns, current trends and available toolkits. The knowledge gathered in the first phase will be utilized toward formulating a preliminary taxonomy of Enterprise BI Mashups. This taxonomy will be enhanced further through interactions with staff members at the partner organization (IBM Cognos) and their experiences with developing mashup based solutions in the business intelligence and analytics domains. The overall thrust of the investigation will be tailored to situate the researcher in a knowledge sharing capacity with key members of the software development kit team at IBM Cognos. This will enable the researcher to better understand the challenges surrounding the development, implementation and adoption of enterprise mashups within a company's overall BI process and technology base.

During the second phase of the project, the taxonomy from the previous phase will be utilized to identify and offer specific solution statements for Enterprise BI Mashups. With an aim to identify a fit between BI mashup use-cases, organizational requirements and various available technologies, the researcher will undertake an extensive exploration of various tools and applications offered by the partner organization. These tools and applications may include products and solution offerings such as Cognos BI, Cognos SDK, Cognos Mashup Service and

Cognos Mashup Center. The findings from this phase of research are expected to yield a substantive foundation for actionable knowledge to facilitate the development of Enterprise BI Mashups to support a variety of business contexts including real-time business intelligence, situational analytics, and self-service BI.

The third phase of the research project will draw upon the findings from the applied investigation and use these as inputs to formulate a *utility framework* for Enterprise BI Mashups. By mapping use cases to specific mashup design patterns, the researcher aims to offer a utility framework that identifies and describes the requirements for suitable component and composition models for Enterprise BI Mashups. Subsequently, the efficacy of the utility framework will be illustrated through a prototype toolkit that demonstrates some commonly requested BI mashups. The fourth phase will consist of knowledge dissemination of the project findings in relevant conferences and to project stakeholders. The master's thesis of the researcher will be written based on the procedures and results of the project.

### **3.3 Methodology of Taxonomy Construction:**

In information systems, taxonomies are part of the foundation upon which an information architecture stands. Being a core component of the whole architecture, taxonomy interrelates with all other components of the information architecture. Taxonomy guides visual design of information navigation and management keeping in line with the standards. At a basic level, taxonomy is derived from analysis of usage patterns and information flow. Originally borrowed from the field of life sciences, taxonomy is essentially a conceptual framework providing a rigid listing of structures in a cascading fashion for a specific topic in question. If applied in the context of the internet, it associates with the effective structuring of content within a defined

scope for facilitating easy and accurate access. Depending on the audience, a definition of taxonomy might be adjusted. In our context, the taxonomy that we are intending to build would provide a classification of concepts for a domain where there is none. Taxonomy consists of three fundamentally different parts : 1) Representation 2) Ordering 3) Nomenclature (de Hoog, 1981; TechRepublic, 2003).

An important part of any taxonomy construction procedure is the description of objects under study. In order to be reproducible, the elements of a description should be maximally simple (Griffiths, 1973). These elements of description are the representation of characters observed. Not only that, these representations maybe interconnected and can also be converted to other ones. In terms of the ordering, dendrograms and nonlinear maps can serve as a basis for taxonomy classification giving it reproducible nature. In this context systems can be postulated as theories by intuition. The researchers are free to choose any criterion that is suitable for his cause backed by consistent logical reasoning. It is also worth mentioning that taxonomies claim to result in representation of relations between objective entities and that taxonomic ordering is an experimental procedure rather than a descriptive science. Lastly nomenclature enables the taxonomists to establish natural laws before the conclusions are deemed to be scientifically justified. Nomenclature is required for converting a system into usable and accessible one. Having said that, without practical quest for manageability of the system, the scientific soundness of the taxonomy can be harmed (de Hoog, 1981).

### **3.4. Taxonomy Construction for Our Research:**

Taxonomies for information systems aims at making explicit the knowledge contained within software applications, and within enterprises and business procedures for a particular domain. In information systems, taxonomies aid in the construction of user interface and application program components among others (Guarino, 1998). As a part of our taxonomy construction process, we have undertaken the task of creating a preliminary functional one which would guide the development of Enterprise BI Mashup tools. Our taxonomy would consist of concepts related to user interface, application program and functionalities as well as BI mashup enablers and drivers. In order to visualize the taxonomy components, we have used mind-mapping techniques. A well-formed representation of the taxonomy can help in better understanding the vocabulary of IS systems. Visualization of domain concepts also aid in explicitly representing knowledge which is implicitly stored in application as well as mapping of conceptual heterogeneous information sources related to data warehousing concepts. This helps by providing ease-of-maintenance, extensibility, and flexibility of the application program, in addition to turning the program into a knowledge base. This eventually helps in increasing the transparency of the application software (Guarino, 1998). We have used the Freemind mind-mapping software which is based on Java technology.

### **3.5 Research Evaluation and Validation:**

For research validation and evaluation purposes we mainly relied on demonstrating our created mashup prototypes and improving them based on the feedback of the Cognos SDK team. We asked the team to rate our mashups on certain criteria. The Cognos SDK team creates the

situational applications that their Cognos BI clients request. By demoing our created mashups to them, we validated the usefulness of our created mashups comparing against their traditionally built solutions. In addition to the criteria set by the Cognos SDK team we also evaluated our mashups on mashup evaluation criteria found in relevant literature. Based on our literature review (Minhas, Sampaio, & Mehandjiev, 2012; Pahlke et al., 2010; Zhao et al., 2011) we have identified four overarching criteria to evaluate a mashup platform. They are further illustrated below:

- I. *Usefulness*: Mashups are designed to benefit both business users and IT professionals in their daily activities. To provide the desired service required by both types of employees mentioned, a mashup platform has to be useful in terms of mashup design and technical features.
  - a. *Mashup Design Features*: The design features of a platform for actual mashup creation can be subdivided in to the following categories:
    - i. *Supported Mashup Activities*:
      - 1. Data Mediation: Involves converting, transforming, and combining data elements from multiple data feeds or API
      - 2. Process Creation: Creates a new process by the choreography for different APIs by presenting the necessary interaction point for the user.
      - 3. Collaboration: Ability to collaborate the created mashups between users.
    - ii. *Mashup Techniques*:
      - 1. Wiring: Facilitated mashup development by supporting connectors between modules, blocks, components
      - 2. Spreadsheet: Ability to load data in a table and process it to a desired format



3. *Programming by Demonstration*: Applying operations from templates instead of programming the operation themselves.
  4. *Script/Language based*: Supports the creation of mashup at code level
  5. *Webpage Customization*: Evaluation of presentation mashups based on the ability to browse, edit and combine different webpages within them
- b. Technical Features*: While evaluating mashup platforms based on their technical features we look at whether it supports the latest technologies related to web 2.0.
- i. Protocols supported for communication with Web services*: Ability to support both SOAP and RESTful web services.
  - ii. Data Retrieval Strategy*: Access to data through the platform that is intended to be mashed up
  - iii. Syndication formats supported*: Support of RSS and ATOM
  - iv. Lightweight process modeling*: Visualization of the process oriented view of the services required or being composed for the mashup application.
- II. Ease-of-use*: The level of ease in terms of using the features of a mashup platform would give us a picture of the overall mashup platform state. The following elements would aid in evaluating whether a mashup platform user, especially a business user, would be consciously be able to operate the platform and make unassisted changes on the spot during the event:
- i. Advanced UI Generation*: Ability to extract technical descriptions from the derived inference of generated UI from participating APIs' service description.
  - ii. Assistance during incompatibility*: A component that monitors the users actions and offers suggestions for changes.

- iii. *Tutorial Element*: An embedded tutorial element to assist the user in the learning process.
- iv. *Learning Curve*: How easy it is to learn the usage of the tool
- v. *User orientation*: The ability to rate, recommend embedded tools with the platform based on end-user requirements of the tool and the task at hand.
- vi. *User goals*: Explicit or implicit support of user goal achievement or task completion, e.g. simultaneous collaboration
- vii. *User requirements*: Whether the tool specifies the user requirements beyond listing the API operation with technical jargon.

III. *Intuitiveness*: The intuitiveness of a mashup system can be improved using the light-weight applications or components of user interface in the form of gadgets, widgets etc (Zhao et al., 2011).

IV. *Cost Reduction*: Higher resource utilization through self-service and collaboration, reusability, as well as lowering IT operating and development costs (Thiele & Lehner, 2012).

# Chapter 4      Results & Findings

---

In this chapter we present our results and finding of our research work. In defining lexical knowledge for enterprise business intelligence mashups we have proposed two frameworks which are the direct result of our research work. The two main outcomes of our research work are a taxonomy and a utility framework for Enterprise BI Mashups. The formulated taxonomy provides a basic framework for understanding the domain of BI mashups and is aimed to aid application development initiatives for creating BI mashups toolkits. Based on our taxonomy framework we have constructed a utility framework. The utility framework draws upon real-world use cases for BI Mashups as well as pertinent software design patterns that can facilitate the development of BI mashup tools and services.

## **4.1 Proposed Functional Taxonomy for Enterprise BI Mashups**

The taxonomy framework presented in this paper is an outcome of the first phase of our applied research investigation that aims to create a utility framework for Enterprise BI Mashups. The taxonomy identifies the high-level components for BI mashups. The undertaken ontological engineering process, mentioned in chapter 3, of this research was geared towards formulating a taxonomy framework for Enterprise BI Mashups where identified concepts were arranged in a hierarchical and easy-to-understand format. . The taxonomy framework is formulated by identifying key enablers and drivers of Enterprise BI Mashups in addition to the business trends that drive user requirements in situational BI uses-cases. The functional range of proposed in the taxonomy consists of the typical user required functionalities to enable self-service development. The taxonomy provides a categorization of the targeted enterprise BI users of mashup toolkits

and the associated development methods which can be undertaken by these users. In the framework we also classify the data source requirements for end-users and how they play their part in the functional capabilities of mashup platforms. The discussion of our taxonomy framework is divided into two sections: The construction of the taxonomy framework, taxonomy component and component-relationship analysis.

#### **4.1.1 Taxonomy Construction:**

The concept and their relationship identification for our taxonomy framework were done through two processes: 1) Literature Review & 2) User Requirements Investigation. The combination of these two processes helped us to associate actual current situational BI needs of clients with literature that is available on mashups.

Through our literature review, which was further elaborated in chapter 2, we gathered our initial data related to Enterprise BI Mashups.

In an attempt to identify the underlying concepts of BI mashups, our literature review process consisted of papers related to enablers and drivers of mashups both at enterprise and consumer level. This included literary work consisting of mashups in the context of SOA, situational data needs of organizations, self-service etc. Once we have identified the fundamental concepts we moved to papers related to enterprise mashup objectives and motivations, target groups as well as mashup design & architecture in order narrow down the intended users of Enterprise BI Mashups and their task characteristics. This included looking into what kind of features and functionalities are offered in traditional mashup products. In order to identify the situational BI requirements our focus was on papers which discussed the ad-hoc BI needs of clients as well as

inefficiencies of current BI applications. Reviewing of publications related to the enterprise data usage helped in identifying the data type, format and data source requirements of enterprise users. We also looked extensively into mashup design patterns for creating solution statements to solve specific situational BI problems. In gathering the data to construct our taxonomy framework our literature review was geared towards what potential benefits an enterprise mashups platform can offer users in order to complement the existing offerings and fill the gaps left by the traditional full-scale BI applications.

While working with the partner organization of this research project, IBM Canada Ltd, we analyzed their undertaken projects in delivering solutions that were requested by their clients in order enhance their BI capabilities. We also investigated sample applications that the organizations had developed for client demonstration purposes. These applications included:

- 1) Sample Cognos SDK and CMS applications for which clients use a guideline to create their own applications
- 2) Customized applications which integrated the Cognos BI setup of clients with the clients' own applications.
- 3) Requested applications which being built at the moment or are in the backlog

In analyzing the projects we specifically tried to answer the following questions in order to identify most commonly requested features of the clients. By narrowing down our use cases we were able to ascertain which capabilities, features and functionalities should be embedded in a mashup platform in order to create these applications through self-service. By answering these questions we identified the use cases to move forward with. Once the identification process was

done, the requirements analysis performed on these projects and applications were mapped to the mashup design patterns. The questions that we tried to answer are as:

- 1) What are the technical requirements of the project/application?
- 2) Can the technical requirements be delivered through a self-service platform?
- 3) Who are the targeted users of the project/application?
- 4) What is allocated timeline in delivering the project/application?

In addition to the above questions, our requirement analysis process also involved understanding what the clients wanted through the requested applications, stakeholder identification, measurable goals, and software requirement specification

The mapping process of use cases to mashup design patterns involved identifying the specific patterns which will be a best fit for creating certain features which the users wanted in their application. We also mapped the requirements of the clients to relevant solution mechanisms found in the literature related to mashup development for both consumer and enterprise level. As there is a lack of literature specifically aimed towards Enterprise BI Mashups, this mapping was done to a whole array of relevant topics related to BI intelligence and mashups. This helped us in pinpointing the components required for the taxonomy framework for Enterprise BI Mashups.

