Engineering For Evolution Of Software To A Production Environment

THESIS

Presented in Partial Fulfillment of the Requirements for the Degree Master of Science in the Graduate School of The Ohio State University

By
Pallavi Chalasani
Graduate Program in Computer Science and Engineering

The Ohio State University
2014

Master’s Examination Committee:
Dr. Jay Ramanathan, Thesis Advisor
Dr. Rajiv Ramanath, Committee Member
Dr. Caroline Wagner, Committee Member
Abstract

Many researchers wish to collaborate within universities and industry. A hindrance to this is that researchers do not have enough information about each other at their disposal to try and form new academic collaborations. The main contribution of this thesis is providing software engineering guidelines to move prototype software to production software for this scenario. This is achieved using a case study. The case study is the process of moving a prototype for the graphic visualization of author networks for a) Extreme Scale Organizational (ExSO) Collaboration to b) The Discovery Tool which is a query based data exploration tool on a university data warehouse called KMData.

The importance of this case study is the frequent pattern in the software development process to convert a proof of concept tool that is barely working to a product that is sustainable, modifiable and secure. The experience for the guidelines is derived from the specific details of moving the discovery tool to production, as part of this research work. To achieve production capability the goals went from presenting relationships between authors etc., for exploration to expecting the user to know the keywords to query.
To provide the right kind of information, the discovery tool must provide a query interface to the most comprehensive information of all the experts in The Ohio State University to help researchers within the university or outside to forge new collaborations. To achieve this goal the important aspects are

1. Functional: A query feature based on the patterns that appeal to the user and constitute the profile of every professor, expert and researcher in The Ohio State University.

2. Non-Functional: Ways to make a software application that works with speed, secure, easy to use and is engaging.

This contribution is to make both these aspects concrete in this thesis through explicit process guidelines and case study illustrations as learning assets. We illustrate the requirement analysis of the discovery tool, and then we show the planning that went to design the application. The design, implementation and testing details are in the next section. Finally we conclude with the evolution examples thus showing the power of the application and future work.
Dedication

This thesis is dedicated to my family, Latha & Pradeep Chalasani, Ravali & Satish and to my uncle Pratap.

You are my strength.
Acknowledgements

I would like to first thank my advisors Prof. Jay Ramanathan and Prof. Caroline Wagner for their continuous support and guidance. The confidence you had in me and the constant encouragement I received from you helped me grow as a student and as a person. I would also like to thank Prof. Rajiv Ramnath for his advice and guidance throughout my graduate studies. I am indebted to Prof. Jay and Prof. Rajiv for making me a part of CETI. I have learnt more software development here than I ever did before. Thank you for the opportunity. I would like to thank Vedu, John, Joan, Bryce, Mohsen and David from the KMData team for their support, insights and for welcoming me into their team. It was a great learning experience for me to have worked with you all.

I would like to thank Krishna for being an amazing friend, helping me with my thesis and project right from the beginning, helping me keep my cool during tense moments, helping me prepare for interviews & classes and above all inspiring me to work harder. I couldn’t have done this without you. Thank you Kalyan for being so nice and so helpful to me and for being such a good friend. OSU feels like HOME because of you. I would like to thank my roommates Akanksha, Ankita, Puja, Janani for being my family away from home.
I would also like to thank my friends Sundeep, Aveek, Naman, Gaurav, Arka, Abhishek, Siddharth, Gautam, Achal and Vikram for your unwavering support and belief in me. Amrutha, thank you for your constant support. I know I can always count on you.

Lastly, I would also like to thank Mr. Sachin Tendulkar for inspiring me to chase my dreams. You will always be an inspiration for millions of Indians like me.

This work was done under the aegis of the Center for Experimental Research in Computer Science at Georgia Tech and The Ohio State University, funded by the National Science Foundation's I/UCRC program (NSF IIP 0753710). This work is funded by the Battelle Center for Science and Technology Policy through the John Glenn School of Public Affairs. We wish to acknowledge the helpful ideas by CETI students, colleagues, members, and sponsors.
Vita

2008 to 2012 ............................................... B. Tech, Computer Science, ANITS,

Andhra University

2011 to present ............................................. M.S. Computer Science and

Engineering, The Ohio State

University

Fields of Study

Major Field: Computer Science and Engineering
# Table of Contents

Abstract ........................................................................................................................................ ii

Dedication ...................................................................................................................................... iv

Acknowledgements .......................................................................................................................... v

Vita ................................................................................................................................................ vii

Fields of Study ................................................................................................................................ vii

Chapter 1: INTRODUCTION ........................................................................................................... 1

  Problem Statement .......................................................................................................................... 2

  Solution Statement .......................................................................................................................... 3

  Primary Contributions of the Thesis ............................................................................................... 4

  Goals of the Thesis and the Discovery Tool .................................................................................. 5

CHAPTER 2: BACKGROUND ........................................................................................................... 8

  Need for a Collaboration Tool: ....................................................................................................... 9

