A Framework for Constructing End User Oriented Service Mashups

A thesis submitted to The University of Manchester for the degree of Doctor of Philosophy in the Faculty of Humanities

2016

Sumaira Minhas
Manchester Business School
# Contents

Glossary ............................................................................................................................................ 11  
Abstract ............................................................................................................................................ 13  
Declaration ..................................................................................................................................... 14  
Copyright ......................................................................................................................................... 15  
Dedication ....................................................................................................................................... 15  
Acknowledgements ...................................................................................................................... 16  
Chapter 1.......................................................................................................................................... 17  
1. Introduction ................................................................................................................................... 17  
   1.1. Motivation ........................................................................................................................... 17  
   1.2. Problem statement, Challenges and Research Questions ......................................... 21  
      1.2.1 Research Challenges ....................................................................................................... 21  
      1.2.2 Research Questions......................................................................................................... 23  
   1.3. Research Methodology ........................................................................................................ 24  
   1.4. Research Contributions ........................................................................................................ 27  
      1.4.1 A Model for the Classification of mashup tools ................................................... 27  
      1.4.2 Goal-Oriented Mashup Development (GO-MaDe) Framework ..................... 27  
      1.4.3 Empirical Validation ....................................................................................................... 29  
   1.5. Thesis Outline ............................................................................................................................. 29  
Chapter 2 .......................................................................................................................................... 32  
2. Background and Literature Review ..................................................................................... 32  
   2.1. End-User Development ....................................................................................................... 32  
      2.1.1 Defining End Users........................................................................................................... 33  
      2.1.2 Goal of End User Development ..................................................................................... 34  
      2.1.3 Motivating Factors of End User Development ....................................................... 36  
   2.2. Web Services and Services Oriented Computing (SOC) ........................................... 37
2.3. Service-Oriented Applications, Web 2.0 and Semantic Web ................................. 39

2.4. Mashup Based Application Development-A new Trend in EUD Driven by Services Oriented Computing ................................................................. 40

2.4.1 Tool-assisted Mashup Development: Surveys and Evaluations ....................... 40

2.4.2 Classifying End-User Mashup Development Issues ....................................... 42

2.5. Goal Oriented Mashup Development ................................................................. 44

2.5.1 Suitability of Goal Based Methods for End-User Mashup Development... 44

2.5.2 Goal-Based Methods in Services Oriented Computing .................................. 45

2.5.3 Existing Goal Based Methods ........................................................................ 47

2.5.4 Recommendation for Selecting a Goal-Based Method ................................ 48

2.6. Meta Design ....................................................................................................... 54

2.7. The Domain Theory ............................................................................................ 56

2.8. WSMO Goals ....................................................................................................... 57

2.9. Discussion ............................................................................................................ 59

Chapter 3 ...................................................................................................................... 61

3. A Classification Model for Mashup tools ............................................................... 61

3.1. Motivation ............................................................................................................ 61

3.2. Characterizing Mashup Development Support ..................................................... 63

3.2.1 Range of Support for Mashup Development ...................................................... 64

3.2.2 General Classification ....................................................................................... 66

3.3. A 3-Dimensional Model for Mashup Platforms .................................................... 68

3.3.1 First Dimension: Mashup Design Features ....................................................... 69

3.3.2 Second Dimension: Usability Features ............................................................ 71

3.3.3 Third Dimension: Technical Features ............................................................. 72

3.4. Analysis and Discussion ....................................................................................... 76

3.4.1 Selection of the Tools ....................................................................................... 76

3.4.2 General Classification: ...................................................................................... 76

3.4.3 Dimensional Analysis ....................................................................................... 77

3.5. Discussion ............................................................................................................ 79
Chapter 4.................................................................81
4. The Goal Oriented Mashup Development (GO-MaDe) Framework............81
   4.1. Redefining the Mashup Development Lifecycle..............................81
   4.2. Why a New Framework?.................................................................83
   4.3. Components of GO-MaDe Mashup Development Framework...........87
      4.3.1 Mashup development Lifecycle and process model...............87
      4.3.2 Knowledge Acquisition and Representation in Mashups (KAReM): A Domain Theory Based Goal Modelling Method.................................89
      4.3.3 Domain ontology.......................................................................89
      4.3.4 WSMO Goals...........................................................................90
   4.4. The Spiral Mashup Development Process Model.............................90
      4.4.1. Redefining the Mashup Process: A Spiral Mashup Development Process Model91
      4.4.2. Detailed Activity Diagram.......................................................94
      4.4.3. Phases of the Spiral Mashup Development Process Model.........99
   4.5. Discussion....................................................................................104

Chapter 5..................................................................................106
5. Knowledge Acquisition and Representation in Mashups (KAReM): A Domain Theory Based Goal Modelling Method........................................106
   5.1 Overview..........................................................................................106
   5.2 Theoretical Foundations of KAReM..................................................108
   5.3 Conceptual Model of KAReM.............................................................109
   5.4 Meta, and Domain Levels.................................................................111
   5.5 Goal Model Overview.....................................................................112
   5.6 Goal Types......................................................................................112
      5.6.1 Goal Structure..........................................................................113
      5.6.2 Goal Templates and Goal Instances.........................................114
   5.7 Syntactic Goal Templates.................................................................115
      5.7.1 Mashup Type Goal Templates..................................................115
      5.7.2 Presentation Mashup with Plain Data Selection.........................116
A Framework for Improving End-User Orientation of Service Mashups

5.7.3 Goal Template for Presentation Mashup with Data Manipulation/Projection .................................................. 117

5.8 Generic Goal Templates ................................................................................................................................. 119

5.8.1 Generic Tasks as Domain Independent Goal Graph .............................................................................. 119

5.9 Meta-Domain Goal Templates based on Generalized Tasks ....................................................... 121

5.9.1 Personalization/Multimedia Mashups based on Modelling Generalized Task 123

5.9.2 Travelling Mashups based on Navigation Generalized Task .............................................. 124

5.9.3 Meeting/Planning Mashups based on Planning/Scheduling Generalized Task 125

5.9.4 Information Mashups based on Information Extraction Generalized Task 126

5.10 KAReM Tool Architecture ......................................................................................................................... 127

5.10.1 System Components .............................................................................................................................. 129

5.10.2 Mashup Profiler .................................................................................................................................... 130

5.10.3 Mashup Template Manager .................................................................................................................. 131

5.10.4 Goal Graph Builder ............................................................................................................................... 132

5.10.5 Goal Specification Generator .............................................................................................................. 135

5.11 Case Studies: .............................................................................................................................................. 136

5.11.1 Data Mashup Example Scenario ...................................................................................................... 136

5.12 Discussion ................................................................................................................................................... 144

Chapter 6 ............................................................................................................................................................... 145

6. Empirical Evaluation ....................................................................................................................................... 145

6.1 Research Methodology ................................................................................................................................. 145

6.2 Experiment Design ....................................................................................................................................... 147

6.3 Evaluation Criteria ......................................................................................................................................... 150

6.3.1 Completion Rates .................................................................................................................................. 150

6.3.2 Usability Problems (UI Problems) encountered (with or without severity ratings) 151

6.3.3 Task Time .............................................................................................................................................. 151

6.3.4 Task Level Satisfaction .......................................................................................................................... 151
List of Tables

Table 2.1 - Web Services Motivational Factors ................................................................. 38
Table 2.2 - End User Mashup Development Challenges ................................................... 43
Table 2.3 - Comparison of I*, Tropos and KAOS ............................................................. 48
Table 3.1 - Classification of tools w.r.t. level of mashup support .................................... 65
Table 3.2 - General Classification of Selected Mashup Platforms .................................... 66
Table 3.3 - The Evaluation Based on the Classification Model ........................................ 74
Table 3.4 - Legends for Classification Table and Mashing Techniques Table ............... 75
Table 3.5 - Analysis of Mashing Techniques ................................................................. 75
Table 4.1 - The Conception Phase ................................................................................. 103
Table 4.2 - The Translation Phase ................................................................................. 104
Table 5.1 - KAReM supporting different Levels of End Users and Complexity of Goal Templates ......................................................................................................................... 107
Table 5.2 - A Summary of Goal Derivation Strategies ..................................................... 134
Table 5.3 - Setting the Objective .................................................................................... 137
Table 5.4 - Parameters List at Step 1 ............................................................................. 137
Table 5.5 - Goal Strategies at Step 2 ............................................................................. 138
Table 5.6 - Goal Strategies at Step 3 ............................................................................. 139
Table 5.7 - Goal Strategies at Step 4 ............................................................................. 140
Table 6.1 - Experiment Setup ....................................................................................... 148
Table 6.2 - Snapshot of Participants and Task Data ....................................................... 152
Table 6.3 - Feedback Form 2 - Mean to Calculate Knowledge Clarity ............................. 158
Table 6.4 - Mean Value for Knowledge Helpful .............................................................. 160
Table 6.5 - Mean Value for Knowledge Comprehension - Feedback Form 2 ............... 160
Table 6.6 – Mashup Perception, Task Level and Test Level Satisfaction - Feedback Form 2 163
Table 6.7 - Requirments and Goals Findings .................................................................. 165
List of Figures

Figure 1.1 - Research Methodology ................................................................. 24
Figure 1.2 – Workflow of the Research Process (steps 1-3) ............................ 26
Figure 2.1 - EUD and its Relationship with Related Frameworks .................. 33
Figure 2.2 – Goal of End-User Development (Fischer et al. 2004) ................. 35
Figure 2.3 - Motivating Factors for EUD ....................................................... 36
Figure 2.4 - Level of Support provided by KAOS, I* and TROPOS in Traditional Software Development ................................................................. 49
Figure 2.5 - Prominent Role of Service Requirements in SOC .................. 51
Figure 3.1 - Growth in Web APIs since 2005 (Programmable Web Research Centre, 2013) 63
Figure 3.2 - Classification of Support Available for Mashup Development .......... 64
Figure 3.3 - The 3D Model for Evaluating Mashup Tools and Frameworks ....... 69
Figure 4.1 – The Goal Oriented Mashup Development (GO-MaDe) Framework Components ........................................................................................................ 81
Figure 4.2 - Mashup Development Scenario Using the Proposed GO-MaDe Approach ...... 85
Figure 4.3 – Redefined Mashup Development Lifecycle in the proposed GO-MaDe framework ........................................................................................................ 88
Figure 4.4 – The Activity Model of the Proposed Spiral Process .................. 93
Figure 4.5 - A visualization of the conception phase ...................................... 95
Figure 4.6 – Spiral Mashup Development Process ........................................ 97
Figure 4.7 - (a) Proposed Vs. Existing Mashup Development Lifecycle (b) Iterative Process ........................................................................................................ 102
Figure 5.1 – Schema of Proposed method .................................................. 110
Figure 5.2 – Conceptual Model- Meta and Domain Levels ............................ 111
Figure 5.3 – Goal Meta Model ....................................................................... 112
Figure 5.4 – Goal Types Defined in the Method ........................................... 113
A Framework for Improving End-User Orientation of Service Mashups

Figure 6.8 - Means and Standard Deviations for Knowledge-Helpful Dimension .......... 160
Figure 6.9 - Means and Standard Deviations for Knowledge-Comprehension Dimension. 161
Figure 6.10 - Mean of Knowledge Clarity, Usefulness and Comprehensiveness Respectively 
............................................................................................................................................. 162
Figure 6.11 - Mashup Perception Task Level, and Test Level Satisfaction - Feedback Form 2 
............................................................................................................................................................ 163
## Glossary

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOM</td>
<td>Document Object Model</td>
</tr>
<tr>
<td>EU</td>
<td>End User</td>
</tr>
<tr>
<td>EUD</td>
<td>End User Development</td>
</tr>
<tr>
<td>EUP</td>
<td>End User Programming</td>
</tr>
<tr>
<td>EUSE</td>
<td>End User Software Engineering</td>
</tr>
<tr>
<td>GO</td>
<td>Goal Oriented</td>
</tr>
<tr>
<td>GO-MaDe</td>
<td>Goal Oriented Mashup Development</td>
</tr>
<tr>
<td>GORE</td>
<td>Goal Oriented Requirement Engineering</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>IS</td>
<td>Information Systems</td>
</tr>
<tr>
<td>ISO</td>
<td>International Standards Organization</td>
</tr>
<tr>
<td>I*</td>
<td>I-STAR</td>
</tr>
<tr>
<td>IT</td>
<td>Information Technology</td>
</tr>
<tr>
<td>KAOS</td>
<td>Knowledge Acquisition in AutOmated Specification</td>
</tr>
<tr>
<td>KAREM</td>
<td>Knowledge Acquisition and REpresentation in Mashups</td>
</tr>
<tr>
<td>KR</td>
<td>Knowledge Representation</td>
</tr>
<tr>
<td>MAST</td>
<td>Mashup Activity Support Tool</td>
</tr>
<tr>
<td>OWL</td>
<td>Web Ontology Language</td>
</tr>
<tr>
<td>P2P</td>
<td>Peer to Peer</td>
</tr>
<tr>
<td>PBD</td>
<td>Programming By Demonstration</td>
</tr>
<tr>
<td>PBE</td>
<td>Programming By Example</td>
</tr>
<tr>
<td>RE</td>
<td>Requirements engineering</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>REST</td>
<td>Representational State Transfer</td>
</tr>
<tr>
<td>RQ</td>
<td>Research Question</td>
</tr>
<tr>
<td>RSS</td>
<td>Real Simple Syndication</td>
</tr>
<tr>
<td>SDLC</td>
<td>Software Development Life Cycle</td>
</tr>
<tr>
<td>SO</td>
<td>Service Oriented</td>
</tr>
<tr>
<td>SOC</td>
<td>Service Oriented Computing</td>
</tr>
<tr>
<td>SOA</td>
<td>Services Oriented Architecture</td>
</tr>
<tr>
<td>SOAP</td>
<td>Simple Object Access Protocol</td>
</tr>
<tr>
<td>SORE</td>
<td>Service Oriented Requirement Engine</td>
</tr>
<tr>
<td>URI</td>
<td>Universal Resource Identifier</td>
</tr>
<tr>
<td>WS</td>
<td>Web Services</td>
</tr>
<tr>
<td>WSDL</td>
<td>Web Services Description Language</td>
</tr>
<tr>
<td>WSMO</td>
<td>Web Services Modelling Ontology</td>
</tr>
<tr>
<td>XML</td>
<td>eXtensible Markup Language</td>
</tr>
</tbody>
</table>
Abstract

Owing to a recent online trend of web 2.0, societies have emerged not only culturally but also technologically/virtually. This coupled with semantic web and Web Services have provided end-users with more opportunities to contribute to the web and consequentially have also multiplied their digital needs. One such requirement of a modern-day end-user is to combine the data, view and/or process presented across the web to suit his/her ephemeral needs by developing a novel application known as a service mashup. The end user development of mashup poses significant challenges to the end users. First challenge addressed in this thesis is the scope vs complexity challenge which refers to the impossibility of fully eliminating the technical barrier between a tool and the end user due to the corresponding complexity creeping in while developing new features in tool and extending its scope to provide added functionality. The second challenge arises from the utility=value/effort equation which implies that effort required in developing a mashup decreases the value and hence neutralizes the utility. Given these challenges and the related issues, I made three contributions. The central theme of my proposed approach - for managing these inherent challenges – is that the end user must be integrated into the process of a service composition application. My approach – A Goal-Oriented Mashup Development (GO-MaDe) framework defines, organizes and addresses the problems faced by end-users while composing their applications by proposing a new style of development in the mashup area in a bid to render irrelevant the underlying inherent tensions of the paradigm. My first contribution in this regard is a classification model that takes into account end user centered usability criteria for calibrating mashup tools efforts. Secondly, this research is the pioneer study about the integration of an agile-style analysis-cum-design phase into the mashup development process. Hence, it presents a redefined process of service-based applications development by introducing a spiral process model that introduces a phased, incremental concept of mashup development lifecycle. Based on the spiral model, I have presented a method (KAReM) annotated with domain theory for better acquisition and representation of user requirements encompassed by the APIs or Web Services that are composed to develop a mashup by incorporating the dynamics of goal decompositions using goal based templates. It serves to enhance the end-user experience by facilitating them to explore the problem space and helping them derive the visual representations of their requirements which can then be translated into service compositions. To evaluate my contributions qualitatively and quantitatively I conducted literature reviews, and validated my framework by scenarios and a controlled experiment.
Declaration

No portion of the work referred to in the thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.
Copyright

The author of this thesis (including any appendices and/or schedules to this thesis) owns certain copyright or related rights in it (the “Copyright”) and s/he has given The University of Manchester certain rights to use such Copyright, including for administrative purposes.

Copies of this thesis, either in full or in extracts and whether in hard or electronic copy, may be made only in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has from time to time. This page must form part of any such copies made.

The ownership of certain Copyright, patents, designs, trademarks and other intellectual property (the “Intellectual Property”) and any reproductions of copyright works in the thesis, for example graphs and tables (“Reproductions”), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.

Further information on the conditions under which disclosure, publication and commercialisation of this thesis, the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University IP Policy (see http://documents.manchester.ac.uk/DoculInfo.aspx?DocID=487), in any relevant Thesis restriction declarations deposited in the University Library, The University Library’s regulations (see http://www.manchester.ac.uk/library/aboutus/regulations) and in The University’s policy on Presentation of Theses.

Dedication

To My Mom, Dad (May their souls rest in peace), my mom like sister, and my significant other Dr. Farhan.............
Acknowledgements

All Praise to Allah Almighty and countless salutations to His beloved and last Prophet Muhammad (ﷺ). As per my experience and observations, Ph.D. is a journey of losing and then finding one’s self again; a big thank you to all those who supported me during this journey. First of all I would like to express my utmost gratitude to my supervisors Dr. Nikolay Mehandjiev and Dr. Pedro Sampaio for all their invaluable support, priceless guidance and above all patience while I was still trying to learn to swim in the this vast sea of research and knowledge. I would like to acknowledge them for being the kind of professors who provide moral support and not just the academic one.

I would also like to thank Prof. Ilias Petrounias for serving on my research committee and providing valuable feedback on my work throughout the course of my doctoral studies and Prof. Trevor Wood Harper for teaching a couple of foundational research courses that proved to be very useful during my research. I want to show my appreciation to the wonderful community at MBS including Prof. Julie Froud, Prof. Stuart Hyde, Prof. Bob Scapens. And a very special thanks to PGR team on the 9TH floor especially Mrs. Lynne Barlow Cheetham, Ms. Madonna Fyne, Mrs Claire Smith and Mr. Paul Greenham. One of the most valuable parts of my doctoral experience was the opportunity to collaborate with highly esteemed researchers during different research events. I humbly thank all the people whom I met during the events and whose ideas helped shape my research especially Ms. Jennifer Bevan, Prof. Muninder P. Singh during SCC 2012 and Professor Patrick Hung for nominating me the chair of a very important and helpful session on services computing in the same conference.

They say behind every successful man, there is a woman and I always say that behind every successful woman, there are lots of men. I am infinitely grateful to my brothers Nayyar Sultan Minhas, Anber Sultan Minhas and Badar Sultan Minhas for all their infinite support in every way possible.

Finally, I would like to acknowledge my sponsor, my employer Fatima Jinnah Women University, Pakistan and Higher Education Commission, Pakistan. I am thankful for their funding without which this thesis would not have become a reality. I also thank the faculty members at Fatima Jinnah Women University especially Dr. Sikandar Hayat and my professor Dr. Naveed Ikram. A very special thanks to all my family and friends!!!
Chapter 1

1. Introduction

1.1. Motivation

The recent increasing trend of domain experts taking up programming tasks seems synonymous with both Boehm’s estimated number of “end user programming performers” (Boehm et al. 1995) and the estimates provided by Scaffidi (Scaffidi et al. 2005) based on another U.S. bureau of Labour Statistics’ survey conducted earlier in 1989. Both these facts point to the popularity and pervasiveness of the idea of “End User Development” (EUD). End-User Development (EUD) refers to the empowerment of users of software systems to update, extend and create a software artefact to satisfy their frequently varying needs (Mehandjiev et al. 2006). Different factors have been contributing to the popularity of EUD. Firstly, since it becomes hard for the professional software developers to provide the assistance partly due to their inadequate domain knowledge and partly due to time-consuming standard operating procedures (SOPs) of the Information Technology (IT) department, EUD addresses the situational needs of end-users (EU) by providing the required set of tools, techniques and methods. Another reason, why the end-users need to create applications for addressing their own needs is the Pareto ’80-20’ Principle or the Long Tail Phenomenon, which suggests that 80% of the efforts of developers will benefit only 20% of the end-users (Ogrinz 2009). There are various examples of end-user development in the daily lives ranging from basic customizations to more intricate scripting (Sutcliffe and Mehandjie 2004). For instance, students writing macros in word processing software to better accomplish their styling objectives, or a teacher writing spreadsheet formulas to grade students.

Software tools are developed for a target class of end-users and services based applications (SBAs) should follow this guideline (Namoun, Wajid, et al. 2010a; Namoun, Wajid, et al. 2010b). In order to specify the target end-user class for this particular thesis, a brief discussion is presented in the beginning for the purpose of clarity and to set the stage for further discussion. Different studies consider different factors to define the term end
user ranging from basic factors such as knowledge about the domain (or application area) and knowledge about the system/tool (Beringer 2004) to consideration of various dimensions such as programming knowledge and experience, domain knowledge, intent of programming, interaction technique and complexity of programming language (Ko et al. 2011). For the purpose of this thesis, which is situated in the context of EU service compositions, I have defined end users as professionals who have little or no computing/programming skills (also sometimes regarded as non-programmers) and who develop programs due to personal and ephemeral reasons (Mehandjiev et al. 2008; Ko et al. 2011; Segal 2007). I have also endorsed and followed the distinction between user and end user (Ardito et al. 2012); as users are all people interacting with the software including the developers themselves and hence have tried my best to use the term EU instead of users. For defining end users I have slightly extended the definition provided by Costabile (Costabile et al. 2007). “End users can be defined as people who are not experts in computer science or related fields and neither are willing to be; rather they use computer systems for their daily activities including work, entertainment and other purposes”. Moreover, this research specifically targets consumer EU rather than enterprise EU. There are other definitions that consider factors such as development for personal or public use but we consider this discussion irrelevant in the wake of current landscape as boundary between personal and public use is becoming increasingly blurred in the wake of personal artefacts being made available through shared platforms (Cabitza et al. 2014).

Before further details are discussed, it is important to define, understand and distinguish the concepts of web services, service mashups and web services compositions. According to w3schools' definition, web services have these characteristics: Web services are self-contained and self-describing application components based on XML and HTTP that communicate using open protocols and which can be discovered using UDDI and can be used by other applications. This definition clearly indicates that web services are suitable for machine to machine communication and is used in the context of SOC/SOA. Similarly the term “web service compositions” is used to define efforts that are directed towards the integration of individual services to meet the needs of complex applications. “Web service compositions aims to provide effective and efficient means for creating, running, adapting and maintaining services that rely on other services in some way” (Benatallah et al. 2005).

1 http://www.w3schools.com/xml/xml_services.asp
Again, this definition is given in the context of Services Oriented Architecture and does not incorporate the concepts of ephemeral applications and/or end users. Although different authors have started to use the phrase ‘service compositions for end users’ to discuss mashups recently, service compositions generally refer to the research area that seeks to work towards the development of new aggregate/composite services to integrate heterogeneous enterprise applications and business-to-business collaborations. Throughout the thesis, we will use the term web services to refer to the application components available on the web and service composition to refer to the aggregation of services.

In the wake of the unprecedented growth of internet and enabling technologies, the idea of end-user development (EUD) has given rise to a new breed of applications: web/service mashups. They both tend to provide a solution to end users; however web services From the broadcast model of web (referred to as web 1.0) to a more participatory form of web characterized by blogs, wikis and folksonomies known as web 2.0, the world has seen colossal growth in the online data and henceforth the opportunities to reuse this data. Mashups provide this opportunity and allow end-users to repurpose the existing data and/or logic to solve their problems which have not been addressed before as well as to create new exciting opportunities. To support mashup development, different tools, techniques and methods have been developed (Caruccio et al. 2016; Mehandjiev et al. 2010). These tools aim to serve the needs of both: general end-users and enterprise end-users thus leading to different types of mashups i.e. consumer and enterprise mashups respectively (Ogrinz 2009). Consumer mashups are considered to be less complex and EUs, while developing consumer mashups target APIs and feeds that are public and well-defined. On the other hand, enterprise mashups are complex and the organizational needs/dynamics have to be considered within the context of enterprise, its tools and its needs (Ogrinz 2009). Considering this demarcation, it can be argued that this thesis is oriented towards the area of consumer mashups since no organizational/enterprise factors are considered.

Mashup development is still considered in its infancy in spite of plethora of Web Services, APIs, support tools and growing interest of end-users towards service compositions that the mashups offer(Daniel and Matera 2014; Lizcano et al. 2008; Daniel 2015; Aghaee and Pautasso 2013). One of the main challenges facing mashup development is the technical barrier of mashup platforms that most of the users do not seem to overcome due to their
non-IT background (Lizcano et al. 2016; Huang et al. 2015; Rodríguez et al. 2014; Namoun, Nestler, et al. 2010b; Namoun, Wajid, et al. 2010b; Namoun, Nestler, et al. 2010b). These technical needs of the mashup platforms comprise the programming needs, use of computer-based concepts and terminologies, lack of visual models, lack of understanding of end-users’ mental models and lack of proper, up-to-date documentation.

This thesis focuses on tool-assisted end-user mashup development aimed primarily at consumers instead of enterprises. The research challenges addressed in this thesis concern the lack of user-orientation during the mashup development and this thesis is an effort to move towards user-centered mashup applications development also referred sometimes in literature as user-centric or user-driven mashup development (Cappiello et al. 2015; Nestler 2008). This style of mashup development is inspired from “user-centered design” where the latter is defined as “a multidisciplinary design approach based on the active involvement of users for a clear understanding of user and task requirements, and the iteration of design and evaluation” (Mao et al. 2001). In mashup development, the term user-centered mashup development is used in contrast to provider-driven mashup development (Cappiello et al. 2015). However, in spite of being used often in mashup literature has not been explicitly defined ((Mroß and Meißner 2014). We define user-centered mashup development as a style of mashup development which is driven by end users and in which end-users are integrated in the design and development of new composite services in such a way that they are easily able to understand features of available services, their integration and are able to compose them without the need of programming knowledge.

The primary motivation of this thesis is to understand the challenges faced by end-users during the development of mashups to improve the end user’s quality of experience (QoE). Different user studies (Namoun et al. 2010b) have been conducted that have tried to investigate and report the issues related to the process. This thesis aims to consider these end user problems while designing my approach. Currently, major efforts in this field have been concentrated in improving the capabilities of the tools, and improving their technical features subsequently ignoring the end-users perspective (Paternò 2013). My intention in this thesis is to devise a method to redefine and augment the existing process of EUD of mashups to improve the quality of the entire EUD activity majorly in terms of quality of experience and motivation. However, one main challenge in this reorganization is to make sure that the inclusion of the concern for end user expectations while developing mashups
A Framework for Improving End-User Orientation of Service Mashups

is least detrimental to the nature of the work or their priorities. The goal is to consider the
differences and priorities of mashup development as compared to that of traditional,
professional software development. Although academically speaking, measuring
motivation and user-experience can be a multi-dimensional and multi-criteria task and
hence cumbersome, but for the purpose of my research I aim to consider few basic factors.
These factors mainly include time to develop mashup, accuracy of developed mashup with
respect to original intent of EU and EU’s personal involvement which are planned to be
collected and analysed during a controlled experiment through a carefully drafted criteria
as part of validation of this research. This will be dealt in detail in later chapters.

1.2. Problem statement, Challenges and Research Questions

The problem statement that I intend to address in this thesis is stated as follows:

While developing situational applications by repurposing the existing data and
logic using mashup platforms, end-users face the dilemma of balancing
between the technical requirements of the mashup tools and their non-IT
backgrounds. Thus, to make the mashup development task more user-oriented,
and mashup tools more efficient and user-friendly (i) usability issues must be
considered and addressed in such a way that they encourage the end-users into
programming for their needs by using and repurposing by
composing/integrating existing sources available in the form of Web Services,
APIs and data feeds rather than putting them off due to the technical
difficulties, (ii) the resulting methodology must be end-user friendly and should
not compromise the quick cycle of development of mashups or shift the focus of
end-users from their intended goals to understanding the complexities of tools
being used (iii) the end-users must be assisted in specifying the goal of their
development tasks to help them achieve their goal without over-emphasizing
on the requirements or design specification activity.

1.2.1 Research Challenges

The main research challenges addressed in this thesis correspond to the dilemma
associated inherently with the EUD’s framework in general and related to the end-users
who want to develop ephemeral applications to address their needs. This dilemma is
explicitly specified in the research literature where end-users having limited technical
knowledge are faced with the complexities of composition of the service composition tools
A Framework for Improving End-User Orientation of Service Mashups

(Huang et al. 2015). For the purpose of ease of understanding, the challenges have been classified into EUD Challenges and EUSE/Meta Design Challenges:

EUD Challenge

CHAL.1. Given the inherent tension of end-user’s non IT background and technical demands of EUD tools, it is impossible to completely eliminate the technical aspects of tools and the corresponding learning difficulties. Therefore, a strategy is sought that motivates users and minimizes the technical burden by using visual support and models.

EUSE/Meta Design Challenge

CHAL.2. Given the quick development cycle of creation of a service based application in ideally a Lego-like style; the emphasis of end-users must be on the goal rather than on the supporting software engineering activities such as requirements or design specifications. To improve the Mashup development activity by incorporating the consideration for end-users’ specifications inevitably tends to affect the agile nature of the development process in this regard and move the end-user, who happens to be a domain expert on his field, away from his main goals. I need to make sure that the overall equation of EUD utility does not create any misbalance in terms of value and effort (Fischer and Giaccardi 2006):

\[ \text{Utility} = \frac{\text{Value}}{\text{Effort}} \]

Where the utility is the net gain or usefulness, value is the benefit and effort is the struggle put into the activity in terms of learning time and task time (Fischer and Giaccardi 2006; Grudin 1992). This challenge seems to be addressed well by incorporating the co-creation and co-designing approaches of meta-design. Hence, instead of taking the predefined and hard-coded rates of utility, value and effort, the “perceived” notions of value and effort are intended to define the utility of personally meaningful tasks.
1.2.2 Research Questions

RQ1: A- What are the issues currently faced by the end users during the development of applications by composing services and repurposing the existing data and process logic?

B- What is the state of the art in the tool assisted mashups and what features are lacking in the existing mashup development support?

C- What mashing techniques have been and are being used by the tools that assist the users in creating their service-based applications?

RQ2: How to integrate the end-user in the development process by addressing the end-user concerns and how to assist them in understanding their own service-based application requirements which can later on help them translate these requirements into models ready to be composed into a mashup-style application?

RQ3: How can I redefine the mashup development process to address the main EUD challenge of inherent tension between non-IT professional end-user’s capabilities and technical demands of the development process of services based applications?