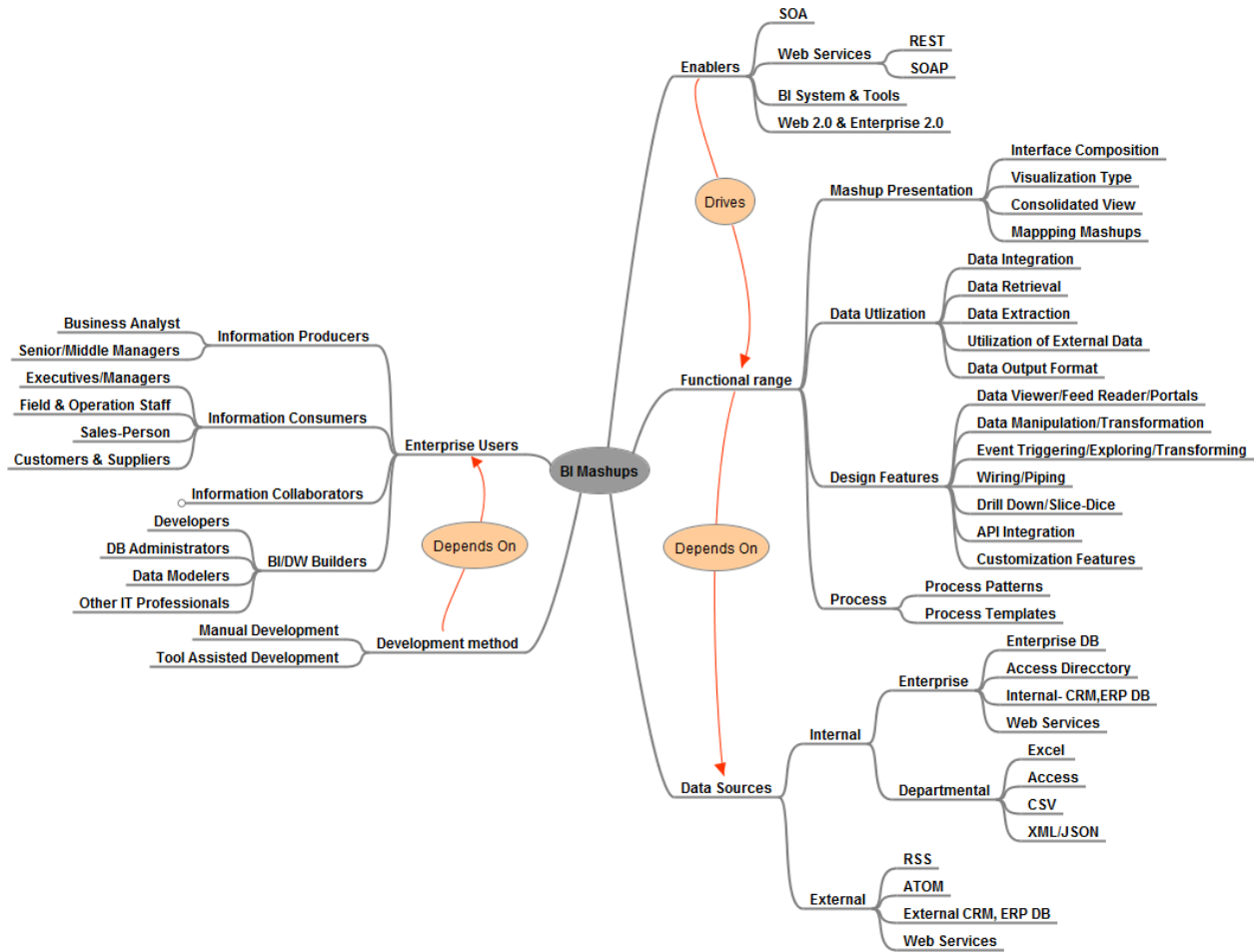


Figure 12: Preliminary Taxonomy Framework for Enterprise BI Mashups

#### 4.1.2 Taxonomy Analysis:

The construction of our taxonomy framework is based on five relevant categories identified through our research methodology. These five categories are further classification trees as highlighted in their sub-taxonomies. Figure 12 shows a visualization of the taxonomy framework as a mindmap. The five high level categories of this framework are: Enablers, Enterprise Users, Development Method, Functional Range, and Data Sources. In following sections we have discussed about our taxonomy framework in detail.

#### 4.1.2.1 Enterprise BI Mashup Enablers

In constructing a taxonomy framework for Enterprise BI Mashups we have to first take into account the enablers for these technologies. Mashups have earned their fair share of popularity in the consumer space. In the enterprise domain, mashups are yet to make their mark because of their collaborative and end-user centric development approach as opposed to the traditional siloed solution delivery approach where different departments have different levels of access to available technologies (Ogrinz, 2009). In the BI domain, the underlying drivers of BI systems would enable the creation of an enterprise mashup platform. The enablers are SOA & web services, BI systems & tools, Web 2.0 & Enterprise 2.0. These enablers drive the development of functional capabilities of Enterprise BI Mashup tools.

##### *i. Service Oriented Architecture (SOA)*

BI system implementation in modern day organizations is greatly driven by Service Oriented Architecture because of the need for integrating business processes with business intelligence. Better decision making requires better interoperability and sharing of resources across the enterprise. SOA's style of loosely coupling services and providing reusability allows BI systems to be extended to other services and minimize their shortcomings. Mashups by their very nature are both a precursor and a beneficiary of the SOA paradigm (Ogrinz, 2009). Their similarities with SOA make them a perfect fit for empowering BI end-users with diverse functionalities to meet their situational requirements.

##### *ii. Web Services*

Regardless of the new trend of using the terms SOA and web services interchangeably, web services by themselves would act as an enabler of BI mashups because of the need of integrating external in addition to internal services. In the API driven development approach, BI applications



are also introducing their own APIs for consumers to integrate or create application using their underlying resources. In a SOA context, web services have become a major method of integrating BI content with external applications (Cappiello et al., n.d.). Mashups can consume web services that SOA-centric applications utilize as well as creating and publishing them as web services for further reusability. The sub-taxonomy of web services consists of the standardized protocols REST & SOAP, REST being the more widely used mechanism because of its simplified integration methods.

### *iii. BI Systems & Tools*

At the center of the Enterprise BI Mashups would be BI tools and applications themselves. The outputs of a BI system would act as the main source of inputs to be used in a mashup platform. The introduction of Software Development Kits (SDK) of widely used BI solutions has made the integration process with mashups possible by making BI content extensible to other platforms.

### *iv. Web 2.0 and Enterprise 2.0*

Web 2.0 technologies enable users to remix/reuse data, web services and micro-applications to create hybrid applications which transforms the mashup creation process from being technically challenging to nearly mainstream (Yee, 2008). While Web 2.0 deals with the consumer internet space, enterprise users' demand for Web 2.0 technologies within corporate walls has given birth to the notion of Enterprise 2.0. In the Enterprise 2.0 concept the rigid structures are removed through collaborative solution delivery. Equal access to technology and flexible yet powerful tools enable every segment of the organization to build their own solutions. In the same manner in which Web 2.0 enables and facilitates mashup creation through self-service, Enterprise 2.0 does it for organizations.

#### 4.1.2.2 Enterprise Users

Development of BI applications is driven by the needs of end users. Based on Imhoff & White's (Imhoff & White, 2011) information worker classification we have identified four types of Enterprise BI Mashups users

##### *i. Information Producers:*

Information producers create the BI related information to be consumed by information consumers. Information producers or power users engage in interactively building and visually exploring complex data models. Their usage of BI mashups will be in a manner which will improve the value of existing solutions. A further classification of information producers would include, but not limited to, business analysts & senior/middle managers.

##### *ii. Information Consumers:*

These are users who support day-to-day business operations utilizing the information delivered by the information producers through BI applications. Through a mashup platform's simplified drag and drop development method information consumers can cater to their own needs when it comes to situational requirements. A sub taxonomy of information consumers consists of executives/managers, field & operation staff, sales-person, and customers & suppliers.

##### *iii. Information Collaborators:*

Information collaborators contribute into the whole BI ecosystem through their activities utilizing collaborative applications to which the notion of enterprise mashups conforms to. Sub categorization of this type of enterprise users consists of motivated information workers, researchers, & subject matter experts.

*iv. BI/DW Builders:*

The final category of Enterprise BI Mashups users are BI/DW builders who are responsible for developing and deploying a BI solutions which consists BI application development and DW construction and maintenance. BI/DW builders can be developers, DB administrators, data modelers or other IT professionals who are not directly related to application or DW development (e.g. network administrators).

**4.1.2.3 Development Method:**

Mashup Development in the context of BI can be categorized depending on the types of users using the platform. The development sub-taxonomy consists of two classifications:

*i. Manual Development*

Manual mashup development requires programming skills and intimate knowledge about the schemes and semantics of data sources or the business protocol conventions for integrating data and applications into a coherent and value-adding application (Liu et al., 2011).

*ii. Tool Assisted Development*

Mashup-specific development tools and frameworks accelerate the mashup development process and enable business and inexperienced users to mash up their own applications and thus are the enabler of self – service for end-users. Powered with visual drag and drop features and widgets, requiring little or no programming, enterprise mashup tools in a BI context are mostly

targeted for the information workers consisting of information producers, consumers, & collaborators (Liu et al., 2011).

Based on the user type and their platform usage criteria and skill level, the development method of Enterprise BI Mashups depends on these user types. The BI/DW builders and some portion of the information producers will undertake a manual developing process of mashups using traditional software development process which is done by using software programming languages. In this process a mashup application will generally built from scratch rather by coding it all the way, while in the tool-assisted development feature is suitable for information consumers, information collaborators as well as information producers. Using the self-service features already embedded in mashup development tools, users with can create mashup applications without having any or minimum programming skills.

#### **4.1.2.4 Functional Range:**

The functional range of a mashup platform for BI should consist of features enabling users to utilize in a manner which would be of best-fit for them. The functional range of Enterprise BI Mashups can be classified into four categories:

### *i. Mashup Presentation:*

The situational BI applications demanded by the users require simple yet diverse user interfaces. Mashup presentation deals with the layout and interfaces of the created mashups. Presentation mashups required by different array of users can be categorized into four classifications (Pahlke et al., 2010).

- 1) **Interface Composition:** The process of designing the layout of the mashup. Different mashup tools tend to have different ways to construct the interface.
- 2) **Visualization Type:** Data visualization is a very important aspect of modern day BI (Negash, 2004). Mashup tools have to have the ability for the users to create or integrate the visualization types they want - be it charts, graphs, diagrams, tree, heat maps etc which traditional BI applications usually do not offer.
- 3) **Consolidated View:** A BI mashup tool can also be classified in terms of its consolidated view construction mechanism. When creating an overall picture of how their business is performing and then taking decisions promptly, business users need to have a consolidated view consisting of all the related internal and external data presented in their mashup interface (Anna, 2011; Kasman & Roder, 2011b).
- 4) **Mapping Mashups:** Overlays integrated data in geographical locations on maps (Sleigh & Johari, 2010).

### *ii. Data Utilization*

Through the self-service capabilities provided by BI mashups, BI professionals of all categories can manipulate data in a manner that they want. Typical use cases of data utilization (Simmen et al., 2008) through Enterprise BI Mashups can be classified into 5 categories:

- 1) **Data Integration:** Integrating organizational internal data with externally available data.
- 2) **Data Retrieval:** Getting hold of data sources or data points which usually exist in a different system or unsupported format in BI tools.
- 3) **Data Extractions:** Filtering or extracting the data needed from huge volume of data.
- 4) **Utilization of External Data:** Users should be able to utilize external data in the same manner they use internal data.
- 5) **Data Output Format:** BI mashup end users should be able to create or transform data in to their desirable format for meeting their situational needs.

### *iii. Design Features*

The actual usefulness of created mashups in a BI context by combining data sources and various other components depends on the mashup functionalities available within the BI mashup tool. In terms of mashup functionality sub-category, we have identified seven generic patterns. They are:

- 1) **Data Viewer/Feed Reader/Portals:** The first type is components for exposing the transformed data for mashing up. These components or widgets can have the functionality of data viewers, feed readers or portals.
- 2) **Data Manipulation/Transformation:** Data manipulation and transformation components usually provide lightweight manipulation and transformation features for end users at the mashup builder side.
- 3) **Event Triggering/Exploring/Transforming** Event triggering, exploring and transforming widgets enable the inter-communication of events between various widgets in a mashup space.

- 4) Wiring/Piping: Wiring and piping functionalities determine the data passing style between different components.
- 5) Drill Down/Slice-Dice: For data drill down and slicing/dicing purposes mashup tools consist of nested data viewer widgets or even more customized components exposing more specific drill-down functionality.
- 6) API Integration: Custom built API integration widgets enable users to utilize external functionalities.
- 7) Customization Features: Customization features allow users to manipulate and modify various resources according to their preference e.g. URLs and strings (Kasman & Roder, 2011b; Simmen et al., 2008; Singh, 2008b; Sleight & Johari, 2010)

#### *iv. Process*

In order to eliminate processes involving repetitive tasks for business users, BI mashup platforms provide the capability of creating process mashups. Process mashups can be further classified in to two categories – process patterns and templates. Process patterns define the nature of workflow which will be automated or created through mashups. Process templates are pre-existing process mashups in the form of templates which can be used to create new process mashups (Vrieze, Xu, Bouguettaya, Yang, & Chen, 2009).

In terms of the relationship between the functional range and other components, the provision and development of the functional range in a mashup tool is driven by the enablers of BI mashups. Building a mashup platform based on SOA will instill sharing and reusing of resources enabling interoperability between different systems within an organization. This would lead to implementing the desired data utilization functionalities; while the usage of web services would

enable the integration of both internal and external APIs. BI systems and tools would allow the development of mashup functionalities in a BI context thus making them the most important drivers of BI mashups. Finally, the concepts of Web 2.0 and Enterprise 2.0 would drive the innovation of features which would bring collaboration in situational BI application development.

#### **4.1.2.5 Data Sources**

In BI context, the extent to which a mashup tool can facilitate the usage of various data sources indicates the usefulness of the tool. Information workers need the capability to utilize a large pool of data sources and formats to meet their situational needs at any given time (V Hoyer et al., 2011). The sub-taxonomy of data sources for BI consists of two elements:

##### ***i. Internal Data Sources:***

Mashup tools utilize the data, which is directly supplied to them by the users. The categorization of internal data can be done in two types: enterprise and departmental. Enterprise data usually consists of a large volume of data stored within multiple databases. The types of this data can be core BI application data in enterprise databases, access directory data or application system data e.g CRM or ERP systems. On the other hand, departmental data usually consists of binary files e.g. excel, access, csv etc or even in standardized web exchange format e.g. XML.

##### ***ii. External Data Sources***

The most common source of external information for BI mashup tools would be RSS and ATOM. In addition to that, external data can also be of similar sources like internal data. For example data exposed through Web Services in XML or JSON format. It can also be directly accessed through plugging in external databases of collaborating organizations.



The fifth high level component of our framework – data sources impact the development of functional capabilities of a mashup platform. What features and functionalities a mashup tool will have will be highly dependent on what data sources need to be accommodated. As the whole concept of BI is centered on data, the functional range of mashup platforms are embedded in a manner which would allow working with the desired external and internal data sources and formats of the enterprise users.

Thus we can see that the relationship between the high level components of an Enterprise BI Mashup taxonomy framework is either dependent or driven by other components.

This proposed functional taxonomy framework aims at filling the current gap in the domain of Enterprise BI Mashups. Regardless of the potential benefits of Enterprise BI Mashups, there is an evident lack of concrete knowledge base or formal BI mashup frameworks for understanding mashup features and capabilities in a BI context. In addition to that, even though there are multiple mashup products available in the consumer space for usage in the Web 2.0 arena and only a few are geared towards enterprise usage, there is an evident lack of mashup products targeted for usage in a BI context. As mentioned in the chapter 2, this results from IT department's focus on high profile BI initiatives and system implementations which leaves many situational BI requirements of clients unaddressed. While in the consumer internet space the notion of self-service has been accepted and put to practice, in the enterprise BI arena self-service is yet to be realized. This leaves the development of BI mashup tools still in the experimental stages. A taxonomy framework such as the one discussed in this thesis, aims to fill this gap in the extant literature and also to potentially serve as a basis for future development of BI mashup platforms.