  Levels of Collaboration: ................................................................................................................... 9

  Scenarios using discovery tool: ....................................................................................................... 11

  Stakeholders and their benefits: ...................................................................................................... 14

  RELATED WORK ............................................................................................................................ 15

  Collaboration Networks .................................................................................................................. 15

  Discovery Tool in Comparison ...................................................................................................... 16

CHAPTER 3: PLANNING ANALYSIS ............................................................................................... 18

  Functional Requirements .............................................................................................................. 18

  Design ......................................................................................................................................... 19

  System Components ...................................................................................................................... 20

  Data .............................................................................................................................................. 21

  Non-Functional Requirements: .................................................................................................... 22

    Usability: .................................................................................................................................... 23

    Reliability: .................................................................................................................................. 24

    Performance: .............................................................................................................................. 25

    Supportability: ............................................................................................................................ 25

    Adaptability: ............................................................................................................................... 25

    Maintainability: .......................................................................................................................... 26

    Portability: .................................................................................................................................. 26

  Capacity, current and forecast: ...................................................................................................... 26

  Security and Privacy: ...................................................................................................................... 26

  Dependency on other parties: .......................................................................................................... 26

  Configuration management: .......................................................................................................... 27

  Challenges Faced: .......................................................................................................................... 27

viii
List of Figures:

Figure 1: Ohio State University and Industry Collaborations 7
Figure 2: Levels of Collaborations 9
Figure 3: Use Case 1-Professor Search 11
Figure 4: Use Case 2- Department Search 12
Figure 5: Use Case 3- Expert Search 13
Figure 6: Non-Functional Requirements Tree 23
Figure 7: ATAM as a graphical representation 30
Figure 8: System Architecture 32
Figure 9: KMDATA Architecture Diagram 35
Figure 10: Software Interaction Diagram 1-Professor Search 37
Figure 11: Software Interaction Diagram 2-Expert Search 39
Figure 12: Profile Building Component 40
Figure 13: Expert Building Component 41
Figure 14: Author Profile page 47
Figure 15: Co-Author network 48
Figure 16: Co-Author network color-coding feature 49
Figure 17: Friend of Friend network 50
Figure 18: Department search results 51
Figure 19: Expert Search Results 52
Figure 20: Feedback Form 53
Figure 21: Word Cloud result for CSE department 55
Figure 22: Pie chart representing the inter department collaborations 57
List of Tables:

Table 1: Related Work 14
Table 2: Technologies used 18
Chapter 1: INTRODUCTION

Before we begin to explore the software development life cycle, we should address the major question: What is the necessity to follow a certain engineering process? The common choice that many students face is to choose whether it is enough to just get the code working or invest in a software engineering practice that would get the code working right. The main aim of an application is to not only get an idea to work. It is to also convert a certain set of requirements into a sustainable application that has quality attributes like performance, security and ability to be developed further. Hence in this thesis we analyze how following good design practices help change a student created prototype to a well functioning production level software application.

The Discovery Tool started as a prototype named Extreme Scale Organizational (ExSO) Collaboration [1] to visually explore the idea of collaboration networks. We attempt to see if it is possible to find future collaborators of an expert. ExSO is a Visualization tool. It was built in an agile model using two steps of software design i.e., Implementation and feedback. Every time a client feedback was given, the application was revisited and the necessary changes were made. In the end, ExSO became an exploration prototype to see if an application like this can be built. On the other hand, the
Discovery Tool is a data exploration tool that follows the agile model with respect to the stakeholders' requirements. It follows the software development life cycle following the process of Requirement Analysis, Planning Analysis, Design Analysis, Implementation and Evolution. It is a query-based application to find meaning in and visualize OSU's data.

**Problem Statement**

The problem addressed in this thesis is the following:

1. What software engineering aspects have to be considered in moving from a prototype to a production system?

2. Is there a methodology that you can follow to aid such a shift?

Useful software systems are often complex. To remain useful they need to evolve with the end users' need and the target environment. In any student research project there is always a choice between “getting the code to run” vs. “getting the code to run right”. This choice transcends beyond the academia to the professional world as well due to various reasons like deadlines, resource availability etc. This decision may hinder the project to achieve quality attributes like performance, security and modifiability. The question addressed here in this thesis is:
Is there a way to culminate the entire process of making a software application into a set of standard guidelines that can be followed by future students while developing a software application?

**Solution Statement**

Given the problem that we are trying to address, the solution should be designed in a way that acts as a methodology. The answer to the kind of methodology to be followed, the aspects of software engineering to keep in mind while developing a product is the major part of this thesis. The basic steps are:

1. Use software development life cycle for analysis following different guidelines at each step of software development
2. Develop a methodology basing on the previous analysis and design a framework
3. Use that developed framework and methodology to test on a new feature of the application being built.

These attributes help to ensure that further project development is smooth. Hence making architectural analysis is important and must be added as a key practice for the project. Architectural Analysis not only satisfies particular quality goals but also provides insight to how these qualities interact with each other and
how they trade off against each other. We use different guidelines in each stage of software development in the case study.