RQ4: How can I manage domain requirements and can I apply domain theory and goal-based methods in achieving the required domain-abstraction for a new improved method?

RQ5: How can I measure the “improved end user orientation” of the framework proposed in this thesis?
1.3. Research Methodology

The methodology used for this research is Design Science Research (Hevner and Chatterjee 2010).

![Research Methodology Diagram]

1-A model for classification of mashup tools
2- Re-defined Mashup Development Process
3- Goal-Based Method for Acquisition of EU requirements
4- Evaluation and proposal for Improvement in the proposed approach based on the results

Figure 1.1 - Research Methodology
Since mashup development is related to an engineering or artificial branch of science rather than pertaining to a natural field of science, a considerably suitable research method is design research (Simon 1996). The benefit of design research is that this approach not only guides the development of an artefact in the field of Information Systems (IS) but also provides the guidelines that help to distinguish the activity of routine design from design research (Hevner et al. 2004). It also allows using existing knowledge base from the relevant research areas and disciplines to propose a new artefact in order to solve the original problem which may be in the form of a theory, framework, method or a new piece of software. Hence, it helps contextualize the problem by anchoring it in the existing environment (fig. 1.1) and provides the flexibility to choose the most suitable evaluation / validation methods (qualitative and/or quantitative) to test the proposed artefact or the underlying theory. Moreover, design research is a suitable approach for supporting a feedback loop from theory to design to experiment and back (von Alan et al. 2004).

Within the design science research methodology, the research methods employed comprise the literature survey, scenario-based research and controlled experiment. The detailed workflow of the research process is explained in the following paragraphs (Fig 1.2).

The first step was “Explorative literature survey” and its purpose was to identify and analyse the relevant literature regarding the mashups and relevant topics to understand the state of the art, which is referred as “Environment” and “knowledge base” in figure 1.1. The literature survey was conducted to fulfil two main objectives (a) To help highlight and identify research gaps and issues and (b) to help formulate the solution. It comprised of several inter-related areas namely End-User Development (EUD), Mashup Development (MuD), Requirement Engineering (RE), Service-oriented requirement engineering (SORE), Goal-Oriented Requirement Engineering (GORE), Domain Theory (DT), Meta-Design (MD) and related themes, theories, frameworks and tools. The main purpose of literature survey was to address research question 1 by identifying the research gaps and issues to help guide the proposed solution. For instance, a classification model encompassing three relevant dimensions was proposed to understand and calibrate the support available for end-user mashups which directed me towards the improved research goal: the need of an
improved end-user orientation framework in mashup platforms. Similarly research in RE, SORE and GORE led me to use goals as the basis of the proposed framework due to their suitability to mashup platforms and representing EU needs.

Different conceptual foundations from the existing literature have been used in the design of proposed framework to address the research challenges: (1) principles of meta-design such as design-for-use, adaptable-design, (2) the libraries proposed by domain theory, (3) the concept of participatory design and goal models. One of the significant themes from comprehensive studies of the related work that fed into my approach is the importance of understanding end users and their skills (Nestler 2008). This drove me to investigate the existing mashup platforms against my first contribution: the user-oriented 3-dimensional classification model. Based on the results of the investigation, I concluded that existing mashup tools lack user-orientation in terms of user goals and requirements since majority of the tools provided no explicit support to end users goals and expectations. Based on these insights collected during literature survey, I developed my approach (step 2 of figure 1.2): the Goal Oriented Mashup Development framework (GO-MaDe) comprising of two elements, the spiral process model and the KAReM method (along with a proof of concept tool architecture for KAReM). Hence, I have proposed a solution to empower end users in

---

**Figure 1.2 – Workflow of the Research Process (steps 1-3)**
their mashup application development by providing them with a space to specify and manage their expectations and requirements.

Scenarios are applied and a controlled experiment is conducted in a university setting to validate the conceptual GO-MaDe framework. For the experiment, I hypothesized that for mashup development to be more user-oriented; an important factor is to integrate the end-user needs explicitly into the development activity. The quantitative analysis of the data obtained through the experiment not only supported my hypothesis but also provided with valuable insights that could serve to further improve my contributions.

1.4 Research Contributions

Given the challenges discussed in section 1.2.3, to improve the state of the art of the end-user mashup development, I made the following contributions.

1.4.1 A Model for the Classification of mashup tools

To assess the end user orientation of the given mashup tool, a conceptual three dimensional classification model is proposed. It can serve both as a foundation guideline for developing new mashup tools as well as for evaluating existing platforms. To know the status of the art of mashup development issues, I have conducted a study based on review of the existing literature. The findings served to guide my framework specifically and will generally serve as an important artefact for researchers working in this area. The model takes into consideration the dimensions that have not been considered before. User orientation has mostly been measured in terms of programming skills requirements in earlier evaluations (Paredes-Valverde et al. 2015).

1.4.2 Goal-Oriented Mashup Development (GO-MaDe) Framework

The major contribution of the thesis is the Goal-Oriented Mashup Development (GO-MaDe) framework which addresses the challenges of EUD by making use of goal models and meta-design principles.

A- Spiral Process Model

First, to improve end-user orientation I redefined the mashup development process by introducing the spiral model of mashup development to emphasize on the earlier phases of conception and translation of a mashup in the proposed Goal-Oriented Mashup Development (GO-MaDe) framework. While the traditional spiral model
(Boehm 1988) was proposed to address the shortcomings of rigid nature of existing models, the spiral model of mashup development serves to integrate end user in the mashup development process. The process model reorganizes the mashup development life cycle and introduces the phases of Conception, Translation, Composition and Execution instead of one monolithic concept of service composition that is currently synonymous with the mashup development. The framework makes use of the visual formalization of concept of goals and makes use of the goal graphs to assist users in deriving their detailed services without exposing them to un-necessary technical details.

B- **KAreM Method**

Secondly, it proposes a method: Knowledge Acquisition and Representation for Mashups (KAreM), which proposes to implement the first two phases of the proposed spiral model: Conception and Translation. It is partially inspired by the principles of meta-design (Fischer and Giaccardi 2006) and makes use of the foundational assumption that future uses and problems cannot be anticipated by the developers during design time which coincides with a typical mashup development scenario. In order to provide the end users with a platform to co-design, the framework works on the EUD guidelines as proposed by Lieberman et al. (Lieberman et al. 2006). The proposed mashup development method allows user to evolve their mashups naturally and provide them with different modification levels thus addressing the Power-of-expression and complexity tension.

a. The framework makes use of the domain theory and thus provides end users with a third level of support to discover and specify their requirements apart from generally used artificial intelligence and domain ontologies. KAreM makes use of the visual formalization of concept of goals and makes use of the goal graphs to assist users in deriving their detailed services by goal-based templates without exposing them to un-necessary technical details. GO-MaDe makes use of the goal templates based on libraries of generalized tasks and generic tasks of domain theory and hence allows the users to reuse the existing knowledge. For proposing goal templates at suitable level of abstraction it introduces and makes uses of the concepts of *mashup genre*, and uses mashup type and other information to determine the levels of expertise of the end user and the maturity of his need/mashup idea.
b. The framework is conceptualized by a template-based architecture that satisfies a key property of EUD-friendly architectures by allowing substantive changes during run-time. The architecture allows for both kind of changes, internally initiated state changes as well as the changes based on users’ explicit requests.

1.4.3 Empirical Validation

For the purpose of qualitative and quantitative evaluation of these contributions, comprehensive literature reviews were conducted followed by two-fold validation. For the purpose of internal validation of the method, an assessment of the proposed conceptual framework is done on a real case study based on a common mashup scenario. For the purpose of end-user validation of the proposed framework, a user study was conducted based on controlled experiment.

1.5. Thesis Outline

This section presents the chapters’ summaries for the remaining chapters:

Chapter 2 – Background and Literature Review:

This chapter discusses the background topics related to the thesis. The first topic is the EUD, its history, importance and impacts on the everyday life. Later, services oriented computing web 2.0 and the manifestation of the participatory aspects of web 2.0 such as blogs, wikis and social media are brought into discussion. Web 2.0 and its participatory and social influence have motivated users to act as content creators rather than passive content consumers. This leads to a myriad of opportunities for end users to make use of the data hence created. This chapter then discusses the third background topic, i.e. Services Oriented Architecture (SOA) and how SOA triggered the development of Services Based Applications (SBA). These two topics will then be discussed as the triggers of end-user service composition applications as the latest trend in the field of End-User Development. This chapter also presents a concept-oriented review of the existing literature on issues of mashups discussed in end-user studies conducted so far followed by other related theoretical topics such as metadesign and the domain theory. Finally, the findings of the analysis of the review are discussed in an objective style to organize the issues and challenges and propose a requirements analysis for representing as a baseline for the proposed Mashup development framework.
Chapter 3 – A 3 Dimensional Classification Model for Mashup Tools: This chapter concerns the thesis goal of understanding the problems with the existing service mashup tools and in a bid to address them proposes a 3 dimensional model that takes into consideration the dimensions that have been ignored in previous evaluation of mashup tools. The model is based on a 3-dimensional evaluation criteria that can provide classification for existing tools and at the same time serve as a guideline for new tool development in the area of service mashups.

Chapter 4 – GO-MaDe – The Goal Oriented Mashup Development Framework: This chapter provides a detailed system analysis and design for the proposed goal-oriented framework (GO-MaDe) and discusses in length about how users’ quality of experience (QoE) can be improved by different components of the new proposed framework. This chapter discusses each aspect of the framework in detail along with the motivations, salient features and examples. The first component which is a spiral model is also discussed in detail in this same chapter to provide the preface to the KAReM method discussed in detail in chapter 5.

Chapter 5 - Knowledge Acquisition and Representation in Mashups (KAReM): A Domain Theory Based Goal Modelling Method: This chapter discusses the details of the second component of GO-MaDe framework: the KAReM method (Knowledge Acquisition and Representation in Mashups abbreviated as KAReM) for representing and acquiring end users expectations. The complete working of the method is discussed in detail in this chapter with a walk-through of the method’s stages for demonstration purpose followed by tool architecture to provide a proof-of-concept for the implementation of the method. The discussion is followed by working examples of the GO-MaDe method by providing scenario-based validation to describe the dynamics of the method and how it helps to manage EU’s expectations and challenges mentioned in previous chapters.

Chapter 6 – Evaluation: This chapter discusses the design of the user-studies to evaluate the proposed method and validate the thesis. The user studies are designed to be conducted in the form of controlled experiments within the premises of University of Manchester. For the purpose of the study which was designed as a controlled experiment, users were recruited from University of Manchester (preferably students) who had little or no programming experience. Each study lasted
120 minutes in which users were required to build goal graphs using the goal-based templates for the given scenarios. The next section of the chapter presents the validation results of the thesis. The validation is carried out by means of qualitative and quantitative evaluation of data collected during the user studies. The evaluation results, apart from providing validation for different aspects of the framework, also provide some valuable improvements for the proposed approach.

Chapter 7 – Summary and Conclusions: This last chapter summarizes the research conducted during the course of this thesis and contributions made. It presents an analysis of the limitations and concludes with the future work.
Chapter 2

2. Background and Literature Review

2.1. End-User Development

End-Users are occasionally referenced as ‘naïve’ or ‘novice’ programmers (Blackwell 2006) and the set of methods, techniques and tools that allow these novice programmers or non-professional software developers to create, update or extend a software artefact is referred to as End User Development (EUD) (Lieberman et al. 2006). These non-professional software programmers develop software as a means and not as an end and use various techniques to program that are referred to as End User Programming techniques. These end user activities can be classified into Parameterization or Customization and Program Creation/Modification (Lieberman et al. 2006). In contrast to EUP, conventional Programming can be defined as a process of transforming a natural, mental plan of solving a problem into a style that is compatible with computers (Lewis and Olson, 1987; Pane et al. 2001). Different authors have tried to research on the difficulties posed by the conventional programming languages to EUs. According to Lewis and Olson, it is considered as a difficult activity and more so for end users due to a reported negligence of HCI issues especially due to problems related to low-level programming primitives (Lewis and Olson 1987). The low-level programming primitives (such as *Char* or *Const*) sound unfamiliar and unrelated to the tasks to be performed and un-relatable to the real world. Nardi on the other hand has argued in favour of domain and task-oriented languages (Nardi 1993). According to Lieberman et.al., an important factor contributing to the difficulty of the programming task can be attributed to the fact that most of the programming languages are designed by people who are themselves professional programmers (Lieberman et al. 2006). All of these arguments imply that it is difficult for the beginner programmers and almost impossible for non-programmers to make use of textual programming languages to solve their problems in spite of their willingness or situational demand. This motivated the researchers to build programming languages and environments to make programming more accessible to novice programmers of all ages (Kelleher and Pausch 2005).
The term EUD has been used synonymously with End user programming (EUP) and End user computing (EUC) and is closely related to the research area of End-User software Engineering (EUSE). Each of these research areas signifies a marginally different scope and coverage thus employing different approaches. The main commonality is that they focus on developing methods while keeping End-Users’ programming needs into account. The relationship can be seen in figure 2.1 which highlights some fundamental differences between these related paradigms (Fischer 2010; Cabitza et al. 2014). End User Software Engineering is an over-arching research area that transcends beyond programming to provide relevant support to entire software life cycle. End User programming is a relatively more focused research domain that intends to empower end users and provides them support to program by techniques such as programming by demonstration/example (PBD, PBE), visual programming (VP), domain specific languages, and scripting languages. Meta design is a solution framework based on design methodology of “design for designers” and defines different principles that facilitate this design methodology such as “design during use” and “design for change”.

2.1.1 Defining End Users

The literature presents different definitions of end users depending on the dimensions taken into consideration. Beringer defines different categories of EUs based on their proficiency across two dimensions: System knowledge (information/computer systems)
and domain knowledge (Beringer 2004). It is important to understand the difference between these two types of knowledge to understand the term ‘end user’ and its different definitions. I have used Zmud’s classification of knowledge to explain this difference (Zmud 1983) albeit by replacing the organizational context with the domain aspect since I am concerned about consumer end users (who are not working or creating mashups in organizational context) rather than enterprise end users. Zmud classified knowledge into six types with the first three referring to organizational/domain knowledge namely organizational skills, overview and target unit. While the last two are specific to the information system in hand namely technical skills, and the product knowledge (in this case knowledge related to the specific mashup tool at hand) this can be used to explain the term system knowledge. The 6th category defined by Zmud is general Information System (IS) knowledge, which is important for being able to use computer hardware and software. Hence, in summary System knowledge refers to the knowledge about the IT system or IS and domain knowledge refers to the knowledge such as terms, business rules, definitions and concepts about application area in question (such as Accounting, Finance). In another attempt of defining EU, another research takes into consideration multiple dimensions such as programming knowledge, programming experience, domain knowledge, intent of programming, interaction technique and complexity of programming language (Ko et al. 2011). While explaining the psychological aspect of end user development, another author (Blackwell 2006) raises a very genuine concern about research on end user tools as most of experiment participants are mainly university students who are chosen due to being non-programmers. This can lead to generalization problems as real end users are normally professionals who have completed their formal education and who can’t spare time to participate in the studies. This motivated me to define the end users for the purpose of my research with a more abstract and wider application. I have opted for a more inclusive definition for end users as professionals, domain experts or casual users who have little or no computing/programming skills (also sometimes regarded as non-programmers) and who develop programs due to personal and ephemeral reasons (Mehandjiev et al. 2008; Ko et al. 2011; Segal 2007) and are not concerned with the tool per se but in doing their work (Fisher and Giaccardi 2006).

2.1.2 Goal of End User Development

The primary objective of EUD is the empowerment of EU in the field of computing albeit different authors have given different perspectives on this goal.
Fisher and Giaccardi have defined this goal in terms of the EU development environments (Fischer and Giaccardi 2006). According to them, EUs face a dilemma of choosing from low-level domain specific environments to over-specialized application systems defined by the two extremes. One extreme is Turing Tar Pit in which everything is possible but nothing of interest is easy and the inverse of Turing Tar pit in which operations are easy but nothing of interest is possible (Fischer and Giaccardi 2006). Developing systems that are easy to develop and are situated somewhere in the middle of this continuum is the real goal of EUD.

Fischer et al. have presented the EUD aim in terms of trade-off between scope and cost of learning of a given environment (language and/or tool) (Fischer et al. 2004). The main aim of EUD, according to them, is to facilitate EUs by finding and defining the balance between expressiveness and usability of the platform/environment at hand by developing and/or employing suitable techniques, tools and methods (Sutcliffe and Mehandjiev 2004). This can be illustrated by the model presented by Fischer et al. in figure 2.2 (Fischer et al. 2004). Repenning and Ioannidou provide a more generic fundamental aim of EUD i.e.: to gain more control over their computers by engaging in a development process for the purpose of empowering users (Repenning and Ioannidou 2006).
2.1.3 Motivating Factors of End User Development

End Users’ needs cannot be anticipated. Consequentially, no commercial pre-packaged software can claim to meet all the end users’ needs. Stemming from this need for flexibility, there have been diverse motives for the end-users to develop, modify and program different artefacts. From the organizational needs of domain professionals to be able to add and modify programs to the increasing amount of embedded software in consumer products the case for the need for EUD has always been strong and the motivation for the EUD has existed mainly on the continuum of needs (personal/professional) and leisure/creative satisfaction (fig. 2.3) until 2013, when another dimension known as the social dimension emerged. Characterized by the rise of web 2.0 and social media, this third force has also contributed towards the promotion and adoption of EUD activities. For the sake of understanding, the EUD motivation has been classified into intrinsic and extrinsic factors and will be discussed briefly in the following paragraphs.

A- Intrinsic Factors

Intrinsic factors correspond to the needs arising within the individual or group’s personal or work environments and involve all those domain experts/professionals working in a work domain such as accountants, architects, doctors, engineers, biochemists, statisticians etc. These professionals need to master certain applications
such as AutoCAD2, spreadsheets, and other custom built applications relevant to their domains being used in their organization. For this group of EU, EUD is a need and not a luxury (Fischer 2010) due to the emergence of new requirements as well as the changes in the work practices of the professionals themselves. Basic examples of this type of EUD are recording/writing macros in word processors, setting up spreadsheets for complex calculations, defining email filters and customizing the reports/forms.

B- Extrinsic Factors

On the other hand extrinsic factors are the factors that do not directly relate to the work environment or personal needs of the end users (Fisher et al. 2004). This corresponds to the factors originating from outside the operating environment of an individual and/or group such as semantic/social or web 2.0. The backbone of these factors is the Internet in general and participatory web in specific (O’Reily 2006). It has given rise to the awareness of the notion, which was underlying the research field of EUD that users are not always passive software consumers; hence breaking down the boundaries between content creator/producer and content consumer/user. While intrinsic factors are driven by personal satisfaction/self-realization, economic or material reward/benefits, the main motivational forces behind extrinsic factors are social capital, reputation, connectedness, and the enjoyment derived from giving things of value away (Fisher et al. 2004).

2.2. Web Services and Services Oriented Computing (SOC)

Web services have been defined differently by different sources ranging from inclusive to restrictive statements. According to Microsoft ‘a Web Service is a piece of business logic accessible via the Internet using open standards’ (Singh and Huhns 2006). A More elaborate definition from W3C defines it as ‘a software application identified by a Universal Resource Identifier (URI), whose interfaces and binding are capable of being defined, described, and discovered by eXtensible Mark-up Language (XML) artefacts and supports direct interactions with other software applications using XML-based messages via Internet based protocols’ (Singh and Huhns 2005). Nonetheless, the idea behind Web Services takes its motivation from certain defined factors defined below (table 2.1) irrespective of

---

2 It is a commercial software application for 2D and 3D computer-aided-design.
the diversity in definitions. Similarly, the emergence of Web Services has also motivated research in different directions.

Table 2.1 - Web Services Motivational Factors

<table>
<thead>
<tr>
<th>Technical Motivations for Web Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>Middleware Placement</td>
</tr>
<tr>
<td>Organizational Autonomy and Confidentiality Issues</td>
</tr>
<tr>
<td>Pragmatic Web</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Financial Motivations for Web Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>For Providers/Vendors</td>
</tr>
<tr>
<td>For Consumers / Customers</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Development/ Process Motivations driven by Web Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service Oriented System Engineering –SOSE</td>
</tr>
</tbody>
</table>

These motivational factors can be divided into three main categories: Financial, Technical, and development/ Process based. Moreover, it’s going to be a worthwhile study to list down these factors for the sake of underlining the importance and understanding of this research. Although these factors are inter-twined to some extent, yet to facilitate the understanding these are listed individually in table 2.1. Table 2.1 lists down the main motivations behind conducting research in Web Services.
2.3. Service-Oriented Applications, Web 2.0 and Semantic Web

The current and continuing trend of using web browser as a uniform application development and deployment platform and not only as a traditional document viewing and distributing environment is altering the landscape of the software practice and research. In the new era of web based applications, application and their components are ubiquitously available on the web as services that can be located anywhere on the web (Anttonen et al. 2011) and have created new opportunities as well as challenges (Dustdar and Li 2014).

With the emergence of Services Oriented Computing (SOC), empowering end users to develop applications by composing individual services has further enhanced the prospects of supporting EUD. This new services innovation trend is further supported by Web 2.0, the Semantic web and associated concepts, protocols and technologies (S. Yu and Woodard 2009; Shuli Yu and Woodard 2009). Semantic web facilitates the machine readability of the web content through semantic technologies (OWL-S, WSMO, WSDL-S) and hence plays a vital role in EUD of Web Service applications by assisting in planning and reasoning about compositions (Singh and Huhns 2006; Domingue et al. 2008a). Moreover, Web 2.0, through the participation concepts (such as blogging, wikis and social networking) and technologies (for e.g. Atom and RSS for web syndication), has further simplified and supported users’ participation and contribution in creating web content and resultantly their empowerment (Singh and Huhns 2006; Patel et al. 2010). As a result, the web is rapidly progressing towards a highly programmable platform and end-user programming has become a very popular and common trend nowadays (Burnett et al. 2004). This facilitates not only the work of developers but also enables end users to leverage different Application programming interfaces (APIs) to create and publish their own content and develop advanced IT solutions to business problems on desktops as well as mobile devices (Chang et al. 2014).

The APIs exposed on the web can be used to create a mashup in two ways. Either programmers can use and combine them by employing their technical details or novice users can employ mashup tools to combine different aspects of these APIs to develop a mashup application (Yu et al. 2008; Patel et al. 2010). This tool-oriented mashup development from the perspective of end-users is the focus of this study.
2.4. Mashup Based Application Development-A new Trend in EUD Driven by Services Oriented Computing

The term Mashup takes its inspiration from pop music where a mashup represents a new derivative work created by mixing different songs (or their parts) (Yu and Woodard 2009). Similarly, EU service mashups are simply new applications that repurpose existing web data and APIs on different levels (presentation, logic and data). According to Maximilien et al., “Mashups combine views, data, and logic from existing Web sites or applications to create novel applications that focus on situational and ephemeral problems” (Maximilien et al. 2008). However, the tools that have been developed for the mashups are not user friendly according to the research conducted (Ma et al. 2015). There has been consensus on the fact that the tools need to impose less programming burden on the end users so that non-technical users can use them more easily due to knowledge about web APIs, parameter settings in widgets, data mappings, and other development efforts (Rodríguez et al. 2014; Namoun, Nestler, et al. 2010b). Moreover, there have been cases where studies present mashup challenges including limited programming knowledge of end users without any scientific evidence (Nestler 2008).

2.4.1 Tool-assisted Mashup Development: Surveys and Evaluations

Service mashups and the related concepts have been researched for over a decade now. The development of mashup tools has benefitted from industry-led research such as Yahoo\(^3\) (yahoopipes.com) and IBM (IBM Sharable Code) (Maximilien et al. 2008) apart from organizational funded projects\(^4\) (such as ServFace (Feldmann et al. 2009), SOA4All (Domingue et al. 2008a), and FAST (Hoyer et al. 2009)). Different surveys and studies have been conducted to study the state of the art of mashup development tools from different perspectives (Beletski 2008; Hoyer and Fischer 2008; Yu et al. 2008; Shuli Yu and Woodard 2009; Grammel and Storey 2010; Patel et al. 2010; Paredes-Valverde et al. 2015) (Yu et al. 2008; S. Yu and Woodard 2009; Patel et al. 2010; Beletski 2008; Hoyer and Fischer 2008). Few of these studies have tried to shed light on the conflicting results of acceptance of mashups as a research field but its poor rate of adoption by end users which seems to be the most likely reason of the current announcement of yahoo to discontinue support to Yahoo Pipes\(^5\). This highlights the need to dig deeper into the end-user expectations.

---

\(^3\) Officially yahoo pipes has been discontinued since Sept. 30\(^{th}\) 2015 http://pipes.yqlblog.net/

\(^4\) The official websites for these funded projects have been discontinued.

\(^5\) Officially announced on www.yahoopipes.com
(Namoun, Nestler, et al. 2010a). Different technology-adoption theories have also been presented to understand the dynamics around end-user development (Fischer et al. 2004). This section presents a summary of the efforts already undertaken in this regard.

Patel et al. presented a study for the mashups development tools and evaluated the top ten mashup tools against a set of seven features (Patel et al. 2010). However, the selection of the top ten tools has not been explicitly explained. Authors have also tried to summarize mashup concepts and most popular APIs according to ProgrammableWeb.com statistics. This study highlighted iGoogle as the most user-friendly platform but there is no explicit proposal for improving the highlighted issues. Besides, it is felt that a distinction should have been made between presentation level (or portals) mashups and data integration or process integration level mashups.

In (Yu et al. 2008), a study was conducted proposing concepts as dimensions to analyse the selected approaches. One interesting contribution of this evaluation is casual use case-style descriptions of how a sample mashup can be developed using each of the selected approaches. The paper also evaluates the approaches based on concept characterization along component model, composition model, development environment and runtime environment. These evaluations are all descriptive and no table is presented for a quick snapshot of the evaluation results. Finally, the authors have concluded with the need of an end user-oriented interface, integration mechanisms and middleware. However, again no concrete future direction or solution is discussed as part of the conclusion.

In (Hoyer and Fischer 2008), a study was carried out on mashup tools with the focus on enterprise mashup tools. The authors have tried to explain the difference between the enterprise mashups and consumer mashups and the evaluation covers both the target end users’ groups. This implies that the study is about mashups in general rather than being exclusively about enterprise mashups. Besides, the sole distinction made between the two types is the additional requirements like security, quality or availability in the enterprise systems. However, it can be argued that these characteristics have also proven to be of importance to the general consumer/user mashups albeit at different level of concern (Namoun, Nestler, et al. 2010a). The study has presented a model for lightweight resource composition and explained the wiring and piping paradigms while explaining three layers in

---

6http://www.programmableweb.com
7http://www.igoogleportal.com/portal/index.php#tab/3
enterprise mashup stack namely Resources, Widgets and Mashups. The tool support has been categorised into two levels catalogues and editors which are further classified into adapter, repository and transformation/aggregation, presentation layers respectively. The representative tools are discussed as case studies to show their functioning and how they fit into the classification model. The classification model is also presented in a graphical format for evaluation of 30 mashup tools including enterprise as well as consumer mashup tools. The issues raised in the conclusion include lack of screen flow design, semantic aspects and coverage of typical enterprise requirements such as security. Like the analyses discussed earlier, this too does not propose a solution or future plan for dealing with the issues.

According to a more recent study (Paredes-Valverde et al. 2015), discusses different tools and tool evaluation efforts followed by a qualitative evaluation according to a new criteria based on 6 aspects. These aspects are: GUI Design, Tool usability, features promoting use, support for heterogeneous data sources integration, support for building graphical interfaces and support and documentation. Finally a quantitative evaluation based on the generated mashup is also carried out. The tool selection is random and comprises of Apatar, DERI Pipes, JackBe Presto, Netvibes, SnapLogic, WaveMaker, and Yahoo! Pipes. The qualitative evaluation ranks JackBe Presto as the most usable tool.

2.4.2 Classifying End-User Mashup Development Issues

The recent advances in the web such as web2.0, semantic web and presence of Web Services and APIs coupled with end-user programming are increasing the potential of web programmability.

Different tools and platforms have been developed to aid end-users in developing situational applications for their ephemeral needs (Nestler 2008). It is a well-known fact the existing tools have been more focused on providing a working platform for satisfying ad hoc needs by creating service based applications in mashup-style rather than systematic investigation of the limited success of mashup development among the end users. My research after analysing a set of selected mashup tools remarked about lack of user-orientation in existing tools since they didn’t explicitly support user goals and requirements. This finding is backed with evidence from, and has been reported in, various other research studies (Mehandjiev et al. 2010; Namoun et al. 2010a, b, c; Väänänen-Vainio-Mattila and Wäljas 2011; Angel et al. 2011; Cappiello et al. 2011). These user studies
have been carried out by the research community to understand the challenges underlying and requirements, expectations surrounding the mashup-style services compositions. I have classified the primary reasons behind the limited success of this research area in three distinct but related groups: End-User Development, service composition and tool capabilities. A brief description of each is listed in Table 2.2.

Table 2.2 - End User Mashup Development Challenges

<table>
<thead>
<tr>
<th>End-User Development Challenges</th>
<th>Motivational Issues: Due to steep learning curve, the users can feel demotivated (Fisher et al. 2004)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Lack of IT skills: EU by definition are domain experts with no or basic IT knowledge, which can prove to be a major hindrance in service compositions. (Angeli et al. 2011)</td>
</tr>
<tr>
<td></td>
<td>Technical Burden: Since there is traditional tension between complexity and support (features), increasing the development support for mashup development in a platform will result in increasing the learning burden on the end users.</td>
</tr>
<tr>
<td>Service Composition Issues</td>
<td>Data complexities: Data should be abstracted to hide the complexities of the data.</td>
</tr>
<tr>
<td></td>
<td>Service complexities and technical terminology: Technical jargon can prove to be a main demotivating factor for the end-user. Hence, technical details should be properly abstracted out using appropriate visual models.</td>
</tr>
<tr>
<td></td>
<td>Service selection and Compatibility details: End-Users find it difficult to understand and improvise the services' compatibilities.</td>
</tr>
<tr>
<td>Tool Capabilities</td>
<td>Complex Process Mashups: The existing tools are not good enough to compose complex multi-page process mashups (Nestler 2008).</td>
</tr>
<tr>
<td></td>
<td>Feedback: The tools should be designed to provide instant feedback (Cappiello et al. 2011).</td>
</tr>
<tr>
<td></td>
<td>Lack of Context and Personalization: The tools should support context and personalization (Mehandjiev et al. 2010)</td>
</tr>
<tr>
<td></td>
<td>Programming Language: Complex programming languages can address a wider array of problems but are hard to learn (Fischer et al. 2004)</td>
</tr>
</tbody>
</table>

For mashup development to become more popular, the three hurdles mentioned above must be addressed. This framework is an effort to address these end users challenges prevalent in this research area.
2.5. Goal Oriented Mashup Development

According to principle of cohesion, systems exist to achieve a goal (Meyer 1985; Yourdon and Constantine 1979), hence goal-oriented modelling is a natural way of representing and abstracting knowledge (Johnson 1992). A goal is a high level statement of system’s and/or user’s intentions or can also be defined as an objective that the system being developed should achieve (Lamsweerde 2001). Goals can be defined and/or identified at different levels of abstraction, from more objective system statements to low-level operational requirements.