The created taxonomy uses the already established concepts in the field of software engineering and information management. Thus it can be easily integrated in any development scenario. The association of mashup drivers, which were identified in the enablers categorization, with conventional software development technologies is expected to keep the BI mashup development cycles up-to-date with the advancement of these technologies. The classification of enterprise users provides a varied target audience for the development initiatives for mashup tools. The functional range and the data sources are expected to fill the gap existing mashup tools in the context of BI.

#### **4.1.3 Mashup Pattern Mapping :**

In this section we will discuss which specific mashup patterns can enable the implementation and fulfillment of the functional range as well as supporting data sources and formats required by the enterprise users. Figure 13 illustrates the mapped patterns with the associated taxonomy elements. The objective of the mapping process here is to relate a pattern to specific enterprise mashup concepts. Associating mashup design patterns with the situational BI requirements will enhance the understanding of why mashups are useful in certain contexts and the overall solution elements at a general level. Identifying specific patterns will also help in fine tuning the functional requirements which are being translated from business use cases to the technical vocabulary. This mapping process would provide a general language that enterprise users, technically skilled users specifically, can use to construct a solution statement while creating a mashup. Specific patterns related to the functional range and the data requirements would bridge

the gap between the enterprise users by providing a language to create specific solution statements for creating mashups in a BI context.

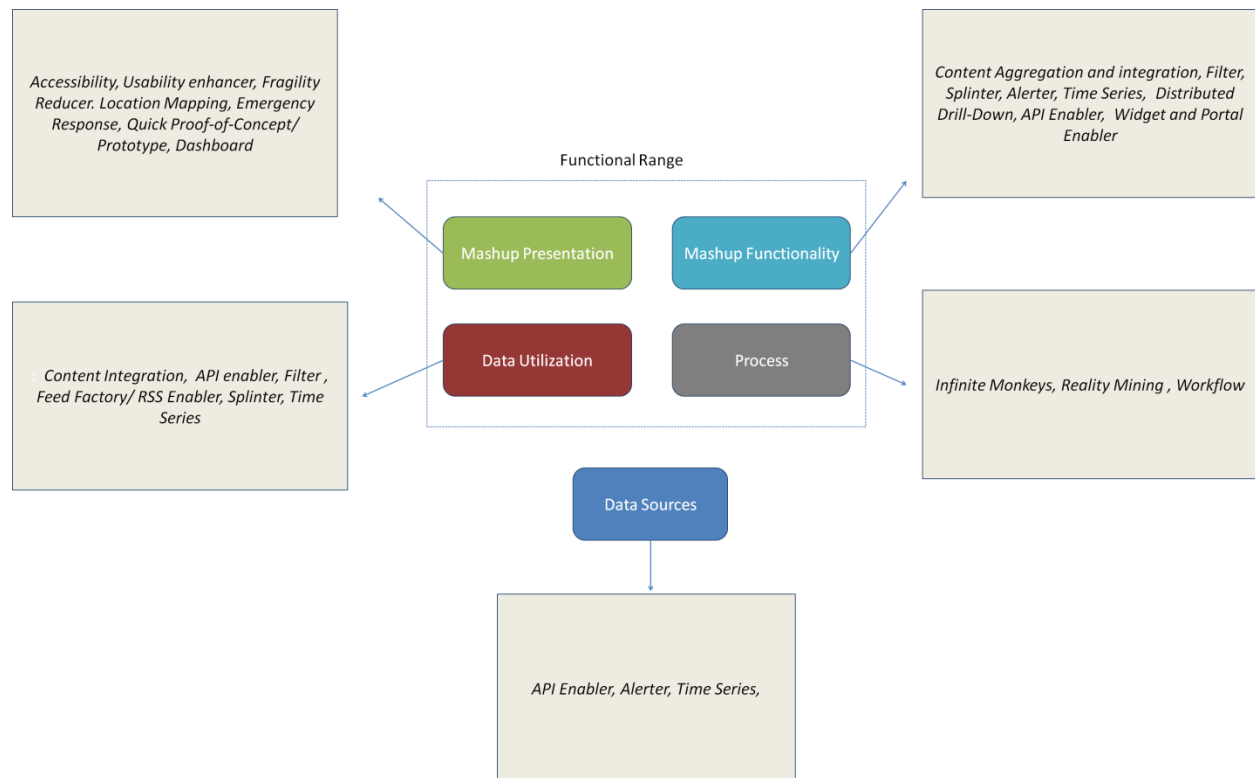


Figure 13: Mashup Pattern Mapping with Taxonomy Elements

#### i. Patterns for Mashup Presentation:

The Mashup presentation component in our taxonomy framework consists of concepts required to compose the presentation interface of mashup. In terms of mashup patterns – accessibility, usability enhancer, fragility reducer, location-mapping, emergency response, and quick proof-of-concept or prototypes would provide means to create effective presentations. The quick proof-of-concepts or prototype patterns allow us to validate a BI situational application solution by creating a rapid presentation utilizing the available resources. The dashboard pattern helps us to

create summaries of the data acquired from multiple sources, while emergency response patterns provides mechanisms to create mashup presentations where the available time is minimum. Accessibility patterns provide means of creating alternative application interfaces where there is a need for different presentation views for different types of users. If the need for implementing a working solution is a bigger priority than making the interface usable, the usability enhancer patterns would provide means to create a wrapping presentation exposing the required functionalities only. And finally the fragility reducer pattern provides solutions in terms of making the mashup presentation less vulnerable to breaking down by utilizing multiple sources (Ogrinz, 2009).

## **ii. Patterns for Data Utilization:**

The data utilization component of our taxonomy framework can be mapped to several mashup patterns i.e. content integration, API enabler, silter, splinter, and time series. The content integration pattern provides means to extend a system that accepts various data sources and combines them in to a single feed which conforms to the systems own standards. The API enabler pattern enables access to data sources locked away in closed or proprietary formats by transforming them to dynamic data sources. The feed factory or RSS enabler pattern allows the creation of RSS or ATOM feeds for websites that don't have one in addition to remixing the new feeds with existing ones. Data sources which are regularly updated, can be tracked using the time series patterns which provides mechanisms to extract and store information at regular intervals. The splinter and filter patterns allows us to transform the data in to a structure that is suitable for a given use case. The splinter pattern is about separating unified data into smaller specialized streams of focused information. Lastly the filter patterns helps us to remove unnecessary data from a feed (Ogrinz, 2009).

### **iii. Patterns for mashup functionality:**

The mashup functionality component of the taxonomy framework is one of the most important aspects of Enterprise BI Mashups. In the case of various functional requirements of situational applications, mashup patterns provide ideas for integration mechanism for various functionalities. In a BI context we have identified the following patterns for enabling mashup functionalities: distributed drill-down, widget, portal, and API enabler, content aggregation and integration, filter, splinter, alterter and time series. The widget enabler pattern provides means to integrate mini-applications or specific functionalities of larger applications. While portal enabler patterns enable data functionalities by allowing content migration without custom coding (Ruhi & Choi, 2013). Distributed drill-down pattern enables the highly drilling down, slicing and dicing functionalities. For accessing external application functionalities, the API enabler pattern allows us to utilize the external SDKs and libraries. Content aggregation pattern is useful where the situational need consists of frequent task switching activities. The content integration, filter, splinter, alterter patterns provides mechanism for integrating various data functionalities as discussed in the previous section.

### **iv. Patterns for Processes:**

For automating repetitive processes mashup patterns like infinite monkey, reality mining, and workflow can aid in effective BI mashup creation. Processes that require a high number of manual monotonous tasks, can be automated through using the infinite monkeys pattern. While

the reality mining and workflow patterns provide mechanisms to understand human interaction with applications and implement them by creating associated workflows (Ogrinz, 2009).

#### **v. Patterns for Supporting required Data Sources:**

In a BI context, it is imperative to support various data sources and formats in order to meet the requirement of the users. Mashup patterns that aid in this process are API enabler, alerter, and time series patterns among others. These patterns enable access locked away data which generally reside in siloed databases. By deploying intelligent agents - the alerter pattern specifically monitors web resources as well as other data of various formats for updates. And the time series pattern in addition to the aforementioned tasks stores the extracted data from the mentioned resources in a mashup specific repository (Ogrinz, 2009).

## **4.2 Proposed Utility Framework for Enterprise BI Mashups:**

The development of web-based enterprise application is driven by frameworks which define the overarching infrastructure, functional components as well as the workflow and relationship of the components of the application. A utility framework provides a comprehensive visual model of utility functions of an application which developers use regularly when building web applications (Oracle, 2013). In order to adopt a feature based development (Tun et al., n.d.) of a mashup platform, a utility framework will guide the development process in a manner which would provide the proper functionalities. In addition to that, in a BI context the features and functions have to be tailored to suit the needs of the enterprise mashup platform which would complement their already implemented BI systems. In this regard, there has to be a fine balance

between the functionalities the BI applications already provide and the ones the BI mashup tools are going to provide. Based on the concepts identified through our taxonomy framework, in our utility framework we have provided an overview of functionalities required to create situational BI applications which are associated with different tasks undertaken by different types of enterprise users. This process also consisted of the reviewing the current mashup platforms available. These mashup tools were not targeted towards any specific field, rather for general application development by business users. By understanding the product features, workflows and their enablers, we identified the components required for Enterprise BI Mashup application. Our reviewed products were IBM Mashup Centre and Jackbe Presto.

The utilized features and functionalities in the framework will enable self-service as well as programming based application development by different types of users. The features reflect the functional range components of the taxonomy framework that we have discussed in the previous section. The framework takes into account the underlying infrastructure required to create a BI mashup platform. The infrastructure components take in to account the required technologies and resources for web based enterprise software development. The BI application and tools as well as BI mashup enablers included in the infrastructure model enables application development specifically for enterprise BI mashups tools. Figure 14 provides a visual model of our proposed utility framework. The model provides an overview of the necessary components and features as well associated user types and their tasks in sequential manner.

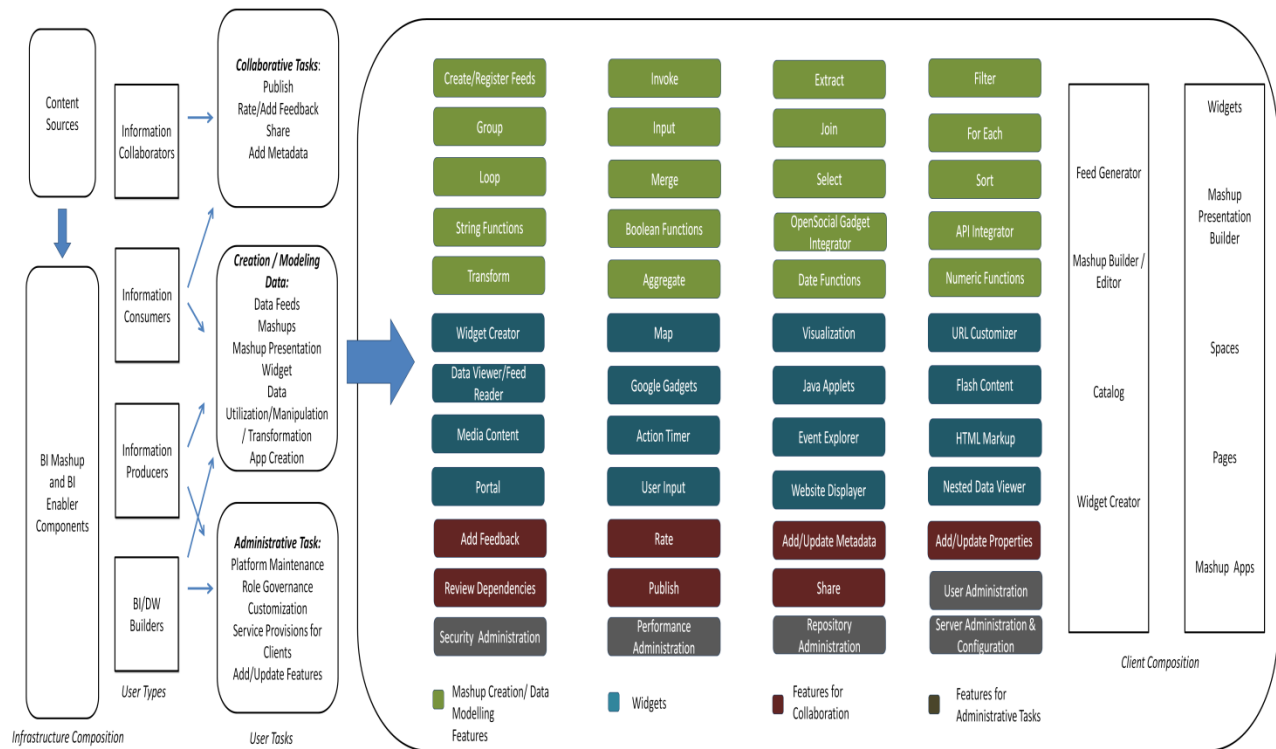


Figure 14: Proposed Utility Framework

#### 4.2.1 Underlying Infrastructure:

The underlying infrastructure of for an Enterprise BI Mashup platform consists of two overarching components: i) Content Sources and ii) BI Mashup and BI Enablers. These two components are the assimilation of a number of software development building blocks which create the whole mashup platform. Figure 15 provides an infrastructure framework which consists of the aforementioned software development building blocks. For the underlying infrastructure we have taken a general approach rather than any vendor specific proprietary applications.

##### i. Content Sources:

In terms of the content sources, as we have discussed in our taxonomy framework, there are internal and external data sources. Organizational internal data mainly lies within enterprise



databases. There can be different types of databases in terms of technical implementation and storage of data type i.e CRM, ERP systems or even from server logs. For the mashup platform usage, there can be other enterprise data sources originating from different parts of the same organization or multiple collaborating organizations which can be enabled by web services. In terms of departmental data, they are mostly files lying in desktop machines of employees. They can be excel, access, csv or xml files. External enterprise data also lie within databases. But they are typically accessed via webs-services provided through APIs. External data sources can also be provided through RSS and ATOM feeds. This data sources are accessed by the mashup application server for further usage as we see in figure 15. The mashup platform's own database consists of all the created feeds, metadata, mashups, applications, user-data, snapshots of resources which are utilized for creating mashups etc.