In this thesis we concentrate on using the case study, the discovery tool, to assert the importance of software engineering.

Primary Contributions of the Thesis

1. **Case Study**: Moving ExSO visual exploration prototype for Scopus files to Discovery Tool, which is a query-based feature to KMDa, as a production system.

ExSO is a framework that provides the architecture for developing the features of a system that helped to accelerate the academic collaboration process. The framework includes intelligent techniques to visualize the vast amount of academic data. ExSO was based on Scopus. Discovery Tool is a software application that is built upon ExSO to handle exclusive OSU data as a feature to KMDa.

2. Using this case study to provide articulate lessons learned and provide a methodology that:
   a. Emphasizes specific aspects of software engineering process critical to production applications
b. The incorporation of functional and non functional requirement features during the deployment process

In this thesis, we illustrate each step of the software development life cycle on the application. We highlight the differences of the same stage in ExSO and Discovery Tool, proving that usage of software engineering over an application makes it more stable and secure.

3. Achieving Scalability for production system: such as sifting through a large number of records to pull out the data that is required.

Data Analysis is the process of systematically applying statistical and/or logical techniques to describe and illustrate, condense and recap, and evaluate data [5]. KMData has more than 2 million records of data exclusive to the Ohio State University [12]. The Discovery Tool, being a data exploration tool, explores these large numbers of records to find meaningful information according to the search query of the user. The contribution of the discovery tool is to serve as an example for a data exploration and visualization application to show collaboration networks exclusive to a particular university.

Goals of the Thesis and the Discovery Tool

The goals of the Discovery tool is to:
1. Find the best expert from OSU in a particular field

   The Discovery Tool is a data visualization application to explore the academic collaboration networks in Ohio State University. The goal is to find the best expert in a particular field basing on the research term, or a department or the expert's name itself.

2. Provide maximum information to aid in successful collaborations

   Another goal of the discovery tool is to help forge successful collaborations. Generally to form collaboration, an expert would have to physically sift through information like authors of various publications in their field of interest. The discovery tool provides the safe information in an automated fashion to experts looking for collaboration, to students looking for experts in an area for an advisor and to industry personnel looking to collaborate with the academia. It provides additional information like co-author network, friends of friends of an expert, the experts in a department etc.

3. The software engineering processes to be followed to convert a prototype to a software product

   As mentioned above, it is now not enough to just implement an idea. It is necessary to create an application that satisfies not only
functional requirements but also more importantly non-functional requirements like usability, modifiability, reliability and performance.
CHAPTER 2: BACKGROUND

Our first step in the software development life cycle is finding and analyzing the requirements of the application, in this case the discovery tool.

First we check some statistics about Industry Collaborations in OSU:

Figure 1. Ohio State University and industry collaborations [3]

This map in figure 1 shows all the collaborations that the Ohio State University has with various industries in USA. This map is taken from the
research website of OSU in which it is specifically mentioned to contact the Industry liaison office to contact an already known collaborator. How would we find a new collaborator then? Also, if industry personnel are collaborating for the first time with the university or with a certain department how would he know who the experts are?

**Need for a Collaboration Tool:**

Scientific research has shifted from individually oriented research to a team based collaborative effort [2]. Teams of different researchers are brought together to solve large-scale problems. Fields like bio informatics, medical computer vision require that academicians from different departments collaborate with each other.

In the given age and time, Many branches of science have shifted from individually oriented research to team based scientific collaborations. It has become a more effective setting to involve people from different research areas together especially for areas like medical vision, bio informatics etc. Hence there is a need for a collaboration tool.

**Levels of Collaboration:**

Now that it is established that we need a collaboration tool, our next question is to find the kind of tool we require. Our main aim here is to connect
people. Macro level collaboration provides insight about broad patterns of scientific collaborations in terms of effect and impact. Micro level collaborations concentrate on forming person-to-person networks, where people are represented as nodes [4]. The figure 2 makes this division more explicit. Hence, developing a tool that concentrates on micro level collaborations, in which nodes represent participant members and link their relationships, satisfies our requirement.

Figure 2: Levels of Collaboration [4]
Ohio State’s research is cited above the world’s average and there are various departments in the university that collaborate with other research institutions out of the university. All these constitute under macro collaborations. Given the academic and research prowess of a university like Ohio State, collaborations within the university will be more cost and effort effective than collaborating with research groups outside the university.

1. In the Discovery Tool, we have co-authors and friend of friend networks that precisely follow the principle of micro level collaborations.

2. We use visualizations to summarize interaction patterns between individuals by examining their communications.

3. In Discovery Tool, we explore the person’s skills (key areas, expertise) to form a profile. This is then used to connect them to other members who share the same profile, thus creating a micro level collaboration.