2.5.1 Suitability of Goal Based Methods for End-User Mashup Development

A- The concept of Goal:

The concept of goal has been defined in different ways depending on the application area and context ranging from domains as simple as ball-games to as sophisticated as artificial intelligence (AI). However, I have looked at the definitions of goal from within the area of computer science/software engineering where goals have been used to derive and define stakeholder’s objectives, requirements for software systems and system behaviour (Santos et al. 2008). In AI, goal is “a description of a world state that is expected to be realized”. In Agent-oriented computing, a goal is “a state with highest utility and an agent must choose the course of action to reach that goal” and “a final state that the agent tries to achieve by moving from its initial state through a defined and finite sequence of intermediary states” (Santos et al. 2008). In system development and RE, goals are the objectives to be achieved by system under consideration (Lamsweerde 2001). Goals are the high level statements or objectives that a system under consideration should achieve by the cooperation of the agents and GORE refers to the use of goals in the requirements engineering activities including elicitation, analysis, specification, validation and management (Lamsweerde 2000). Before GORE, main emphasis of RE was on what and how questions but GORE provided the rationale for requirements and Yue was the first one to comment on the completeness of requirements by suggesting to link sufficient requirements with goals in order to establish the objective as a check of completeness (Yue 1987). The use of goals provides other benefits that directly correspond to the requirement engineering objectives defined in standards such as IEEE 830-1998 (Davis 1993). GORE is reported to provide better completeness, pertinence, rationale, traceability, readability,
stability and derivation of requirements including improved conflict management and better level of abstraction to help choose between alternatives during the requirements elaboration process (Van Lamsweerde 2001; Yu and Mylopoulos 1998). I have merged the agent-oriented definition with that of requirements engineering to come up with my own concept of goal: “Goals are high level objectives/states that the application must achieve by helping end users moving from the initial state through a defined and finite set of intermediate states”.

The concept of goals is considered as a naturally suitable way of doing RE since unlike requirements, goals are concrete and less volatile statements that naturally provide a justification and organization mechanism for requirements (Yu and Mylopoulos 1998).

Goals happen to be the high level statements and operationalizing goals into requirements help satisfy the completeness objective of requirements specifications; since to satisfy goals, sufficient level of elaborations are required (Yue 1987). This goal derivation mechanism also helps in ensuring that requirements are pertinent to the system (Lamsweerde 2001) and thus it keeps the scope creep under control. The tree structure of goal graphs apart from providing the visual picture of goals also help in traceability of requirements as well as in increasing the readability of requirements document (Yu and Mylopoulos 1998).

All of the above reasons contribute to the growing trend of Goal oriented requirement engineering and goal-based techniques, methods, tools and techniques in the software research community. A literature based survey of goal-based approaches in SOC is given in the next section.

### 2.5.2 Goal-Based Methods in Services Oriented Computing

Goals have been used in services oriented computing (SOC) and services oriented architecture (SOA) for matching requirements of services requestors and functionalities provided by services i.e. service descriptions. There have been attempts in SOC to apply goal models for specifying services and their requirements, as well as for services discovery and composition. Gehlert et al. (2008) have presented a method that uses TROPOS goal-models to align services and requirements (Gehlert et al. 2008). Certain other initiatives have also been presented in SOC that use the concept of goals in Web Service discovery.
and composition such as WSMO\(^8\) (Web Services Modelling Ontology)(De Bruijn et al. 2006) and GoalMorph (Vukovic and Robinson 2005). Although, services composition and discovery can be regarded as a commonality between SOC and end-user mashup programming, however where the initial requirements for SOC might be specified by the requirement engineer, in mashup programming there is by far no separate phase for collecting, eliciting or specifying requirements (a research gap which I have addressed in my research).

**A- Goal-Based Methods in Mashup Programming**

The evidence of using goals, goal-models and goal-based methods such as Tropos (Giunchiglia et al. 2003; Bresciani et al. 2002) and I* in SOC can be considered as a motivational factor for applying goals in mashups; however, following differences must be kept in mind between SOC and mashup development:

**i. No source of early requirements**

While Services based applications can use the concept of a requirement engineer to specify the initial goals, mashup programming is an on-the-fly assembling of services.

**ii. Domain Ontologies**

The scope and frequency of composition of services based applications and that of mashup development varies. While, concentrated efforts can define the domain-specific task ontologies in the services based applications, the platforms for mashup development should support the composition of services from various domains which makes it difficult to incorporate the domain-dependent descriptions of concepts. For instance, a mashup composition tool like Yahoo pipes can be used to develop applications including a wide variety of range from video clips (YouTube API) to geographical maps (Google Maps API).

**iii. Technical Difficulties Exacerbation**

It has been mentioned time and again that end-user mashup development should be user-friendly and technical details should be hidden to encourage end-users for using the assisted composition tools. Hence, any attempt to handle the issues with greater degree of automation should be traded off with user-friendliness.

---

\(^8\)http://www.w3.org/Submission/WSMO/
2.5.3 Existing Goal Based Methods

A- **I* (I-STAR)**

I-STAR is a goal-oriented methodology that is based on the idea of social modelling (Eric 2009). The main motivation of I* is the incorporation of social and human aspects in the software development process specifically in the early phase requirement engineering (RE) (Yu et al. 2011). For this purpose, I* includes a social ontology in its main modelling constructs comprising human actors thus drifting away from a mechanistic view of RE. In contrast to the existing modelling standards such as Unified Modelling Language (UML) and IEEE⁹ std. 830, which depict static dynamic behaviour of the entities, I* specify the intentional and motivational behaviour of actors by the use of intentional modelling and hence tends to specify the working of the system in terms of actors and their collaboration in the problem space rather than in the solution space. The claimed advantage of I* intentional modelling is the much neglected focus on social aspects rather than technical and mechanical aspects which impoverish the models.

I* modelling framework has been developed to mainly address the early phase requirements engineering, however in later research studies, I* models have been uses in conjunction with other techniques to provide different level of support during system analysis and design. I* framework originally supports 2 models: Strategic Dependency Model (SD) and the Strategic Rationale Model (SR). These models cater for the needs of early phase requirements engineering to identify the relationship among stakeholders and their goals. The SD model of I* depicts external relationships among actors and acknowledges the autonomy of actors in the social world. Technically, SD model is a network of directed dependency relationship among actors by supporting 4 types of dependencies namely: Goal Dependency, Task Dependency, Resource Dependency, Softgoal Dependency. The strategic Rationale (SR) Model is the other main type of goal graph that in which goals, tasks, resources and softgoals are attributed and related to each other. I* Ontology makes use of concepts such as goals, actors and agents in a special context. Actors: In I* actors can be physically embodied entities such as humans or they can be abstract logical actors such as roles. Agents: I*  

---

⁹ Institute of Electric and Electronic Engineers standard
specifies those actors as agents that are physical human actors. An agent with multiple roles is classified as a position in I* framework.

Table 2.3 - Comparison of I*, Tropos and KAOS

<table>
<thead>
<tr>
<th>GOAL CHARACTERISTICS</th>
<th>I*</th>
<th>TROPOS</th>
<th>KAOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFINITION OF GOAL</td>
<td>&quot;a condition or state of affairs in the world that the stakeholders would like to achieve.&quot;</td>
<td>&quot;a condition or state of affairs in the world that the stakeholders would like to achieve.&quot;</td>
<td>&quot;a non-operational objective to be achieved by the composite system&quot;</td>
</tr>
<tr>
<td>DISTINGUISHING PHILOSOPHY</td>
<td>Actor autonomy and intentionality, Social Modelling</td>
<td>Agent-based, Special emphasis of Req. Analysis</td>
<td></td>
</tr>
<tr>
<td>SUPPORT</td>
<td>Early phase RE</td>
<td>Early Phase RE to Detailed Design</td>
<td>Late RE (Req. Specification)</td>
</tr>
<tr>
<td>GOAL MODELS</td>
<td>Strategic Rationale (SR) and Strategic Dependency (SD) Model</td>
<td>Actor Models and Goal Models</td>
<td>1-Goal Model</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2-Object Model</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3-Agent Responsibility Model</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4-Operation Model</td>
</tr>
<tr>
<td>LANGUAGE/TOOL SUPPORT</td>
<td>GRL (Goal-Oriented Requirements Language)</td>
<td>Tropos Modelling Language</td>
<td>GRAIL tool</td>
</tr>
<tr>
<td>GOAL CONCEPTS</td>
<td>Softgoal, belief, Constraint</td>
<td>Softgoal, belief, constraint</td>
<td>Achievement Goal, Maintenance Goal</td>
</tr>
</tbody>
</table>

B- Tropos

The Tropos project, initiated by John Mylopoulos, is an extension of the I* project and extends from the early phase RE until the detailed design (Giunchiglia et al. 2003). It builds on the concepts of I* such as actors, roles and positions and is categorised as an agent-oriented framework and provides extended support for detailed design (Bresciani et al. 2002).

C- KAOS (Knowledge Acquisition in Automated Systems)

The KAOS project assumes that a basic understanding of the system’s objectives is available and supports the further specification and derivation of the main objectives. Consequently, it provides support for later stages of requirements engineering (Kavakli 2002; Regev and Wegmann 2005; Dardenne et al. 1993).

2.5.4 Recommendation for Selecting a Goal-Based Method

It was revealed during the research that I* and Tropos can be classified as belonging to the same family of Goal-oriented methods as they share basic goal concepts and philosophies (Regev and Wegmann 2005). As can be seen from the figure 2.4 and is presented in the table-2.3, all 3 methods provide different levels of support and support different goal
concepts. It has also been noted that while I* and Tropos explicitly support the early phase Requirement Engineering (RE) which involves the understanding of the existing system, the KAOS majorly concerns the future goals and to-be system (Kavakli 2002).

Keeping in view the typical mashup development scenario, it can be argued that a method that supports early phase requirements such as I* could be a better fit. Since, I* also emphasizes on the intentional autonomy and sociality of software development, the mental models of end users could be better represented using the strategic dependency (SD) goal models. Alternatively as suggested by Kavakli, a method integration approach could also be used, if the SR and SD models of I* are deemed insufficient to model the mashup requirements needs (Kavakli 2002).

However, since the mashups and end-user requirements while developing a mashup underline a different process as compared to a normal distributed or multi-agent system, the selection of the right goal-oriented method from the existing set of methods seems challenging. The fundamental differences between Mashup applications and that of traditional software applications triggered the need of a new goal based method.

![Figure 2.4 - Level of Support provided by KAOS, I* and TROPOS in Traditional Software Development](image)

A Criteria for Proposing a Goal-Formalism for Mashup Programming

In this section, I am presenting tri-dimensional criteria based on the earlier research for specifying the conditions that could help me propose a new goal-based formalism for representing and managing the end user expectations.
1- **End User Service Compositions (EUSC) Compatible**: The proposed goal-models should be aware of and consequently compatible with End-User Service Compositions style of development. This dimension considers the needs of end-users embedded within the service composition framework.

2- **Services Oriented Requirement Engineering (SORE) Aware**: Services-led development of applications requires a service-oriented solution that is able to meet the requirements of service-oriented computing. This requires that the solution is sensitive to the requirements engineering needs of a typical service application scenario.

3- **Mashup Process Centric**: The proposed method should be compatible with general features of Mashup development life cycle. End User Service Mashups are a special case of service oriented computing and have conditions that are specific to mashup and external to service based applications. The rapid Lego-style development of applications, which are a unique characteristic of mashup, requires that the process-specific details of a standard mashup development scenario be considered while developing a new method.

The detailed criteria of all of these dimensions are summarized as follows:

**i. End-User Service Composition Compatibility**

After analysing the results of existing research on mental models and expectations of end-users with respect to service composition, following main sub-criteria has been developed:

a) **Abstraction of technical details**: There has been considerable research on service compositions and developing tools for mashup development. While experienced users skilled in IT might still prefer programming over tool-based compositions, it is considered as a difficult task for end users. Hence, it is required that mechanisms that hide the technical details are employed to address this issue. This should also include Operational Abstraction of data (i.e. treating sets of data as one item).

b) **Resolution of dependencies**: This requires the tool to hide the technical dependency between different elements so as to ease away the burden from the end user of technical decisions. Two special dependencies that
A Framework for Improving End-User Orientation of Service Mashups

should be resolved are the control flow dependencies and data dependencies.

c) Services Integration support: The tool should provide the right kind of support for integrating the services at the interface level using visual modelling elements.

d) Continuous feedback: This requirement applies to general Tool/ User Interface (UI) design and requires that tools should provide live continuous feedback to the EU in the form of messages, alerts and tool tips.

ii. SORE Aware

What is SORE
The emergence of Services Oriented Computing (SOC) has not only changed the original concept of software development life cycle (SDLC) but at the same time has opened doors to myriad research opportunities in SOC including requirements engineering thus giving way to specialized field for RE in SOC named as SORE (Tsai et al. 2007). Being the first, foundational and fundamental step of software development methodology, its importance can never be stressed enough as various studies have shown that getting the requirements right not only is one of the main factors of the successful development of the resulting software (Bell and Thayer 1976; Lutz 1993; Ibanez and Rempp 1996; Group 1995), but getting them right earlier saves developmental costs as well (Boehm 1981).

The role of requirements in SOC becomes even more important with the fact that functional and non-functional attributes of services in Service-based-Systems are an integral part of entire - often disjoint - services development life cycle. This fact is shown in figure 2.5 where the perpendicular axis comprising Quality of Service attributes, semantics and functional and non-functional characteristics cuts across all the layers of service oriented computing research road map (Papazoglou et al. 2007).

**SORE and Mashup Development**

SORE, at the level of services descriptions and compositions can be used as an alternative model for users’ expectations and requirements in mashup programming. However, since this research relates to improving the end user orientation of mashup development, I am replacing services with user expectations/requirements in the guidelines. Hence, the literature has been analysed concerning SORE to draw and outline the requirements-related expectations of mashups along 3-sub categories:

**Service Discovery**

a) **Iterative discovery process**: The proposed solution should support an iterative process to discover the users’ expectations needs. This is due to the fact that EU requirements are prone to ambiguity.

b) **Locating Right service according to user requirements**: Mashup development requires the target services to be composed to develop
an application that serves the situational demands of EU. End Users, at the time of composition might not be aware of the right service or choosing the service that is not a best fit for his/her requirements. This condition will make sure that the right service is chosen and recommended to the EU.

c) **Automated dynamic service discovery:** The proposed tool/technique should employ the mechanism that ensures the automated and dynamic service discovery. Automated here enables the EU to complete his composition task without getting the prior knowledge of all available services and dynamic enables the discovery process to be adaptable to the user changing requirements.

**Knowledge Management**

a) It refers to the need of possessing knowledge of Previous Compositions to increase the efficiency of central repository. This implies that ontologies as well as existing composition knowledge about the domain and users’ expectations should be reused. The benefits will be two-fold. Firstly, it will make the task simpler for the EU; secondly the system will be able to support an adaptable and adaptive environment hence increasing its efficiency.

b) Grouping of services according to their functionality to improve the search procedures. This requires the similar services to be grouped based on common domain functions resulting in search efficiency.

**Composition and Semantic Issues**

a) Web services dependencies should be discovered and resolved. The System should able to handle the compatibility issues between the services.

b) Bridge the semantic gaps as services are brought together from hybrid environments. Selected services might not be able to meet all the users’ need and there might be a requirement for an additional service for addressing that gap. The system should handle these composition issues by tracking the users’ expectations and service requirements.
iii. Mashup Process Centric

Keeping in view the short development process and run-time compositions factors of mashup development, it may not be appropriate to identify requirements engineering as a timed phase for mashup development process. However, the benefits of identifying end-user’s objectives, their derivation to sub-goals and ensuring the completeness of requirements are activities that are still relevant to mashup programming like any other software development paradigm. In the following section, the mashup requirements are discussed based on a typical/standard composition scenario:

a) In mashups, the initial requirements description of a user will always be fuzzy and inaccurate due to myriad reasons. One of the main reasons is that the users’ knowledge about existing services can never be complete and secondly, albeit the situational nature of the target application which is being developed/composed, all the requirements can never be known in the beginning. Hence, the system should support a natural, user-friendly mechanism of deriving users’ expectations/needs.

b) The initial requirements should be complemented according to the knowledge repositories based on similar functions. This condition necessitates the reuse of requirements at a higher abstraction level than that of domain.

2.6. Meta Design

It is a conceptual framework that is proposed as a solution to a constant need of change in the design of existing systems and has been prescribed to meet the future EUD needs (Fischer and Giaccardi 2006). It comprises of principles, guidelines and models that are aimed at defining social and cultural systems and creation of infrastructures that enable and promote collaborative design and co-creation. The conceptual underpinning of this framework, as is the motivating force behind EUD, is the assumption that future uses and problems of a given system cannot be anticipated during design-time leading to mismatches during use-time resulting in new insights that can be employed for the benefit of end users (Fischer and Giaccardi 2006). It is a concept that believes in cultures of participation, thus fostering a mind-set that moves end users from the passive consumerism side to a more active designer’s role ultimately making them owners of their problems (Fischer 2009). Hence, it seeks to extend the traditional system design phase
beyond the system development of that system, and enables the users to become the co-developers or co-designers of the system by making use of a co-adaptive and collaborative process between user and the system.

Meta-design systems have also been termed as open systems and require technical as well as social conditions for realizing the goal of large scale participation by user communities in design activities. It is a multi-dimensional framework and has been discussed in context of various socio-technical systems pertaining to different domains such as interactive art, open source software development, domain oriented design environments (DODEs), and learning communities (Fischer and Giaccardi 2006). Meta design primarily tends to attend to three necessities of socio-technical environments: 1- flexibility 2- Co-designing at the hand of EU and 3- design for evolution. To achieve these necessities, following objectives have been outlined (Fisher and Giaccardi 2006; Fischer et al. 2004)

The first objective is “design for change”. It implies that a culture of participation be built around users in such a way that it encompasses not only their professional lives but also their personal lives. Hence, it seeks to invent cultures where humans are able to express themselves in meaningful activities. The main challenges posed in this context relate to the “How” mechanisms, i.e. 1- How to provide the right kind of support to domain experts, 2- How to create Co-adaptive environments and 3- how to manage the ubiquity of the participation culture.

The second objective tends to blur the line between “design time” and “use time”. This objective requires the meta-design systems to be under-designed by creating environments that are not providing the solutions themselves but that supports design spaces in which the users or “owners of problems” can create the solutions for themselves. This does not necessarily mean that the user is pushed from a consumer to a designer; it rather requires providing a support that helps the end users to migrate from being a passive consumer to a meta-designer on a continuum.

The third objective seeks to extend the Notion of User-Centered and Participatory Design. In User-centered design, solutions are generated by designers that place users in the reactive roles and in the participatory design, users are empowered to propose design solutions, however none of these approaches tend to involve users beyond the original development of the system and mainly focuses on the design time. This does not satisfy the main condition of flexibility of meta-design, which is becoming the prime requirement
of modern day systems. Hence, meta-design takes these approaches further by proposing to enable users to co-evolve these open meta-design systems.

The process model proposed in the context of meta-design is known as “Seeding, Evolutionary Growth and Reseeding Model” (Fischer and Ostwald 2002; Fischer 1998). It supports the vision of meta-design which tends to support “users as designers” thereby conceptualizing their activity of meta-design. The term seed, here, refers to an open initial system that can be evolved over time by small contributions of a large number of people. Evolutionary growth is the decentralized growth phase which does not involve developers rather users are focusing on framing and solving the problem. In the final phase of reseeding, a concentrated and formal effort is carried out to incorporate the information created in evolutionary growth phase.

The final goal of meta-design is to foster an unself-conscious culture of design. This means to facilitate informed participation in which participants from a wide range of professions collaborate and contribute to the solution of a given problem. While explaining this concept, Fischer and Giaccardi have cited the examples of planned cities (Canberra) vs unplanned cities (London, Paris), to be used as metaphors to emphasize on the need of new interactive information systems that provide a forum for social debate and discussion and do not concentrate on repositories of given information (Fischer and Giaccardi 2006).

2.7. The Domain Theory

Inspired by different factors such as saving time and effort, the area of reusability is being researched over for more than 30 years now and has had an impact on many industries such as electronics and computer hardware; however A. Sutcliffe claims that the software reuse is still not a success story (Sutcliffe 2002). To address this, the domain theory is a hybrid theory of knowledge representation proposed by Alistair Sutcliffe in computer and cognitive science. It addresses the knowledge reuse problem in computer science and software and tends to tap into the potential of reusable components. It is based on the principles of granularity and abstraction and defines three different levels of reuse in relevance to the Software development life cycle (SDLC) namely, Requirements Reuse, Design Reuse and Software components reuse. Each of these concepts is summarized in the following paragraphs.
Domain theory proposes the software reuse by resting on the principles of abstraction and granularity. Abstraction, to be achieved by generalization in the field of computer science is defined as: “the loss of detail to model a more general view of the world”. According to Sutcliffe, abstraction increases the number of potential reuse and the higher the abstraction, the wider is the range of the reuse (Sutcliffe 2000). In contrast to Abstraction, Granularity is calculated as a measure of complexity by the use of function points. Hence, higher the complexity higher is the granularity, and lower the abstraction. In order to support higher level of reuse, it is ideal to have smaller-grain components with high level of abstraction since large-grain components will be less flexible owing to containing more functionality consequently decreasing the abstraction and potential of reuse.

Grounded Domains, generic tasks and meta-domains form the main components of domain theory. Grounded Domains are the main core of domain theory (Sutcliffe and Maiden 1998) which can be located in the real world but have distant association with the physical structure. They are classified into Object System Models (OSMs) and Information System Models (ISMs). Examples include, hiring and monitoring applications. Meta Domains are composed of generalized tasks and operate on grounded domains to achieve a complex goal. Meta domains make use of cognitive tasks and examples include Education, Management and Research. The final component, generic tasks represent a small unit of activity and are independent procedures that achieve a comparatively simpler goal than the once achieved by meta-domains. Generic tasks represent the behavioural components in both meta and grounded physical domains. The examples of generic tasks are comparing, evaluating and identifying. Two or more generic tasks are composed to make up a generalized task. In the main contribution of this thesis, reusable libraries of generic and generalized tasks are used and applied to the design of mashup templates to reuse design knowledge in the form of goals and sub-goals. However, I do not use The Domain theory to the system-level details.

The domain theory has been applied successfully in various projects and case studies (Sutcliffe 2000).

2.8. WSMO Goals

Web Services aim to resolve the limitations of conventional Enterprise Application Integration (EAI) middleware intended for business-to-business (B2B) communication (such as that provided by Ariba and CommerceOne) for executing organizational business
processes (Elonso et al. 2005). Besides, they also promise a more customer-friendly way of finding and executing required services for facilitating end users in accomplishing their everyday tasks in a conventional business-to-customer (B2C) setting. Hence, Web Services and related semantic technologies have great potential in developing scalable, dynamic and cost-effective enterprise applications. Owing to these reasons Services Oriented Computing (SOC) and Services Oriented System Engineering (SOSE) are being taken as an emerging trend in the Business Systems (Tsai 2005). In the direction of semantic technologies, two most important and noticeable efforts are Web Ontology Language for Services (OWL-S)\(^{10}\) (Martin et al. 2004) and Web Services Modelling Ontology\(^{11}\) (WSMO)(De Bruijn et al. 2006). These efforts are aimed at describing the semantic Web Services so as to support the service invocation, discovery, composition and interoperation. WSMO (De Bruijn 2005)is based on the WSMF (Web Services modelling framework) and OWL-S is based on DAML-S (DARPA Agent Mark-up Language for Services) (Lara et al. 2004). The purpose of discussing these efforts is to justify the selection of WSMO as the language for the final goal specifications in my contribution of a new approach discussed later in the thesis. Both the approaches follow a top-down approach and are based on a set of principles but I am considering WSMO for the goal specifications in the proposed approach due to following reasons (De Bruijn et al. 2006; Lara et al. 2004):

1- WSMO is comprised of 4 components namely: Ontologies, Web Service Descriptions, Mediators and Goals. Ontologies provide the terminology whereas the Web Service descriptions are used to describe different elements of Web Services. One of the main reasons of selecting WSMO is the concept of goal which describes the intention of the user. Finally, mediators manage the compatibility issues during interoperation.

2- WSMO identifies and acknowledges the differences between users and services not only in terms of ontological roles but also their technical descriptions. Web services are described by their capabilities and they may differ from user intentions or goals. Hence, WSMO clearly differentiates between the point of views of requester (or user) and the provider.

\[1\] 10 http://www.daml.org/services/owl-s/1.0/owl-s.pdf

\[2\] 11 http://www.w3.org/Submission/WSMO-primer/
3. WSMO follows the design principle of loose-coupling and hence can allow the goals, ontologies and service capabilities to exist independent of each other unlike OWL-S where each resource is tightly coupled with other resources.

2.9. Discussion

In spite of the popularity of mashups and their suitability for end-user development of situational applications, more emphasis has been laid on the technical platform capabilities rather than on addressing the end-user needs (Yu et al. 2008; Namoun, Nestler, et al. 2010a). Consequently, end-user Web Services compositions research is still considered to be in its infancy (Namoun, Nestler, et al. 2010a). Several studies have reported the research gaps in this area (Yu et al. 2008; Patel et al. 2010; Hoyer and Fischer 2008; Beletski 2008); but none have specifically addressed the role of end user requirements or users goals in enabling mashup development in much detail. Most of the discussions have highlighted issues such as the need of programming skills or an understanding of technical knowledge (S. Yu and Woodard 2009; Yu et al. 2008; Patel et al. 2010; Lin et al. 2009; Tuchinda et al. 2008) as the impeding factor for end-user mashup development, or lack of user orientation (Namoun et al. 2010a; Nestler 2008) in current tools. However, most of the solutions end up in proposing, but not beyond a certain abstract level, to enhance user orientation without detailed empirically tested conceptual solution.

My proposal is an explicit, goal driven development of mashups, since goals are a technical yet natural way of deriving a system based on the underlying requirements of users (Van Lamsweerde 2001). Although most of the work done on goals pertains to requirement engineering, given the benefits of goals discussed in the chapter and commonality of centrality of user requirements in goals as well as during developing mashups, the use of goal-models for developing situational applications is promising. This idea can help improve the user-orientation of the mashup systems since end users can relate to the service descriptions by mapping them with their goals and requirements (if supported explicitly in the system) which is believed to maintain their interests as well during the whole technical activity of composing an application. Coupled with visual models and suitable mechanism to represent and manage domain knowledge, the idea of goals leading to mashups can be a reasonably practical solution.

Secondly, domain theory provides the right level of abstraction and granularity for hiding the technical details that has been the most discussed problem so far in the research field.
of service-based situational applications. The libraries of generalized tasks of Meta
domains and generic tasks are to be reused for this purpose.

Thirdly, I propose to motivate end users to understand the concept of services and the
necessary compositions by making them “the owners of the problem” (Fischer and
Giaccardi 2006). I propose to use a set of meta-design principles that could lead to an
adaptive and adaptable system thus making it easier for the EU to complete their service-
based tasks. In this way the inherent technical nature of the work involved in developing
end-user service based applications and the possible low learning curve due to high set of
features (due to complexity vs. learnability trade-off) could be managed since the emphasis
will be on providing a design space rather than ready-made design solutions. However,
keeping in view the scope of the work, creating a social space for design evolution and
instantiating the SER model that make up an important part of meta-design seems
unrealizable goals. Hence, it is not claimed that meta-design framework would be
implemented in totality.
Chapter 3

3. A Classification Model for Mashup tools

3.1. Motivation

This chapter presents the first contribution: a model for classifying the support available for end user service mashup development. As the title of my thesis suggests, the main theme of this thesis focuses on improving the end-user orientation of service mashups by focusing on user-to-service scenarios. To enhance the consumer experience and empower the users to benefit from the service compositions, it is crucial to identify the end-user goals, properties and corresponding metrics and from them to infer the end-user friendliness of not only existing tools but also new mashup development solutions. Since usability plays a central role in the development of any software tool in general and its relevance increases manifold in the area of end user mashup development in particular, a more general motivation to develop this model was the need to validate EU-friendly mechanisms to ensure that mashup development platforms and methods are meeting the expectations of EU.

According to Brinkkemper, agreeing on terminology is central to scientific development (Brinkkemper 1996). Hence, before going into the details of my model, it is necessary that a distinction is made between these relevant terms: method, framework, tools and techniques. This will not only contextualize my contributions being discussed in this chapter but will also clarify the application of these contributions. These distinctions shall be applicable to entire thesis and are discussed within the scope of software engineering research. 1- Framework: Usually a framework is defined in the context of a specific field under discussion such as a framework for software process model (Mnkandla 2008). Generally speaking a framework is a set of abstractions and concepts that serve as a solution to a number of similar problems (Mnkandla 2009). 2- Method: A method follows a specific way of thinking to perform a systems development project (Brinkkemper 1996). It consists of systematic directions and rules that discuss the development activities of the project with corresponding development products. 3- Tool: A tool is a possible automated

---

12 Published in the IEEE Services Computing Conference, 2012, DOI: 10.1109/SCC.2012.19
means (a software program or an application) to support a part of a development process such as Microsoft Visio\textsuperscript{13} is a tool that helps in analysis and design of the system by enabling users to draw unified modelling language (UML) and other analysis and design diagrams. The term tool in the context of mashup development in this thesis is used interchangeably with platform, technology and solution and refers to the automation that allows end users to create, modify, share and deploy web applications with minimalistic programming requirements by assembling information/data from multiple sources. The scope of the tool depends on the level of support provided by the tool (Brinkkemper 1996).

4- Technique: A technique is a sequence of steps or a procedure to perform a development activity and usually has a notation associated with it. For instance, a technique to develop an entity relationship diagram for modelling the data specifies both the steps and the notation.

I have proposed this classification model with the goal of tackling the lack of end user support while focusing on well characterized and defined properties and goals in the state-of-the-art end user mashup tools. In this classification model, the end user friendly properties are specified explicitly and driven by quality attributes such as those defined by researchers of interfaces in general (Shneiderman and Plaisant 2004) and that of EUD (Repenning and Ioannidou 2006) and service mashups (Cappiello et al. 2011) in particular. The 3 dimensional classification model provides: (i) a basic criteria that is useful to classify mashup development support; (ii) a compendium of usability quality attributes to evaluate end-user orientation; (iii) a set of mashup design features including the mashing technique being used and the supported mashup development activities; and (iv) technical profile. I applied this model for characterizing and evaluating selected mashup development approaches. Table 3.1 represents the basic characterization of the mashup development support on a sample set of selected tools including the tools providing the minimalistic support to the comprehensive mashup development environments. Table 3.2 presents further characterization of the mashup development environments that provide the extended support for developing web mashups. Finally, table 3.3 provides the detailed application of my classification model on the set of selected tools. Following sections present detailed discussions on these characterization and the related dimensions.