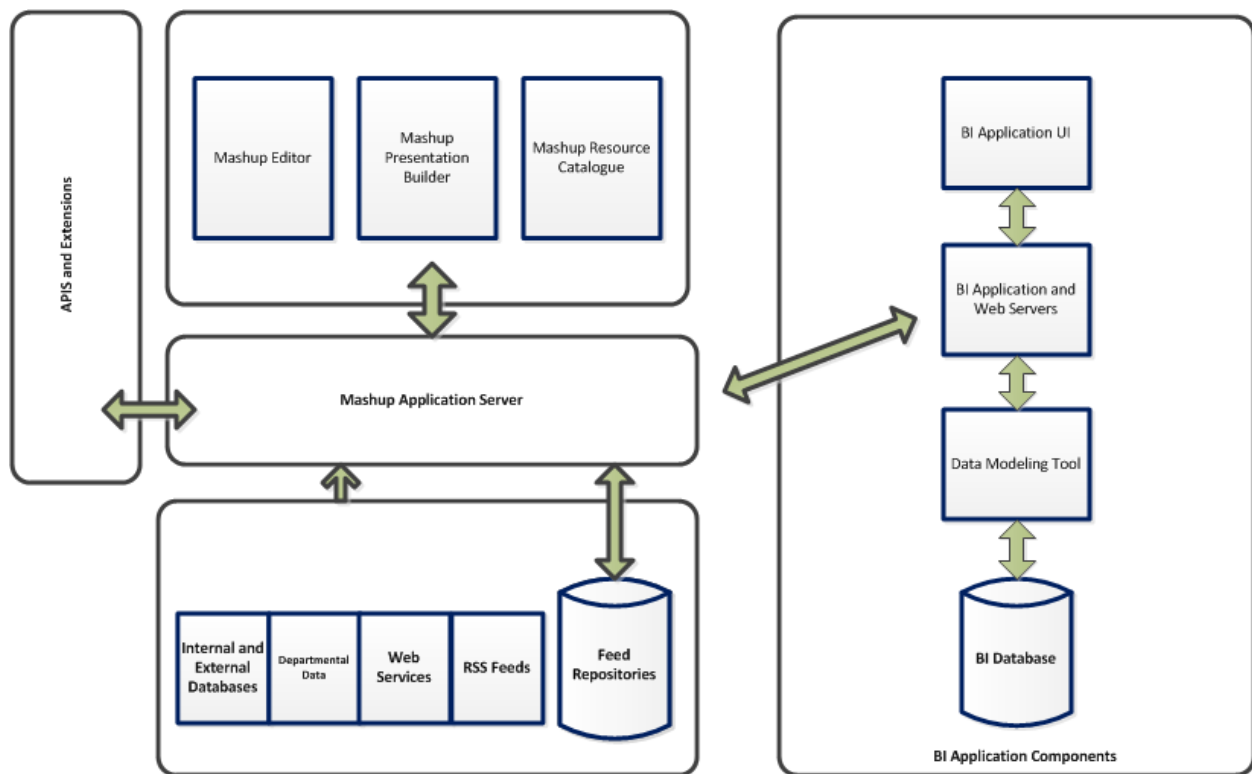


Figure 15: Underlying infrastructure for a Enterprise BI Mashup Platform

As for the BI components of the infrastructure setup which we are going to discuss in the next section, the data sources tend to ROLAP or Relational databases. The data lying in these databases are further modeled in order to ensure that metadata is presented in a manner that business users understand. An example of tools which does this processing is IBM's Framework Manager.

## **ii. BI Mashup and BI Enabler Components:**

The BI mashup enablers section consists of all the components which allow the usage of mashup and BI tools for our purpose. In terms of the mashup tool, it consists of the mashup application server and the clients of the platform. The application server supports the design, discovery and governance of all the client platform components. In addition to that, the server supports a wide range of data sources. The communication method for mashup server with its clients is through a secure HTTPS connection through the exposure of REST API. The clients of the mashup platform supported by the application server consist of three main components: mashup editor, mashup presentation builder, mashup resource catalogue. All these three components are web based tools providing both easy to use features for non-technical users as well as well skilled users. The components are integrated visual drag and drop enabling enterprise users to create and modify mashups. Through the mashup editor users can perform custom and real-time data utilization tasks. The different sets of features within the mashup editor allow different types of users to perform tasks consisting of different expertise levels. The presentation builder component is used to compose the view of the mashup through a collection of widgets. The resources required for composing this view generally come from within the other components of the mashup platform i.e. the result sets of the mashup editor or from external sources as well. The general features of the mashup builders are exposed through pre-built widgets which we will

further elaborate in the ‘required functionalities’ section. The mashup resource catalog enables the enterprise users to search, rate, share existing feeds and mashups. It also acts as a launching point for creating and editing new mashups and feeds. In addition to these client components, a mashup platform can include a widget creator for coding new widgets by BI/DW builders.

The BI tool implementation for our purpose is similar to any standard implementation. The implementation consists of four main components. The data sources components is where the data is stored and from where it is provided to the other components for usage. The data modeling tool converts the data in a BI usable format by ensuring metadata is presented in a manner understandable by business users. The next level components are the web servers and BI application servers for the BI implementation. The BI application server takes care of running and supporting the requests related to all the tasks that users undertake. The web server on the other hand transfers the BI web application requests to the BI application server. This is typically done through gateways residing in the web server. The component which is directly exposed to the BI user is the BI application user interface. Through this interface the enterprise users perform their tasks. Now in order to enable the extension of the BI resources to another external application, an API of the implemented BI system is required. This is generally done through a SDK (software development kit), discussed in chapter 2, of the system. The extension of BI resources to other application is enabled through the usage of Web Services between the application servers of different systems. In our case this is done between the BI application server and the mashup application server through a secure protocol as shown in the figure 15. Through the usage of the BI application API, we can integrate the data from the BI intelligence application to the mashup platform and use it to build the situational applications.

### 4.2.2. User Types:

As we have discussed in section 4.1.2.2, the enterprise users can be of four types : 1) Information Producers 2) Information Consumers 3) Information Collaborators 4) BI/DW Builders. The utility framework takes into consideration these user types and provides the functionalities required by them.

### 4.2.3 User Tasks:

The tasks undertaken by the enterprise users can be mapped to three overarching categories. These tasks can be overlapping between the user types, meaning each user type can undertake any of these tasks. Having said that, a user group will be focusing on any one or two of the task-categories. The categories are as follows:

#### *a) Creation and Modeling Tasks:*

The goal of creation and modeling tasks within a mashup platform is to create actual mashups, the artifacts and components required for mashup creation and to model the data according to the needs of the users. These tasks are generally undertaken by information consumers and producers, but are not limited to them only. The self-service features enabled by the functional range of the mashup platform allow information consumers and producers to create and model data according to their needs. The creation process involves creating data feeds from various data sources (see section X). These feeds can be created directly from the plugged in data sources as well as through web services. In addition to that feeds can be created by the enterprise users using RSS and ATOM feeds. Once the feeds are created, they can be modeled according to the needs of the users. The modeling of data is done through the data utilization as well as

through data transformation and manipulation features. Once transformed into the desired structure, these artifacts or building blocks for mashup creation can be stored and accessed for future usage. The mashup creation process is done through utilizing the modeled data as well as utilizing external resources, i.e. external APIs and external data. The created artifacts can be a data mashups created from integrating separate data feeds. It can also be the more recognizable mashup view or mashup presentation which enterprise users create by utilizing the widgets available. In terms of complementing the BI systems, both data level mashups and presentation level mashups play important roles. The widgets which are utilized for the creation for mashup presentations are generally built in. But in addition to that a mashup platform might have separate components within the client environment for creating widgets. Users with programming knowledge can utilize this component for creating widgets which would be used by the business users. In addition to that a mashup platform may contain specific application building features for creating applications targeted for usage in mobile devices.

#### ***b) Collaborative Tasks:***

Collaboration being at the heart of Web and Enterprise 2.0, all the enterprise users undertake some sort of collaborative tasks. One of the important characteristics of a mashup tool is that the created artifacts such as data feeds and data mashups are reusable. These components are stored in the mashup repository and can be accessed through the mashup catalogue. Once a data feed or a data mashup is created they can be published in order to be accessed by a certain user group according to the enterprise governance policies. Users who have permission to access these components can further validate their usability by rating them. In addition to that, they can update these artifacts and share them as well. Another collaborative task is adding relevant metadata or tagging feeds and data mashups. This process would make the mashup building

blocks more usable and increase their searchability. By tagging them with relevant keywords the data feeds and data level mashups can be organized in categories.

### *c) Administrative Tasks:*

Administrative tasks consist of the activities relating to maintaining the mashup platform. These tasks are undertaken by the BI/DW builders or the IT department of the organization. A web based mashup tool would be used by a vast number of users. The mashup tool, as a web application, has to be reliable in terms of service provision, delivering of requested upgrades and customizations, reliability, performance, compliance with standards and organization policies, and being up to date in terms of features and functionalities covering both the mashup application client and application server. Most of these tasks are routine based tasks which need to be performed in order to make sure the platform is running smoothly. The upgrading and maintenance tasks are more technical activities, while governance of the platform is to make sure that users are complying to the agreed standards. The governance is done setting various access control measurements and auditing of user tasks. The responsibilities are down to the application owners and system administrators. Performing these tasks requires mechanisms i.e. administrative functionalities within the mashup platform. In addition to the administrative functionalities of the mashup platform, the maintenance of the underlying infrastructure is also an integral part of administrative tasks. Since the implementation of the mashup platform depends on the underlying infrastructure i.e. the data sources, external APIs, BI system – the administrative tasks are also extended to these components. Typically the IT department is responsible for maintaining the underlying infrastructure as a whole.

#### **4.2.4 Mashup Platform Client Composition:**

The mashup platform client can be composed of several modules. From our exploration of current mashup tools available for consumer usage (Jackbe, 2012; Kasman & Roder, 2011b; Singh, 2008a), we have identified ten client components.

##### ***1. Feed Generator***

A feed in a mashup platform is XML data which is used for creating mashups. Any data that is provided to the mashup tool is converted to XML format and then utilized for the required purposes. A feed generator is the component which actually generates the XML document from the data provided. Once the feed is generated, it is registered in the catalogue.

##### ***2. Catalogue:***

The catalogue facilitates sharing and discovery of mashup building blocks, i.e. data feeds, data level and presentation level mashups. It is a repository of all the mashup objects.

##### ***3. Data Mashup Builder/Editor:***

The data mashup builder and editor is a graphical interface embedded with self –service features for the enterprise users to rapidly create data mashups.

##### ***4. Mashup Presentation Builder:***

The mashup presentation builder is an interface which enables mashup presentation creation by creating, assembling, configuring and designing mashup pieces. The working mechanism of the presentation builder is based on widgets.

##### ***5. Widget Creator:***

Widget creator allows building custom widgets by skilled users through programming initiatives. Widget creator also allows to export or import widgets to and from external files.

#### **6. Pages:**

A particular mashup presentation is created in a page. After creating a mashup presentation, that page can be published for access by other users.

#### **7. Spaces:**

Spaces are hierarchies of pages which contain pages with mashups of similar nature. Spaces allow the organization of mashups in a fruitful manner.

#### **8. Mashup Apps:**

Mashup Apps allow the created mashup presentations to be accessed in different external destinations and devices. The different destinations can be enterprise or consumer web applications i.e. Microsoft Sharepoint.

### **4.2.5 Required Functionalities:**

In alignment with the user tasks mentioned in the previous section and based on our reviewing of IBM Mashup Center and Jackbe presto (Jackbe, 2012; Kasman & Roder, 2011b; Singh, 2008a), we have identified the required functionalities to perform them. The functionalities are provided through various features in a mashup tool. The features are exposed through various blocks. For creation and modeling data tasks we have identified the following features:

1. *Create/Register Feeds*: For creating data feeds from the desired data sources and registering them in the platform catalogue
2. *Invoke*: For adding data sources from the internet or through URLs
3. *Extract*: For fetching the desired items from a data feed
4. *Filter*: For selecting a specific item from a result set based on some conditions
5. *Group*: For identifying repeating items based on the unique values for one or more fields.
6. *Input*: For providing dynamic input fields and to be used by other components



7. *Join*: For combining repeating items of multiple block based on a condition
8. *For Each*: For replacing values in a specific feed with repeat values from another feed based on a condition
9. *Loop*: For adding blocks in a loop
10. *Merge*: For combining several similar results sets
11. *Select*: For select only specific items from a group
12. *Sort*: For sorting repeating items in a feed based on field
13. *Transform*: For changing the structure, organization or data result sets to a desired format
14. *Aggregate*: For performing simple calculations on a set of repeating items. The calculations type include:
  - a) Average
  - b) Count
  - c) Maximum
  - d) Minimum
  - e) Sum
  - f) Variance
15. *Date Functions*: For performing various functions to format the date as desired
16. *Numeric Functions*: For performing a complex range of mathematical operations on a result set
17. *String Functions*: For building or modifying strings
18. *Boolean Functions*: For performing Boolean operation on a value
19. *Widgets*: Widgets are miniature applications embedded in a web page. Types of most popular widgets in a BI context are:

- a) Map: For displaying content which includes fields of geographic information
- b) Visualization: For displaying numeric content in a desired visualization
- c) Feed Reader: For displaying text based RSS or ATOM feeds
- d) Data Viewer: For displaying data from a binary file in a grid view
- e) Google Gadgets: For wrapping and displaying Google Gadgets in a widget
- f) Java Applet: For displaying Java based content typically deployed in an external web resource
- g) Flash Content: For displaying external Adobe Flash or Shockwave content
- h) Media Content: For displaying pictures or videos
- i) Action Timer: For setting up a timer to control the frequency of an event passing from in between widgets
- j) Nested Data Viewer: For organizing data in table and drilling down
- k) Event Explorer: For publishing event data from a different widget
- l) HTML Markup: For displaying a webpage based on the provided HTML code
- m) Portal : For displaying specific portal pages
- n) User Input: For providing simple control inputs to be taken as parameters
- o) Website Displayer : For displaying websites

20. *API Integrator*: For integrating external web functionalities exposed through APIs. In our case the resources provided by BI system, i.e. reports, are accessed by the mashup platform by using the API of BI system.

21. *OpenSocial Gadget Integrator*: For integrating similar social gadget tools to Google Gadgets.

For collaborative tasks we have identified the following features:

1. *Add Feedback:* For providing comments for data feeds, data mashups, and presentation mashups
2. *Rate:* For rating data feeds, data mashups, and presentation mashups
3. *Add / Update Meta data:* For adding and modifying the description, category, provider & tags.
4. *Add /Update Properties:* For adding/updating attributes of blocks and artifacts
5. *Review Dependencies:* For reviewing the artifacts depending one on artifact before deleting it.
6. *Publish:* For publishing data feeds, data & presentation level mashups as well other artifacts like widgets for usage by an assigned user group
7. *Share:* For sharing resources via email or other mechanisms.

Administrative tasks can be done with the following features:

1. *User Administration:* For configuring user repository , integrating access directories, managing user and users groups etc
2. *Security Administration:* For ensuring the conformance of policies and access control for users
3. *Server Administration and Configuration:* For tasks related to configuring and managing the application server
4. *Performance Administration:* For ensuring the application is reliable in terms of performance.
5. *Repository Administration:* For overlooking all the databases connected, connection pools etc

In the following chapter, we are going to present the implementation of two mashups which we have done using IBM Mashup Center.