**Scenarios using discovery tool:**

In order to grasp the value to be gained by providing the innovation, we need to enumerate several scenarios that quantify the benefit in detail. At the high level, we list three key user scenarios of the system. These scenarios summarize the role of the three primary users as listed below:
**Scenario 1**: This scenario/use case shown in figure 3 depicts a user using the name search feature of the discovery tool to search for an already known professor. Once the user enters the name of the expert he’s looking for, the database is searched for an expert with the same name and his/her profile is returned.

![Diagram of Scenario 1: Professor Search](image)

Figure 3: Use Case 1: Professor Search

In the end, he saved months of time that he can now use for research, and this is where the key value of the newly implemented system lies.
**Scenario 2:** This use case shown in figure 4 depicts a user using the department search feature to explore a list of all experts in a particular department. It is not required that the user enter the full name of the department.

![USE CASE 2: Department Search](image-url)

Figure 4: Use Case 2- Department Search

This is specifically useful for new students who are looking to see the experts in a particular department.

**Scenario 3:** This use case shown in figure 5 depicts a user using an expert search feature. In this, if a user enters a keyword, a list of experts in that area is shown.
The experts are arranged in a way so that the person with the maximum research in that area shows up first.

**Stakeholders and their benefits:**

We also analyze as to who the stakeholders are and how they would benefit from this application. The stakeholders for such kind of collaborations tools are professors, industry personnel, PhD students and potential students. Owing to team based collaborative research, finding collaborators to work with increases their productivity. Applications like discovery tool help to speed up the process of finding one. Also, a PhD student or a potential student can look up to
see who the experts are in their area of interest, see their publications to find out their current research area. They can also see a list of all professors in their department helping them to choose a potential advisor.

**RELATED WORK**

**Collaboration Networks**

These are the other applications that function like how the discovery tool works but have their own drawbacks. Microsoft academic search and Google scholar for that matter fall under the macro collaboration category as they give information about the broad area of research going to in a particular field rather than being specific. Our main comparison lies with ExSO, as it is the prototype on which the discovery tool was built. The various existing applications built which explore academic collaborations are given in the following table.
<table>
<thead>
<tr>
<th>NAME</th>
<th>WHO</th>
<th>FUNCTIONALITY</th>
<th>TECHNOLOGY</th>
<th>BOTTOM LINE</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXSO – Expert Finder</td>
<td>OSU</td>
<td>-OSU as a starting point</td>
<td>-Open Source</td>
<td>Overall best solution for university needs to capture value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-Profile Based Expert Finding</td>
<td>-Java Based Spring Framework</td>
<td></td>
</tr>
<tr>
<td>Microsoft Academic Search</td>
<td>Microsoft</td>
<td>-Search Engine</td>
<td>-Proprietary</td>
<td>Too generalized to be useful</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-Good Visualizations</td>
<td>-SilverLight</td>
<td></td>
</tr>
<tr>
<td>VIVO</td>
<td>Cornell</td>
<td>-Similar to OSU:PRO</td>
<td>-Open Source</td>
<td>Not profile based</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-Good Visualizations</td>
<td>-Semantic Web</td>
<td></td>
</tr>
<tr>
<td>Harvard Catalyst</td>
<td>Harvard</td>
<td>-No good Visualizations</td>
<td>-Open Source</td>
<td>Limited influence</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-Limited to Harvard</td>
<td>-C# .NET</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Related work [1], [10]

**Discovery Tool in Comparison**

In comparison to the other applications, the discovery tool functions better due to the following reasons. As said ExSO was the prototype on which the discovery tool was built. Hence the following sections of this thesis would largely draw a comparison between both applications and push the point that following the software design practices as explained later will eliminate the drawbacks that ExSO had.

The Discovery Tool has transformed from a prototype to a software application that has:
1. Speed: Response Time is around 5 sec when compared to ExSO’s 120 sec.

2. Data: The KMData has around half million records exclusively of Ohio State data.

3. Visualizations: The discovery tool has minimalist visualizations showcasing only the necessary information.

The discovery tool showcases micro level collaborations that help a potential student as well as an industry investigator hence making it an ideal university specific micro collaboration software application.
CHAPTER 3: PLANNING ANALYSIS

In the planning step of the software design life cycle, we develop a framework for production software engineering. We explore and compare the functional and non-functional requirements of ExSO and discovery tool. This application is built on a prototype; hence we see what are the drawbacks of the prototype and why were they caused. Next we put forth a set of guidelines to be followed during the design phase to develop a perfectly functioning application.

ExSO was the visual collaboration prototype on which Discovery Tool is built. It was designed to help the user explore the query result. Discovery Tool aims at providing minimalistic visualizations with highly comprehensive data, thus putting the availability of the right data to the user before visualizations. It resulted in the user role being greater in forming the exact query.

Functional Requirements

It is then important to identify the functional and non-functional requirements of the application. In Software Engineering, a functional requirement defines a function of a system or its component. Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed
to accomplish [6]. In the following section we compare how functional requirements were considered and met in ExSO and the Discovery Tool. Here ExSO is referred to as the initial prototype.