\textsuperscript{13} https://products.office.com/en-us/visio/flowchart-software
3.2. Characterizing Mashup Development Support

This section presents the base model for the characterization of the end user mashup development software. According to a McKinsey survey (Bughin and Manyika 2007), mashup technology is reaching acceptance within enterprises owing to the active involvement of the actual service consumer and the resulting empowerment of end-users (Carl 2008; Heltewig 2007). The programmableWeb.com\textsuperscript{14} statistics show that the average rate of increase of mashups was 7% in 2010 (Patel et al. 2010). According to Programmable web research centre (ProgrammableWeb 2013), number of active APIs has passed the 10,000 mark, following an upbeat trend since 2005 (fig. 3.1). Consequently, number of supportive web 2.0 platforms such as web service discovery portals and mashup tools is also increasing (Hoyer and Fischer 2008; Mehandjiev et al. 15; Nestler 2008; Paredes-Valverde et al. 2015). These tools provide different specialized functionality and varied support based on their objectives (fig. 3.2).

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{growth_in_webApis.png}
\caption{Growth in Web APIs since 2005 (Programmable Web Research Centre, 2013)}
\end{figure}

\footnotesize{\textsuperscript{14} http://www.programmableweb.com}
3.2.1 Range of Support for Mashup Development

In order to provide a mechanism for classification, it is imperative to define criteria to classify the mashup development platforms. The model takes into account three factors for evaluating the mashup development platforms, namely: the development goal, inputs and outputs.

The development goal refers to the main objective behind the tool. They are divided into three groups based on the degree of support they provide for mashup development: 1) Vertical portals consisting of catalogues or directories that provide list and search facilities for APIs and mashups (for e.g. programmableweb.com). 2). Mashup Activity Support Tools (MAST) which are developed to accomplish a single or selected set of mashup activities instead of entire process of mashup development such as data extraction or annotation. 3). Mashup Development environments/platforms that comprise editors and frameworks for supporting entire phase of developing a mashup.

\[\text{www.programmableweb.com}\]
A preliminary classification of different tools was done based on this spectrum (table-3.1). This classification helped me in selecting the appropriate mashup platforms for the final classification (the last column of table-3.1) since the main objective of this classification model is to focus on comprehensive tools that provide maximum mashup development support. Hence, making the table 3.1 as baseline, the mashup activity support tools were excluded and only the development environments (listed in the last

<table>
<thead>
<tr>
<th>Catalogue/Directory</th>
<th>Mashup Activity Support Tool</th>
<th>Mashup Development Environments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Data Extraction Tool</td>
<td>Annotation Tool</td>
</tr>
<tr>
<td>ProgrammableWeb.com</td>
<td>KOALA</td>
<td>Ubiquity</td>
</tr>
<tr>
<td>Strikelron.com</td>
<td>Dynvoker</td>
<td></td>
</tr>
<tr>
<td>Mashable.com</td>
<td>Clip-Connect-Clone</td>
<td></td>
</tr>
<tr>
<td>RemoteMethods.com</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wsoogle.com</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XMethds.com</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
column of table 3.1) were selected. These tools were further shortlisted based on further detailed sifting. The details pertaining to criterion for the selection of tools is discussed further in the analysis section.

### 3.2.2 General Classification

To get an outlook of the tools selected for classification, a general classification was carried out (Table-3.2) based on target user group and tool type.

**Table 3.2 - General Classification of Selected Mashup Platforms**

<table>
<thead>
<tr>
<th>TOOLS16</th>
<th>Target User Group</th>
<th>Tool Type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Enterprise Oriented</td>
<td>Consumer Oriented</td>
</tr>
<tr>
<td><strong>Yahoo Pipes</strong></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td><strong>IBM Sharable Code</strong> <em>(Maximilien et al. 2008)</em></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td><strong>SOA4All</strong> <em>(Domingue et al. 2008b)</em></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td><strong>Intel</strong></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td><strong>MashMaker</strong> <em>(Ennals et al. 2007)</em></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td><strong>ServFace</strong> <em>(Feldmann et al. 2009)</em></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td><strong>FAST</strong> <em>(Hoyer et al. 2009)</em></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td><strong>Karma</strong> <em>(Tuchinda et al. 2008)</em></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td><strong>CMU</strong></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td><strong>Marmite</strong> <em>(Wong and Hong 2007)</em></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td><strong>Microsoft Popfly</strong> <em>(Loton 2008)</em></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td><strong>Vegemite</strong> <em>(Lin et al. 2009)</em></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td><strong>Dapper</strong> <em>(Koschmider et al. 2009)</em></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td><strong>iGoogle</strong> <em>(Casquero et al. 2008)</em></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

16 Original sites hosting different tools such as Dapper, Popfly have been discontinued. Whenever the tool was not available, the documentation was consulted for analysis purpose.
A. Target User Group

There are two types of target user groups defined in literature, however they are not mutually exclusive (Hoyer and Fischer 2008; Hoyer et al. 2008; Hoyer et al. 2009). Moreover, for this classification, the intent behind the development of the tool is considered as a base for classification.

i. Enterprise-Oriented: They combine data, content or application resources in an enterprise environment and are to be used by users from the business units (Hoyer et al. 2009). In addition, enterprise mashup tools implicate additional requirements such as security and reliability and employ a stack of components for streamlining the development process for organizational end users. (Hoyer et al. 2008). The stack consists of a base layer of resources followed by widgets with a final layer of mashup at the top. Widgets are microservices (Nicolaescu and Klamma 2015) and according to W3C Web Apps specifications they encapsulate a well-defined functionality and are client-side applications. Due to their structure, they can be reused, shared and personalized according to the needs.

ii. Consumer-Oriented: These tools are aimed at individuals for creating mashups for private and casual use by casual users (Sarraj and Troyer 2010) who are not using it as part of official or professional job within an enterprise (Hoyer and Fischer 2008).

B. Tool Type

Various types of tools have been defined in the literature such as editors, catalogues (Hoyer et al. 2009), but for the purpose of this analysis, the following two types have been identified (Yu et al. 2008):

i. Mashup Editing Tool: The mashup editing tool provides an interface for the combination/composition of different APIs to develop the desired mashup application. However, it is not normally (or even required to be) elaborate enough to include any other components such as domain specific language, process or task modelling editors or any other extended support.

17 http://www.w3.org/TR/widgets/
**ii. Framework:** Frameworks provide the maximum support for the users to combine their mashups and APIs and also gives the language and coding facilities for extended support and enhanced experience.

However, in spite of the growing number of Web Services and mashups, several shortcomings have been reported in the existing mashup solutions that limit their capability and call for improved mashup development solutions (Nestler 2008). Mashup development mainly inherits issues as well as motivations from two fields: End User Development (EUD) and Services Oriented Computing (SOC). One of these traditional problems is highlighted in Figure 3.2 where complexity and the features supported by mashup tools are shown as forces working in opposite directions. While increasing the features facilitates the support provided by the mashup tool, on the other hand it also hinders the end-user productivity by imposing a learning burden (Lausen and Steinmetz 2008).

### 3.3. A 3-Dimensional Model for Mashup Platforms

The model for classification takes into account basic and advanced characteristics of a mashup solution along three main dimensions: 1) Mashup Profile, 2) Usability and 3) Technical Profile (fig. 3.3). The model’s technical features and mashup design features are inherent and specific to the mashup domain mainly derived from the literature; whereas usability dimension is inspired from standards ISO 9241-11\(^{18}\) and ISO 13407\(^{19}\). The former defines usability while the latter provides the guidance for designing usability (Jokela et al. 2003). The standard ISO 9241-11 defines usability along the dimensions of efficiency, effectiveness and satisfaction in a specific context of use. 1-Efficiency: Resources expended in relation to the accuracy and completeness with which users achieve goals. I have specified this by Learning Curve and Advanced UI generation. 2-Effectiveness: Accuracy and completeness with which users achieve specified goals. For this purpose I included goals and requirements support. 3- Satisfaction: Freedom from discomfort, and positive attitudes towards the use of the product. The tutorial element and argumentation support has been included to meet the satisfaction needs.

The following paragraphs will explain the necessary details of the dimension of the criteria and the associated concepts.


\(^{19}\) [http://www.iso.org/iso/catalogue_detail.htm?csnumber=21197](http://www.iso.org/iso/catalogue_detail.htm?csnumber=21197)
3.3.1 First Dimension: Mashup Design Features

Mashup tools available currently provide one or more of the following activities:

![Diagram of Mashup Design Features]

A. **Supported Mashup Activities:**
   - **Data Mediation:** involves converting, transforming, and combining data elements from one or more data feeds or APIs. The mashup can thereby create a new data object or satisfy an API’s operational needs.
   - **Process Creation (or protocol mediation):** creates a new process by providing the choreography for different APIs by presenting the necessary interaction points to the user.
   - **User Interface Customization:** elicits user information and displays the intermittent process information to the user. It can involve web forms and static page units with dynamically generated content or more complex Ajax interfaces that refresh content or dynamically generate new pages based on user interactions.

B. **Mashup Techniques:**

Different tools use different mashing up techniques (or data processing techniques) for developing the situational applications (Domingue et al. 2008b; Patel et al. 2010). These techniques are not exclusively designed for mashup development. For instance, programming by demonstration or programming by example has been in use in general
end user development and is not exclusive to mashup development (Lieberman et al. 2006). These techniques have their own pros and cons which are presented in table 3.4 and discussed in detail in section 3.3.3. More details can be

i. **Wiring:** facilitates the mashup development by supporting connectors between modules, blocks, components (popularly known as widgets, gadgets or badgets). It is a type of visual language technique where diagrammatic representations rather than textual ones are encouraged (Blackwell 2006; Patel et al. 2010).

ii. **Spreadsheet:** is the most widely used technique and a popular paradigm for end user development (Blackwell 2006) and enables the data to be loaded into tables and then processed into desired format. It helps the end-users by eliminating much of the programming effort and allows users to see the initial and desired state of the data.

iii. **Programming by Demonstration/Example:** This technique allows users to apply the operations from templates instead of programming the operations themselves. It also helps eliminate/minimize the programming burden. In programming by demonstration, the user demonstrates the intended operation on a set of objects that will later be applied to another set of data objects (Cypher and Halbert 1993). It helps the user in registering procedures and doing repetitive tasks by using known functions of the tool. Programming by example is slightly different in which EU provides several examples of desired program behaviour which is then translated into a generalized program to be used later on (Blackwell 2006).

iv. **Script/Language based:** It supports the creation of the mashups at the code level. Normally XML scripts or a domain-specific language (DSL) define the services and operations to be used by a mashup. They also define the actions to perform as a result of service responses to construct the results of the mashup. A DSL can be defined as a mini language that defines the domain concepts and behaviours using common syntax and semantics (N. Mehandjiev et al. 2010).

v. **Webpage Customization:** Implemented both as an add-on to an existing browser or a complete separate application, this category allows the users to browse, edit and combine different web pages or the contents contained within them (Domingue et al. 2008b). As the combination takes place at the content or presentation level, hence this type is also regarded as a presentation mashups occasionally (Maximilien et al. 2008).
3.3.2 Second Dimension: Usability Features

Keeping in view the user’s focus in the mashup development tool for this study, this dimension has been added and composed using usability features discussed with respect to mashups in literature. This dimension is inspired by the “Reflection in Action” phenomenon (Patel et al. 2010). Reflection in Action occurs when one is able to consciously evaluate and make changes on the spot during an event.

Key usability features are discussed below:

A. Advanced UI Generation:
Mashups involve the execution of the participating APIs. Therefore, many mashup tools focus on the generation of user interfaces by deriving their inference from the services description like Web Services Description Language (WSDL) or Web Application Description Language (WADL). WSDL (Christensen et al. 2001) is a document written in XML that specifies the location and methods exposed by a web service. Due to XML syntax, it is machine processable. Similarly, WADL is also a machine readable description of web based applications (Hadley 2006). However, there are details that are not easy to infer from these technical descriptions, especially for an end-user who does not want to develop applications from scratch.

B. Signal/Prompt Incompatible Situations/Provide Argumentation:
This refers to a component (or a knowledge base) that monitors the user’s work and offers suggestions for changes.

C. Tutorial Element:
Different user studies have consensus on the presence of an embedded tutorial element that assists the user in learning about the application and the domain.

D. Learning Curve:
Also known as learnability, this feature indicates how easy it is to learn a particular tool. The classification for this feature is done on a 3-point rating scale: high, medium and low. The easier it is to learn a particular tool, the higher is the learnability and vice versa.

E. User Orientation:
When mashups are developed, they are composed of different Web Services, APIs, or scrapped web data which serve some purpose or fulfil specific user requirements. The
current requirements standards for Web Services are more technical rather than user-oriented such as those specified by WSDL (Christensen 2001) or WADL (Hadley 2006). However, these descriptions are meant for machine communication and are too technical to be understood by end users without a significant learning effort. Hence, this feature is included to acknowledge the importance of user goals and requirements by the tools under review. In the literature, the requirements have been discussed at different levels of abstraction: End-User requirements or expectations regarding the tool; and End-User requirements regarding the task at hand such as sending a text to a friend. This model is dealing with the latter type and rate whether the tools support the users by linking their functionality with users’ tasks, goals and requirements, a feature that is experimentally proven to be a recommendation for end-user service composition systems (Singh and Huhns 2006):

i. **User Goals:** This feature evaluates the explicit or even implicit support of user goals or tasks by the system/tool under review where the goal defines any objective in user terms such as arranging a hang out with friends that might consist of several low-level requirements.

ii. **User Requirements:** Technical jargon intimidates users from developing mashups or composing services through tools (Singh and Huhns 2006). This criterion will determine whether tools specify the user requirements beyond listing the APIs operations and other technical descriptions.

### 3.3.3 Third Dimension: Technical Features

While the above mentioned dimensions provide an insight into the mashup tools based on their mashup profile and the support provided to the user, this dimension seeks to evaluate the tools based on the technical feature set. Although, the users are not desired to have a direct relationship with the inner technical details (Singh and Huhns 2006), this dimension is formulated to understand the technical build-up of tools to see if they are considering the latest technologies in their design or not. Moreover, it will also enlist the technical components of the tools and might also help inform the design of a potential future solution. Lastly, it is also meant to highlight the compatibility of the mashup tools with emerging standards such as a mashup tool supporting only SOAP protocol for communication with Web Services won’t be of use to APIs communicating using a different protocol such as REST.
A. Protocols supported for communication with Web Services:
   For communication with Web Services, different protocols and standards have been defined that deal with the message passing. One of these is XML-based SOAP (originally it meant Simple Object Access Protocol but later on is considered an acronym for Services Oriented Access Protocol). A recent development in this regard is an architectural style REST (Representational State Transfer) that views a web application as a network of web pages.

B. Data Retrieval Strategy:
   One important step in mashup development is the access to the data that is intended to be mashed up. While the composition tools normally rely on widgets for this purpose, there are other scrapping tools that directly access the web pages through their DOM (Document Object Model). Apart from widgets and DOM, a third category has been included in the criteria to deal with all other possible data retrieval strategies.
### Table 3.3 - The Evaluation Based on the Classification Model

<table>
<thead>
<tr>
<th>DIMENSION</th>
<th>Mashup Design Features</th>
<th>Usability Features</th>
<th>Technical Features</th>
<th>Syndication Formats</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Mash Up Activities</td>
<td>Mashup Design</td>
<td>Usability Features</td>
<td>Technical Features</td>
<td>Syndication Formats</td>
</tr>
<tr>
<td></td>
<td>Mashup Design</td>
<td>Usability Features</td>
<td>Technical Features</td>
<td>Syndication Formats</td>
</tr>
<tr>
<td></td>
<td>Mashup Design</td>
<td>Usability Features</td>
<td>Technical Features</td>
<td>Syndication Formats</td>
</tr>
<tr>
<td></td>
<td>Mashup Design</td>
<td>Usability Features</td>
<td>Technical Features</td>
<td>Syndication Formats</td>
</tr>
<tr>
<td></td>
<td>Mashup Design</td>
<td>Usability Features</td>
<td>Technical Features</td>
<td>Syndication Formats</td>
</tr>
<tr>
<td>Yahoo Pipes</td>
<td>X</td>
<td>WRE</td>
<td>L2M</td>
<td>X</td>
</tr>
<tr>
<td>IBM SharableCod</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SOA4All</td>
<td>X</td>
<td>X</td>
<td>WRE</td>
<td>M2H</td>
</tr>
<tr>
<td>Intel MashMaker</td>
<td>X</td>
<td>X</td>
<td>WPC</td>
<td>M2H</td>
</tr>
<tr>
<td>ServFace</td>
<td>X</td>
<td>X</td>
<td>WRE</td>
<td>M2H</td>
</tr>
<tr>
<td>FAST</td>
<td>X</td>
<td>X</td>
<td>WRE</td>
<td>M2H</td>
</tr>
<tr>
<td>Karma</td>
<td>X</td>
<td>PBD</td>
<td>M2H</td>
<td>X</td>
</tr>
<tr>
<td>CMU Marmite</td>
<td>X</td>
<td>X</td>
<td>SPR</td>
<td>MED</td>
</tr>
<tr>
<td>Microsoft Popfly</td>
<td>X</td>
<td>WRE</td>
<td>MED</td>
<td>X</td>
</tr>
<tr>
<td>Vegemite</td>
<td>X</td>
<td>PBD</td>
<td>MED</td>
<td>X</td>
</tr>
<tr>
<td>Dapper</td>
<td>X</td>
<td>PBD</td>
<td>MED</td>
<td>X</td>
</tr>
<tr>
<td>iGoogle</td>
<td>X</td>
<td>WPC</td>
<td>HGH</td>
<td>X</td>
</tr>
</tbody>
</table>
### Table 3.4 - Legends for Classification Table and Mashing Techniques Table

<table>
<thead>
<tr>
<th></th>
<th>(a)</th>
<th>(b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>WRE</td>
<td>Wire paradigm</td>
<td>MED</td>
</tr>
<tr>
<td>WPC</td>
<td>Web Page Customization</td>
<td>HGH</td>
</tr>
<tr>
<td>PBD</td>
<td>Programming by demonstration</td>
<td>M2H</td>
</tr>
<tr>
<td>SPR</td>
<td>Spread sheet</td>
<td>L2M</td>
</tr>
<tr>
<td>SCR</td>
<td>Script-Language based</td>
<td>PI</td>
</tr>
</tbody>
</table>

### C. Syndication formats supported:

Different websites publish their regular content using different syndication formats. I have picked two important and common ones for this study namely Real Simple Syndication (RSS) and Atom.

### D. Light-weight Process Modelling:

A Process view helps the user in capturing the overall picture and understanding the goals/tasks visually (Hoyer and Fischer 2008). In a flow chart-like style, this specific view helps in capturing the process-oriented view of the services required or being composed for the mashup application.

### Table 3.5 - Analysis of Mashing Techniques

<table>
<thead>
<tr>
<th>Programming Needs</th>
<th>SPR</th>
<th>PBD</th>
<th>WRE</th>
<th>SCR</th>
<th>WPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOW</td>
<td>LOW</td>
<td>LOW</td>
<td>M2H</td>
<td>M2H</td>
<td>LOW</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>User Adoption Barrier</th>
<th>SPR</th>
<th>PBD</th>
<th>WRE</th>
<th>SCR</th>
<th>WPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>L2M</td>
<td>L2M</td>
<td>MED</td>
<td>HGH</td>
<td>HGH</td>
<td>LOW</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Learning Difficulty</th>
<th>SPR</th>
<th>PBD</th>
<th>WRE</th>
<th>SCR</th>
<th>WPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>L2M</td>
<td>L2M</td>
<td>LOW</td>
<td>HGH</td>
<td>HGH</td>
<td>LOW</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Power (Feature Set)</th>
<th>SPR</th>
<th>PBD</th>
<th>WRE</th>
<th>SCR</th>
<th>WPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>L2M</td>
<td>L2M</td>
<td>MED</td>
<td>HGH</td>
<td>HGH</td>
<td>LOW</td>
</tr>
</tbody>
</table>
3.4. Analysis and Discussion

3.4.1 Selection of the Tools

The tools listed in Table 3.1 that have been evaluated in Table 3.3 are selected on the basis of their relevance with the research i.e. they support the whole idea of mashup development rather than a single activity. Consequently, vertical portals and MAST are excluded due to their unsuitability to the research objective of this study. Moreover, in case of multiple solutions from the same corporation (for instance IBM offers more than one solutions for mashup development), only one (IBM Sharable Code) has been selected on the basis of criteria presented earlier in section I. Similarly, one tool is selected from platforms having a similar technique or implementation idea. For instance, as Intel Mash Maker (Yu et al. 2008) and Mozilla Ubiquity (Erlewine 2009) are both implemented as browser add-on tools, only one (Intel Mash Maker) is selected. The main theme behind the selection of tools has been to include as many features, concepts and implementation techniques as have been possibly reported in the literature. Thus, the selection of tools does not represent any intention to ignore or emphasize on any specific project/tool. A brief introduction of the tools is given below.

Yahoo Pipes provide wired compositions and supports discussion groups where users can discuss and share their newly created applications. However, frameworks such as IBM Sharable Code, SOA4All, ServFace and FAST are multi-year research projects that not only provide the front end for composing different Web Services but also comprise components that define and explain supporting process flows and ontologies. For instance, the IBM Sharable Code framework provides comprehensive solution for the creation, reuse, deployment and management of web API mashups. Similarly FAST provides a multi-phase and multi-level approach to developing mashups that are based on an ontology that can be defined within the system for each solution. Besides, Microsoft Popfly has also been included despite the fact that Microsoft abandoned the project at the end of 2008. Dapper is an enterprise tool that has now been acquired by yahoo and is used to create data feeds for the enterprises.

3.4.2 General Classification:

Tool Type suggests a mixed trend in enterprise and consumer mashups (Table 3.2). Although enterprise mashups such as FAST and IBM Sharable Code demand additional
capabilities, the fact that they are still intended to be used by non-programmers explains the inclusion of enterprise mashups in this analysis. As is evident from the Table-3.2, apart from FAST and IBM, no other tool is exclusively developed for the enterprise users. However, other commercially available tools such as Yahoo Pipes and Karma can be used by both enterprises as well as non-enterprise consumers.

3.4.3 Dimensional Analysis

A. Mashup Design Features:
Currently mashup development focuses mainly on the data aggregation or data mediation. i.e. most of the tool support for mashup development allows data mediation activity support which is consistent with earlier reported results (Namoun, Nestler, et al. 2010a). Another interesting analysis visible from Table-3.3 is the traditional tension between learnability and expressiveness or power and complexity (Fischer et al. 2004). From the analysis of different mashing up approaches, it was noticed that they render specific impacts on other factors and thus affect the whole mashup development activity. Hence, selection of suitable mash-up technique is foundational in developing an effective tool for service mashups. For instance, wiring paradigm normally implies the connection of blocks such as widgets which increase the learning burden of the user. But on the other hand, they can offer the maximum features as different widgets can be developed to encapsulate different operations and/or data. A full analysis is presented in Table-3.3. Besides, the actual mode of application also affects the challenges of mashup development. If an application can be developed and used as a browser add-on (for e.g. Intel Mash Maker), it satisfies the users’ security concerns due to no involvement of servers. This is hard to provide in case of server-based mashup tools such as Yahoo Pipes.

B. Usability Features:
Different user oriented studies have been revealing the need of more concentrated efforts for understanding user’s needs (Namoun, Nestler, et al. 2010a; Mehandjieva et al. 2010). In most cases, developing a new mashup application would require a significant manual programming effort which would deter the users from adopting the mashup technologies and tools (Daniel and Matera 2014; Rodríguez et al. 2014). It is confirmed from the analysis that almost all of the tools lack the much needed user orientation in terms of user requirements and goals. Even the tools that support users’ goals and requirements do so implicitly rather than providing any explicit task-driven or goal-
driven development approach. In spite of this analysis, although, it will be an unscientific claim to make that this is the only reason of poor adoption of mashup technology by end users in the context of a growing trend of APIs and mashup tools. However, given the theories regarding end user development (Mehandjiev et al. 15) and the programming needs of the current mashup tools, users requirements and users goals can be considered as a viable solution. Besides goals and requirements form two most important dimension of the usability standard ISO 924120 (Jokela et al. 2003). Learnability is an important feature to be considered in EUD systems and is difficult to objectively define; as the most comprehensive definition relies on subjective factors as defined by Michelson “The system should be easy to learn by the class of users for whom it is intended” (Michelsen et al. 1980). Although learning can be calculated in terms of a formula, the classification done in this analysis is on the basis of individual experiences. Platforms that used WPC as the mashing technique (iGoogle) have a better learning curve due to no programming needs as WPC belong to the class of meaning preserving EUD technique :Customization (De Souza and Barbosa 2006). However, for other systems, this was calculated on the basis of three main factors: programming needs, need of technical knowledge and visual interfaces as the first two inversely affect learnability and the last factor supports learnability.

Tutorials and system prompts in case of incompatible situations or proposed suggestions by the system were almost present in all tools notwithstanding the tutorial strategy used. Advanced User Interface (UI) generation is currently not a widely supported feature since it requires special annotation techniques and hence is supported only either main frameworks (such as ServFace and FAST) or tools that can access the web content and directly manipulate them (Intel MashMaker). It can be an important feature for EU service composition platforms as it can be used to increase the user-friendliness of the content and interfaces.

C. Technical Features:

Technical features have been included in this classification to understand the underlying technical components and range of protocols supported for communication with and within services. This dimension suggests an encouraging trend in terms of protocols and syndication formats since majority of the platforms support the latest trends, REST and
ATOM, respectively. This implies that new Web Services that are using latest technologies are compatible to be used with the majority of the platforms.

Visual components such as a process modeller are only supported by a few of the platforms (mostly frameworks). While studying and working on different platforms, this component seemed an important and user-friendly part of the mashup programming due to its visual and model-based approach.

Data Retrieval strategy is a fundamental step in mashup programming since it makes the data available to be combined through mashups. It was observed during classification that tools that used screen scraping instead of using APIs (Intel MashMaker, Vegemite, KARMA) accessed the DOM of the webpages. Another important technique was through widgets; however it is reported to have an adverse impact on learnability since knowledge about widgets is normally presented in technical terms and is hard for the user to understand (Tuchinda et al. 2008).

3.5. Discussion

The 3-dimensional model for classification of mashup tools constitutes the first contribution of this thesis. It allowed me to study about end-user orientation of current mashup tools. Several mashup tools evaluation attempts have tried to discuss different dimensions but none of them has discussed about the user’s requirements and task/goal support explicitly which form an important dimension of usability standard. Most of the evaluations end with a general conclusion of a need of a more user-oriented platform by highlighting issues such as programming needs. In an end-user programming environment where code-less development is sought to assist the end users in developing applications to fulfill their needs, the support for users tasks becomes imperative (Namoun, Nestler, et al. 2010a; Nikolay Mehandjiev et al. 2010). However, it can be seen from the evaluation that both user goals and requirements/task specifications are only implicitly supported in a couple of cases.

Concentrated efforts have been reported in this field of end user service mashups. Several big names of web industry and technology such as IBM and Yahoo have already come up with their platforms to stay in competition. However, the fact that service user mashups are still not popular among general users, calls for an in depth study of the concepts and issues surrounding this area.

The analysis presented in this chapter confirms the needs for a more user-oriented approach towards mashup development. The criteria set out in this chapter can serve as a
A Framework for Improving End-User Orientation of Service Mashups

guideline to inform the future design and serve as the methodological foundation especially the consideration of user goals and user requirements. From this perspective, it can be a valuable addition to the knowledgebase of managers and developers working in industries and academia, as it can serve as a requirement template for launching new mashup development platforms projects.

Keeping in view the general tension between complexity and learnability, which has also been confirmed from Table 3.4 and figure 3.1, future directions need to be specified. The two most discussed problems in the area of EUD of mashups are programming needs and lack of end user assistive mechanisms. I propose solutions to address both these problems in the next chapter.

The next chapter presents the proposal for motivating end users to understand the concept of services and the necessary compositions by using the concepts of abstraction and generalization. In this way the technical nature of the work involved in developing end-user service based applications and the possible low learning curve due to high set of features (due to complexity vs. learnability trade-off) is made reusable by using visual models that help end users operate on a generic layer by using domain theory constructs (Sutcliffe 2006). Another potential solution applied is an explicit, goal driven development of mashups, since goals are a technical yet natural way of deriving a system based on the underlying requirements of users (Van Lamsweerde 2001). Although most of the work done on goals pertains to requirements engineering, given the benefits of goals and centrality of user requirements during developing mashups, the use of goals for developing situational applications has served to be a promising step in improving user-orientation of web application development by composing data and/or application logic from disparate sources. This idea can help improve the user-orientation of the mashup systems since end users can relate to the service descriptions by mapping them with their goals and requirements (if supported explicitly in the system) which is believed to maintain their interests as well during the whole technical activity of composing an application. Coupled with visual models and suitable mechanism to represent and manage domain knowledge, the idea of goals leading to mashups can be a reasonably practical solution. The detailed working of these components is explained with motivating scenarios for an initial validation. A more detailed evaluation of the proposed goal-inspired mashup application development is done in chapter 5.
Chapter 4

4. The Goal Oriented Mashup Development (GO-MaDe) Framework

4.1. Redefining the Mashup Development Lifecycle

This chapter discusses the GO-MaDe framework which encompasses the conceptual model, design and working details of the framework including one of its components: the spiral process and summarizes the technical underpinnings of the functionalities and the components of its general architecture. According to the summary of previous chapter, end-user mashup development environments should be designed to be more user-oriented and explicitly support end-user requirements and goals. In order to realize this idea of goal-oriented mashup development, I am presenting a framework that comprises of different components: a redefined mashup development process, a method to assist end users in modelling their mashup applications without the need of complex formal specifications, and tool architecture to enable the creation of mashup models required to derive services from goals. In short, I am proposing to improve the user orientation by incorporating user goals and user requirements in a redefined mashup development process. The details of the proposed framework, its components and the underlying theory and motivations are discussed in detail in this chapter.