# Chapter 5 Prototypes & Evaluation

---

This chapter presents two prototype mashup created through created through IBM Mashup Centre. The implemented use cases were selected from the projects undertaken by the IBM Cognos SDK team as well as solutions already requested but not yet delivered to the clients. These projects consisted of specific situational applications that clients requested to be developed for them. The need for these applications resulted from changing business requirements of the clients. In order to make the best usage of their available BI data at a given time, the business users experienced the need for certain application services. Out of these two client projects, one of them has been already delivered as separate applications, while the other was in the requirement analysis phase. The unavailability of these requested services in the clients' BI tool resulted in the clients requesting these applications. For our purpose we analyzed the requirements of these applications and created them with IBM's own mashup platform – IBM Mashup Centre.

## 5.1 Prototype 1: Airplane Seat Information

### 5.1.1 Overview:

The purpose of this application is to extend the usage of airplane seat information by overlaying it on an airplane seat map. In general, through the Cognos BI application the clients can generate the reports selecting a specific seat with the application. The report consists of data regarding seat arrangement of airplane. But through the reports the users are only able to see a specific seat data and make their decisions on that. In order to be able to have a bigger picture, the reports

need to be complemented with an airplane seat image where a user can select a specific seat from the picture and get the associated data. This is a situational requirement of the users which Cognos BI doesn't provide. The solution has to be custom built in order to provide this service.

### **5.1.2 Application Requirements:**

The requirements of this situational application are as:

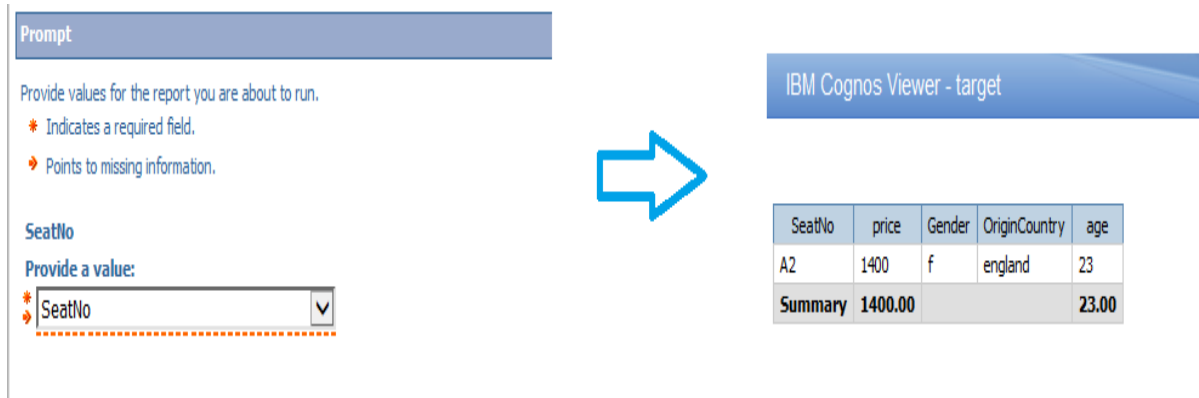
1. Provide a visual outlay of an airplane seat arrangement where user can select specific seats according to their needs
2. Output the associated seat information from the Cognos BI in the same visual pane.

### **5.1.3 Prototype Mashup Implementation:**

The mashup creation process for the above mentioned airplane seat information application consists of two main steps:

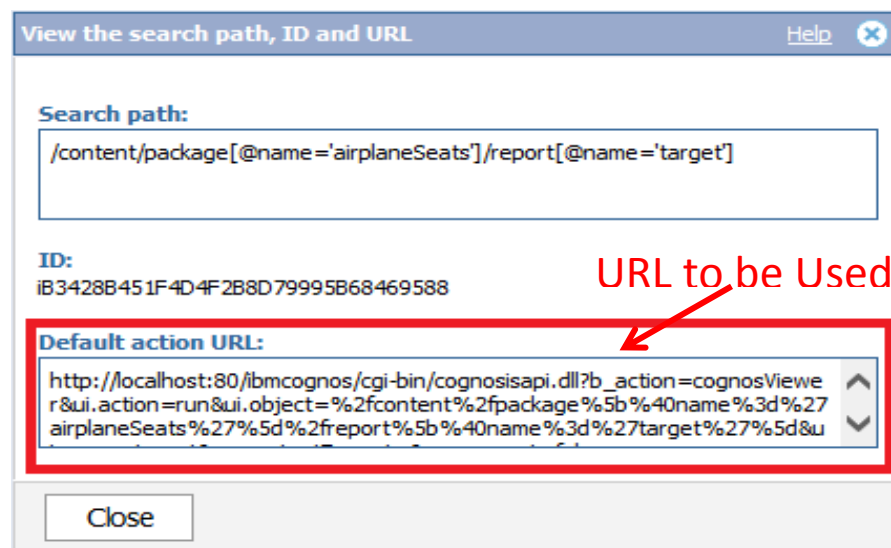
#### **5.1.3.1 Cognos Report Creation and Access to the Report:**

The first step of the mashup creation process is the report generation and fetching the report from Cognos BI. The data of the airplane seats resides in the BI data base and is already modeled. In the Cognos BI application, the user has to select the specific report and input the seat number in order to create a report containing the information of that specific seat. Figure 16 illustrates the process.



*Figure 16: Report Prompt Selection and Report Generation*

In order generate this report from IBM Mashup Center, we have to access the report through the default URL. This default URL is provided in the report -properties section in Cognos BI. Figure 17 shows the access mechanism of this report provided through its properties.



*Figure 17: Report URL for Mashup Center Usage*

### 5.1.3.2 Mashup Composition :

The composition of the mashup through IBM Mashup Center requires working in the mashup presentation builder as there is no data modeling need for this specific situational application. For composing this mashup we need four widgets: user Input, URL customizer, and two website

displayers. Once we have added these widgets in the mashup page, we start our composition process by configuring the URL customized widget. In the settings mode of the widget we load the URL of the report. But before doing that we have to trim out the special characters from the URL and add the prompt parameter with a default value which is *p\_seat=A1*. In figure 18 we can see the loaded reformatted URL and the other parameters required. The reformatted URL is:

*http://localhost:80/ibmcognos/cgi-bin/cognosisapi.dll?b\_action=cognosViewer&ui.action=run&ui.object=/content/package[@name='airplaneSeats']/report[@name='Seat Prompt']&ui.action=run&run.prompt=false&p\_seat=A1*

URL Customizer

Enter or paste a URL:

Values to customize URL request:

Enter or replace custom default values:

b\_action

ui.action

ui.object

ui.action

run.prompt

p\_seat

Default parameter to receive single value:

*Figure 18: Configuring the URL customizer widget*

Once we have configured the widget, we need to setup an input mechanism to feed into the URL customizer widget. For that purpose we have setup the user input widget with 'Seat No.' field. The next step is to wire the widgets. In figure 19 we can see that user input sends the submitted seat number as any data to the url customizer. The URL customizer on the other hand sends its customized URL to the website displayer as a URL.



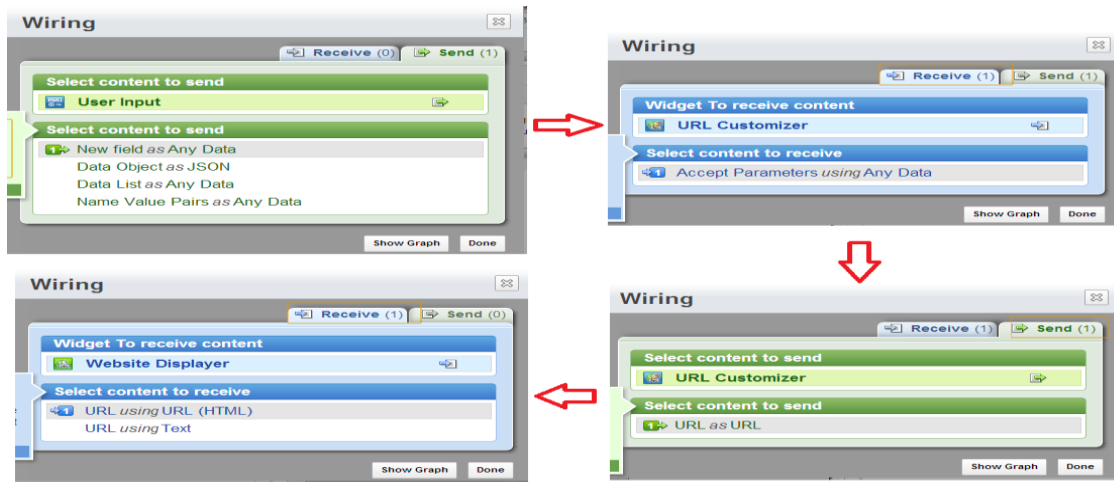


Figure 19: Wiring configuration of the widgets

The final step is to add the airplane seat image from which the user will select the desired seat. After configuring all the widgets, the mashup composition is completed. In figure 20 we see the created airplane seat information mashup. Here the users can select a seat from the image and input the number of the seat which will fetch the relevant data from Cognos BI application

The screenshot shows the IBM Mashup Center interface with the 'Airplane' mashup selected. The main content area displays an airplane seat map, a table of seat information, and a user input field for the seat number.

**Website Displayer**

IBM Cognos Viewer - Seat Prompt

SeatNo	price	Gender	OriginCountry	age
A2	1400	f	england	23

Sep 30, 2013 1:09:07 PM

**URL Customizer**

```
http://localhost:80/ibmcognos/cgi-bin/cognosisapi.dll?
b_action=cognosViewer&ui.action=run&ui.object=/content/package%5B@name='airplaneSeats'%5D/report%
5B@name='Seat%20Prompt'%5D&run.prompt=false&p_seat=A2
```

**User Input**

Seat No.:

Figure 20: Airplane Seat Information Mashup

## **5.2 Prototype 2: Organizational Hierarchy Mashup**

### **5.2.1 Overview**

Cognos BI consists of numerous built in chart functions, which the users can utilize to generate charts in their reports. But with the advancement of data visualization techniques, the need for new representations of data becomes more frequent. In this mashup prototype, we have created an organization hierarchical chart which can be drilled down to see the element inside. The chart consists of hierarchical positions of an organization. It is arranged in such manner that clicking a certain position would show the positions underneath it. Simultaneously the selection of the elements from the chart would show the related information i.e. position name and salary information from Cognos BI. This situational requirement of a specific visualization type was requested by the client to be developed for them, which is in process now. In the meantime, we have created a mashup prototype of the application according to the client requirements and successfully demoed it to the Cognos SDK team.

### **5.2.2 Application Requirements:**

The requirements of the situational application in question are as follows:

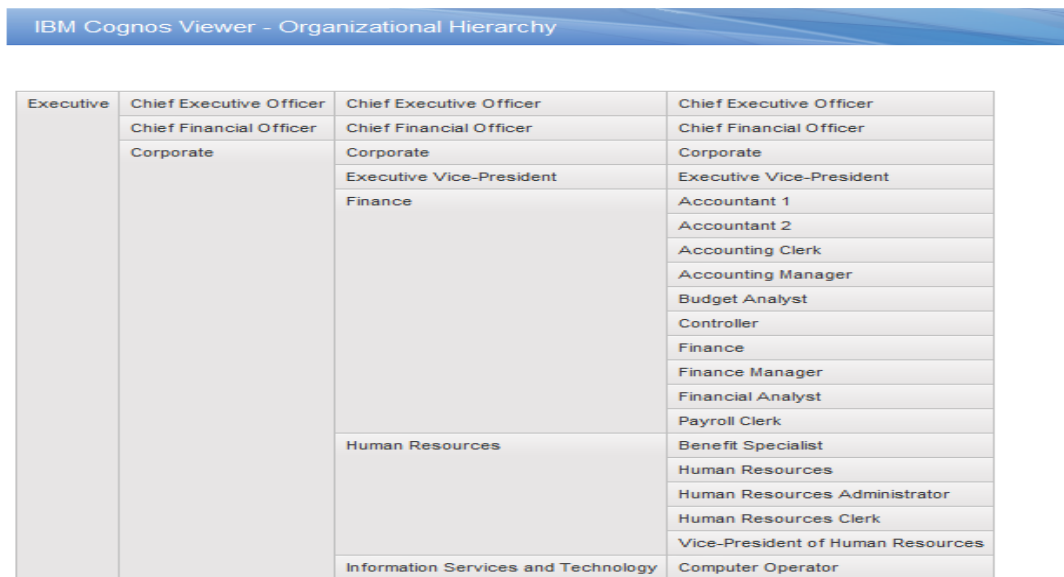
1. Overlay the Cognos BI organizational report data in hierarchical chart.
2. Embedded drill down feature in order navigate into the elements of the chart.
3. Generate reports from Cognos BI based on the elements selected from the chart.

### **5.2.3 Prototype Mashup Implementation:**

The organization hierarchy mashup creation process consists of the following steps:

### 5.2.3.1 Cognos Report Access through CMS and Direct URL:

Two reports are utilized for creating the organizational hierarchy mashup. One is report which contains the hierarchical information of the organization and the other one consists of the related pay-scale information. The hierarchy report is to generate the chart in Mashup Centre and is accessed through CMS using Restful Web Services. Figure 21 shows the report. The data of this



IBM Cognos Viewer - Organizational Hierarchy

Executive	Chief Executive Officer	Chief Executive Officer	Chief Executive Officer
	Chief Financial Officer	Chief Financial Officer	Chief Financial Officer
	Corporate	Corporate	Corporate
		Executive Vice-President	Executive Vice-President
		Finance	Accountant 1
			Accountant 2
			Accounting Clerk
			Accounting Manager
			Budget Analyst
			Controller
			Finance
			Finance Manager
			Financial Analyst
			Payroll Clerk
		Human Resources	Benefit Specialist
			Human Resources
			Human Resources Administrator
			Human Resources Clerk
			Vice-President of Human Resources
		Information Services and Technology	Computer Operator

*Figure 21: Organization Hierarchy Report*

report is accessed via web services in the Mashup Center in order to be modeled. For this purpose we are using the Report ID, which is a mechanism to integrate the report to other external applications. Using this Report ID we can construct a URL which will access the data in a Layout Data XML format (Sleigh & Johari, 2010). Accessing via the LDX format allows us to model the data in the mashup center in order to transform the structure so that it can be used to generate a hierarchical chart. By using the CMS we can construct the URL using the following parameters:

Resource type: reportData

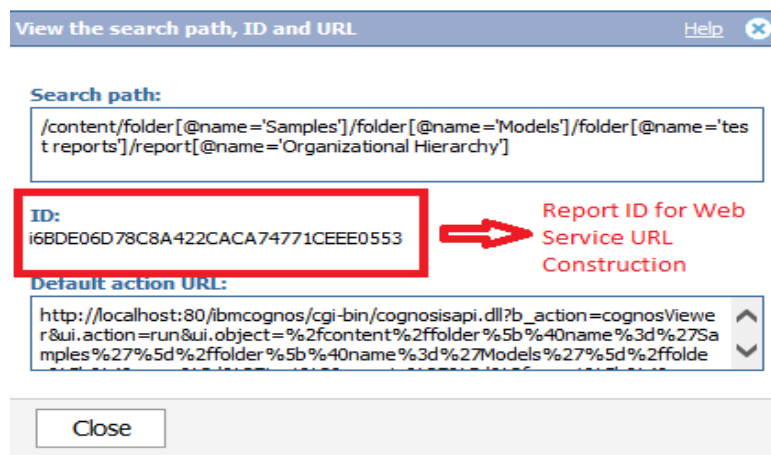
Source type: report

Resource\_id: report store id: i6BDE06D78C8A422CACA74771CEEE0553

The constructed URL:

*http://localhost/ibmcognos/cgi-*

*bin/cognosisapi.dll/rds/reportData/report/i6BDE06D78C8A422CACA74771CEEE0553*



*Figure 22: Accessing the Report data through Web Service*

As for the report containing the information position pay-scale, the process is similar to the report accessing process mentioned in the first prototype. The report will be URL access directly from the mashup server. Figure 23 shows the generated report. It should be noted here, that because of inconsistent data in the database, the paid hourly column value is 0.