1. Design

The main requirement of any large-scale software system is that it should be modular. Any software application is built in different modules that in turn depend on each other for the functionality. This helps us in the long run, if we wish to add new functionalities to the software. There is no necessity to change the entire code, instead we can add to the specific module where the change is required. An ideal software system has low coupling and high cohesion. Coupling and cohesion are the primary attributes that led to procedure oriented quality systems. Coupling is a measure of the interdependencies between different modules and cohesion is a measure of the binding of the elements within a single module. To be rated as well designed, a procedure-oriented system has to have low coupling properties in terms of few interdependencies between modules and high cohesion properties in terms of strong bindings between the elements within a single module [5].
a. Initial Prototype-ExSO: EXSO was less modular. There was a high coupling. The application was designed to handle in memory data and could not handle large amount of data.

b. Final Application-The Discovery Tool: The discovery tool is more modular when compared to ExSO. There is also high cohesion and low coupling. Each module like visualizations, the profiles, and back end are all individual entities.

2. System Components

The initial prototype and the final in-production version of the application are designed using the software as shown in table 2:

<table>
<thead>
<tr>
<th>VENDOR</th>
<th>TECHNOLOGY</th>
<th>VALUE PROVIDED</th>
<th>USAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SpringSource</td>
<td>SpringSource MVC API</td>
<td>Provides a very capable framework with to structure this product</td>
<td>HIGH</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>PGAdmin3</td>
<td>GUI to use Postgre database</td>
<td>HIGH</td>
</tr>
<tr>
<td>OSU</td>
<td>KMDATA</td>
<td>Provides the basis for our system’s data</td>
<td>HIGH</td>
</tr>
<tr>
<td>VMware</td>
<td>TC Server</td>
<td>Java / SpringSource Tomcat execution environment</td>
<td>HIGH</td>
</tr>
</tbody>
</table>

Table 2: Technologies used [9]
a. **SpringSource**: Spring helps build simple, portable, fast and flexible JVM-based systems and applications. Spring framework is an open source application framework for the Java platform. It is a programming module for enterprise Java applications.

b. **PostgreSQL**: PostgreSQL is a powerful, open source object-relational database system. As a database server, its primary function is to store data, securely and supporting best practices, and retrieves it later, as requested by other software applications. The PostgreSQL essentially manages KMDATA.

c. **KMDATA**: KMDATA provides a virtual repository to enable analytics and foster the sharing of research data. Using the KMDATA we can access over a million records of academic and professor related data exclusive to OSU [12].

### 3. Data

**Initial Prototype-ExSO**: ExSO used data from Scopus. Scopus is a bibliographic database containing abstracts and citations for academic journal articles. It consists of data from major peer reviewed journals [9].
Final Application-The Discovery Tool: The discovery tool uses KM Data. The Knowledge Management Database (KM Data) provides the Ohio State University publication records.

Non-Functional Requirements:

Non Functional Requirements describes aspects of the system that are not directly related to the functional behavior of the system. It is a constraint on the system that is not directly related to the operation of the system. They answer how software will work. Another important aspect of the non-functional requirements is that architectural modifications are done basing on non-functional requirements. Non-functional requirements include a broad variety of requirements that apply to many different aspects of the system, from usability to performance.

The non-functional requirements in general can be categorized as follows:
Out of the non-functional requirements mentioned above, we consider only those that are specific to Discovery Tool.

**Usability:**

Defines the ease with which a user can operate a system. Discovery Tool is very easy to use. The user has to just enter his search query that then gives the corresponding result. From there, it is very easy to navigate
around different sections. Being a tool that has minimalistic visualizations, it is an engaging application and has a high emotional value as it has university specific data and can be used by all experts, students and future collaborators related to Ohio State University.

**Reliability:**

Ability of the system to perform its required functions for a specific period of time

a. Frequency/Severity of Failure:

ExSO could handle at the most ~20 queries after which it would crash. On the other hand, discovery Tool can handle any number of search queries. It has a rigorous testing system before deployment and also another system that monitors its response times to prevent failures. Discovery Tool functions on a KMData that has a backup database, in case of any failures

b. Robustness: The degree to which a system can function correctly in presence of invalid inputs

c. ExSO would crash without any error message and it required that the server be restarted to get it to work again. Discovery Tool catches the error and shows a error message to the user, generally correcting the input, and functions normally again
Performance:

Concerned with quantifiable attributes of the system

a. Response Time: How quickly the system reacts to a user input
   i. ExSO: Close to 120 seconds
   ii. Discovery Tool: 5 to 6 seconds

b. Throughput: How much work a system can accomplish within a specified amount of time. Discovery Tool has high throughput when compared to ExSO

c. Accuracy: ExSO had accuracy issues with people having similar last names, as it was a last name-based search. Hence, it had ambiguous results. Discovery Tool is a full name based search that eliminates the confusion

Supportability:

Concerned with the ease of changes to the system after deployment

Adaptability:

ExSO is written in a highly coupled fashion. Due to this, addition of any new feature requires the whole code to be modified. Discovery Tool is modular. The changes to be done are limited to that module itself. The modules are the visualization modules, the query module, controller module.
Maintainability:
Ability to change the system to deal with new technology
As mentioned, with Discovery Tool being modular it is easier to adapt to changes within the system as well as in the introduction of new technology

Portability:
Portability can be defined as the ease with which a system can be transferred from one environment to the other.
ExSO and Discovery Tool are both highly portable. Both the tools do not function well if used in browsers with older versions of IE.