Figure 4.1 – The Goal Oriented Mashup Development (GO-MaDe) Framework Components
I propose GO-MaDe framework to address the concerns of lack of end user orientation in the current mashup development approaches. The framework is designed to place the end user at the centre of the process of mashup development and is inspired from the principles of meta-design. In my framework, end user goals and requirements are specified explicitly and driven systematically until the end user is able to select from a set of candidate services and compose an application. For this purpose, the GO-MaDe framework enables a visual model-based transition from domain-independent goal models to domain-specific goal models consequentially leading to final set of services and data sources that are required to be composed for developing the required mashup application to address the end-user’s need(s). For the purpose of this research, I am defining the term domain dependent and domain independent as defined in The Domain Theory (Sutcliffe 2002). Generally in software engineering, the term domain refers to the application areas (Sametinger 1997) such as airline reservations systems or financial applications. Domains are particularly helpful in improving reuse since for a class of similar systems I can identify the common properties. However, the scope of domain can be variable. The concept of scope helps in identifying the concepts of domain independence and domain-specific within the proposed GO-MaDe framework. The domain independent goal graphs are comprised of templates that are based on the concept of meta-domain of the domain theory and which are not physical domains. The typical example of a meta-domain is a borrowing system which can be instantiated into a physical domain by specifying the details such as cars for a car hire domain or a book or a video disk for a library domain. Hence, the framework has been developed to be customizable to support different application domains. It makes use of templates and meta-models based on goal-modelling principles and reusable domain theory constructs that can be instantiated by plugging the right domain concepts from the respective domain ontologies. I have proposed the GO-MaDe framework :(i) to shift the ownership of the problems to end users by redefining the mashup development process in an attempt to provide the end users with necessary analytical and design space to explore the problem area. It serves (ii) to provide different levels of support to End Users based on their skills in developing the situational applications by representing their mashup problem in the form of goals that eventually get translated into WSMO goal specifications. It also tends to (iii) assist End Users in gathering, organizing and maintaining their reusable mashup templates by exploiting existing knowledge of domain ontologies and generalized tasks of Meta domains as explained in the domain theory. These models can be reused and modified at a generalized level by
making use of goal-oriented model-based visual mechanisms that hide the un-necessary
data and process complexities of underlying functionality.

This chapter is organized as follows. Section 4.2 reiterates the need of a new approach and
presents the EU challenges that GO-MaDe is intended to address which is followed by the
process specifications of my framework in section 4.3. It presents the details of different
phases of the underlying spiral model that serves to improve the emphasis on the end-user
mental models as compared to the direct technical exposure of end user to the solution
space with all the technical intricacies.

4.2. Why a New Framework?

The current trend of consumer web is synonymous with Ben Shneiderman’s claim
(Shneiderman 2003):

“The old computing was about what computers could do; the new
computing is about what users can do”

The number of services is on the rise (Namoun et al. 2010), and can be a driving force for
dependants’ empowerment from mere consumers of web to become active producers of
services. Currently, the literature hints towards an underlying problem regarding existing
tools. According to Cappiello et al., the research has mainly focused on enabling
technologies and standards with little attention to easing the mashup development
process(Cappiello et al. 2011). This has led researchers to conduct different user studies
and propose user-centered mashups (Väänänen-Vainio-Mattila and Wäljas 2011; Mikkonen
and Taivalsaari 2010). Hence, it can be deduced that mashup development, specifically
among end users, does not represent a success story in spite of the high interest of end
users in the idea of developing novel applications in a mashup-style (Namoun, Wajid and
Mehandjiev 2010b). Besides, a group of researchers is also emphasizing on the need of
bridging the gap between software engineering and web engineering software
development practices (Mikkonen and Taivalsaari 2010). Different user studies have
suggested that the main problem is the lack of focus on the human factors of mashup
development as most of the research has been carried out to deliver and improve the
technical aspects of developing applications by combining services(Namoun, Nestler and
Angeli 2010; Mehandjiev et al. 2010; Namoun et al. 2009). User studies into this research
area suggest a more comprehensive development environment with a thorough support,
which should be presented to the users in a progressive way (Koschmider et al. 2009; Namoun et al. 2010). Keeping in view the state of the art of mashup development and studying about different mashing techniques (Blackwell 2006), I decided to provide the extended support for mashup development activities. The purpose of this research has been to revisit the problem as well as solution space since proposing a new tool using the existing (or even proposing a new) mashing techniques - such as scripting, visual languages, wiring/piping, spreadsheets, programming by demonstration - would only contribute to the existing research into the tools and technical aspects. Hence, this research aims at bringing forward an important aspect of mashup development: exploring and understanding end-users’ expectations and requirements while composing an ephemeral application.

The major objective of the framework defined in this thesis is to provide user-friendly techniques to elaborate the EU’s objective, by applying semi-automated techniques in a collaborative way, into more concrete, ready to be designed mashup requirements and further assist the assignment of these concrete requirements to target Web Services that satisfy the main objective and hence the sub-goals.

Different approaches focusing on goals and their specification, reduction and discovery have been introduced. The originality of this approach lies in the following aspects:

1- The proposed framework (GO-MaDe framework) does not only provide technical details for composing services but provides detailed process-level guide. The spiral process model attempts to breakdown the mashup development process and identifies and defines the phases that have not been previously discussed by the researchers’ community. For the purpose of this framework, I have redefined the mashup development process to incorporate the concept of end-users’ expectations and requirements and identify it separately from the actual composition of services. The composition of services refers to the searching and integration of disparate services to form a composite service and is an integral part of SOA paradigm (Singh and Huhns 2006). In the context of mashup development in which end users compose different services, we hereby emphasize that end users expectations should also be explicitly considered and specified and hence have included service compositions as a phase of our proposed process model.
2- The second component of GO-MaDe framework is the method for mashup knowledge acquisition KAReM (Knowledge Acquisition and Representation in Mashups). KAReM attempts to achieve goal discovery, service discovery, service dependency and goal specifications in a top down fashion starting from the main EU’s goal and deriving the final goals that can be translated into services. Hence, the proposed method is unique in the sense that the focus of goals is not only compositions but also the goal specification that are composable.

3- The typical mashup development scenario is listed below in figure 4.1 and it helps in understanding the proposed framework.

- The EU wants to develop a mashup with an objective about the proposed mashup application.
- The EU creates a new mashup project
- The system in a bid to help EU specify and explore the requirements asks the user to input some information.
- System matches the information and recommends a suitable generic task template.
- EU starts to fill in the template and gives the template a final form of a goal graph until all parameters are elicited.
- EU moves to translation phase and completes the generalized task GG which gets translated into WSMO goals
- The WSMO goals specifications are later on moved to the composition engine for services search and composition.

Figure 4.2 - Mashup Development Scenario Using the Proposed GO-MaDe Approach

The method employs the AND/OR goal graphs (AOGG) to be developed collaboratively by the system and the EU. The AND/OR graph were introduced in the context of search problem by Nilsson (Nilsson 1971). The proposed method employs AND/OR graphs to reach a specification of goals leading to concrete services and/or data sources and information necessary to execute their compositions. The AND refinement link between two sub-goals at the same level implies that all these sub-goals need to be satisfied in order to achieve the corresponding parent goal. Whereas an OR refinement link suggests that satisfying one of the sub-goals is sufficient for satisfying the parent goal. However for
supporting the specific needs of mashup development scenario the OR contributions are recommended to be implicitly handled resulting in the final goal model to be completely an AND contribution goal graph. However, in other intermediate goal models, AND/OR refinement shall be used and suggested via notation. Hence the goal graph (GG) containing the final goals with candidate services will be considered as an AND contribution only GG.

In GG, the end user specifies the operations through a query reformulation process and the attributes are provided based on domain ontology recommendations and previously developed mashups. For structuring the goals, domain ontology and verb ontology are used in the form of structured natural language.

The method takes the initial input by user and later on works in close collaboration with the EU to develop the goal graph. The entire GG can be seen as the solution process, however the final composition information is entailed in the atomic/terminal goals (or the leaf nodes) of the GG. The entire composition specification can then be used as input to any tool that takes WSMO goals as input to give the required mashup.

The capture of requirements, their structuring and their operationalization is the basis of requirements engineering (Kavakli 2002). Mapping these phases onto mashup development identifies two important aspects:

Since, the main advantage of the mashup development process is the quick lifecycle; my proposed approach tends to support this agility. Hence, two distinct activities that can be regarded as pertaining to analysis and design respectively for conventional software development have been merged in a single interface to support the agility: the structuring of requirements in the form of goal graph and the assignment of requirements onto suitable Web Services, APIs and data sources. The proposed method adopts a goal model that is realized by an attributed GG and tends to merge the requirements and design to cater for the WHAT, HOW and WHY questions all in a single developmental phase. The approach is collaborative instead of fully automated to derive the requirements, however the generation of design of mashup in the form of detailed goals specifications, ready to be composed is proposed to be automated.

The goal model to be adopted is designed to be compatible with WSMO goals and hence it can be deduced that the model takes inspiration from the WSMO goal model. The central concept of the entire method is the concept of the goal, which translates the objectives of
the client EU and is meant to be achieved by a combination of different data sources. The data sources may or may not be known in the beginning of the mash-up development process and the proposed method serves to elicit and specify the goals until the concrete services/APIs and data sources are reached. For the goals to be properly specified in the context of mashup development, I list down the major requirements of the goal descriptions:

1- The goal model should be developed collaboratively and the process should be semi-automated to work as collaboration between EU and the system.
2- The goal descriptions should be expressive enough to be formulated on the knowledge/intention level rather than on the technical level.
3- The goal model should not be a complex and formal one due to the low technical knowledge and background of the EU in focus. Rather, the goal model should make use of the semi-formal and structured language notations presented in a graphical interface.

4.3. Components of GO-MaDe Mashup Development Framework

The key components of the GO-MaDe framework are as follows and the detailed discussed of each is presented in the following sections. The method uses WSMO goal specifications for formal representation of the goals as defined by Stollberg and Norton extended goal model for the specification purpose where required (Stollberg and Norton 2007). The main reason for using the WSMO extensions proposed by Stollberg and Norton is the requirements analysis performed which emphasizes on the client-side goal formulation and providing visual modelling aid for clients who wish to describe their objectives that can be later mapped to the Web Services in the services oriented architecture.

4.3.1 Mashup development Lifecycle and process model

The existing mashup development process which is underlining most of the tools and their underlying methods does not consider end users’ requirements explicitly. Different tools provide a front end that allows users to structure different widgets and wires to develop a mashup such as Yahoo Pipes (www.yahoopipes.com). There are other tools that employ other techniques such as programming by demonstration and work by using the steps recorded earlier in a similar task. Notwithstanding the mashing techniques used, mostly these tools assume that the end users requirements are implicitly translated into the final mashup by using the
technical components of the interface. I, however, argue that the non-IT background of end-users require an explicit emphasis on the users’ requirements.

This should, then, be followed by a translation driven by visual models, into concrete services. Hence, GO-MaDe framework requires a redefinition of the mashup development activity. For this purpose, I have redefined the mashup development life cycle (fig. 4.2) and proposed a spiral process model (fig. 4.3) that introduces different phases of the mashup development process in contrast to the single composition phase underlying existing tools. The framework employs the principles of “meta-design” as the premises for the new process and uses different goal templates, goal graphs and domain ontologies for its working.
4.3.2 Knowledge Acquisition and Representation in Mashups (KAReM): A Domain Theory Based Goal Modelling Method

The domain theory based goal-driven modelling method has been developed to incorporate the benefits of goal-based mechanisms to systematically expand the end users’ objectives in a visual and user-friendly manner. The goal modelling method uses direct manipulation techniques for creating and modifying the goal models. The direct manipulation of the method operates on data structures representing domain-independent and domain-specific models. The organization, working and constraints of these models are governed by goal meta-models and goal templates. The goal meta-model (fig-14) and goal templates define general constraints on the goal model and is domain-independent and is intended for end-users with vague requirements of their mashup applications. Whereas generalized goal templates are for end-users with concrete requirements and are derived from domain theory constructs of generalized tasks. The generalized goal templates cannot be regarded as domain-oriented; however they help specify the genre of the template and are more detailed. For the ease of reference, the meta-models based on domain theory generalized tasks are known as Meta-domain goal templates. Although the proposed framework supports reusability by creating and reusing new templates, however it is governed by the constraints of the goal meta-model. The GME meta-models and templates discussed above specify the semantic and syntactic constraints of the resulting goal models.

4.3.3 Domain ontology

The domain constraints are defined by the domain ontologies. The meta-domain meta-models are finally instantiated for the target domain by using the domain ontology concepts.

The final and completed goal graph is thus achieved by coupling the meta-domain Meta models to domain concepts as described by the respective domain ontologies. Ontologies make up an important part of the final mashup application but developing ontologies for a particular problem domain is a time-taking task. Hence, I rely on importing ontologies from the online ontology libraries. The final instance of a goal graph will specify the concrete requests in terms of real APIs or data sources with the necessary composition information. However, the actual composition is beyond the scope of my research thesis.
4.3.4 WSMO Goals

For the purpose of supporting end-to-end mashup development, I am proposing a WSMO Goal specification generator that can be used to generate the specifications of the final concrete goals and contain the data sources binding for each goal.

4.4. The Spiral Mashup Development Process Model

In conventional software development, a software process defines different stages, their order and the transition criteria to move from one stage to the other (Boehm 1988). The software process, hence, is considered as a critical factor in the final software product (Acuña and Ferré 2001). The concept of mashup development life cycle and consequentially of the mashup development process in web mashups has not been researched explicitly. I believe the main reason behind this research gap is the underlying assumption regarding mashup development. Most of the researchers have equated mashup development with service compositions. Mashup development is a Do-It-Yourself approach that aims to empower EU to create their own mashups which requires that the resulting expenditure of time and effort should be reasonably below than that of traditional compositional development in conventional SOA (Lizcano et al. 2016). While in traditional Service compositions, advanced programmers focus on integrating and organizing services at the back end on the technical layer, mashup development requires non-programmers to reap the benefits of service compositions on the front-end layer by employing flexible and user friendly interfaces. Resultantly, mashup tools should focus on composing services that lead to a novel application that is capable of serving the situational need of the end user. Although, it is understood that mashups represent a fast development time and hence the conventional software development phases cannot be and should not be replicated fully while talking about the former, I believe that not considering the presence of the activities pertinent to users’ analytical and design thinking is the main reason of lack of end-user orientation in mashup development. It is to be noted, that most tools (for instance Yahoo Pipes\(^{21}\)) present the end users with an interface that implicitly is providing some design space in rapid iterations of design and implementation. These design activities vary depending on the underlying mashing techniques. For instance for wires and pipes based tools, this design activity involve

---

\(^{21}\)Http://www.yahoopipes.com
resolving the wires and pipes between widgets. However, the analytical (pertaining to user requirements and expectations) space is altogether absent in the present day mashup tools. I also argue that although the needs of the end users can be regarded as concrete due to the idea of web mashup, the needs are not always fully known and complete. Different characteristic examples of mashup development can be cited here. Apart from the need-based mashups, it is also reported in the literature that software service compositions are not only need-based but can also be used for getting out the creativity of end users (Mehandjiev et al. 2010). This, more than ever, leads to the need of a software process that provides an explicit, though intertwined with design and development cycle, analysis phase to help in achieving the completeness of the end user requirements. Lastly, I believe that the proposed process can also contribute towards the enterprise mashup development since developing mashup by mashing up resources from service repository apart from the scenario where expert developers create mashups and end users only use the developed mashup (Cappiello et al. 2011).

Keeping in view these arguments, I have proposed a spiral process model by phasing the mashup development life cycle. Following sections discuss the details including characterizing of the end user.

4.4.1. Redefining the Mashup Process: A Spiral Mashup Development Process Model

One of the attractive features of SBA for EU is its light-weight and agile development style. A mashup development activity normally comprises of the “Design-Build-Run” cycle as can be deduced by analyzing the underlying process model of the mashup platforms available. For instance, Yahoo Pipes (www.yahoopipes.com) provides a single integrated screen for designing and running the composition application. This model assumes and implies that end-users requirements need not be managed through a dedicated phase. This assumption seems to align well with the quick development cycle of a situational application which is developed by composing different services and/or data sources together. At the heart of this assumption lies another assumption that end-users with limited technical/programming knowledge and having no prior knowledge of software development activity are in a ready-to-compose position at the onset of their mashup endeavors. I tend to argue to differ from this position during the course of this thesis.
Researchers have also argued regarding the web applications requirements analysis, that website end-users are likely to have requirements clarity problems owing to an imprecise objective (Bolchini and Mylopoulos 2003). Bolchini and Mylopoulos (2003) describe three main problems with the website requirements: 1) Ill-defined goals, 2) Missing goals and 3) Restriction on exploration of alternatives. Mashup applications tend to face these similar difficulties exacerbated by no or little IT knowledge required to produce meaningful applications. In contrast to taking mashup development as a monolithic activity I propose to breakdown the mashup development into 4 distinct phases. Thus, I propose an updated process model for the standard mashup development project. Figure 4.3 presents the detailed lifecycle expressed as activity diagram and figure 4.5 presents the actual process model with the main phases involved in the proposed process. The KAReM method discussed in chapter 5 follows this spiral model to guide the development of a mashup application.
It involves acquiring the initial requirement-set of the end user preferably through a set of system dialogues.

Progressive enrichment of the goal graphs by adding goal attributes (inputs and outputs parameters).

Mediating the candidate services and composing them.

Executing involving testing and deployment.
4.4.2. Detailed Activity Diagram

The figure (4.3) above presents the activity model for the proposed spiral process model. As can be anticipated from the name, the proposed process model takes its characteristics from the Boehm Spiral model of traditional software development, one of the most practical and accepted methods in software engineering. However, the area of focus and hence the scales of time and needs are completely different since traditional software development projects’ timelines are in the months or years whereas the mashup development projects are typically expressed in terms of days or at most weeks. The distinguishing feature of the proposed process model from the current trends in mashup development is its template-driven (Which in the case of my thesis are goal-based templates), iterative (due to goal models) and incremental (reusable) nature.

My model is template driven and incremental since it takes the life cycle of mashup development beyond a single mashup application and isolated end users’ needs. It does not take a single mashup into account and takes mashup development as a continuous incremental activity. Templates are the building blocks of the process model and the first step in generating the mashups that drive the activity of deriving services from the objectives or goals at the beginning of the mashup development process.

The process is iterative and incremental in nature since within each activity or phase, goal templates are developed iteratively, typically focused on different phases of the mashup development and at the end of each cycle, the mashup is incrementally enhanced in terms of features and addressing additional end users’ needs.

The proposed process model is composed of phases and cycles. Each cycle of the spiral consists of 4 phases (conception, translation, composition and execution) and at the end of each cycle, a new tested and deployed mashup is released (if the mashup needs not to be published depending on the intent of the end user, testing and deploying can be skipped). The enhancements in the mashups take place through templates which can be detailed and extended. In the context of my thesis, I am proposing to extend the template by adding additional goals from the library of goals, which bring extended feature set to the new mashup.

The activity diagram explains the process with the detailed workflow.
Capturing the initial requirements of the mashup makes the conception phase of the mashup development which can be mapped to early requirements of the traditional software development. It consists of specifying the basic mashup information including the name, purpose, type, genre and additional comments regarding the details. The end user can also specify the services in the beginning depending on his preferences. If the end user starts a new mashup project and selects the template to iteratively develop a goal model, it will be considered as a new start of the new spiral in the process model. However, if the end user continues with the filled template and updates an existing filled template with parameters and services specified to add a new goal component, it will be considered as the next cycle of the parent mashup.

Figure 4.5 - A visualization of the conception phase

The translation phase is oriented towards the elaboration and refinement of the early mashup requirements identified in the previous phase using appropriate goal based templates in a top-down fashion. The goals are considered a natural mechanism for
derivation of requirements. Hence the goal hierarchies are believed to be a user friendly way of guiding users from their objective to the final candidate services. The translation phase for cycle 1 will start from an unfilled template. However for cycle 2, the end user may want to start with the filled template and modify the goal components. The workflow for transition phase consists of instantiating the goal template by filling in the parameters followed by selecting the candidate services or data sources.
First phase of the proposed process to determine and define the objective, type, and potential data sources of mashup. Also involves the domain-independent goal graph development.

**Input:** Basic Mashup Information  
**Output:** Initial GG

Elaborate the objectives, requirements, constraints and alternatives while identifying the matching services that meet the requirements and develop the domain-oriented goal graph based on generalized tasks. **Input:** Early GG, **Output:** WSMO Goal Specifications

---

**Figure 4.6 – Spiral Mashup Development Process**

**Conception**

**Translation**

**Goals to candidate Services**

Goal Graph 1, Template 1. Incremental goal graph development based on the template.

Goal Graph 2, Template 2. Goal Graph development based on customized needs.

**Basic Mashup Profile**

Mashup Type and Genre

**Services Composed**

**Execution**

Debug and if required deploy the final composed solution.

**Composition**

Finalized the services and develop the mashup by composing the services identified earlier.

**Situational Need of End User**

At the end of first spiral, First Version of mashup is published. Example: Basic San Fran. Craigslist.com email list with no maps.

At the end of Second Spiral, Advanced Version of mashup is published. Example: Craigslist.com with maps listings.

The Spiral continues
Mashup development is concerned about making the disparate data available on the Internet more meaningful according to one’s needs by integrating it. The integration has to be user friendly based on visual techniques to facilitate end-users who have little or no IT knowledge. I argue and propose that though mashup development is done in a rapid-style and is specific to a short-lived, ad-hoc need of an end-user, the mashups can be abstracted and reused using a template-based development and evolve spirally. For this purpose, consider the following scenarios.

Different iterations and rounds of the spirals can be explained using the help of an example. I will apply spiral model to Craigslist.com, one of the very first and commercially successful mashups. Craigslist was originally developed in 2005 as a San Francisco mailing list featuring local events in San Francisco Bay area. Now, it is an expanded commercial online business and provides classified ads related to different categories including but not limited to jobs, housing, for sale, and items wanted. The evolution of Craigslist could be cited as a good example of how template-based development can be used to facilitate enhancements into an existing mashup. Currently according to Craigslist.org, Craigslist over 700 cities in 70 countries have Craigslist sites. Let’s suppose in the first iteration, Craigslist used a basic data mashup template to create basic version of the site to list the events in San Francisco bay area. Next, housingmaps.com decides to publish the events on the map according to their locations (Housing maps is the first mashup that used the Google maps API but is now closed since Craigslist has added the maps component within their site). Instead of developing the mashup from scratch, the template for the Craigslist can be reused and customized to add the map component in the second spiral leading to the second round of spiral.

Another example is the Trendsmap mashup (Trendsmap.com). Originally Trendsmap listed famous twitter trends classified based on their locations and presented using Google Maps. It could be listed as the first spiral of a presentation mashup with data selection. Lately, it has been integrated with YouTube videos based on the locations on the map which could be marked as the second spiral of the mashup giving a YouTube trends map at the end of spiral 2 (an example of YouTube Trendsmap: http://infosthetics.com/archives/2013/05/youtube_trends_map_explore_the_most_popular_videos_by_location.html).
This model is inspired from meta-design principles, where end users are given an analysis and design space to involve them in the design of the artefact and the collective community is used to use, reuse and enhance the artefact.

### 4.4.3. Phases of the Spiral Mashup Development Process Model

The redefined mashup development process consists of four phases namely conception, translation, composition and execution (fig. 2, 4.6a). Based on these phases, a spiral mashup development process model is proposed (fig. 4.5) reflecting the incremental nature of the mashup development.

#### A- Conception

This phase may be regarded as the ideation phase and should deal with the idea generation and clarification in terms of early requirements and goals in form of domain-independent Goal Graph (DIGG). In traditional software development it can be regarded as the early requirements phase. In the context of mashup development, this phase deals with the purpose of the mashup and aims to clarify and elicit the initial set of requirements from the end-user encompassing the situational need(s). The domain independent goal graphs are simple and/or goal graphs with semantic and syntactical constraints that govern the process of derivation of goal models. The conception phase helps the end user in understanding the problem and through a series of user-system dialogues help in gathering the initial set of requirements for the target application.

#### B- Translation

This middle phase help end users translate their need into models which could be further translated into candidate services/data sources. Hence, this phase undertakes an important transition from goals to services. Practically, it is designed to help the end-users transfer their ideas onto a medium for communication and specification purpose which could be read by both humans as well as could be suitable for further processing by the computers. The idea of empowering the end users to specify and design their own needs is inspired from one of the meta-design principles which states that users should be involved into the project at design time to give them the ownership of the problem (Fischer and Giaccardi 2006). This phase corresponds to requirements modelling/early design (since the candidate data/service components will be identified) in the traditional S/W engineering and aims to model the initial requirements from the end-user. The end product of this phase shall be a domain-specific goal graph. The terminal goals in
the goal graph will be able to get transformed into Web Services Modelling ontology (WSMO) goal specifications for the ease of composition phase. The goal modelling for translation phase is carried out at meta domain as well as domain level by the help of domain theory templates and domain ontologies. The goal derivation process is required to be elaborate enough to help in assigning the candidate data sources/APIs to terminal (or atomic) goals.

C- **Composition:** This phase involves selecting the final service/data components from the candidate services/sources and followed by the aggregation of these data sources. This phase is the only explicitly defined phase in the mashup development lifecycle as is evident by most of the contemporary tools and involves the composition of data sources and APIs needed to fulfill the end user situational need. In this proposed process, this should also accomplish the resolution of any/all conflicts between the finally selected service/data components. Ideally, the composition level decisions should be handled by a composition engine such as SOA4All (Domingue et al. 2008a) to facilitate the end users. It is to be noted that here that this proposed framework only provides the handling of first two phases i.e. conception and translation.

D- **Execution:** SBAs are being reused; famous examples are Craigslist\(^{22}\) and Housingmaps. This requires the applications to be tested and debugged before being deployed and reused. Quality for the mashup development is being considered as an important aspect recently and models for ensuring quality are being researched (Daniel and Matera 2014). Hence, I propose that this phase tests the data and performance of the composed application for consistency with the original intention of the EU. Lastly, it involves the final building of the application after debugging and may involve deployment depending upon the application type, intention of the EU, quality and utility.

The mashup development process begins with the identification of the end user need. The end user can have a very clear objective but the final services and data components might be vague. For instance an end user developing a personal mashup might change his requirements given the new and exciting data sources available such as an end user developing a personal sports site with the original intention of including the images and

\(^{22}\)https://www.craigslist.org/about/sites
\(^{23}\)http://www.HousingMaps.com
news might also want to include videos and Facebook/Twitter statuses. In the first phase conception, requirements elicitation is carried out and is the early requirements phase. This phase is completely ignored in the existing tools. Once the basic information is gathered, I have proposed the end user should be guided to the next step of translation. It is proposed that this transition should be intelligent to analyze the set of early requirements. For instance, if end user is presenting a very vague set of initial requirements with no knowledge of the target services, the end user should be given a mechanism to explore the requirements further. On the other hand, if the end user is advanced and knows the type of mashup and the final services, a more specialized interface is recommended. This is in line with the Lieberman’s recommendation of flexibility and helps in achieving the “gentle slope” (Lieberman et al. 2006). This marks the end of the conception phases. The iterations in this phase is achieved by giving the end users the facility to change their mashup type and services selection.

Once the end user has specified the initial set of requirements, the next step is presenting the analysis and design space preferably through easy-to-understand visual models to let the end user translate the domain independent objectives and subsequent goals into requirements that can be mapped and translated to concrete services/ data sources hence the name translation phase. I propose the domain ontologies to be used in this phase. This phase can be regarded as the requirements modelling phase in conventional software engineering terminology. The end of this phase is marked by the identification of candidate services and data sources.

This marks the onset of the composition phase which is employed by most of the tools and gives end users the ability to compose the selected services. The main product of this phase is a data/service composition with no compatibility issues. Since the services and data sources are disparate; there should be a mediation mechanism to resolve the conflicts.
Finally, the composed mashup is tested and depending on the discretion of the end user, deployed and published on the web. Testing is necessary to ensure that these mashup applications available on the web are error-free. This final phase should also include the facility to generate and customize interface(s) of the developed mashup.

Keeping in view the quick development lifecycle of a typical consumer mashup application, it is proposed that this should be supported by user-friendly interfaces, easy-to-understand tutorials and should have reusability support. The spirals of the mashup indicate that the mashup designs could be reused and updated (fig. 4.6b). The spirals of the mashup development process are understood to be quick as compared to a traditional software development project. For instance, if the analysis phase of a student registration system lasts for few months, the conception phase of a mashup application might last for few hours to a couple of days depending on the level of complexity of the mashup and the IT knowledge of end user.
i. **Examples**

a. **Scenario-1**  
Consider mashup for creating a personal news page as an example scenario to understand the proposed process model. The main objective of the end user is to create a personal page with news feeds, images, and videos about South Asia. The attributes mentioned in table 4.1 are assumed and arbitrary albeit logical. The motivation and priority are assumed to play an important role in the mashup process since they will identify the urgency of the mashup. These attributes can also help in creating a library of the mashups to be reused in future. At the start of the process, table 4.1 describes the snapshot of the mashup. The knowledge, genre and target services help identify the end user’s knowledge of the mashup that needs to be developed as well as in the complexity of the next phase. This phase should ideally be accomplished through system dialogues that are well-annotated and provide help services such as on various mashup properties.

<table>
<thead>
<tr>
<th>Objective</th>
<th>To create a personal page about south Asian country news updates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>To receive news from the native region while studying abroad.</td>
</tr>
<tr>
<td>Type</td>
<td>Data mashup</td>
</tr>
<tr>
<td>Genre</td>
<td>Information Mashup</td>
</tr>
<tr>
<td>Target services</td>
<td>Unknown in the beginning</td>
</tr>
<tr>
<td>Priority</td>
<td>Normal</td>
</tr>
<tr>
<td>Motivation</td>
<td>Personal</td>
</tr>
</tbody>
</table>

Table-4.2 presents the summary of the translation phase. The next phase involves deriving the initial requirements given in table-2.1 further. The main characteristics of translation phase:

- The initial requirements are further derived into more concrete requirements.
- Domain ontology has been specified and will be used to get more requirements
- The specific data sources have been recommended and selected.
Table 4.2 - The Translation Phase

<table>
<thead>
<tr>
<th>Objective</th>
<th>To create a personal page about south Asian country</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motivation, type and Genre</td>
<td>To receive news from the native region while studying abroad, Data mashup, Information</td>
</tr>
<tr>
<td>Domain Ontology</td>
<td>News ontology</td>
</tr>
<tr>
<td>Requirements</td>
<td>Get the News headlines and relevant images and videos from different news websites. The news should include the headlines, details and relevant images and videos.</td>
</tr>
<tr>
<td>Requirements</td>
<td>Unknown in the beginning</td>
</tr>
<tr>
<td>Data Sets</td>
<td>TheNews.com</td>
</tr>
<tr>
<td></td>
<td>BBCNews</td>
</tr>
<tr>
<td></td>
<td>TheNation.com.pk</td>
</tr>
<tr>
<td>Candidate Services</td>
<td>RSS News feeds from</td>
</tr>
</tbody>
</table>

The next step is composition of these services in the process to make a mashup. Once the candidate services have been identified, the final service selection starts based on the compatibility of these data sources. Final compatible services are composed to get the desired mashup application.

In the final step, the mashup is tested for any bugs and is published on the web.

4.5. Discussion

The spiral process model is the first main component of the proposed goal oriented framework - GO-MaDe - that serves to improve the end user orientation of end users while developing service mashups and the second main contribution of this thesis after the classification framework. It allows me to advance towards the accomplishment of my research goals by addressing research questions 2 and 3 by helping me in addressing challenge 1. The spiral process model provides end users with a model that is centered on and controlled by the end user to specify their expectation and design their own mashup applications followed by their composition and execution. This chapter discussed the motivations behind a new framework, its theoretical underpinnings and constituent components. The chapter later discussed the first component of the framework which is a

24 Fhttp://data.press.net/ontology/asset/
A Framework for Improving End-User Orientation of Service Mashups

spiral process model in the details along with the validating scenarios. The chapter discussed the process model in detail and how it (i) constitutes an end user centric model of mashup development to improve end user orientation and help address the technical barrier challenge by applying meta design principles; (ii) empowers end users by providing them with the analysis and design space through interactions driven by visual models (iii) provides a new concept of continuous, spiral and incremental development of mashups. Therefore, in contrast to existing mashup application development monolithic and ad hoc process, the proposed spiral process model helps address.