Position name	Minimum salary	Maximum salary	Paid hourly
Benefit Specialist	22000	50000	0

*Figure 23: Employee Position and Pay-Scale Information Report*

The search path for accessing the report can be found from the report settings. As before the special characters have to be trimmed out. The url for directly accessing the report is:

```
http://localhost:80/ibmcognos/cgi-  
bin/cognosisapi.dll?b_action=cognosViewer&ui.action=run&ui.object=/content/folder[@name  
='Samples']/folder[@name='Models']/folder[@name='test reports']/report[@name='Prompt  
report for poission info']&run.prompt=false&p_pos=Human Resources Clerk
```

### 5.2.3.2 Data Modeling and Transformation

To create an organizational chart through the ‘Nested Data Viewer’ widget, we have to model the data and transform it into a structure that can be read by the widget. For that purpose we will first create a data mashup in the data mashup builder module of IBM mashup center. Once we are in the create mode in the data mashup editor, we will require three operators: source, transform, and publish. Figure 24 shows constructed data level mashup which will be used in the mashup presentation.

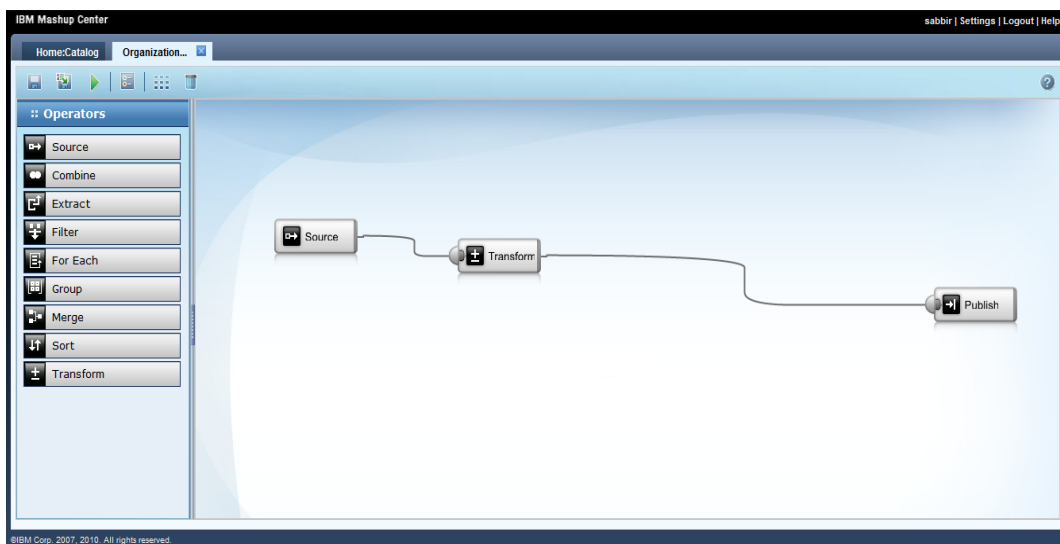
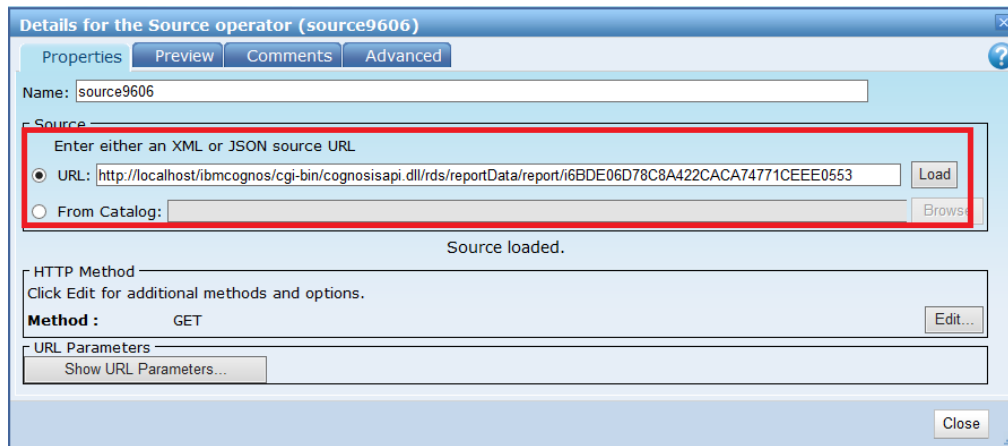


Figure 24: Organization Hierarchical Data Level Mashup

In the source operator we have defined the source of the data which will be modeled. Here we inputted the web service URL of the organization hierarchy report. Figure 25 shows the

configuration of the source operator. Once the LDX document has been fetched it is ready to be modeled which is done by the transform operator. In the transform operator edit mode we can



*Figure 25: Source Operator Configuration*

see the LDX document. This LDX document will act as the input as we see in figure 25. As for the output, we have created a new XML tree. The LDX document accessed via CMS web service has a lot of overhead in its DTD as well as in other elements which embedded in the Cognos report. But these components are not required for our purpose. In the output structure we can only choose the values that we need from the LDX document. Since the organization hierarchy has four levels (figure 21), we have to create four elements as well in the output structure. Once the placeholders are created, we manually input the first value as “CEO”. For inserting the rest of values we can drag and drop the values from the input structure in those placeholders or under each ‘values’ in the output structure. Figure 26 elaborates the process of the output structure creation. In transferring the values from the input to the output structure we have to create only one instance as the transform operator will automatically generate the other similar elements from the iterating values of the input document. Our required values, which we have dragged and dropped in the output structure, reside in the following locations in the input tree :

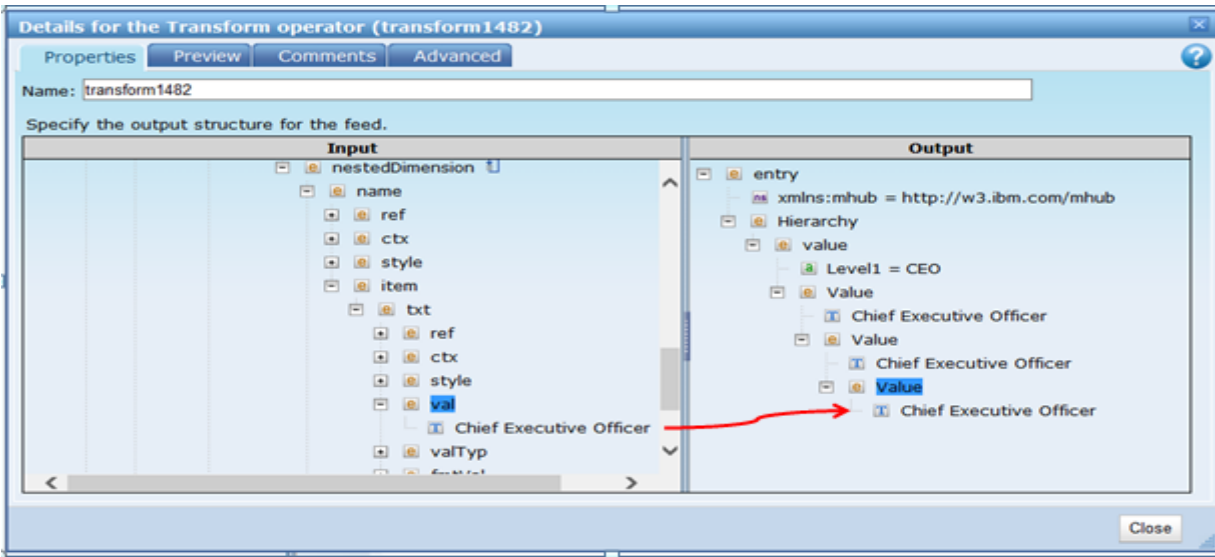


Figure 26: Data Modeling and Transformation

Level 2: *document>pages>page>body>item>row>name>item>txt>fmtval*

Level2: *document>pages>page>body>item>row>nestedDimension>  
name>item>txt>fmtval*

Level3: *document>pages>page>body>item>row>nestedDimension>nestedDimension>  
name>item>txt>fmtval*

The last step in the data mashup creation process is publishing it through the publish operator as an XML document. Once published the XML document looks like as:

```

<?xml version="1.0" ?>
- <root>
- <item>
- <Hierarchy>
- <value Level1="CEO">
- <Value>
Chief Executive Officer
- <Value>
Chief Executive Officer
<Value>Chief Executive Officer</Value>
</Value>
- <Value>
Chief Financial Officer
- <Value>
Chief Financial Officer
<Value>Chief Financial Officer</Value>
</Value>
- <Value>
Corporate
- <Value>
Corporate
<Value>Corporate</Value>
</Value>
- <Value>
Executive Vice-President
<Value>Executive Vice-President</Value>
</Value>
- <Value>
Finance
<Value>Accountant 1</Value>
<Value>Accountant 2</Value>
<Value>Accounting Clerk</Value>
<Value>Accounting Manager</Value>
<Value>Budget Analyst</Value>
<Value>Controller</Value>
<Value>Finance</Value>
<Value>Finance Manager</Value>
<Value>Financial Analyst</Value>
<Value>Payroll Clerk</Value>
</Value>

```

Figure 27: Organizational Hierarchy Data Mashup in XML format

### 5.2.3.3 Mashup Presentation Composition:

In order to compose the organization hierarchical mashup we require three widgets: Nested Data Viewer, URL customizer, and Webiste displayer. In the nested data viewer widget we have used the created data level mashup described in the previous section as source. Since the data is already modeled in accordance to the nested data viewer widget requirements, no configuration is required here. As for the showing the employee position pay-scale information, the configuration of the URL customizer widget is showed in figure 28

URL Customizer

Enter or paste a URL:

Values to customize URL request:

Enter or replace custom default values:

b_action	<input type="text" value="cognosViewer"/>
ui.action	<input type="text" value="run"/>
ui.object	<input type="text" value="/content/folder[@name]"/>
ui.action	<input type="text" value="run"/>
run.prompt	<input type="text" value="false"/>
p_pos	<input type="text" value="Marketing"/>

Default parameter to receive single value:

Figure 28: URL Customizer Widget Configuration



The URL to be load here is :

*http://localhost:80/ibmcognos/cgi-*

*bin/cognosisapi.dll?b\_action=cognosViewer&ui.action=run&ui.object=/content/folder[@name  
='Samples']/folder[@name='Models']/folder[@name='test reports']/report[@name='Prompt  
report for poistion info']&run.prompt=false&p\_pos=Human Resources Clerk*

Once the URL is loaded it can be wired to website displayer where it sends the URL as URL. Figure 29 shows the completed organizational mashup where the users can drill down into a specific position or dept. Once an element has been selected the associated pay-scale information is fetched as a Cognos report from the BI application.

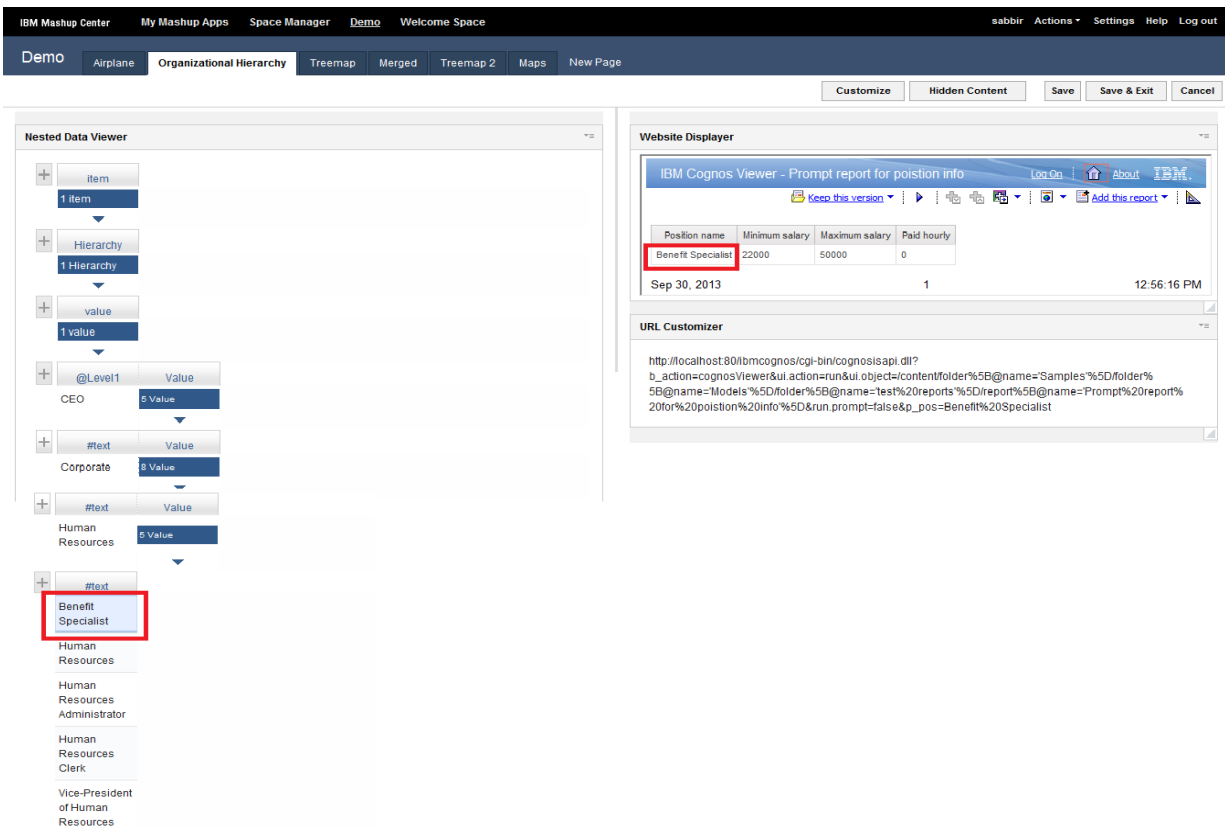


Figure 29: Organization Hierarchical Mashup Presentation

### 5.3 Evaluation of the Prototype Mashup Applications

In order to evaluate the mashups we have created, we demonstrated our mashups to the Cognos SDK team, specialized in delivering mashup products to their BI clients. This team consisted of 13 members, specializing in different areas of software development. We asked them to rate our mashups based on four criteria which had further sub-criteria, which were discussed in our literature review chapter. We also compared them against the current available solutions or the solutions that are being built. Below is the detailed description based on their feedback:

*i. Usefulness:*

When it comes to satisfying the needs of both business users and IT professionals, the feedback that we have gathered indicated that mashups will be useful for both user groups. In terms of the mashup design features, our mashups support data mediation. In the airplane mashup we have used data conversion and for the organizational hierarchy mashup both data conversion and transformation have been used. In addition to that both process creation and collaboration activities are supported. In terms of mashup techniques used, we have utilized wiring as well as loading the data from spreadsheet. While enquiring about the usage of programming by demonstration technique and webpage customization capability, the Cognos SDK software developers supported the validity of these techniques in our mashups. But as for the supporting mashup creation at code level our mashups were not deemed to be script/language based.