Capacity, current and forecast:
We can access around 1.3 million records in the Discovery Tool. The number of records will increase as more data sources get added. This will not affect the working of the application.

Security and Privacy:
We are only dealing with the public data provided by the authors themselves.

Dependency on other parties:
All software used for the application is open source.
**Configuration management:**

The code is stored in github. Before the deployment of a version Jmeter load testing is done on the application to check its performance.

**Challenges Faced:**

Before developing the discovery tool, it was necessary to test exso rigorously to find out where it failed and why. We explain the drawbacks in this section. How these drawbacks were approached is answered during the implementation phase of the discovery tool. The planning phase is to find out the errors and then to sit with the stakeholders and plan the most efficient way to remove them.

1. Application crashing

   Every time something is searched for, the database is queried and the results are returned. In case that search does not return a result, the application should show a message. Instead exso would just crash and the connection between the front end and the database would be lost.

2. Speed

   The KMDdata is a huge database pulling in data from different sources. Hence views of all the necessary information were created. Each of these views
involved over 17 joins over them and each of the queries did pattern matching over the search query. This is a very wrong coding practice that increased the search time substantially to almost 120 sec. The server couldn't take this load as well which often resulted in the crashing of the application

3. Duplicate Data

   The data used for EXSO was from Scopus. Scopus has different information and is categorized into tables quite different from KMData. The data in KMData is manually entered which resulted in duplicate data, multiple entries, wrong data, holes in the table, slight spelling mistakes of the same data etc.

4. Legacy Code Issues

   The initial software application started out as a prototype to test an idea. Hence the code was structured so as to test the functionalities. Issues with the code have been found. They are as shown. These legacy code problems took a long time to understand and solve which wouldn't have been the case, if there were proper documentation

   a. No proper commenting

   b. It was structured in a way to process in memory code

   c. Couldn’t handle large amounts of data

   d. It was designed to handle scopus specific data structures

   e. It couldn’t handle the holes in the new database structure
5. Constraints of the existing environment

It took meeting with the client and a demo to the client to understand that Battelle uses IE 8, which is the older version that is not supported, by the visualization library used in this application. Visualizations are not effective with older versions of Internet Explorer. e.g. IE-8. Battelle is a high security environment hence migration to newer versions of browsers was a problem.

Judging from the challenges faced, it can be observed that most of them could have been avoided if a certain design process was followed from the beginning. This design process is highlighted in this thesis. Writing code just to get it right is no longer an option to create good last software. It was observed that the problems with the front-end application were majorly because of the code structure that was not flexible enough to handle a different and larger back end database.

If the front end is not flexible enough to provide different functionalities to the user, there is no use even if the backend contains all valuable information. Hence the main requirements were a step-by-step process to design a proper software engineering framework following certain rules, to design the application and follow MVC principles to code in a way so as to maintain modularity, which takes us to ATAM.
Architectural Tradeoff Analysis Method (ATAM)

ATAM is a risk mitigation process used early in the planning phase of the software development life cycle. The Architectural Tradeoff Analysis Method is a method for evaluating software architectures relative to quality attribute goals [2]. ATAM evaluations expose architectural risks that potentially inhibit the functioning of a software application. ATAM not only gives a methodology to perform risk mitigation but also specifies the tradeoffs between them. The Discovery Tool required a process like this to ease the shift from a prototype to a software service.

ATAM formally consists of the following steps:

1. Present business drivers:

   Everyone in the process presents and evaluates the business drivers for the system in question. The stakeholders i.e., Battelle, ODEE and CETI evaluated the existing system and presented their requirements.

2. Present the architecture:

   Following the conclusion of the meeting, time is given to the developer, to present the architecture of how the whole application would work. The main goal was to put down on paper the modules that would be a part of the application and how they would interact with each other.
The architect presents the high level architecture to the team, with an appropriate level of detail. Here, a high level architecture highlighting the application’s interaction with the data was identified and given to the stakeholders.

3. Identify architectural approaches:

The architecture then underwent revisions depending on the requirements after team meetings. The main goal of the architectural approaches is to design a way to make the application robust and to eliminate the errors discussed previously.