The next chapter presents the second main component of my proposed GO-MaDe framework: the KAReM method. This method is my proposal of a goal based method that helps end users to model their mashup knowledge in the form of goal hierarchies and using these models to derive their service compositions using domain theory abstractions.
Chapter 5

5. Knowledge Acquisition and Representation in Mashups (KAReM): A Domain Theory Based Goal Modelling Method

5.1 Overview

This chapter discusses in detail the second component of my Goal oriented mashup development (GO-MaDe) framework: KAReM; a method for acquiring and modelling the knowledge related to mashups. KAReM is developed based on the idea of goal modelling and uses Domain theory for developing the constructs that support reuse. Specifically, it can be regarded as one of the pioneering works of service based compositions in the context of mashup technology and can be placed in a broader area of Goal-oriented service engineering.

The Domain theory based goal modelling method KAReM of the proposed framework – GO-MaDe - is derived from the mechanisms of goal-based methods with underlying details and rationale provided by the Domain Theory (Sutcliffe 2002). It provides the end users with a semi-formal visual modelling environment to creatively explore and specify their mashups requirements. The proposed method is aimed at guiding the EU to his/her mashup application using the DT templates modelled as AND/OR goal graphs. The KAReM uses goals models and goal templates to derive the goals-to-services transition. The method achieves this transition via different goal-based sub-models that are governed through inter-model consistency rules. The end user is guided to the right goal template based on the initial knowledge of requirements of the end user that defines the readiness of end user in accomplishing the task. This fulfils one of the guiding principles of end user of providing different levels of complexity based on the level of end user (Table-5.1).
Table 5.1 - KAREM supporting different Levels of End Users and Complexity of Goal Templates

<table>
<thead>
<tr>
<th>Type of End User/Complexity</th>
<th>Goal Template</th>
<th>Domain Theory Construct</th>
<th>Syntactic Rules of Goal Template</th>
<th>Semantic Rules of Goal Template</th>
<th>Resulting Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic End User</td>
<td>Create Generic Goal Graph</td>
<td>Generic Tasks</td>
<td>Basic Mashup Templates based on type of mashup</td>
<td>Basic Goal structure and Generic Task</td>
<td>Basic Goal Graph</td>
</tr>
<tr>
<td>Intermediate End User</td>
<td>Create Hypo-domain Goal Graph</td>
<td>Meta Domain Generalized Tasks with No Domain Ontology</td>
<td>Meta-domain goal templates</td>
<td>Basic Goal Structure and Generalized tasks</td>
<td>Generalized Domain-Independent Goal graph</td>
</tr>
<tr>
<td>Advanced End User</td>
<td>Create Domain Oriented Goal Graph</td>
<td>Meta Domain Generalized Tasks with Domain Ontology</td>
<td>Domain-Oriented goal templates</td>
<td>Basic Goal structure, Generalized Tasks and Domain ontology</td>
<td>Generalized Domain oriented goal graph</td>
</tr>
</tbody>
</table>

The goal models are iteratively developed based on the rules defined in the respective goal template. The method supports three goal-based templates namely, generic goal graphs, generalized meta-domain goal templates and domain-oriented goal templates in the increasing order of complexity. The basic generic goal template is a basic goal-based derivation template that is governed by the syntactic rules as defined by the type of mashup (presentation or data mashup). The semantic rules of generic tasks are defined by an abstract collection of domain theory constructs known as primitive or generic tasks. These generic tasks are basic abstract reusable tasks that are domain-independent and achieve single goals. They have a one-to-one relationship with goals. The next and comparatively more complex goal graph template is based on the domain theory defined generalized tasks. These generalized tasks are composed of generic tasks and provide...
A Framework for Improving End-User Orientation of Service Mashups

typical example of an activity pattern (Sutcliffe 2002). Since these generalized tasks are not
directly domain-specific, relate to reusable composite tasks but are more specialized than
generic tasks and relate to an activity pattern that help in resolving the domain, I have
termed them meta-domain goal templates. Finally, the domain specific goal graphs are
created by appending the target domain concepts after resolving the domain ontology. The
domain ontologies, for the purpose of this thesis are reused from the existing set of
domain ontologies defined on the web. The proposed KAReM method implements the
phases of the proposed spiral process model to provide support for user requirements and
provide them with analysis and design space. Further details including the foundations,
conceptual framework, architecture and detailed working are explained in the following
sections.

This chapter is organized as follows. Section 5.2 presents the theoretical background of
GO-MaDe and explains the idea of using the Goal formalisms and the domain theory
constructs. Section 5.3 explains the conceptual model of the GO-MaDe framework and
discusses various components in details including the Meta and domain models, the goal
structure and the goal Meta model. Section 5.4 presents the GO-MaDe generic
architecture and how it makes use of various components discussed earlier in this chapter.
It also presents the working of the method and that of the various components including
the Mashup Profiler, Mashup templates manager, The Goal Graph Builder and the Query
Analyser. Section 5.5 explains the generic templates followed by example scenarios
presented as proof of the concept.

5.2 Theoretical Foundations of KAReM

The KAReM method instantiates the proposed spiral mashup development process in 2
distinct cycles. The first cycle comprises of the conception and translation phases of the
mashup development process. The conception-translation (CT) cycle is supported by the
GME. The second cycle is composition-execution cycle which is currently not supported by
GME of my proposed framework. These cycles intertwine the analysis-design and design-
develop phases of the mashup applications development to provide the swift, agile-like
development process that is the inherent need of end users mashups.

The purpose of introducing goals for mashup development is to enable EU to develop Web
Services on the knowledge-level. Instead of facing the technical difficulties of the existing
tools and techniques, clients of my proposed method shall be able to specify their mashup
needs in terms of objectives to be expressed as goals. This helps in abstracting the technical details while carrying all the information, through various defined attributes (discussed later in the report), which is necessary in translating the users’ requirements into service discovery, specification and composition.

The Domain Theory (DT) is a hybrid theory that bridges cognitive sciences and software engineering and draws on different underlying theories such as Rosch’s theory of natural categories, Gartner’s structure matching theory on analogy, Schank’s theory of ecological memory and theory of modularity based on cohesion. It was developed by Neil Maiden and Alistair Sutcliffe (Maiden and Sutcliffe 1994, 1996; Sutcliffe and Maiden 1998) to facilitate cross-domain knowledge reuse at requirements specification level. The main purpose of using domain theory is to make use of the reusable libraries defined by the theory that can work across different domains. The underlying inference of the theory that has been used in the KAREM is the generalization of the problem at an abstract level through models in the problem domain. In order to facilitate the reuse of the models by the end users in the mashup development, I have replaced the object-oriented approach of the domain theory with the goal-oriented approach.

DT supports reuse across different domains and is focused on the problem domain rather than the solution domain. The use of DT can help incorporate domain-independent goal specifications by abstracting the mashup categories in the problem domain. The proposed method makes use of the three components of domain theory for proposing three different goal modelling templates namely 1- Grounded Domains, 2- Meta-Domains, 3- Generic Tasks. The concept of grounded domains as the inspiration to categorize different genres of mashups so as to provide the abstraction helps in cross-domain reuse. The generalized tasks of meta-domains are used to model the mashup genres, whereas the library of generic tasks is used to provide the goal verbs. These components represent a generalized view and hence can be used as bases for templates for EU and can help them derive the requirements for their proposed mashup application.

### 5.3 Conceptual Model of KAREM

In the proposed approach I am dealing with the concept of goals and goal graphs which is independent of the platform specific model of services. However, it is intended that the final specification of goals is capable of being converted into ready to compose services.
Figure 5.1 presents a schema model of KAReM. A goal is composed of a phrase consisting of a goal verb and noun. The goal verb is derived from the generic task and the noun is derived from the domain ontology. Goal graph comprises of goals and follows the mashup type template and meta-domain templates for syntactic and semantic rules respectively. Goal graphs finally is translated into a service model consisting of candidate services ready to be composed into a mashup application and deployed in the composition and deployment phases.
5.4 Meta, and Domain Levels

My proposed method (GO-MaDe) involves two different levels of template-based modelling (Fig. 5.2).

![Conceptual Model - Meta and Domain Levels](image_url)

**Figure 5.2 – Conceptual Model - Meta and Domain Levels**
The Meta (or meta-domain) level involves the abstract view of the method and describes domain-independent entities. This relates to the domain independent modelling and is specialized by selecting the target domain which then leads the EU to the domain-level modelling. This conceptual model also explains the mapping of the domain-independent models onto the domain-specific goal models. While the domain-independent models are defined by the syntactical rules of the mashup types, however the expansion of the goal models is loose as compared to the domain-specific models as the latter are well defined and more restricted based on a target domain.

### 5.5 Goal Model Overview

An overview of a goal is given in fig. 5.3 in the form of a class diagram. I have defined different goal models in my framework keeping in view the requirements of the mashup application area. The details of these goal models are given in the following paragraphs:

![Figure 5.3 – Goal Meta Model](image)

### 5.6 Goal Types

For the purpose of my research, I specify three goal types (fig. 5.4, 5.5):

1- Abstract goals are the highest level goals and summarize the objective of the target mashup. They are characterized by the target domain ontology and preconditions and results. This forms the root node of the goal graph.
2- Atomic goals are the sub-goals that cannot be further divided into sub-goals and are characterized by concrete inputs and outputs pertaining to the particular domain ontology in focus. They are ready to be realized by a web service/API or data sources. Hence, they can be regarded as the leaf or terminal nodes in my goal graph and all invocations and consumption of services are performed via these atomic goals.

3- Composite goals can be further decomposed into sub-goals (either atomic or composite goals) and are characterized by the necessary control and data flow information.

Figure 5.4 – Goal Types Defined in the Method

5.6.1 Goal Structure

The basic structure of the goal is a verb-noun pair to emphasize the action-oriented nature of the goal. This structure is common to all goal types and to further systematically derive the goal names, two libraries are defined which guide the user for naming the goal. The verb ontology is derived from the domain theory constructs comprising the generic tasks whereas the noun ontology is derived from the target domain. The domain ontology is not necessarily redefined for the purpose of this thesis since different comprehensive domain ontologies are already defined and present on the World Wide Web (WWW). Apart from these compulsory components of a goal structure, different additional parameters have
also been defined. These parameters are optional if they are user-defined or derived from the library of optional parameters, or mandatory if they are recommended by the templates. Figure 5.5 represents the goal structure for my method.

5.6.2 Goal Templates and Goal Instances

The method makes distinction in two types of existential goal models: goal templates and goal instances: goal template and goal instance. A goal template is a generic goal description and governs the syntactic or semantic rules of the goal graph. It is a predefined template for guiding the goal derivation created at the analysis and design-time and stored in system. A goal instance represents the ready-to-compose request of the client end user defined at run-time by instantiating the goal templates with concrete user inputs. A goal template is further divided into syntactic goal templates and semantic goal templates. A syntactic goal template defines the syntactic rules and restrictions of the resulting goal graph such as and/or cardinalities of the sub goals. A semantic goal template governs the semantic rules such as the types, names and parameters of the sub-goals. 

*Mashup Type template* and *Generic goal template* are examples of syntactic templates and are directed for end users with vague knowledge of the mashup requirements. A *meta-domain* or *generalized template* is an example of a semantic template and is for the end users with advanced knowledge of the requirements of their mashup application. The next section discusses these templates and their ontology in further details.
5.7 Syntactic Goal Templates

5.7.1 Mashup Type Goal Templates

A. Mashup Types

To come up with a generic model for the end-users to utilize in order to register their initial requirements, I need to define the mashup types. This categorization (fig. 5.6) coincides with the mashup categories already defined (Yu et al. 2008) but are more elaborate for the purpose of my research:

1- Presentation Mashup (GUI Mashups): When the end-user wants to assemble and organize different data or GUI elements from different sources to satisfy the individual needs. This is further divided into 2 types:
   a. Presentation mashup with plain data selection:
   b. Presentation mashup involving data projection or data manipulation

2- Data Mashups: This type of mashup involves the components that act as pure data source such as those composed by yahoo pipes.

![Figure 5.6 - Mashup Categorization based on Data Input/output](image)

The main purpose of defining mashup types is to facilitate the end users in choosing the right goal template to develop their mashup. These templates help define the cardinality rules between parent and child goals. The diagrams below explain the multiplicity relationships between different levels of goals for all the three mashup types. Rn stands for root node, * stands for defining a “many” relationship. These rules need to be
implemented while development to restrict the goal decompositions. For every syntactic template, there will be 1 to * many relationship between objective (level 0) and level 1 goals.

5.7.2 Presentation Mashup with Plain Data Selection

a. Presentation Mashup Scenario:
Mark is a sports enthusiast and wants to learn about the latest happenings in the world of sports. He wishes to see the latest news, pictures and videos for cricket, hockey and football. For this reason, he gets the news in the form of RSS feeds from espn, crickinfo and bbc. He wishes to search the pictures and videos from flicker and YouTube respectively apart from the sports headlines about all the major sports from all around the world.

APIs:
1- RSS feeds from ESPN-cricinfo\textsuperscript{25}
2- Flicker\textsuperscript{26} API
3- YouTube\textsuperscript{27} API

5.7.3 Goal Template for Presentation Mashup with Data Manipulation/Projection

The main goal can be decomposed into multiple goals.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure5_8.png}
\caption{Generic Goal Template for Presentation Mashup with Data Manipulation}
\end{figure}

b. Presentation Mashup with Data Manipulations Scenario

\begin{itemize}
\item \textsuperscript{25}http://www.espncricinfo.com/
\item \textsuperscript{26}https://www.flickr.com/
\item \textsuperscript{27}https://www.youtube.com/
\end{itemize}
Ahmed is a new student at university of Manchester and he wants to have a look at all the restaurants that serve Pakistani/Indian food on the Google maps. The data sources would need to be filtered through appropriate use of operators which could be combined as filters.

**B. Goal Template for Data Mashups**

Since it’s the complicated mashup, the cardinality rules can be custom defined by the end users during the creation of a data mashup.

An important example in this regard could be planning a day out with your friends in which you would not only need to finalize the event and time but also would need to disseminate the information to the group of friends by using preferred communication method such as email or text message. This clearly indicates that all the required data cannot be simply mashed up by specifying sources, rather a series of steps need to be taken before the desired application can be developed to give the desired output.
The example scenario for this template is presented as a detailed case study for validation purpose in the next section.

## 5.8 Generic Goal Templates

### 5.8.1 Generic Tasks as Domain Independent Goal Graph

For the purpose of my proposed method, the reusability of the model will depend on the abstraction of the recurrent tasks. This has been made possible by annotating the goal graphs with the generic tasks of domain theory (Sutcliffe 2002). The complete list of domain theory’s generic tasks is listed below for an insight into the idea of reusability in the light of domain theory:

ii. **Plan:** This task requires an intention sub-goal to be present and ensures that a procedure is present in the working memory after the task terminates.

iii. **Interpret:** It ensures that a state change or a change in the environment is comprehended. It begins when a change is perceived and ends when the change is understood or at least decided to be incomprehensible.

iv. **Evaluate:** It has two subtasks, one evaluates the state change and the other assesses the properties or qualities of an object as a result of the state change. Hence, it as a whole assesses the implications of a state change.

v. **Assemble:** It can be taken as an assembly of both physical and conceptual objects. However, for the sake of my research, it is taken as a visual assembly of conceptual objects. It begins when the assembled objects are available and as an end result, the given parts are organized to form a new whole.

vi. **Disassemble:** In contrary to the “assemble generic task”, this works to break the association of the constituent tasks which are present in the beginning as a whole.

vii. **Transform:** It works to change the expression or representation of a problem from one form to another. In the context of my research, this will refer to data transformation.

viii. **Model:** It refers to enable reasoning by semantically creating a model of known but organized facts.

ix. **Sort:** It is a grouping generic task that seeks to rank or prioritize the given items of facts according to the given criteria.

x. **Classify:** It works to group the items based on the commonalities of items and organized them in categorizes.
xi. Associate: It tends to conceptually bind the given objects using a relationship and is strongly related to “modelling” generic task.

xii. reSelect: As a precursor to this task, a set of two or more objects are given and this task helps in selecting an instance based on the selection criteria expressed at the beginning of the task. Contextually speaking, this task is intended to be used for selecting the target Web Services based on the requirements expressed by the end user.

xiii. Identify: An object or item is the process of establishing the name, label and category of a given item.

xiv. Compare: It is a special case of “evaluate” task and requires the judgement to be made between two or more given phenomena.

xv. Decide: It assesses an object or action as good or bad.

xvi. Test: Testing refers to the validation and verification of a hypothesis, model or proposition to assess its consistency with a set of given facts.

xvii. Record: It specifies the given facts or models into a permanent medium such as writing text or drawing diagram.

xviii. Communicate: Given facts are communicated in the form of message or email to another party; irrespective of the medium.

xix. Explain: It can be associated with communicate and involves advising by providing facts and information to a requesting agent with reasons and related information.

xx. Monitor: It is stimulated with the environmental events and requires to detect and interpret those events and send the information to a monitoring agent.

xxi. Search: The given objects are located. With reference to my research, this refers to the database search according to given conditions.

xxii. Locate: Locate returns the spatial position of the given item. From the perspective of computing systems it refers to the physical memory addresses.

xxiii. Orient: It refers to orienting in terms of finding a location for travelling with respect to a reference point.
5.9 Meta-Domain Goal Templates based on Generalized Tasks

The Domain theory framework describes the meta-domain component composed of generalized tasks that emphasizes the problem solving and creative design but is not directly related to the physical world. Generalized tasks follow a goal hierarchy structure and are composite components having two or more generic tasks. Inspired from the concept of meta-domain and goal-based hierarchy of generalized tasks, I have defined the concept of meta-domain templates that do not directly relate to the domain but are an important step towards the domain orientation followed by the complete goal specifications with candidate services. In order to identify the meta-domains, the recurring mashup applications were studied to identify the patterns that I have termed as mashup genres. These mashup genres are recurrent categories of mashup applications that share a common set of requirement patterns and help in abstracting the problem domain. While supporting reuse, they also helps in classifying the problem domain and are specialized templates aimed for providing advanced goal modelling features to users with well-defined set of initial requirements. Based on the initial literature study, I have identified a set of 4 mashup genres (Wong and Hong 2008; Goarany et al. 2010) and have used one-to-one mapping with the domain theory generalized tasks. However, the list is not exhaustive and the extensibility of my proposed method allows for the definition of new, customized meta-domain templates based on user-identified/defined mashup genres. The users can also customize the system-defined templates. All the goal templates defined below have AND contribution links unless otherwise specified. Following is a more detailed description of the initial mashup genres identified (Wong and Hong 2008):

1- **Personalization/Aggregation Mashups – Information Modelling Generalized Task**: This genre relates to one of the most common categories of mashup applications in which the user-specific data is extracted through multiple sources. In most of the cases, this data is in the form of multimedia applications. The main motive of this mashup is the personalized view of the specific domain concepts, often times involving multimedia content such as images and videos, by taking the required data from disparate sources. The generalized task used for creating a template for this category is the information modelling.

2- **Travel – Navigation**: This genre of mashup relates to the trip planning for official or leisure purpose and is modelled using the navigation generalized template of the domain theory. It is a specialized case of Real-Time Monitoring mashup
applications defined by Wong and Hong (Wong and Hong 2008). It include hotel, flights booking scenarios.

3- **Meeting / Planning – Planning/Scheduling:** This is another specialized case of Real-Time Monitoring mashup category (Wong and Hong 2008). It is oriented towards mashups that involve meeting irrespective of the nature and motive. This can involve official meetings such as conferences or seminars as well as informal get together of friends and families.

4- **Information – Information Retrieval:** Derived from the parent class of Focused View of mashup types of the categorization of Wong and Hong (Wong and Hong 2008), it defines those mashups that return a subset of data from an API or data source based on a query. The information retrieval generalized task goal hierarchy has been customized to provide the template for this genre of situational applications.

While simple mashups can be made by invoking the respective category once, complex mashups might require a combination of these generalized tasks to solve the problem at hand.
5.9.1 Personalization/ Multimedia Mashups based on Modelling Generalized Task

This category involves mashups that deal with personalization scenarios and often end users wish to include maps, images or videos relevant to their domain. The generalized task for this category is selected to be Modelling (although in original context of domain theory, it is referred to as Analysis/Modelling). This generalized task involves organizing and assembling facts (streams of data, photos, and videos) in a coherent representation.

![Modelling Generalized Task Goal Hierarchy for Multimedia mashups](image)

The personalization meta-domain goal model can be instantiated selecting multitude of physical domains such as Sports News and can be further specified using following attributes:

- **Data Feeds:** Sports/News/Show Business/Add New
- **Filters:** Less than / Greater Than / Equal to / Contains / Does not contain
- **Orientation:** Grid / List / Small Icons / Large Icons
5.9.2 Travelling Mashups based on Navigation Generalized Task

All the travelling related generalized tasks involve planning the journey and stay. It is further divided into two main sub-goals for the flights and hotels sub-tasks. The terminal goals list all the input requirements of this kind of mashup application.

The travelling meta-domain can be further specialized by specifying the mode of travel and selecting the relevant domain ontology. The input/output parameters of the terminal goals are as follows:

- **Journey ID:** Unique identity of the journey
- **Purpose:** Official/Personal
- **Mode:** Air/Water/Road
- **Source:** City/Country
- **Destination:** City/Country
- **Transport Co.:**
- **Departure:** Date/Time
- **Arrival:** Date/Time
- **Class:** Business/First/Economy/Other
- **No. of Persons:** Numeric Data
- **Hotel:** Name
- **Rooms:** Number
- **Room Type:** Text
5.9.3 Meeting/Planning Mashups based on Planning/Scheduling Generalized Task

This is derived from the *Real Time Monitoring mashup* of Hong and Wong categorization (Wong and Hong 2008). This mashup genre involves the organizing of an event to happen at a specific date and time. The planning-Scheduling generalized task is used to derive the goal template for this genre. The planning in this context refers to carrying out an activity in future and is different from Artificial Intelligence planning. This template basically considers the core requirements of venue, activities selection followed by selecting the participants and communicating to them. Further additions to the template could involve any changes in the plans due to environment of place or people.

![Figure 5.12 – Meeting/Planning Generalized Task Goal Hierarchy for Planning/Scheduling Mashups](image)

This domain can be specialized using the following requirements from the library:

- **Venue:** Place
- **Date:** dd/mm/yy
- **Time:** 12 hour/24 hour format
- **People:** Names
- **Contact:** Email/Text Messages/
5.9.4 Information Mashups based on Information Extraction Generalized Task

This genre of mashup targets the informational needs of the EU such as financial data or business data analysis and is derived from the focused view mashup category of mashups (Wong and Hong 2008). It involves identifying the information needs, formulating search queries using appropriate search criteria, and evaluation of the final search results.

![Diagram of Information Acquisition Generalized Task Goal Hierarchy for Information Extraction Mashups]

This meta-domain can be linked to different physical domains such as sports, current affairs, show-business etc.

The information needs and filters could be expressed by following set of requirements:

- Information Need: Sports/News/Show Business/Add New
- Filters: Less than / Greater Than / Equal to / Contains / Does not contain
5.10 KAReM Tool Architecture

The KAReM method is based on templates based development principle that can be created and updated by end-users to define and register their requirements (Fig. 5.14). The main task in this regard is the designing of templates. For the purpose of defining templates, mashup types have been defined in order to define a representative template for that mashup category.

![Figure 5.14 – The proposed Lifecycle Activity Model](image-url)
In order to explain the working of KAReM, I am presenting the algorithm for listing down the main steps of the method:

START

INPUT Mashup_Profile

IF Mashup_Profile InComplete *for basic end user

GOTO GenericTemplate

LOOP Until

Develop GenericGoalGraph

ELSEIF Mashup_Profile Information Complete *for advanced end user

GOTO MetaDomainGeneralizedTemplate

LOOP Until INPUTParameters=Complete

Develop GeneralizedGoalGraph

ASSIGN CandidateServices to TerminalGoals

GENERATE WSMOGoalSpecifications

END

The end user specifies the basic mashup profile attributes either by setting different attributes (preferably in a wizard style series of dialog boxes). The mashup profile includes selecting attributes from a drop down list about the type, priority, motivation of mashup. If EU doesn’t understand the attributes, or for some reason want to bypass the wizard, they can directly go to generic domain independent goal graphs and start to specify their requirements using visual goal models by directly manipulating the goal graph. Once, the EU has completed the generic goal graph and is ready to fill the mashup profile, they then go to the generalized goal templates based on the information provided in the mashup profile. So, if the user wants to develop a data mashup for navigation purpose, the corresponding generalized template with the syntactic rules of a data mashup will be recommended. The EU can then complete the visual model of the generalized goal graph. The end user can enter inputs until all the required attributes are provided and finally selects from the set of candidate services recommended based on the set of attributes specified.
The Goal Oriented Mashup Development (GO-MaDe) Editor (fig.5.15) is the proposed tool to assist the proposed framework mentioned earlier in this chapter. The purpose of the tool is to provide design-time support for non-IT end-users who want to develop mashups. The following paragraphs of this section provide a brief description of the overall architecture and explain the details of components.

![System Architecture](image)

**Figure 5.15 – System Architecture**

### 5.10.1 System Components

The GO-MaDe Editor has four main components (fig. 5.15): a **mashup profiler**, a **Mashup Template Manager**, a **Goal Graph Builder**, and a **Goal Specifications Generator**. The **Mashup Profiler** communicates with the EU to record basic information about the EU’s target mashups at the time of initiation of a new mashup project through system
dialogues. It is also responsible for gathering the information about the new mashup project to confirm the type of mashup which is to be used in later steps. The next component is the Template Manager which analyses the basic mashup profile generated by the mashup Profiler and uses that information to suggest and generate a customized goal graph template which is later used to develop goal graph with the help of next component, the Goal Graph Builder. The Goal Graph Builder uses the help of two of its sub-components: Query Manager and Service Matching to reformulate the queries for each goal graph and help the user develop his goal graph that is a model of the EU’s requirements and design of his/her target mashup project. The last component Goal Specification Generator takes input of the complete goal graph and generates a complete set of mashup goal specifications that represents a complete set of EU requirements and design including the concrete services and data sources along with the required manipulations and processing.

5.10.2 Mashup Profiler

This component is responsible for eliciting, creating and managing the initial mashup profile for the EU’s target mashup project including identifying the mashup type based on the initial data gathered by the user. It consists of two sub-components and determines the type of the target mashup by taking the input regarding the EU’s target mashup’s potential data sources and objective. For instance if the EU specifies that he wants a couple of data sources merged into a single interface and specifies the sources that do not require any processing, the mashup profiler with the help of type validator sub-component suggests that the EU is trying to develop a basic presentation mashup.

A. Mashup Types

To come up with a generic model for the end-users to utilize in order to register their initial requirements, I need to define the mashup types. My categorization coincides with the mashup categories already defined (Yu et al. 2008) but are more elaborate for the purpose of my research:

1- Presentation Mashup (GUI Mashups): When the end-user wants to assemble and organize different data or GUI elements from different sources to satisfy the individual needs. This is further divided into 2 types:
a. Presentation mashup with plain data selection:

b. Presentation mashup involving data projection or data manipulation

2- Data Mashups: This type of mashup involves the components that act as pure data source such as those composed by yahoo pipes.

5.10.3 Mashup Template Manager

Once the ‘mashup profiler’ suggests/recommends the mashup type, next comes the responsibility of Mashup Template Manager: to suggest a goal template based on the mashup type suggested by the Mashup Profiler. The customization can also be incorporated before generating the template such as for a plain data selection template the number of data sources to be mashed up can be customized in the goal template.

A. Generic Goal-based Templates

The generic goal-based templates for each mashup category are defined below:

Rationale

Mashup activities can be from diverse domains and of diverse nature. A given objective of a user can go through several decompositions before it can be mapped onto concrete Web Services and web APIs. Hence, one of the main tasks in the proposed method for mashup development is building a goal-graph (or technically speaking a goal-decomposition mechanism). Pragmatically, dealing with every goal graph will require separate insights into end-user activities pertaining to the domain. However, reusability can abstract these complexities. To practice reusability at the domain level, I have proposed the use of concepts of generic and generalized task models as presented by domain theory.
5.10.4 Goal Graph Builder

The responsibility of Goal Graph Builder is threefold: (1) Initiating and finalizing a goal graph development activity (2) Query Reformulation for each goal of the goal graph (3) Supporting the identification of target data sources and/or services/APIs. The strategies used for query reformulation are mentioned in table 5.2.

![Query Formulation Process States and Strategies](image-url)
A. Goal Query Formulation

According to the underlying premises of my proposed approach, the end-user might not be well aware of the final form of intended mashup. Hence, I have introduced the query formulation process to help end-user develop the goal graph.

The query formulation process is adapted from the work of Assar and Aljoumma and is presented in the form of a map, where intentions are presented as nodes and strategies as edges (Aljoumaa et al. 2011; Assar et al. 2012).

The process consists of following states:

1- Start
2- Formulate the Root Node
3- Formulate the Sub-nodes
4- Formulate the Leaf Nodes
5- Terminate
6- Select Services

Different strategies are implemented to support the transitions and query formulation process within and across the states while following the constraints of the process:

a) Constraints:
   i. Specify in structured Natural Language (Generic Task+Domain Task): This is the first constraint according to which the EU specifies the goal statement by following the structured Natural Language rule by appending the suitable target domain concept (product) with the generic task (verb)
   ii. Goal Template Contributions and Degrees: Following this constraint, EU follows the guidelines embedded in the goal graph template suggested during initial mashup profiling and generated thereafter. The basic template contains the rules to specify each goal at different level of goal graph thereby governing the degree of goal graph and contributions and multiplicities of goals at each level.
### b) Strategies

Table 5.2 - A Summary of Goal Derivation Strategies

<table>
<thead>
<tr>
<th>Strategy code</th>
<th>Strategy</th>
<th>Definition</th>
<th>Procedure</th>
<th>Outcome</th>
</tr>
</thead>
</table>
| GE            | Goal elaboration          | A higher level goal is refined into sub goals by qualifying non-func. attribute | 1- Right Click a Goal to be elaborated.  
2- Click Elaborate  
3- Select the NFR to qualify  
4- Enter the names of sub goals (by using strategy RGCT and RDO) | A main goal is elaborated into sub-goals                                                   |
| GD            | Goal decomposition        | A higher level goal is divided into sub goals by functional decomposition    | 1- Right click a goal to be decomposed  
2- Select the number of sub goals to be generated  
3- Enter their names (by using strategy RGCT and RDO) | A goal is decomposed into sub-goals.                                                        |
| RGCT          | Reuse Generic Tasks       | Generic Tasks of The Domain Theory are Reused                                | 1- Right click the goal in focus  
2- Select the verb field  
3- Click on Generic Tasks  
4- Select the appropriate generic task | Goal with verb specified                                                                  |
| RGDT          | Reuse Generalized Tasks   | Generalized Tasks of The Domain Theory are Reused                           | 1- User selects the Genre while Mashup Profile Creation  
2- Selects the appropriate template. 3- Supplies the parameters to complete the goal graph | A complete goal graph                           |
| RDO           | Reuse domain Ontology     | The Domain Ontology is Reused.                                               | 1- Right click the goal in focus  
2- Select the Noun field  
3- Click on Domain Ontology  
1- Select the desired ontological noun. | Goal with ontology specified                                                               |
i. Decomposing into sub-goals: The goal in focus is aggregate and is decomposed into simpler smaller goals.

ii. Searching/selecting the candidate services: This strategy works by initiating a search of candidate services for goal-in-focus or by selecting a candidate service.

iii. Suggestion by system: The system magnifies the target semantics and looks into synonymous, related concepts to replace the concept of goal-in-focus.

iv. By displaying results: The results of the query of the particular goal in focus are displayed.

v. Rewriting the query: If the user feels unsatisfied with the query, he/she can reformulate the query by starting over again at the goal level.

vi. Terminating: The search for the particular goal ends.