In terms of the technical features the mashups supported, both SOAP and REST protocols can be used. The data retrieval strategies include web services or direct url access. Both RSS and ATOM syndication formats can be used to access an external feed. At the data mashup

construction level we also demonstrated the involvement of light weight process modeling features.

*ii. Ease-of-Use:*

While ensuring whether the mashup creation process can be carried out by a business user, we asked the experts whether the mashup creation process conformed to the criteria in order to make it easily usable. The interfaces of both the mashups are descriptive enough to comprehend minimum technical requirement and features of integrated APIs. The team was also satisfied with the tutorial element embedded in the mashup tools we used and agreed that the error finding components can provide a satisfactory level of assistance during incompatibility. In addition to that, the availability of tools that specifies the user requirements without the usage of technical jargon and the ability to rate those tools would make the learning curve relatively easy. However there were no mechanism available to support user goal achievement or task completion

*iii. Intuitiveness and Cost Reduction:*

The usage of pre-built widgets has made the process of creating mashup presentation very intuitive. We have also shown the ability to utilize social gadgets i.e. Google gadgets, which enable users to use popular consumer internet space tools into their enterprise level applications. As for reducing the situational application development costs, the Cognos SDK agreed that the usage of mashup tools would decrease the costs for their clients. We have further elaborated the cost reductions in the next section where we have compared our mashups against that of the Cognos SDK team's

## Comparison with current available solutions:

Based on the feedback from the Cognos SDK team and analysis of their solutions we have compared our created mashups with theirs. Table 12 summarized the comparison points.

	<b>Prototype 1: Airplane Seat Mashup</b>		<b>Prototype 2: Organization Hierarchical Mashup</b>	
<i>Evaluation Criteria</i>	<i>Solution Developed by the Service Provider</i>	<i>Mashup Application</i>	<i>Solution Developed by the Service Provider</i>	<i>Mashup Application</i>
<b>Meeting the Requirements</b>	Meets the client requirements	Meets the client requirements	Development in Process	Meets the client requirements
<b>Application Feature Richness</b>	Feature Rich	Not feature Rich	Development in Process	Not feature Rich
<b>BI Resource Integration</b>	Yes	Yes	Yes	Yes
<b>Application Development Time</b>	1 Week	1 Hour	2 Weeks	1 Day
<b>Effort and Skills first application</b>	Programming and Database administration skills required	Business Users (without programming and Database administration skills) can build the application	Programming and Database administration skills required	Business Users (without programming and Database administration skills) can build the application
<b>Evaluation Criteria</b>	Solution Developed by the Service Provider	Mashup Application	Solution Developed by the Service Provider	Mashup Application
<b>Application Development Cost</b>	Cognos BI License cost + Customized Solution Development + Application Support Cost	Cognos BI License cost + IBM Mashup Center License Cost	Cognos BI License cost + Customized Solution Development + Application Support Cost	Cognos BI License cost + IBM Mashup Center License Cost
<b>Modification through Self-Service</b>	No	Yes	No	Yes

*Table 11: Comparison of mashup prototype implementations with available IT developed solutions*

As we can see, the mashup prototypes meet the requirements that the clients put forward to the service provider. To be relevant in a BI context, they successfully integrate the BI contents. The significant advantage of these mashup prototypes over IT built solutions is the required development time. In addition to that, programming and database administration skills are not required for developing these situational applications and once they have been developed users can also modify them. Table 1 summarized the comparison between solutions developed or being developed with our mashup prototype applications. One other advantage is the cost factor. Traditionally client organizations purchase their BI systems from a specific vendor. On top of that, for their situational demands they have to purchase the services of a vendor regardless of having a dedicated IT team, the reason being the need to access and understand the APIs of the BI system which has to be done through the vendor. In addition to that, the long term support for these applications would cost extra. On the other hand, for using a mashup platform, the clients have to purchase the product license only in addition to their current BI costs. This significantly reduces the cost of developing situational BI services for the clients.

# Chapter 6      Conclusions

---

## 6.1 Summary of the Research

Today's data driven organizations, even though comprehending the need for better utilization of their data and BI resources, are faced with tight IT budgets. Mashup driven self-service BI liberates the under-resourced IT department of organizations from having to respond to an unrelenting backlog of user-requests. Through an enterprise BI mashup platform users can take charge of solving those situational problems and needs themselves without having to change the complex data models and structures. Regardless of these obvious benefits there has not been any significant development undertaken to provide organizations with the right mashup toolkits which would aid in removing the bottlenecks in providing situational BI services. The current literature has not addressed the development of BI mashup tools in much detail. In addition to that there is a lack of formal knowledge base in the domain of Enterprise BI Mashups.

In this research, we attempt to provide a basic utility framework as the preliminary steps in defining lexical knowledge in the domain of Enterprise Business Intelligence Mashups. In constructing the utility framework we first formulated a taxonomy model of BI Mashups by identifying key business trends and common end-user requirements in situational BI use-cases. Utilizing this taxonomy framework we formalized the components of BI mashups by constructing a utility framework for enterprise BI mashup toolkits. In order to prove the efficacy of a mashup platform, we also implemented the traditionally built mashups with through enterprise level mashup tools available. The validation of our created BI mashups was done

through demonstrating them to the Cognos SDK team IBM who built mashups for their BI clients through programming initiatives.

## **6.2 Contributions of the Thesis:**

In our effort to create a utility framework and advance the understanding of Enterprise BI Mashups domain, the resulted benefits of our research work are as follows:

***Pattern Identification of Situational BI requests of Clients:*** Through reviewing relevant literature and analyzing various projects undertaken by the IBM Cognos SDK team, we were able to identify the generic trend of client requests in terms of their situational BI need. This process helped us to identify the business drivers of BI mashups. The assembled patterns acted as the concepts and constructs for the creation of a taxonomy framework of enterprise BI mashups

***Taxonomy Framework:*** Our taxonomy model aims to fill the gap in current literature in the domain of BI mashups by providing a basic framework for understanding the enablers, drivers, user characteristics & requirements as well as situational data requirements of organizations. The formulated taxonomy framework provides a classification of targeted BI mashup users, the development methods, functional range and associated data needs. The framework also identifies the enablers and drivers of Enterprise BI Mashups.

**Utility framework:** Utilizing the taxonomy framework, our constructed utility framework offers Enterprise BI Mashup application development guidelines for vendors. This framework provides a detailed description for composition of Enterprise BI Mashup tools in terms of underlying infrastructure, user types & tasks, and features & functionalities required to perform those tasks. The goal of the framework is to the application development process of BI mashup toolkits which will empower different types of users with self-capabilities, thus increasing the adoption and usage of BI applications as well removing the bottlenecks associated with delivering situational BI services to the end-users.

While implementing the situational BI use cases with available mashup tools some other benefits of our research works are:

**Integration of BI and Mashup products:** We have integrated the Cognos BI application with IBM Mashup Center, which are both products of the partner organization. This integration of the two different products which are aimed towards providing different services to the clients will enable process efficiency across the organization and improved visibility of offered services in a large organization like IBM. The integration also offers potential IT time and cost savings. The biggest benefit this integration process has to offer is the enablement of user driven innovation by empowering users with self service capabilities when it comes to their situational BI needs. The integration was done using the Cognos Mashup Service (CMS).

**Product enhancement and novel use cases:** Through the implementation of situational applications in a mashup platform utilizing the BI resources, we have introduced new use cases



of BI mashup development tools such as Cognos Mashup Service (CMS) and Cognos Software Development Kit (SDK). More specifically we have utilized the CMS of IBM Cognos to enable the usage of BI data in a mashup platform for self-service application development. In terms of CMS, the new use cases would be beneficial for attracting more clients and accelerate the growth of the product. We have also demonstrated rapid prototype creation of situational BI applications for purposes like client demonstration. Through a mashup platform, as the one we have used in our research, business users are able to apply their functional expertise to tailor processes and applications in the realm of situational BI in a way that improves performance.

## 6.3 Limitations

In terms of the limitation of our work, the lack of available mashup tools at enterprise level has restricted the implementation of use cases to a limited amount. For the implementation purpose we have used only one mashup tool – IBM Mashup Centre. We had limited access to Jackbe Presto because of licensing issues. For our implementation, we chose the use cases which were the most popular requests of the clients. There is a need to implement more use cases which are varied in nature in order to validate a mashup platform's efficacy for delivering situational BI services. In addition to that, we need to implement the use cases with more mashup tools. This is necessary in order to find out what the current products in the market offer in terms of meeting the data needs of clients. As for the BI application used in this research, we have opted for Cognos which is a very popular application among the BI clients. But we have to increase the scope to other leading vendors as well. Moreover, open source BI tools are also making their way among the BI application user base. This research doesn't take into account these other BI applications.

Another limitation of this research is related to testing. The lack of a prototype toolkit didn't allow us to test our frameworks. As these frameworks were intended to create a preliminary knowledge base in the domain of Enterprise BI Mashups, which the current literature lacks, testing the frameworks extensively was out of the scope of this research at its current stage. Having said that, not thoroughly testing both the frameworks and the prototype implementations is a major limitation of this research. In terms of test categories, we only performed acceptance tests for our prototypes. These prototypes were created as a proof-of-concept and aren't ready to be used at a production level. The underlying infrastructure of our setup was a very simple one to support the initial proof-of-concept implementations and does not focus on scalability of the applications. As we mentioned previously, because of the unavailability of enterprise level mashup tools, we couldn't use any high performance mashup platform. The IBM Mashup Centre was intended for experimental application development by less tech savvy users, thus making it untested for application development in a BI context. Typically any medium or large organization using BI tools has large number users. So the performance and the reliability of a BI Mashup tool are of high importance.

As for the validation and testing of our frameworks, we depended on the feedback of Cognos SDK team. The taxonomy framework lacks formal ontology validation. Even though our constructed taxonomy framework was tested and improved according to the feedback of the software development experts in the mashup domain, formal ontology validation is necessary. Moving on to the utility framework, the absence of a prototype toolkit didn't allow us to test the framework in an end-user environment from a technical setup point of view. Our frameworks also do not emphasize on the governance structure of mashup tools which is very important in an

enterprise environment. Our focus was creating a preliminary knowledge base which would validate the efficacy of Enterprise BI Mashup tools.

## 6.4 Future Work and Research Directions

As part of our future work, we plan to guide our research in a manner which will address the aforementioned limitations. This will include developing a prototype toolkit and testing it with a client setup. Thus we can test our utility framework. For the taxonomy we plan to undertake both accuracy and completeness validation of the framework. We also plan to expand our currently used products by reaching out to other vendors in terms of BI and mashup tools. This will allow us to apply our frameworks in a different setting. In addition to different technical settings and products, we also plan to focus on specific clients types in terms of their functionalities and apply our frameworks on them. Our future plans also include publishing the full results in a relevant journal. We have already published the first part of our work, the taxonomy framework, in the 2<sup>nd</sup> International Conference on Informatics and Application in Lodz – Poland.

Other research that can be fruitful in the domain of Enterprise BI Mashups is studying the self-service application development with the advancement of web technologies. *Self-service* being one of major offerings of mashup tools will be continue to evolve with the advancement of technology as well as end-user needs. Studying the evolution of self-service application development would allow us to improve the efficacy of mashup platforms. In terms of Business Intelligence, the proliferation of data within organizations as well as in the internet means BI tools have to evolve rapidly to cater to the enterprise client needs. This will make analytic

service delivery more complex and make BI implementations more prone to failure. Studying the usage patterns of current and future BI applications would help us in understanding the user expectations from these tools and thus come up with better solution delivery mechanism through mashup platforms.