Different architectural approaches of the application were then presented and discussed.

d. Generate quality attribute utility tree:

Then the research stakeholder and the developer meet to generate a quality attribute utility tree. In this meeting the core business and technical details of discovery tool are discussed. The architecture was modified to include only the required visualizations that translated to using only the required tables/views in the database. This meeting was very important because it helped reduce the number of things the application had to do, thus leaving time and effort to concentrate on only what was important. The core business and technical requirements of the system are defined, and are mapped to an appropriate
architectural property. The stakeholders met again to determine what was needed in terms of the functionality of the Discovery Tool. It was decided to decrease the number of visualizations to only what is required and to introduce new features like color-coding the departments etc.

e. Analyze architectural approaches:

Once it was decided about what was required, it is necessary to prioritize them according to importance and time available. The first preference was given to build the profile of an author and to implement co-author and foaf visualizations. This was done earlier compared to auto complete, feedback feature which were not a part of the main implementation. Each scenario is analyzed, and is rated by priority. The architecture is then evaluated against each scenario. The main scenarios identified were Author search and expertise search.

f. Brainstorm and prioritize scenarios:

Each of the modules of the application was discussed with the stakeholders and modifications were done accordingly. The scenarios are then presented to the stakeholders, following their discussion, are then designed.

7. Analyze architectural approaches:

A team meeting with the developers at ODEE who reviewed the code and gave suggestions to improve it followed the implementation of every module. This helped to recognize some errors like using joins, using like, reason of
crashing of the application etc. Before deploying each version, the team helped to test the application rigorously. A team of developers (ODEE, in this case) verified the implementation of the architectural approaches. The problems, errors were identified and the final application was deployed.

8. Present results:

To avoid legacy code errors, the code this time is properly documented and commented and knowledge transfer is done when required.

a. All documentation is provided to the stakeholders

b. Knowledge Transfer is done to the next developer continuing on the project

Figure 6 shows the graphical representation of the ATAM process. All these steps were followed in an agile model. All these steps were followed in an agile mode involving all the stakeholders every time a decision was taken.
A functionality would be implemented, tested, shown to the stakeholders, take their suggestions and follow this process once again to modify/develop the functionality or to develop a new one.
Chapter 4: DESIGN

System Design

The main modules of the discovery tool are searching, visualizations and the back end. In this once the user searches for some information, it passes on to the database which first checks if the query is valid and if that corresponding data is available. If it is, then the information related to that query is created on the fly and the corresponding json objects for the visualizations are created. This is then passed to the visualization module, which then creates the visualizations basing on the json objects.

Figure 8: System Architecture
In the system, as shown in Figure 7, the functionality can be divided into three distinct parts. The architecture here is divided into three modules depending on the functional decomposition of the application.

**Profile Building**

Data is acquired and accumulated for each registered principal investigator that we find through several avenues of characterizing the data. As the profile is built automatically and stored, we will be able to reference this accumulated data when the user searches for a particular research area or individual. Think of this data as summarizing the raw unrefined data previously existing only in unstructured and untrusted data sources.

**Expert Finding**

This function will functionally accomplish the aggregation functions of the profile data. Essentially it will assist in summarizing it by using particular metrics created by our project team. Additionally, the component will draw out relationships between individuals’ data that are stored in the profiles.

**Visualization**
Finally, we need to have a functional piece that will facilitate rendering this complexly linked data to the user in a succinct way. This portion of our solution will essentially take a stream of simplified data and show it to the user in a number of ways that they specify. Of course, the option will always be there to see more detailed data, but we assume a high level starting point.

Data Sources

For all practical purposes, KMData serves as the primary source of publication dataset that is used by the Discovery Tool. However, the modular MVC structure of the Discovery Tool makes it easy to seamlessly transfer to other data publication data sources such as research in view dataset.

The primary fields of the dataset are:

- Author Last Name
- Author First Name
- Title
- Department Name
- Keywords
Deployment/Back End Architecture:

Figure 9: KMDATA Architecture Diagram
The diagram in figure 8 shows the deployment diagram of the discovery tool and it’s functioning as a feature of KMDATA. The Discovery Tool is a KMDATA UI interface that is then linked to KMDATA Postgresql views, in this case, they are called CETI views as a part of KMDATA Athena. These views are then linked to the KMDATA core called HERMES that contains the KMDATA core database infrastructure.

**Database Design**

The KMDATA is a database that pulls in data from different heterogeneous sources. The different sources are Research in View, Academic Analytics and Web of Science. RIV is the database, which consists of manually entered information by professors. Hence we can expect to find spelling mistakes, holes in the data, multiple entries, minor differences which would result the input to be counted as two etc. The data-cleaning task is the second problem addressed in this thesis.

**Data Deduplication**

Along with the team at ODEE, we came up with an algorithm that would help create a strong database. Duplicate data can give wrong results. Given that the tool is pulling data from different sources, it is important to clean the data and remove duplicate data. We check the weight of each word in a given article,
journal or paper name. While comparing two articles, we check the combined weight the article name. If it crosses a certain threshold value the two articles are deemed to be similar and one of them is deleted. This removed all the major duplications. The work is still ongoing in case of this step. The developers are working on removing non-alphanumeric characters and are trying to make it more semantic.
CHAPTER 5: IMPLEMENTATION

In order to understand the interaction among the system components of the Discovery tool, we provide the following software interaction diagrams below:

Figure 10: Use Case 1- Professor search
Figure 9 depicts the professor search interaction diagram. In this, the primary user enters the author’s name into the search area. This author name is sent to the profile-building component. This is caught by the database component that searches the entire database for the author name and formulates the corresponding results. The author profile complete with his publications, co-author network and friend of friend network is formed. This is sent to the visualization component that visualizes these results in the form of graphs and returns it to the user.