The following table provides a list of strategies to derive the goals. Please note that each strategy can be used in special context and is directly related to the options selected while creating a mashup profile.

c) Trigger to Move to Translation Phase

The main motive behind the conception phase is to allow users to elicit the early phase requirements through a model-driven semi-automated goal-based technique. Hence, this phase needs to exercise a signalling trigger to stop with the conception phase and move to translation phase. To incorporate this, I have introduced the parameterization technique that requires the EUs to fill the parameters with values related to task at hand. Once, the parameters are provided, it serves as a signal to move the EU to the next phase of my proposed process i.e. translation.

5.10.5 Goal Specification Generator

This component gives the output of the goal in the form of WSMO goals but can also generate the goal specification in structured language. These specifications can be used for the service compositions.
5.11 Case Studies:

This section presents the example for demonstrating the goal descriptions as defined in the above method. I have considered a scenario from the university renting domain, which is often used as a representative example.

5.11.1 Data Mashup Example Scenario

One of the major concerns for such migrating students is finding a suitable accommodation. A part of the student population wishes to live in the student accommodation. However, a large part prefers private accommodation and searching a suitable private accommodation is a difficult task in spite of various online search facilities. Besides, different groups of students will have different requirements for a house. For instance, different faith groups would like to have houses near to their respective worship places. Similarly, different groups prefer different places of interest near their accommodation. This information is not available through a single source and students would need to integrate data and feeds from different sources. One of the possible ways of doing this:

1-   Find a place near university with suitable rent.
2-   Find a place with least crime rate.
3-   Find a place with preferred places of interest near the university.

Goal Objective

Search_Accommodation-Near_Uni (goal)
has-input-role :value has_Uni_Name
   :value has_Start_Date
   :value has_Crime_Rate<
   :value has_Places_of_Interest
has-output-role :value has_Required_Flat

A. Basic Mashup Profile (through System Dialogues)

Mashup Name: Search Accommodation near University

Objective: To search for a private accommodation near university that meets certain preferences.

Domain Ontology: University House Renting
Mashup Type: Data Mashup/ Mashup Data Sources: Not confirmed in the beginning of mashup development activity

**B. Building Generic Goal Graph**

**a. Step 1 – Set (By Default) Objective as Goal 0 (Level 0 of Goal Graph)**

Table 5.3 - Setting the Objective

<table>
<thead>
<tr>
<th>Generic Task</th>
<th>Target Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search</td>
<td>University Accommodation Renting</td>
</tr>
</tbody>
</table>

Table 5.4 - Parameters List at Step 1

| Known Parameters | 1-Direction  
|                 | 2-Time: September 2014  
|                 | 3-Quantity: 01  
|                 | 4-Beneficiary  
|                 | 5-Location: University of Manchester, UK |

| Unknown Parameters | 6-Suitable  
|                   | 7-Filters |
**Parameters after Step 1**

b. **Step 2: Decomposing Goal-0 Level-1 by Applying Suitable Query Reformulation Technique**

![Diagram](image)

**Figure 5.17 - Step 2**

**Table 5.5 - Goal Strategies at Step 2**

<table>
<thead>
<tr>
<th>Strategy no. 1</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decomposing Into Sub Goals</td>
<td>Goal-1: Search for Least Crime Rate</td>
</tr>
<tr>
<td></td>
<td>Goal-2: Search for Accommodation with POI nearby</td>
</tr>
</tbody>
</table>

c. **Step 3: Decomposing and Elaborating Goal 1 (G1) through Suitable Query Reformulation Technique**

![Diagram showing step 4]

**Figure 5.18 - Step 4**

**Table 5.6 - Goal Strategies at Step 3**

<table>
<thead>
<tr>
<th>Strategy no. 3 &amp; 4</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decomposition</td>
<td>Goal 3-G3: Search Accommodation with No murders</td>
</tr>
<tr>
<td></td>
<td>Goal 4-G4: Search Accommodation with low robberies</td>
</tr>
<tr>
<td>Parameterization</td>
<td>Possible Optional Parameters:</td>
</tr>
<tr>
<td></td>
<td>1- Direction</td>
</tr>
<tr>
<td></td>
<td>2- Time</td>
</tr>
<tr>
<td></td>
<td>3- Quantity (0 murder crimes, &lt;10 robberies)</td>
</tr>
<tr>
<td></td>
<td>4- Beneficiary</td>
</tr>
<tr>
<td></td>
<td>5- Location: Hulme, Manchester</td>
</tr>
<tr>
<td></td>
<td>6- Means</td>
</tr>
</tbody>
</table>
d. **Step 4: Decomposing and Elaborating Goal 2 through Suitable Query Reformulation Techniques**

**Figure 5.19 - Step 4**

<table>
<thead>
<tr>
<th>Strategy no. 3 &amp; 4</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decomposition</td>
<td>Goal 5-G5: Search Accommodation with a mosque nearby</td>
</tr>
<tr>
<td></td>
<td>Goal 6-G6: Search Accommodation with Tesco nearby</td>
</tr>
<tr>
<td>Parameterization</td>
<td>Possible Optional Parameters:</td>
</tr>
<tr>
<td></td>
<td>1- Direction</td>
</tr>
<tr>
<td></td>
<td>2- Time</td>
</tr>
<tr>
<td></td>
<td>3- Quantity (mosque is within 1 km, Tesco is within 1 miles)</td>
</tr>
<tr>
<td></td>
<td>4- Beneficiary</td>
</tr>
<tr>
<td></td>
<td>5- Location: Hulme, Manchester</td>
</tr>
<tr>
<td></td>
<td>6- Means</td>
</tr>
</tbody>
</table>
e. **Final Goal Graph at the end of Conception Phase**

![Final Goal Graph](image-url)

**Figure 5.20 - Final Goal Graph**
A Framework for Improving End-User Orientation of Service Mashups

A. Generalized Task Goal Hierarchy for Translation Phase

Since, this mashup is an information retrieval mashup, therefore the information acquisition goal hierarchy template is selected:

Generalized Task GG Template for Information Acquisition

![Diagram of Generalized Task Goal Hierarchy for Information Acquisition](image)

Figure 5.21 - Goal Template Development by using Goal Template
B. Goal Specifications:
The last step is the WSMO goal specifications generation. Once full details are acquired through the Generic and Generalized goal graph, the details can be passed on to WSMO goals specification generator module which generates the WSMO goals that can be used to search the target services and compose them keeping in view the dependency needs.

goal SearchAccommodationNearUni
capability
inputVariables c?City, poi?PlacesOfInterest, sd?StartDate, cr?crimerate
outputVariables f?f,
precondition definedBy
?home memberOf city and ?University memberOf city and
?sd memberOf dateandtime and ?cr memberOf NumericData.
?poi memberOf places (List:WorshipPlaces, ShoppingStores)
postcondition definedBy
?f memberOf flat and
?f [ Uni hasValue ?UniName,
StartDate hasValue ?sd, EndDate hasValue ?ed,
PlacesOfInterest hasValue ?poi1(WorshipPlaces), PlacesOfInterest hasValue
?poi2(ShoppingStores)] and
after (?od, ?leave) and
?f memberOf flat and
?f[Uni hasValue ?UniName, CrimeRate hasValue ?cr,
StartDate hasValue ?sd, EndDate hasValue ?ed].

interface
orchestration
stateSignature
shared {city, dateandtime}
in { PlacesOfInterest , CrimeRate }
controlled { cs1, cs2, ... }
transitionRules
forall {?cs } with (?cs[value hasValue cs1] memberOf control) do
perform applyMediation ppNearbyPOI_input
perform subgoal1 achieveGoal NearbyPOI
update cs[value hasValue cs2]
endForAll
5.12 Discussion

Mashup development is based on the idea of developing applications for short-lived needs that arise episodically. The main beneficiary of these mashup applications are domain professionals who are not IT-literate to know programming. Thus, mashup development tools and frameworks must support the abstraction to hide away the technical details and should support user-friendly visual mechanisms. Nevertheless, the trade-off between expressiveness and difficulty is always an issue. Similarly, cost and scope tension also poses a great difficulty for end user mashup development researchers (Fischer et al. 2004). On one hand, developing technically efficient tools hinder the end user’s adoption as they focus on the technology side rather than the end user requirements. On the other hand, very basic domain-specific languages that allow customization offer very little power of development such as Office applications and query writers. To achieve an appropriate balance between expressiveness and learning curve, I investigated the application of goal-based method, based on domain theory constructs in the light of meta-design principles. I have proposed the spiral process in the previous chapter that recommends that mashup development activity should be engaging end users in their problem domain by helping them in specifying and modelling their needs with the aim of inferring implicit services from the modelled requirements to be composed for fulfilling the end user needs.

Consequently, I presented the framework the GO-MaDe framework based on a spiral process that redefines the mashup development activity and a knowledge representation and modelling method (KAReM). The process highlights the importance of acquiring end users’ needs and modelling them thus offering a substitute to the view of mashup development as a monolithic activity majorly focusing on service compositions. In the method, I proposed the goal templates based on the domain theory components of meta-domains, generalized tasks and generic tasks. These templates help end-users specify their needs in a goal-tree structure, starting from an abstract view, while gradually deriving the more concrete, domain-specific information in a top-down fashion by direct manipulation of the goal hierarchies. My method can be applied to multiple domains by selecting the right domain ontology and is scalable as it supports modification and creation of new goal templates.

The next chapter presents a controlled experiment as part of the validation of my contributions. The evaluation is based on a story board of my proposed modelling templates, and validates the process as well as the method comprising the framework.
6. Empirical Evaluation

This chapter presents the systematic assessment of the framework –GO-MaDe- and its component parts, the spiral process and the KAReM method, presented and discussed in the preceding chapters 4 and 5 respectively. The empirical evaluation is carried out by a controlled experiment followed by a quantitative analysis and is conducted to evaluate various aspects of the contributions of this thesis which are listed below:

1- Does the proposed process help users in creating mashups?
2- Abstracting the technical details can help reduce the technical burden of the end users.
3- Visual goal modelling can help end users better present their situational applications tasks.

GO-MaDe framework combines the benefits of semi-formal goal models with the templates of domain theory with partial application of meta-design principles to assist the non-programmer domain experts in composing their mashup applications. Key aspects in the proposed method are improvement in user-orientation. The semi-formal goal representations are used to derive the requirements-to-services transition in a user-friendly way and domain theory templates are intended to support the reusability and cross-domain usage of the method. The primary research hypothesis, hence, can be: “improving end user orientation of web mashups through abstraction and visual modelling using goal models and domain theory constructs”. The main reason of selecting controlled experiments as the approach for the empirical evaluation is its suitability for the research problem at hand and is explained in the following section.

6.1. Research Methodology

Evaluation is central to the Design Science Research Methodology (DSRM) and comprises one of the seven guidelines of DSR (Hevner et al. 2004):
“The utility, quality, and efficacy of a design artefact must be rigorously demonstrated via well-executed evaluation methods”

The checklist presented by Hevner and Chatterjee clearly states that the evaluation should be performed during the design cycle of DSRM and the feedback should be used to improve the artefact design (Hevner and Chatterjee 2010b; Hevner and Chatterjee 2010a). Hevner et al. also present a list of design evaluation methods along with the guidelines for application of each of the method. The selected evaluation method must be consistent with the designed artefact and therefore selecting a suitable method is of prime importance (Hevner et al. 2004). After analysing the design problem at hand and studying the guidelines for each evaluation method, two distinct evaluation methods have been identified as best-fit for the research validation of this thesis: Experimental evaluation (Basili 1996) and Descriptive evaluation.

The descriptive evaluation is carried out by employing two methods, Informed or structured argument (Ghetiu et al. 2010) and scenario based validation (Ryser et al. 1998; Ryser and Glinz 1999; Ryser and Glinz 2000). The validation by these methods is attempted in the previous chapters (chapter 4 and chapter 3), where domain theory and visual goal models are presented as a viable solution to the current challenges faced by EU mashup development. In addition different scenarios are presented in chapter 4 as an attempt to present the workability of the proposed method. Scenarios describe the working of a proposed system at example or instance level (Sutcliffe et al. 1998) and are defined differently depending on the research area in focus. It has been widely used in the research areas of business and management strategy (Lindgren and Bandhold 2002), software engineering (Jacobson 1992) and technology foresight. Scenarios are considered central to software and systems engineering and scenarios like use cases and personas are considered central to object oriented system analysis and design (Rosenberg and Scott 1999; Jacobson 1992). Although, I have not yet developed the implementation details of my proposed framework to identify different objects and their working in detail, I can warrant that the innovative idea underlining my proposed system can be validated using the scenario based validation technique. However, since the underpinnings of the proposed method are in the form of a hypothesis that seeks to improve the design and consequently implementation of the EU service mashups, it is imperative to use the classical scientific method for testing the hypothesis by identifying the cause-effect relationships: a controlled experiment (Sjøberg et al. 2005). Shadish et al. define
experiment as an empirical study which is deliberately intervened to study the effects (Shadish et al. 2002).

For the purpose of the experiment, story boarding technique (Beaudouin-Lafon and Mackay 2003) is used to validate the GO-MaDe method and the underlying hypothesis “empowering end users in developing mashups by improving the end user orientation through goal-based abstract visual models and reusable components of the domain theory”. It was considered that the development of the prototype as “proof-of-concept” does not serve the purpose of the validation and that paper-based story boards will be sufficient enough for the purpose. Accordingly, developing a prototype is not a primary contribution of the research since the main research contribution was the evaluation of the existing literature and proposal of a new method. Moreover, since a major novelty of the method lies in the redefined process, developing a GUI interface with minimal features would be inequitable as it would not have represented the research contribution at a reasonable level beyond the interface level interactive design. Lastly, the main objective of the evaluation does not intend to measure any performance-related metrics of temporal kind or compare the working of my proposed method with the existing tools. Therefore, I opted for the simplest method that could provide the greatest insights into the design and process level working of the proposed framework. Keeping in view these factors, a paper-based story board is selected to evaluate the workability and suitability of the proposed method- GO-MaDe- for developing end user service mashups in a bid to facilitate the end-users’ in dealing with the problems arising due to their non-IT background. The approach to be used for the purpose of study is logical positivist approach.

6.2. Experiment Design

Usually, consumer mashup tools aimed at facilitating end users do not use explicit requirements acquisition method. These tools force all end users to rely on implicit handling of requirements and use the components given via the tool such as widgets or pipes to translate their intention into a services-based-application. I hypothesize that a separate phase for explicit handling of users’ expectations, in which the end users can specify the requirements of their mashup application will have a significantly higher impact on the adoption of mashup tools by decreasing the technical burden that is a reported cause of end user problems in the literature.
A total of 10 participants were selected for the evaluation with no, basic or limited IT knowledge. The key group for the study was identified from the University of Manchester students and will be referred to as end users or participants/respondents in the remainder of the chapter. It is acknowledged that the university students cannot be a perfect representation of organizational domain experts with little or no prior IT knowledge, the Masters level students are practically oriented towards their scheme of studies and are given problems during their degree program that can be considered as examples of situational problems. Besides, since my research and the method proposed therein do not exclusively make a distinction between organization and non-organizational end users, it does not necessitate the compulsion of recruiting the real domain experts. It is therefore concluded that valuable insights can be obtained by studying their responses.

<table>
<thead>
<tr>
<th>STEP</th>
<th>DURATION</th>
<th>DETAILS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Setup</td>
<td>10 minutes</td>
<td>Setting up the room, experiment material, stop watch and projectors followed by welcoming the participants</td>
</tr>
<tr>
<td>Tutorial</td>
<td>20 minutes</td>
<td>The tutorial presentation lasted for 15 minutes and then all the participants were given time to read the handout.</td>
</tr>
</tbody>
</table>
| Task-1              | Max time: 40 min | **Scenarios Selection:** The pair was given the scenario list and was asked to select the scenario of choice with mutual consent.  
**Scenario Modelling:** One participant was given a goal oriented template (the treatment) and the other participant was given a template with no goals in plain outline format (the control group). (All scenarios are presented in the appendix) |
| Task 1 Feedback     |           | At the end of the first task, the participants were required to provide feedback on the feedback forms. |
| Task-2              | Max time: 40 min | **Scenarios Selection:** The control and treatment roles were swapped and both the participants were asked to select a scenario without consultation with each other.  
**Scenarios Modelling:** During the second task, the templates were swapped and the participants were asked to complete the modelling task. |
| Task-2 Feedback     |           | At the end of task 2, the participants were required to provide feedback about task 2 as well as the whole experience. |

**TOTAL EXPERIMENT TIME: 120 minutes (2 hours)**

The experiment was designed to be conducted with two participants at one time so as to facilitate the control-treatment experimentation style (table 6.1). All the participants were carefully recruited based on the results of an initial selection form. Before the initial selection, the interested participants were interviewed to minimize the time wasting on worst-fits by asking few short questions about the major subject of their last degree and ensuring that they don’t have a major in computer science/software engineering or related discipline. The design of the experiment included a tutorial in the beginning of the session.
including the questions and answers session regarding the subject area or the experiment itself. The tutorial had two parts, one was a presentation and the other was in the form of carefully drafted handouts. The presentation consisted of three parts. The first part introduced the participants about the background with some practical applications of the mashups. The second part shed light on the examples with screenshots of Yahoo Pipes. The last part explained the tasks to be conducted during the experiment. The tutorial was conducted by the same person for all the sessions. For all the participants, same presentation and handouts were used. This helped me in guaranteeing that each participant went through the same treatment and hence helped me in reducing any biases related to the introduction. During the experiment, the participants were free to ask questions.

The duration of the experiment was 120 minutes at the most excluding the time for initial setup. However, in actual most of the participants finished their tasks in lesser time. The experiment was conducted at the University of Manchester (the details are provided in table-6.1).

The evaluation was done on the basis of a survey designed especially for the purpose of evaluation and classification of the end users and is attached in the appendix. The survey consisted of two sections:

Section I – Feedback Form 1: End users were given a series of situational problem scenarios to be completed using the given prior training on the day and the story boards provided.

Section II – Feedback form 2: This section comprised of the feedback of the participants regarding their experience while developing the scenarios.

1- This section was intended to get several useful insights regarding the components of the method such as the suitability of visual models and the extent to which end users make use of reusable templates.

2- This section also tends to seek feedback on the idea of providing the problem-solution space instead of directly introducing them to a tool.

A total of 5 scenarios were given to the participants. All the scenarios are listed in the appendix and were carefully chosen so that the students can relate to them. 3 scenarios belonged to each of the 3 goal template namely Travel/Navigation, Planning/Scheduling
Meeting, and personalization. Remaining 2 scenarios belonged to Information Acquisition goal templates and final 2 scenarios required them to create their own template.

Two set of evaluations were carried out. One set of evaluation consisted of the suitability of visual goal models and domain theory constructs for use/reuse of existing templates.

6.3. Evaluation Criteria

This section discusses the prime objectives of the evaluation that integrate to form the main hypothesis of this study:

1- Abstracting the technical information
2- Improving user quality of experience by helping them specify more requirements
3- Providing a reusable set of constructs to develop the library
4- Provide visual goal models to increase the involvement of end-users in the process of developing mashups by providing them a space to explore the problem-solution space

These objectives encompass both the method design model and the process model of GO-MaDe and have been extracted from the main hypothesis to enable a detailed evaluation of each aspect of the proposed method. The next paragraphs discuss the evaluation metrics that were selected to assess the evaluation objectives.

The evaluation is planned to be oriented towards following metrics derived a seminal work on evaluation of workability of both process and method (Gould and Lewis 1985) and relate to the standard usability definitions (Jokela et al. 2003):

6.3.1. Completion Rates

It's generally recorded as binary metric (1 for Task Success and 0 for Task failure) but due to nature of my task, I have chosen a Likert scale-like scoring. For the purpose of this experiment, the completion rate will be depending on the number of requirements specified by the participants. Although keeping in view the scenarios and the ambiguity contained in the term “requirements completeness”, I can argue about task completion. Moreover, the exciting requirements can be considered as additional to main requirements, but I am having a set of requirements that must be specified in order for the services to be specified and the application to be composed. Scores are planned to be awarded to goal modelling tasks based on the following:

1- 5 marks for complete and correct goal model with maximum set of requirements
2- 4 marks for minor errors and sufficient set of requirements
3- 3 marks for flawed goal models and missing main requirements mentioned in scenario
4- 2 marks for minimal effort till level-1 goal models
5- 1/0 mark for the little or no attempt

6.3.2. Usability Problems (UI Problems) encountered (with or without severity ratings)
In order to measure the usability problems, I included different questions related to different aspects such as requirements elaboration problems, goal templates problems in feedback forms. This will help me in improving the specific aspect of the method.

6.3.3. Task Time
Total task duration is considered the standard measure of efficiency and productivity. I had planned to record how long the end user takes to complete a task in minutes. Special care was taken in noting the start and finish times. The start of the task time was noted after the scenario was decided once users finished reading task scenarios and end the time when users have finished all actions (including reviewing).

6.3.4. Task Level Satisfaction
We included different questions related to the tasks in each of the feedback forms. After users attempted a task, they answered a few or just a single question about how difficult the task was. Task level satisfaction metrics were important since I wanted to know the level of satisfaction of end users with my proposed goal-based models.

6.3.5. Test Level Satisfaction
At the conclusion of the experiment and at the end of feedback form 2 I included both closed ended and open-ended questions. These questions were specially designed to inquire end users about their impression of the overall ease of use of mashups generally and their level of interest in particular in the exercise.

6.3.6. Errors
Errors tend to give excellent diagnostic information and, in my experiment, were mapped to design problems. The errors were mainly recorded at the template level once the experiment was over and the analysis began as the unintended action, slip, mistake or
omission a user made while attempting the task. The two main error categories were wrong goal-template selection and flawed goal decomposition.

### 6.3.7. Custom Metrics

Keeping in view the requirement of this experiment, I classified questions along comprehension, helpfulness and ease of use to rate my method based on the users feedback. These metrics were particularly useful as they could provide valuable insight into the completeness of the method.

### 6.4. Results and Analysis

This section reports on the lessons learned while conducting the experiment including quick observations, basic statistical analysis of the feedback and any pitfalls that can be improved while replicating the experiment in future.

<table>
<thead>
<tr>
<th>Table 6.2 - Snapshot of Participants and Task Data</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PERSONAL AND PROFESSIONAL DETAILS</strong></td>
</tr>
<tr>
<td><strong>M=MALE / F=FEMALE</strong></td>
</tr>
<tr>
<td><strong>NAME</strong></td>
</tr>
<tr>
<td><strong>PAIR 1</strong></td>
</tr>
<tr>
<td>SL (F) Student MA (Edu.)</td>
</tr>
<tr>
<td>RI (M) Student MA (Eco.)</td>
</tr>
<tr>
<td><strong>PAIR 2</strong></td>
</tr>
<tr>
<td>FT (F) STUDENT MA (Interpr)</td>
</tr>
<tr>
<td>MH (F) STUDENT M.Sc (Mkt)</td>
</tr>
<tr>
<td><strong>PAIR 3</strong></td>
</tr>
<tr>
<td>GD (M) STUDENT PHD (FIN.)</td>
</tr>
<tr>
<td>YK (M) STUDENT PHD (INOV.)</td>
</tr>
<tr>
<td><strong>PAIR 4</strong></td>
</tr>
<tr>
<td>MU (F) STUDENT PHD (INOV.)</td>
</tr>
<tr>
<td>YT (F) STUDENT MSC (Mgmt.)</td>
</tr>
<tr>
<td><strong>PAIR 5</strong></td>
</tr>
<tr>
<td>CY (F) MSc. (Law)</td>
</tr>
<tr>
<td>JR (F) MSC (Ind. Eng.)</td>
</tr>
</tbody>
</table>
A detailed discussion of the results is also presented which shows that my approach and the underlying motive are a viable alternative to the existing mashup development method and models. (Table 6.2 presents the fact sheet and task times for the experiment).

6.4.1. Initial Observations

Different observations were made during the course of experiment and were being noted down during the whole exercise. I am listing all such quick remarks before presenting the detailed discussion on the quantitative analysis of the feedback.

A. Requirements

One of the most prominent observations made was regarding the timings of questions. All the users with no exception asked different task-specific questions in the beginning of experiment i.e. after the tutorial and before beginning the scenario modelling; once their choice of the scenario was confirmed. This could lead one to do further experimentation and conclude that the requirements elicitation problems that exist in traditional software development where the roles of end users and analysts/developers are differentiated also exist in mashup development.

B. Scenario Selection

65% of the participants selected either scenario 1 or scenario 4 (15 out of 20) with only 15% selecting scenario 3 and 20% selecting scenario 2. Although, the users were asked not to consult the selection in the second task with the other member, the end users preferred to select the scenarios that they had discussed as a possible selection in the beginning. This also points towards an important proposal of my method: Meta design. One of the proposals of meta-design is to share the design among the end users and could lead to an important research prospect in mashup development.

C. Services Knowledge

All the participants did not have any idea of Web Services or APIs and could only relate to the services at a user-level. Most of the participants asked about services that were to be identified in the templates and came up with vague services functionalities to identify the services required in the particular scenarios.

D. Task completion time

Certain observations were made regarding task completion time, and they were obvious even without comparing the actual, precise times.

The treatment group for task 1 took less time than the control group for task 1. Whereas for Task 2, the completion time for control group was less than that of the
treatment group. There are only 2 exceptions to this observation when actual times are noted and both involve scenario 1.

All the participants took less time in the task two as compared to task 1. This makes me believe that the tutorial and experience is an important part of the mashups for end users. In actual, the completion time could not be taken as a comparison between the control and treatment groups. Since, the templates need to be measured for completeness and errors.

### 6.4.2. Detailed Analysis

For the analysis purpose, I have used terms feedback form 1 and control group synonymously for participants who were not using my goal-oriented templates and were listing the requirements for the scenario in plain textual format drafted in outline style. Whereas terms feedback form 2 and treatment group will be used for participants that used my proposed templates.

**A. Task Completion Rates**

For calculating task completion rate, I followed the key and guidelines mentioned in section 6.3. Figure 6.1 shows the graphical representation of the means of the participants in pair form for task 1. The data clearly shows that apart from pair 5, for all the rest of cases, the completion rate which was based on the number of requirements specified by each participant was better for treatment group. The comparison for task 1 can be justified as the measure of the usefulness of the goal templates, since both the participants were working on the same scenario so the bias of the case can be ruled out.

![Task 1 Completion Rate](image-url)
For Task 2 completion rate (fig. 6.2), I used the same formula; however I cannot directly compare the data since the participants were not required to work on the same scenario. However, in terms of number of essential requirements specified, like task 1, the treatment group showed a better score than the control group for task 2 as well. These figures clearly indicate that goal models have been effective in the presentation and acquisition of requirements.

According to the graphs displayed above I can comfortably say that the effectiveness of goal templates is better than the other method. The average of the task 1 and task 2 means and standard deviations are given in the chart shown in figure 6.4. The results are synonymous with the other trends that show the improvement in the task 2 in
terms of means as well as standard deviations which show that the drift from the mean value for the treatment data is less than that of the control data.

![Task 1 vs Task 2 Mean & Standard Deviation for Completion Rate](image)

Figure 6.4 - Task 1 and Task 2 Cumulative Standard Deviation and Means for Control/Treatment Groups

### B. Task Completion Times

Task Completion time is a good measure of the efficiency (Jokela et al. 2003) of a method. The task completion times also endorsed my hypothesis. The trend of control and treatment in task 1 endorsed my hypothesis that using templates aids in specifying the requirements which in turn increases the efficiency and hence reduced the time. However, analysing task 2 completion times led me to believe that task 2 completion times were not only less than the task 1 completion times but task 2 control times were surprisingly lower than treatment times. This raised the following important points:

1. One of the explanations could be experience factor. Since doing task 1 already introduced them to the concept of requirements, goals and services albeit using different methods (control and treatment). The treatment group was introduced to the idea of goals, services and requirements, and it was easy for them to fill in the no-goal method. However, one important point to be noted here is that the time was less for control group but given the completion rates discussed earlier, it cannot be regarded as efficient since the effectiveness of the control group was less than that of the treatment group.
The standard deviations for the control and treatment groups are a very interesting observation but it should be considered with the completeness rates. The task 2 control shows the minimum standard deviation which can be related to the experience factor and simple and straightforward approach for the control group. This can be used to explain the proposal for a spiral life cycle model of the mashup development. Over different iterations, the end users will be performing better with the simpler and quicker methods of mashup development which confirms my idea of allowing the users to use pre-filled templates for the subsequent cycles of their mashup development endeavours instead of starting from scratch. This fact is further confirmed by comparing Task 1 and Task 2 over all standard deviations where it’s lower for task 2.
Custom Metrics for the KAReM Method

Different questions included in feedback form 2 (the treatment group) were specially designed to get feedback on different aspects of the proposed goal based method KAReM. The feedback provided a valuable feedback towards improvement in the method along different lines particularly goal templates and tutorial element.

We intended to explore three dimensions of the method:

i- **Knowledge Clarity**: 4 questions were included in the questionnaire to get the feedback on how clear the knowledge presented in the form of goal templates was. The questions corresponded to each of the 4 themes of ease of filling goal templates, understanding, goal-modelling rules and goal hierarchies. The questions and the responses are mentioned in table 6.3.

ii- **Knowledge Useful**: This dimension sought feedback on goal templates selection, goal templates role in getting the requirements, and tutorial elements.

iii- **Knowledge comprehensive**: The third dimension analysed the completeness of the goal templates and sought users’ responses on templates’ ability to get, derive and represent the necessary set of requirements.

**Knowledge: Clear**

Table 6.3 lists the mean values for the questions that represent the clarity of knowledge. The table below clearly shows that respondents disagreed with the idea that understanding the templates was hard or learn the goal modelling rules and hierarchies was difficult. I was not expecting users to strongly disagree and the means of 2.6 and 2.8 for the negative questions whereas 3.1 and 3.4 for the positive questions is a positive sign that the idea of visual goal models wasn’t an alien or hard concept for the users.