## References

- Alnoukari, M., Alhawasli, H. A., Alnafea, H. A., & Zamreek, A. J. (2012). Business Intelligence: Body of Knowledge, 1–13.
- Anna, O. (2011). Tutorial: Introduction to creating mashups using IBM Mashup Center 3.0. Retrieved from [http://www-10.lotus.com/ldd/mashupswiki.nsf/dx/Tutorial\\_Introduction\\_to\\_creating\\_mashups\\_using\\_IBM\\_Mashup\\_Center\\_3.0](http://www-10.lotus.com/ldd/mashupswiki.nsf/dx/Tutorial_Introduction_to_creating_mashups_using_IBM_Mashup_Center_3.0)
- Berners-lee, T. I. M., Hendler, J., & Lassila, O. R. A. (2002). THE WEB. *Scientific American*, (April).
- Bitzer, S., & Schumann, M. (2009). Mashups : An Approach to Overcoming the Business / IT Gap in Service-Oriented Architectures, 284–295.
- Bozzon, A., Brambilla, M., Facca, F. M., & Carughu, G. T. (2009). A Conceptual Modeling Approach to Business Service Mashup Development. *2009 IEEE International Conference on Web Services*, 751–758. doi:10.1109/ICWS.2009.24
- Browne, D., Desmeijter, B., Dumont, R. F., Kamal, A., Leahy, J., Masson, S., ... Keen, M. (2010). *IBM Cognos Business Intelligence V10.1 Handbook*. Retrieved from <https://www.redbooks.ibm.com/redbooks/pdfs/sg247912.pdf>
- Cappiello, C., Daniel, F., Matera, M., Picozzi, M., Weiss, M., & Milano, P. (n.d.). Enabling End User Development through Mashups : Requirements , Abstractions and Innovation Toolkits, 9–24.
- Carney, M. (2012). Traditional Business Intelligence Training Models Must Adapt to Support End-User Adoption. *BeyeNETWORK*,. Retrieved from <http://www.b-eye-network.com/view/16151>
- Carrier, N., Deutsch, T., Gruber, C., Heid, M., & Jarrett, L. L. (2008). The business case for enterprise, (August).
- Curbera, F., Duftler, M., Khalaf, R., & Nagy, W. (2002). Unraveling the Communication : SOAP, (April).
- Daniel, F., & Matera, M. (n.d.). Turning Web Applications into Mashup Components : Issues , Models , and Solutions.
- De Hoog, G. S. (1981). Methodology of Taxonomy. *International Association for Plant Taxonomy (IAPT)*, 30(4), 779–783.

- DellaPorta, L. (2012). Business Intelligence vs. Business Analytics. *RJMetrics Blog*. Retrieved March 07, 2013, from <http://blog.rjmetrics.com/business-intelligence-vs-business-analytics/>
- Eckerson, W., & Shermann, R. (2008). Strategies for managing spreadmarts. *TDWI Research Report*, 13.
- Elliott, T. (2011). Business Analytics vs Business Intelligence? *Business Analytics*. Retrieved from <http://timoelliott.com/blog/2011/03/business-analytics-vs-business-intelligence.html>
- Evelson, B. (2010). *Agile BI Out Of The Box*. Retrieved from <https://www.wherescape.com/getattachment/resource-library/white-papers/agile-bi-out-of-the-box/Agile-BI-out-of-the-box.pdf/>
- Evelson, B. (2011). *Trends 2011 And Beyond : Business Intelligence*. Retrieved from [http://www.mxisoft.com/Portals/53068/docs/Forrester trends\\_2011\\_and\\_beyond\\_business\\_intelligence%5B1%5D.pdf](http://www.mxisoft.com/Portals/53068/docs/Forrester_trends_2011_and_beyond_business_intelligence%5B1%5D.pdf)
- Gartner. (2009). Gartner Reveals Five Business Intelligence Predictions for 2009 and Beyond. *Gartner Business Intelligence Summit*. Retrieved from <http://www.gartner.com/newsroom/id/856714>
- Griffiths, G. C. D. (1973). Some fundamental problems in biological classification. *Syst. Zool.*, 22, 338–343.
- Grimes, S. (2010). *Nimble Intelligence : Enterprise BI Mashup Best Practices BI mashups unite diverse data and* (pp. 1–3). Retrieved from [http://mdc.jackbe.com/downloads/nimblebi\\_grimes.pdf](http://mdc.jackbe.com/downloads/nimblebi_grimes.pdf)
- Guarino, N. (1998). Formal Ontology and Information Systems, (June), 3–15.
- Hassanzadeh, O., Duan, S., Fokoue, A., Kementsietsidis, A., Srinivas, K., & Ward, M. J. (2011). Helix : Online Enterprise Data Analytics. *Information Storage and Retrieval*, 225–228.
- Hildebrand, C., Shankland, P., & Baya, V. (2012). Consumerization of APIs: Scaling integrations. <http://www.pwc.com/>. Retrieved from <http://www.pwc.com/us/en/technology-forecast/2012/issue2/features/feature-consumerization-apis.jhtml>
- Hinchcliffe, D. (2007). The 10 top challenges facing enterprise mashups. <http://www.zdnet.com/>. Retrieved from <http://www.zdnet.com/blog/hinchcliffe/the-10-top-challenges-facing-enterprise-mashups/141>
- Hirschheim, R., Welke, R., & Schwarz, A. (2010). Service-Oriented Architecture: Myths, Realities, and a Maturity Model. *MIS Quarterly Executive*, 9(1).

- Hoyer, V, Stanoevska-Slabeva, K., Kramer, S., & Giessmann, a. (2011). What Are the Business Benefits of Enterprise Mashups? *2011 44th Hawaii International Conference on System Sciences*, 1–10. doi:10.1109/HICSS.2011.490
- Hoyer, Volker, & Fischer, M. (2008). Market Overview of Enterprise Mashup Tools, 708–721.
- Hoyer, Volker, Stanoesvka-Slabeva, K., Janner, T., & Schroth, C. (2008). Enterprise Mashups: Design Principles towards the Long Tail of User Needs. *2008 IEEE International Conference on Services Computing*, 601–602. doi:10.1109/SCC.2008.88
- Imhoff, C., & White, C. (2011). *SELF-SERVICE Empowering Users to Generate Insights*.
- Işık, Ö., Jones, M. C., & Sidorova, A. (2013). Business intelligence success: The roles of BI capabilities and decision environments. *Information & Management*, 50(1), 13–23. doi:10.1016/j.im.2012.12.001
- Jackbe. (2012). Presto Library. Retrieved from <http://mdc.jackbe.com/prestodocs/v3.6/index.html>
- Kasman, A., & Roder, K. (2011a). *IBM Mashup Center Need-to-know tips and tricks , Part 1 : Work with feeds and build data mashups* (pp. 1–29).
- Kasman, A., & Roder, K. (2011b). *IBM Mashup Center Need-to-know tips and tricks , Part 2 : Building mashups and leveraging widgets*.
- Kobielus, J. (2009). Mighty mashups : Do-it-yourself business intelligence for the new economy. Retrieved from <http://www.corda.com/pdfs/mighty-mashups-article.pdf>
- Liu, Y., Liang, X., Xu, L., Staples, M., & Zhu, L. (2009). Using architecture integration patterns to compose enterprise mashups. *2009 Joint Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture*, 111–120. doi:10.1109/WICSA.2009.5290797
- Liu, Y., Liang, X., Xu, L., Staples, M., & Zhu, L. (2011). Composing enterprise mashup components and services using architecture integration patterns. *Journal of Systems and Software*, 84(9), 1436–1446. doi:10.1016/j.jss.2011.01.030
- Löser, A., Hueske, F., & Markl, V. (2009). Situational Business Intelligence. *Lecture Notes in Business Information Processing*, 27, 1–11.
- Luo, X., Xu, H., Song, M., & Song, J. (2008). Research on SOA-based platform to enable mobile mashups. *2008 11th IEEE International Conference on Communication Technology*, 607–610. doi:10.1109/ICCT.2008.4716146
- Makki, S. K., & Sangtani, J. (2008). Data Mashups & Their Applications in Enterprises. In *2008 Third International Conference on Internet and Web Applications and Services* (pp. 445–

- 450). IEEE. Retrieved from [http://ieeexplore.ieee.org/xpl/freeabs\\_all.jsp?arnumber=4545653](http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4545653)
- March, B. S. T., & Storey, V. C. (2008). DESIGN SCIENCE IN THE INFORMATION SYSTEMS DISCIPLINE : AN INTRODUCTION TO THE SPECIAL ISSUE ON DESIGN SCIENCE RESEARCH, 32(4), 725–730.
- Mcafee, A. P. (2006). Enterprise 2 . 0 : The Dawn of Emergent Collaboration, 47(3).
- Merrill, D. (2009). Mashups : The new breed of Web app An introduction to mashups Mashup genres, 1–13.
- Minhas, S. S., Sampaio, P., & Mehandjiev, N. (2012). A Framework for the Evaluation of Mashup Tools. *2012 IEEE Ninth International Conference on Services Computing*, 431–438. doi:10.1109/SCC.2012.19
- Mohammadi, S., Khalili, A., & Ashoori, S. (2009). Using an Enterprise Mashup Infrastructure for Just-in-Time Management of Situational Projects. *2009 IEEE International Conference on e-Business Engineering*, 3–10. doi:10.1109/ICEBE.2009.11
- Negash, S. (2004). Business Intelligence, 13, 177–195.
- Nelson, G. S., Edia, M., & Ntertainment, E. (2010). Business Intelligence 2 . 0 : Are we there yet ? *SAS Global Forum 2010 Business Intelligence / Analytics*, 1–10.
- Nestler, T. (2008). Towards a mashup-driven end-user programming of SOA-based applications. *Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services - iiWAS '08*, 551. doi:10.1145/1497308.1497408
- O'Reilly, T. (2006). Web 2.0 Compact Definition: Trying Again. <http://radar.oreilly.com/>. Retrieved from <http://radar.oreilly.com/2006/12/web-20-compact-definition-tryi.html>
- Ogrinz, M. (2009). *Mashup Patterns - Design and Examples for the Modern Enterprise* (1st ed.). Crawfordsville, Indiana: Pearson Education, Inc.
- Oliver, D., Livermore, C. R., & Farag, N. A. (2009). Self-Service in the Internet Age. In F. Sudweeks, C. Romm Livermore, & D. Oliver (Eds.), *Self-service in the Internet age: expectations and experiences* (pp. 257–274). London: Springer London. doi:10.1007/978-1-84800-207-4
- Oracle. (2013). *Oracle Utilities Application Framework Software Development Kit Developer ' s Guide*.
- Pahlke, I., Beck, R., & Wolf, M. (2010). Enterprise Mashup Systems as Platform for Situational Applications. *Business & Information Systems Engineering*, 2(5), 305–315. doi:10.1007/s12599-010-0121-9



- Popescu, C. (2011). *IBM Cognos Proven Practices : Hands-on IBM Cognos Software Development Kit Programming* (pp. 1–14). Retrieved from [https://www.ibm.com/developerworks/data/library/cognos/development/how\\_to/page565-pdf.pdf](https://www.ibm.com/developerworks/data/library/cognos/development/how_to/page565-pdf.pdf)
- Popovic, A., Coelho, P. ., & Jaklič, J. (2009). The impact of business intelligence system maturity on information quality. *IR Information Research*, 14(4). Retrieved from <http://informationr.net/ir/14-4/paper417.html>
- Ruhi, U., & Choi, D. (2013). Enterprise Mashups for Knowledge Management.
- Scientifique, C. (2003). Using Users ' Expectations to Adapt Business Intelligence.
- Simmen, D. E., Altinel, M., Markl, V., Padmanabhan, S., Singh, A., & Jose, S. (2008). Damia : Data Mashups for Intranet Applications, 1171–1182.
- Singh, H. (2008a). *IBM Mashup Center and the InfoSphere MashupHub , Part 1 : Get started with InfoSphere MashupHub* (pp. 1–20).
- Singh, H. (2008b). IBM Mashup Center and the InfoSphere MashupHub , Part 2 : In-depth look at Feed Mashup Editor within IBM Mashup Center ' s InfoSphere MashupHub Feed mashups, 1–34.
- Sleigh, C., & Johari, I. (2010). *Get started with the IBM Cognos Mashup Service* (pp. 1–11). Retrieved from <https://www.ibm.com/developerworks/data/library/techarticle/dm-1001cognosmashup/dm-1001cognosmashup-pdf.pdf>
- TechRepublic. (2003). Understanding information taxonomy helps build better apps. Retrieved from <http://www.techrepublic.com/article/understanding-information-taxonomy-helps-build-better-apps/>
- Thiele, M., & Lehner, W. (2012). Real-Time BI and Situational Analysis. *Business Science Reference*, 285–287. doi:10.4018/978-1-61350-038-5.ch013
- Tun, T. T., Chapman, R., Haley, C., Laney, R., Hall, W., & Keynes, M. (n.d.). A framework for developing feature-rich software systems \*.
- Vaishnavi, V., & Kuechler, B. (2004). Design Science Research in Information Systems. *Association for Information Systems*. Retrieved from <http://desrist.org/design-research-in-information-systems/>
- Vrieze, P. De, Xu, L., Bouguettaya, A., Yang, J., & Chen, J. (2009). Process-Oriented Enterprise Mashups. *2009 Workshops at the Grid and Pervasive Computing Conference*, 64–71. doi:10.1109/GPC.2009.20

- Watt, S. (2007). Mashups -- The evolution of the SOA, Part 2: Situational applications and the mashup ecosystem. *IBM Developer Works*. Retrieved from <http://www.ibm.com/developerworks/webservices/library/ws-soa-mashups2/>
- Wu, L., Barash, G., & Bartolini, C. (2007). A Service-oriented Architecture for Business Intelligence for assessing IT service management per-.
- Xie, L., Xu, L., & de Vrieze, P. (2010). Process modelling in process-oriented enterprise mashups. *2010 2nd IEEE International Conference on Information Management and Engineering*, 650–654. doi:10.1109/ICIME.2010.5477893
- Yakovlev, I. V. (2007). Web 2 . 0 : Is It Evolutionary or Revolutionary ?, (December), 43–45.
- Yee, R. (2008). *Pro Web 2.0 Mashups - Rmixing Data and Web Services*.
- Young, G. O., Gualtieri, M., Daley, D., Shey, H., & Ashour, M. (2008). *The Mashup Opportunity – A Social Computing Report*.
- Yu, J., Benatallah, B., Casati, F., & Daniel, F. (2008). Understanding Mashup Development. *IEEE Internet Computing*, 12(5), 44–52. doi:10.1109/MIC.2008.114
- Yu, S., & Woodard, C. J. (2009). Innovation in the Programmable Web : Characterizing the Mashup Ecosystem, 136–147.
- Yu, T., Chen, Q., Li, Q., Liu, R., Wang, W., & Liu, W. (2009). A System for Web-Based Interactive Real-Time Data Visualization and Analysis. In *2009 IEEE Conference on Commerce and Enterprise Computing* (pp. 453–459). IEEE. Retrieved from [http://ieeexplore.ieee.org/xpl/freeabs\\_all.jsp?arnumber=5210759](http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5210759)
- Zhao, Z., Bhattarai, S., Liu, J., & Crespi, N. (2011). Mashup Services to Daily Activities – End-user Perspective in Designing a Consumer Mashups, 5–7.
- Zou, J., & Pavlovski, C. J. (2007). Towards accountable enterprise mashup services. *IEEE International Conference on e-Business Engineering (ICEBE'07)*, 205–212. doi:10.1109/ICEBE.2007.12
- Zur Muehlen, M., Nickerson, J. V., & Swenson, K. D. (2005). Developing web services choreography standards—the case of REST vs. SOAP. *Decision Support Systems*, 40(1), 9–29. doi:10.1016/j.dss.2004.04.008