<table>
<thead>
<tr>
<th>Easy to fill goal template (Question 1)</th>
<th>Too long to understand templates (Question 2)</th>
<th>Goal modelling rules easy (Question 3)</th>
<th>Hard to understand goal hierarchies (Question 4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>26</td>
<td>34</td>
<td>28</td>
</tr>
<tr>
<td>3.1</td>
<td>2.6</td>
<td>3.4</td>
<td>2.8</td>
</tr>
<tr>
<td>Neutral</td>
<td>Disagree</td>
<td>Slightly over neutral</td>
<td>Near neutral</td>
</tr>
</tbody>
</table>
An important observation here is standard deviation. While for requirements and tutorials, the standard deviation is below 1 which means, the response can be taken generally as being close to the mean value and hence unanimously representative of the mean. The answer to the goal template selection shows a standard deviation of greater than 1 which is confirmed after analysing the data as 40% of the participants selected extreme values of 4 and 1. This is synonymous with the analysis on errors while selecting the templates. This clearly shows the need of improving the templates. Moreover, the reasons why the responses were not strongly in favour of the method and values not prominently endorsing the ideas of templates can be a good point for the method improvement.

![Knowledge Clarity](image)

**Knowledge: Helpful**

The second custom defined variable to be calculated from feedback form 2 data was how much the knowledge represented in the templates and disseminated through the tutorial was helpful. The participants again gave disagreement responses to the negative questions which is an endorsement of my hypothesis that visual goal modelling helps users better represent their tasks. Similarly, the above neutral value for the role of goal templates in completing the task is a sign that participants responded positively to the idea. This depicts that the goal models played role in enhancing the understanding of the respondents. One last question is regarding the tutorial and it confirms the idea that tutorials are not only important but features that can help the participants during their activity is also desirable.
The overall value coming out to be neutral can be ignored here due to the negative questions.

Table 6.4 - Mean Value for Knowledge Helpful

<table>
<thead>
<tr>
<th>Difficult to select Goal Template (GT) (Question 5)</th>
<th>Hard in beginning the task, GT helped (Question 6)</th>
<th>Understood Scenario and GT as waste of time (Question 7)</th>
<th>Tutorial helpful and Need for more help (Question 8)</th>
</tr>
</thead>
<tbody>
<tr>
<td>26</td>
<td>36</td>
<td>19</td>
<td>37</td>
</tr>
<tr>
<td>2.6</td>
<td>3.6</td>
<td>1.9</td>
<td>3.7</td>
</tr>
<tr>
<td>Near neutral</td>
<td>Slightly Near agreement</td>
<td>Disagree</td>
<td>Slightly Near agreement</td>
</tr>
</tbody>
</table>

The trend can be further confirmed by looking at the standard deviation values represented in a graphical format in figure 6.8.

![Figure 6.8 - Means and Standard Deviations for Knowledge-Helpful Dimension](image)

**Knowledge: Comprehensive**

The degree to which the knowledge represented in templates was comprehensive moved more closer to agreement as it can be seen in the table 6.5 below:

Table 6.5 - Mean Value for Knowledge Comprehension - Feedback Form 2

<table>
<thead>
<tr>
<th>Question 9</th>
<th>Question 10</th>
<th>Question 11</th>
<th>Question 12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goal models helped in getting additional requirements</td>
<td>Had more inputs than requirements</td>
<td>Systematic requirements derivation useful</td>
<td>Problems in input labels</td>
</tr>
<tr>
<td>36</td>
<td>31</td>
<td>38</td>
<td>27</td>
</tr>
<tr>
<td>3.6</td>
<td>3.1</td>
<td>3.8</td>
<td>3</td>
</tr>
<tr>
<td>Slightly Near agreement</td>
<td>Neutral</td>
<td>Near agreement</td>
<td>Neutral</td>
</tr>
</tbody>
</table>
The feedback on the completeness of goal models again has endorsed my hypothesis of using goal models to improve the representation of user expectations. The first question was closer in agreement which shows that users had taken the idea of a goal method which seemingly requires an effort to understand it positively. This is further confirmed by their responses about the goal decomposition for the purpose of deriving and representing the requirements. It is a very important metric for KAReM method. The significance of this result can be considered high since users were aware of both of the methods by the end of task 2 i.e. the task performed with and without the goal model. However, the idea of neutrality for the inputs and goal graph labelling needs further investigation.

![Figure 6.9 - Means and Standard Deviations for Knowledge-Comprehension Dimension](image)

**Cumulative Means for the Custom Metrics**

Figure 6.10 shows the cumulative means after the feedback values for negative questions have been converted into positive values by reversal method. The graph below gives the detailed result of the feedback form 2 according to which all the variables pointing to the fact that the knowledge is clear tend to be closest to agreement in comparison to the clarity and usefulness. Overall, all the respondents mean calculation for all the three
dimensions shows above-neutral responses. It is to be considered as a positive sign since for most of the respondents, it was their maiden first-hand experience with mashups.

![Figure 6.10 - Mean of Knowledge Clarity, Usefulness and Comprehensiveness Respectively](image)

**Errors**

**Template Selection:** One of the most useful feedbacks came from the errors in selecting templates. The scenarios given to the participating students had overlapping in travelling cases. Most of the students who selected travelling cases could not differentiate between navigation, Planning/scheduling and information acquisition templates. For instance, hanging out a trip with friends should have been specified using planning/scheduling template as friends need to be informed about the venue, time. On the contrary, participants selected navigation templates. This was an important realization that it is actually confusing for the end users to look beyond the dominating theme. I intend to use this information in improving the templates as well as their descriptions so that end users can improve their template selection.

**General Perception about Mashups**

Different questions were asked about the idea of mashups to see the responses of the participants at the end of the whole exercise for task and test level satisfaction. Following table (6.6) shows the responses. The overall response of the users for the “understanding of mashup” (Question 13) was dominantly positive exceeding 4 for the mean value (4.3). This was consistent with their outlook and excitement during the exercise. All the non-negative questions about the task being exciting and relevant to participants’ needs scored
a mean of over 4 in both the feedback forms 1 and 2. This strongly confirms the claims that end user mashup development is an important and popular trend among the end users. This underlines the need of suitable tools yet once again.

<table>
<thead>
<tr>
<th>Question Number</th>
<th>Questions About Mashup Experience</th>
<th>Mean Values</th>
<th>S.D</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>Understanding of mashup has increased</td>
<td>4.3</td>
<td>0.67</td>
</tr>
<tr>
<td>14</td>
<td>I can create a mashup design using templates</td>
<td>3.3</td>
<td>0.82</td>
</tr>
<tr>
<td>15</td>
<td>Will prefer tool over this exercise</td>
<td>2.5</td>
<td>0.97</td>
</tr>
<tr>
<td>16</td>
<td>Need of a more detailed tutorial</td>
<td>3.9</td>
<td>0.88</td>
</tr>
<tr>
<td>17</td>
<td>Boring and Tiring Exercise</td>
<td>1.8</td>
<td>0.79</td>
</tr>
<tr>
<td>18</td>
<td>Mashups are useful and I will look at different mashup tools after today’s exercise</td>
<td>3.9</td>
<td>0.74</td>
</tr>
<tr>
<td>19</td>
<td>Will be able to teach about mashups to others</td>
<td>3.6</td>
<td>0.84</td>
</tr>
</tbody>
</table>

Each green column represents each of the 7 questions and the blue bars represent the standard deviation value. The low standard deviation values clearly indicate that the participants were unanimously agreeing.

![Mashup Perception](image_url)
Important Data Observations

Goals and Requirements: The main emphasis of my thesis has been to improve the process by improving the end user centrality. Hence, in this section, I have listed the observations on data made regarding the questions concerning goals and requirements. If I look at the questions concerning the goals and requirements in control and treatment groups, I find some very interesting trends that endorse my hypotheses (table 6.7).

1- Requirements are hard to get but goals are easy to specify for control group (Mean for easy to get requirements= 2.5; Mean for easy to get goals = 3.7). Similarly, a very strong message came from question 6 of form 1 about the need of help to get requirements (Mean: 4).

2- Goal based derivations helped the end users get additional requirements, and as compared to 2.6 mean of requirements, end users found it better to get requirements using templates not only in the beginning (3.6) but as an overall experience as well. This is consistent with the demand of user of having a help of getting requirements and goals available.

Services: The general users these days are familiar with the internet; however, I cannot say that this decreases the importance of availability of Web Services to choose from in a mashup scenario. This fact has already been reported in the literature and is further confirmed by the data:

1- The question whether services should be available (feedback form1-Question 8) while doing the exercise got an overwhelming mean of 4.1 although the participants responded positively to the question about the having prior knowledge of services (Mean of 2.3 to the negative question)

Composition Tools vs Specification Exercise: Another very important response was related to the use of a composition tool vs the exercise. The participants were given the idea of composition tools and were shown Yahoo Pipes interface along with a 2-minute briefing about widgets. The mean response was 2.5 with 50% neutral, 40% disagreement and just 10% responding to agree to prefer the composition tool over specification exercise.
### Table 6.7 - Requirements and Goals Findings

<table>
<thead>
<tr>
<th></th>
<th>Control Group</th>
<th>Treatment Group</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Easy to get requirements</strong></td>
<td>2.5</td>
<td>3.1</td>
</tr>
<tr>
<td><strong>Easy to specify goals (in an outline style)</strong></td>
<td>3.7</td>
<td>3.6</td>
</tr>
<tr>
<td><strong>Need of help to get requirements and goals</strong></td>
<td>4</td>
<td>3.6</td>
</tr>
</tbody>
</table>

**6.5. Guidelines and Conclusions based on Results**

The evaluation results discussed in the previous section clearly indicate that end users when driven systematically by their goals take more keen interest in the process of mashup development and feel less daunted by the technological nature of the task. This validation confirms not only the idea of splitting the mashup development process into four well defined phases but also the involvement of end users during the whole process. This is a very valuable validation of the proposed spiral process model and points towards the need of a more rigorous experimentation in this direction to further confirm and improve the model. Similarly, the use of goal based models and the consequential positive impact of these on end users task completion rates and time is another promising development. Overall, the results of the experiment indicate that the idea of involving end user in the mashup development beyond the technical phase of composing services helps improve the end user orientation of the activity and increases the end user satisfaction. These key findings after evaluation have enabled me to formulate the following guidelines:

1- The mean values of the completion rates and completion tasks demonstrate that goal based methods can be effective and efficient approach for the specification of user expectations. The proposed goal oriented method can provide support to end users with little or no programming experience.

2- End users should be provided with an easy-to-understand and easy-to-perform mechanism for deriving requirements that is natural language based and preferably in a graphical format to help them specify their expectations explicitly. No matter how much clear idea a user has of his/her need, converting it into a program requires effort and interesting visual models can help balance the utility=value/effort equation.

3- End users tend to use and reuse the templates. As the experience with a particular tool is increased, users prefer quicker ways such as pre-filled templates and should
be supported in the tool. This endorses my viewpoint of iterative process model for development of mashups in which the end users can create their own templates as well as reuse the pre-filled templates for on-point modification.

4- The tutorial element plays a substantially important role in any tool and especially in end user development of service mashups. The tutorial should have examples and walk-throughs of the scenarios. The selection of scenarios by the participants in this experiment was greatly affected by their mutual discussion and examples cited during the tutorial.

5- A new scenario that students can relate to can be intriguing for students but they may not be able to specify all requirements related to the scenario as was suggested by the scenario selection in this experiment. All the participants enjoyed the experience, could relate to the scenario and reported no problem in specifying goals but their response on specifying their expectations demanded support. This completely endorses my process model that proposed and specifies separate phases of conception and translation. Most of the participants limited their choices to travelling related scenarios although university accommodation could be the first choice since all the participants were international students.

6- The GOAL templates should be more descriptive and annotated properly for the users to differentiate between seemingly-similar scenarios. If this rule is generalized then all the visual elements, be it pipes or wires (referring to piping and wiring mashing technique respectively), should be having helpful information about their usage preferably in a click-and-display manner.

7- Services should be made available within the tool not only in the form of widgets like micro-services but physical services so that users can choose from them.

6.6. Discussion

This chapter detailed an empirical study to present the validation results of a controlled experiment. The controlled experiment was conducted to validate different dimensions regarding the contributions presented in this thesis. The research contributions such as the one made in this thesis require expensive validation efforts and is a challenging task. The main reason behind this is the nature of contribution which is process as well method based. Hence, the validation was spread across two chapters. Validation by using different scenarios has already been presented in the previous chapter in which the proposed
process and method have been applied to different scenarios. This chapter empirically evaluates the methodological aspects of the proposed approach.

The design of experimentation was a tough task since I proposed a new process-based method for developing mashups and I needed to measure different aspects of the proposed approach. First, the measurement of process aspect was limited since there was no documented process underlining the existing tools. Secondly, I needed to get the End User feedback on the goal-based templates and their suitability for developing mashups. Hence, I designed this experiment in a control-treatment style and conducted the experiment in pairs. I wanted to see the response of the participants against a control group. According to the results, the shift suggested in my thesis towards a more meta-design inspired, goal oriented approach based on principles of abstraction is promising and compels the research community to conduct further research in this area. The improved completion rates and completion times with the proposed idea of templates implies that the approach is effective and efficient. GO-MaDe approach highlights some very important issues in the existing development practices of mashups and recommends that the analysis phase be explicitly supported in the activity and be made user-friendly and interesting by using visual elements that are easy to understand and manipulate by end users. The GO-MaDe approach is a concentrated effort in this regard and presents a novel method that is goal-driven. Finally, the exhaustive validation of the framework requires their development to a full-scale system followed by the user-evaluation. This process probably requires a time period of several years and a concentrated effort of the research community in this direction.
Chapter 7

7. Summary and Conclusions

This chapter presents the summary of the thesis by revisiting and concluding the research in terms of its goals, outcomes, contributions and limitations. At the end of the chapter, the possible areas of future research are also discussed.

7.1. Thesis Summary

Mashups can be regarded as the last mile of Services Oriented Architecture (SOA). End User Service Mashups- the research subject of this thesis- seems to be a great example of “Innovation in assembly” and “end-to-end” principles (O’Reilly 2007) which highlight the importance of creating value by repurposing the existing web components rather than by creating new components from scratch. The need of providing lightweight models to facilitate these principles is further enhanced by the increasing number of internet users\(^{28}\) and availability of internet in the form of mobile internet to masses by ensuring city-wide internet access such as the recent project of LinkNYC\(^{29}\) (Cnet 2015). The needs of end users who might not be IT-expert or even IT-literate in the wake of these recent developments cannot be undermined. However, the issue of enabling these end users to develop meaningful service mashups for themselves by providing the right support cannot be traced to the availability of APIs or Web Services or even availability of technical tool support. The issue is the provision of new development models keeping in view different aspects of end-user service mashups development and considering the needs that this paradigm entails.

The prime motivation driving my thesis has been the improvement of end-user quality of experience by understanding the main problems faced by them while developing applications for themselves. This also helped in defining my second motivation which is another important aspect of service mashups: the guidelines for new tools for development of service mashups. The main research problem discussed in the literature

\(^{28}\) http://www.internetworldstats.com/
\(^{29}\) http://www.link.nyc/
and addressed in this research is to manage the gap between the technical tools and the capabilities of end users.

In a bid to understand the end-users problems, existing literature was consulted to study the state of the art. The vast number of efforts culminating in developing tools for creating mashups was an indicator that the issue is not pertinent to the availability of technical capabilities. This led me to believe that the incumbent technological models need to be redefined and reorganized.

### 7.1.1. Challenges Addressed and Goals

In an attempt to present my research, I have classified the challenges as well as contributions into End User problems and the required capabilities of a new model/system. I am listing the challenges in the following paragraph and goals in the subsequent paragraphs:

1. The main challenge is the apparent impossibility of fully eliminating the gap between the tools involving the technical jargon and requiring technical know-how to a certain extent and the domain expert end users who normally do not have the required IT-related technical expertise. How to mitigate the problems faced by end users due to this and what are the effective strategies to manage the learning curve that is present due to this challenge. This challenge could lead me to both sides of the problem: end user problems and capabilities that a tool supporting the service-based applications should have.

The work reported in the thesis achieves following goals in a bid to address the challenges specified above:

1. To propose a reference framework for evaluating the existing tool support for the development of situational applications. This model also works to guide the future development of tools. The 3-dimensional model of the framework not only considers technical aspects but also takes other aspects that are user-oriented in order to quantify the user-orientation problems.

2. To propose a new model for the mashup development in a bid to highlight the importance of explicitly supporting the end user-related activities of analysis and design. The new model reorganizes the mashup development activities and which,
assisted by the user, supports mashup requirements specification and design by linking goals to service requirements.

3. To propose reference architecture for the goal-based method - GO-MaDe - this supports goal-oriented mashup development by introducing the idea of goal-based templates with the help of the constructs from the domain Theory.

4. To evaluate the process and methodical aspects underlining proposed approach by doing user studies.

7.1.2. Contributions

My research in this thesis focused on the end user orientation problems and how to propose effective methods to deal with the problems. The problems identified in the literature posed different challenges; one of the recurring themes in most of the problems was technical barrier. In a bid to address these challenges and formulate effective strategies, different contributions were made which are discussed in this section. The research reported in this thesis is based on goal-oriented methods and the domain theory which is used for applying the principles of abstraction and generalization to the area of service mashups. The research is also inspired by the principles of Meta-Design which is reportedly the future of end user development research. This thesis used the knowledge encompassed by these components to present its novel contributions. Contributions C1 is related to the classification model and helps in calibrate the state of the art support available for mashup development. C2 and C3 address the improvement of end user mashup development and are part of the bigger framework referred in this entire thesis and GO-MaDe framework. C2 is related to the process of mashup development while C3 and C3.1 are related to the method KAREM developed to represent and model mashup related knowledge. Collectively, C2, C3 and C3.1 serve to improve the end user quality of experience while developing a mashup. Finally, contribution C4 presents some guidelines for mashup development scenarios.

C1: The Mashup Tool Classification Model

To conduct the systematic assessment of existing service composition solutions, I formulated a conceptual model that uses different properties applicable to the mashup tools and classifies them in a bid to understand the state of the art. Different efforts have been made in this direction and tool evaluations have covered different aspects of the mashup tools. These evaluations help in understanding different technologies and
techniques used for these mashup tools. I proposed a 3-dimensional classification model that serves to classify the existing mashup tool support from not only technical aspects but also custom defined mashup criteria and from the perspective of end-user. The classification model also proposes to distinguish the mashup tools based on the level of support provided for developing a composition application. This classification can help in mapping and classifying the research work in this field also making it convenient for both the developers and the end users to present and search for the right tool respectively.

C2: Spiral Mashup Development Model

Currently, mashup development is regarded as a monolithic activity. I have proposed a spiral process model for the development of applications that repurpose the existing data/application logic from multiple sources. The Spiral mashup process model takes into account the lifecycle of the mashup from its conception and continues till after its deployment. Different phases have been proposed in the spiral model to distinguish different activities carried out during the composition process that is the characteristic of end-user applications. These activities are conception, translation, composition and execution. The conception phase relates to the early requirements engineering and relates to the basic mashup information such as the type, genre and if possible the services involved. The second activity is the translation and should lead the end-users from their objective to the candidate services. In the case of goal-oriented method of my research, I am proposing the translation from goals to the detailed requirements that can eventually be fulfilled by physical services. The composition phase is proposed to be regarding selecting from the candidate services based on the compatibility issues and their composition. Finally, in execution phase, the composed mashup is tested for any bugs and deployed, if the end user wants it to be available to general public.

C3: Knowledge Acquisition and Representation in Mashups (KAReM): A Domain Theory Based Goal Modelling Method

After quantifying the end-user orientation problem and proposing a spiral model to explain the mashup development activity, it becomes imperative to support the analysis and design tasks explicitly in the application development process. Hence, following the spiral model (C2), I have proposed a method for supporting the representation of end-user requirements which supports the conception and translation phases of the proposed process. Meanwhile, I have also discussed the challenges that can be addressed by
proposing a goal-based method and the suitability of goals as the deriving mechanism for the end users from main objective to the concrete services. The method, as mentioned in the name of the method, is based on goal-decomposition techniques and adapts its knowledge components to implement the generalization and abstraction principles to help the users derive the analysis and design models of their applications. To the best of my knowledge, this proposed approach is the first proposal to address the mashup development problems by explicitly integrating the end users expectations into the mashup development process as a separate activity. Similarly, with the spiral process proposal, this is the pioneering research in the community of end user development of situational applications of proposing a model that reorganizes the development activity and includes an intertwined instrumentation for explicitly supporting end user requirements-cum-design activity as an inherent feature of the process.

C3.1: Tool Architecture

The last contribution of this thesis is the implementation of the general architecture of KAReM method which is a template based architecture and is based on the components such as query engine and template manager. The architecture presents a proof-of-concept for the method which enables me to demonstrate the practical applicability of my GO-MaDe framework in general and the goal-based method in particular.

C4: Evaluation

In order to understand the end users’ reception of the principles underlying the GO-MaDe framework, I carried out an evaluation. The evaluation which was carried out in the form of a control-treatment experiment confirmed my viewpoint regarding the lack of requirements and goal specifications as one of the main hindrances. According to RE literature in traditional software development, lack of communication standards is the main problem with the requirements engineering process as it involves the technical people who are the system analysts and the non-technical stakeholders, who nonetheless are the domain experts. I believe, this evaluation has confirmed that this barrier also exists between the technical tool support available and the non-technical end users while the latter are developing applications for themselves by reusing the existing services.
7.1.3. Limitations and Future Directions

This subsection sheds lights on the aspects that could be regarded as the limitation of this research. Most of these aspects are not only limitations but could be a potential future work. The GO-MaDe framework and the Mashup Tools Evaluation framework described in this thesis can be extended in different directions to enhance the effectiveness of both the frameworks.

**F1: Extending the classification framework**

The classification model for mashups could be extended into an empirical evaluation framework by interpreting the results of the classification and providing concrete recommendations for end users about a certain tool. The usability criteria could be enhanced by linking with the usability standards and a final future direction could be automating the evaluation of mashup tools such that it takes an input for the tool characteristics and provides a recommendation.

**F2: Extending the Experimental Evaluation**

The empirical evaluation could be further extended to IT-experts and the feedback could be used to incorporate their expert opinions and further enhance the framework. Since the ultimate goal of End user mashup development is to provide automated support to end users, the technical insights of the IT-experts, including the developers and architects could be valuable to provide a seamless experience to the end users.

**F3: Implementation of the GO-MaDe Framework and KAReM Method**

One of the main future extensions could be to implement a user interface for the KAReM approach proposed in this thesis to enhance the usability of the framework. The implementation could be used to automate the goal representation and template generation. The next extension linked to this previous extension can be the development of the GO-MaDe tool that provides support for the all the four activities that have been proposed as part of the spiral process model. This could be done either in the form of an integrated development environment (IDE) with built in support for conception, Translation, Composition and Execution phases. Alternatively, a tool could be developed to provide mechanisms for initial two phases namely conception and translation while providing support for the last two phases namely composition and execution by handling integration with other existing composition tools such as SOA4All.
F4: Extending the Goal Oriented Framework for Enterprise Mashups

Another very important future direction could be the adaptation of the framework for the enterprise mashups. Since, enterprise mashups have well-defined phases to differentiate the roles of component developers and component aggregators, this framework could be extended amicably and support the development of components at the developer level as well as the composition and execution at the end-user level.

F5: Meta-Design based Mashup Development Workspace

One of the main visions behind the proposed GO-MaDe models is the principles of meta-design which proposes to provide a space for the end-users for collaborative design or design-during-use. This method can be used to extend this vision for the mashup development by providing a collaborative space for the creation and modification of the mashup templates and final end user applications. This space could be used for creating new mashup templates or using the existing templates in order to fully specify the requirements for their ephemeral applications.
References


Ardito, C. et al. (2012). End users as co-designers of their own tools and products. Special issue dedicated to Prof. Piero Mussio, 23(2), pp.78–90.


Appendix

INITIAL SELECTION FORM

Name: ___________________________________________ _________

Occupation: _____________________________________________

Last completed Degree: _____________________________________________

IT Knowledge Level: □ Beginner □ Intermediate □ Advance

Programming Experience: Please select the most appropriate option in the following table:

<table>
<thead>
<tr>
<th>Never Programmed</th>
<th>Do some fun programming occasionally</th>
<th>Only macro-level programming in Word and Excel</th>
<th>Written small programs</th>
<th>Developed some big projects</th>
<th>I am an expert programmer</th>
</tr>
</thead>
</table>

How would you define your current domain of expertise? __________________________
________________________________________________________________________

What programming tools have you been using and please explain the level of usage? Also mention any other computer-related course studied and duration of the course?

<table>
<thead>
<tr>
<th>Programming Language/Computer or Software course</th>
<th>How Often Used?</th>
<th>Beginner/Intermediate/Advanced</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Have you heard about web/service mashups before? | ☐ YES | ☐ NO
---|---|---
Have you ever developed a mashup before? | ☐ YES | ☐ NO
Which mashup editors have you used before? | 1- | 2-

Please let us know if you would like to be notified of the result by providing your email below:
Scenarios

From the following 5 scenarios, select 1 and complete the task as told.

1- You are selected for a conference in a city that you have not been to before. You need to find out about different things before travelling. You need to find about different tourist attractions related to the city and the temperature/weather and a lot of other things.

2- You are selected for a university abroad and want to find out about living there before arriving there. You need to select the right kind of accommodation with required number of rooms and want to have your places of interest within the required distance.

3- You want to create your own personal sports page. You are fan of a particular sports (let’s say X) and want to get the right kind of news, and multimedia content related to sports X.

4- Suppose a group of friends want to travel to France for a holiday. They want to get the right kind of hotel and flights and want to make sure that all their needs are met (airlines, business/economy class/cheapest fare).

5- You have a personal website setup. You want to have a section where the latest news about your home country is available on the web site.

Select the scenario that you want to proceed with, put its number below and please mention if you can describe the mashup type.

<table>
<thead>
<tr>
<th>TASK 1</th>
<th>TASK 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCENARIO NUMBER: _______</td>
<td>SCENARIO NUMBER: _______</td>
</tr>
<tr>
<td>Mashup Type □ Data □ Presentation □ Can’t tell</td>
<td>Mashup Type □ Data □ Presentation □ Can’t tell</td>
</tr>
</tbody>
</table>

Control Exercise

GOAL 1

SERVICE 1

REQUIREMENT 1

REQUIREMENT 2

REQUIREMENT 3

GOAL 2

SERVICE 2

REQUIREMENT 1

REQUIREMENT 2

REQUIREMENT 3

GOAL 3

SERVICE 3

REQUIREMENT 1

REQUIREMENT 2

REQUIREMENT 3

GOAL 4

SERVICE 4

REQUIREMENT 1

REQUIREMENT 2

REQUIREMENT 3

**************Thank you for your valuable time and feedback***************
Feedback Forms

Name (optional): ____________________________________________

Age: _______________________________________________________

Email Address (optional): ____________________________________
Feedback Form 1
On scale from 1 to 5 (1=strongly disagree, 5=strongly agree) please provide your feedback below.

<table>
<thead>
<tr>
<th>QUESTIONS</th>
<th>Strongly Agree</th>
<th>Agree</th>
<th>Neutral</th>
<th>Disagree</th>
<th>Strongly Disagree</th>
</tr>
</thead>
<tbody>
<tr>
<td>1- I found it very easy to get all the requirements.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2- It took me too long to understand the scenario.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3- I understood the scenario but found it hard to identify the subsequent requirements.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4- I do not know of any services available online.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5- Specifying goals was easy.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6- There should be some kind of help to begin filling the requirements and goals fields.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7- I know about a lot of websites that offer the particular services but have no idea about the individual services provided by them.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8- Having a list of services to choose from with a brief description of each service will help greatly.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9- Overall, I enjoyed the experience of doing the task.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
A Framework for Improving End-User Orientation of Service Mashups

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>10- The idea is interesting but the exercise was too boring.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11- I can relate to the scenarios and needs that were specified in the scenario.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12- It was a good and simple exercise with no complications.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Can you suggest how can this task be improved?

__________________________________________________________________________

__________________________________________________________________________

For any additional comments, please feel free to use the space provided below regarding any aspect of today’s exercise. Please ask for extra sheet, if required.

__________________________________________________________________________

__________________________________________________________________________

__________________________________________________________________________

__________________________________________________________________________

__________________________________________________________________________

__________________________________________________________________________

__________________________________________________________________________

__________________________________________________________________________

__________________________________________________________________________

__________________________________________________________________________

**************Thank you for your valuable time and feedback***************
## QUESTIONS

<table>
<thead>
<tr>
<th></th>
<th>Strongly Agree</th>
<th>Agree</th>
<th>Neutral</th>
<th>Disagree</th>
<th>Strongly Disagree</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-</td>
<td>I found it very easy to fill the goal template with the right requirements from given scenario.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-</td>
<td>It took me too long to understand the idea of templates.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3-</td>
<td>The goal modelling rules were easy to learn.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4-</td>
<td>It was very hard for me to understand the goal hierarchies.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5-</td>
<td>It was difficult to map the right template for the selected scenarios.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6-</td>
<td>I hardly understood the problems at beginning but then drawing goal models helped in generating ideas.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7-</td>
<td>I had understood the entire scenario in the beginning and goal models were a waste of time.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8-</td>
<td>Tutorial in the beginning helped and it would have been more enjoyable if there was more help available during the task.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9-</td>
<td>Goal models helped me in getting additional requirements which were not known previously.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10-</td>
<td>I had additional inputs which I couldn’t specify using the given templates.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11- Specifying my requirements one by one in a goal fashion helped me systematically think about my final mashup.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12- I could easily fill all my requirements with the given templates but had problems in understanding the inputs labels.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Feedback Form 2

On scale from 1 to 5 (1=strongly disagree, 5=strongly agree) please provide your feedback below.

<table>
<thead>
<tr>
<th>QUESTIONS</th>
<th>Strongly Agree</th>
<th>Agree</th>
<th>Neutral</th>
<th>Disagree</th>
<th>Strongly Disagree</th>
</tr>
</thead>
<tbody>
<tr>
<td>13- I feel my understanding of mashups has increased after today’s exercise</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14- I could create a mashup design for myself easily by using goal models and templates</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15- I would create a mashup directly by using a tool rather than having to do this exercise.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16- I believe a more detailed tutorial would have helped me better understand these models</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17- I got tired and bored of the entire exercise</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18- I believe mashups are very useful and I will go have a look at different tools after today’s exercise.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19- I will be able to teach these concepts to others</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Which mashup template you chose and why?

________________________________________________________________________
________________________________________________________________________
________________________________________________________________________
________________________________________________________________________
________________________________________________________________________
________________________________________________________________________
For any additional comments, please feel free to use the space provided below regarding any aspect of today’s exercise. Please ask for extra sheet, if required.

***************Thank you for your valuable time and feedback***************