Figure 10 depicts the expert search interaction diagram. In this there are 4 components. Initially the user enters the keyword in the search area. This keyword is passed to the database component directly in, which the corresponding information is searched for. The list of publications related to that keyword is compiled. We then look for authors related to those publications. Depending on the number of times the keyword appears in their profile, they are arranged as experts. This list of experts is the passed to the user interface for the user to explore.
Profile Building Component:

Analyzing his publication records can accurately summarize a researcher’s academic profile. An author’s profile is built using the people he collaborated with in the past. His citation index is taken into account for the profile construction. Figure 11 depicts this process.
When the client sends in a name search request, the controller calls the function that creates the author profiles. The author profile service first checks if the searched author exists and then sends the call to the author data access layer. This layer then queries the database that sends back the query result. This result is sent back to the create profile service which converts this result to a

Figure 12: Profile Building Component
json object, after arranging the co-authors according to highest number of collaborations. This is sent to the controller that sends it to the visualization module. The visualization module then constructs the co-author and friend of friend visualization. The co-author visualization is designed according to the department and the number of collaborations with the co-author.

**Expert Building Component:**

Another key feature of the Discovery tool is expert search. The user can enter any “word” for example solar, wireless, cancer etc., to find who are the top researchers at OSU in this area of research.

![Figure 13: Expert Building Component](image-url)
The user sends the search query. The controller sends this query to the service layer which functions in the following way, as shown in figure 12:

It locates the experts based on keywords and locates experts based on friends list. These experts are then arranged according to the number of times this keyword appears in their profile. The list is arranged in descending order. This result is returned to the user. Each person’s profile in this result set is linked to his profile.

**Implementation Features: Solutions To The Challenges Faced**

In the planning phase, we came across code challenges faced while testing ExSO during the conversion of ExSO to Discovery Tool. This section explains how these challenges were met and the solutions for them.

1. **Application crashing**
   
The connection between the front end and the database would be lost.

   **Solution:** It was found that there was no exception handling while querying the system. The solution was to use try-catch blocks. Catch the error and return the error message.

2. **Speed:** The KMData is a huge database pulling in data from different sources. Hence views of all the necessary information were created. Each of these views involved over 17 joins over them and each of the queries did pattern matching over the search query. This is a very wrong coding
practice that increased the search time substantially to almost 120 sec. The server couldn’t take this load as well which often resulted in the crashing of the application. A code review session with the team helped recognize the problem that was then solved by using materialized views and EQUALS in the queries. This highlights the importance of SCRUM and team meetings.

3. The response time of the application was 120 sec

Solution: This was because the entire table was searched to give the output for one query. Also, multiple joins were created which slowed down the application. We use materialized views. A materialized view is a replica of a target master from a single point in time. The master can be either a master table at a master site or a master materialized view at a materialized view site. Whereas in multi master replication other master sites continuously update tables, materialized views are updated from one or more masters through individual batch updates, known as a refreshes, from a single master site or master materialized view site [5].

LIKE: The LIKE expression returns true if the string matches the supplied pattern. It matches patterns and is case sensitive [5].

ILIKE: The keyword ILIKE functions same as LIKE with the difference being that ILIKE is case insensitive [5]
EQUALS: The keyword EQUALS makes literal comparisons with the keywords. In the case of our application we use EQUALS to speedup the search process and decrease the response time.

4. Duplicate Data
The data used for EXSO was from Scopus. Scopus has different information and is categorized into tables quite different from KMDa.
Solution: Modify queries to search in KMDa database.

5. The data in KMDa is manually entered which resulted in duplicate data, wrong data, holes in the table, slight spelling mistakes of the same data etc.
Solution: Data Cleaning and Data Deduplication process.

6. Constraints of the existing environment: It took meeting with the client and a demo to the client to understand that Battelle uses IE 8 which is the older version that is not supported by the visualization library used in this application. Battelle is a high security environment hence migration to newer versions of browsers was a problem.
Solution: Battelle agreed to use Firefox for the systems that would host Discovery Tool.

7. Legacy Code issues: The initial software application started out as a prototype to test an idea. Hence the code was structured so as to test the
functionalities. Several issues with the code have been found. They are as shown. These legacy code problems took a long time to understand and solve which wouldn’t have been the case, if there was proper documentation

- No proper commenting
- It was structured in a way to process in memory code
- Couldn’t handle large amounts of data
- It was designed to handle scopus specific data structures
- It couldn’t handle the holes in the new database structure

Solution: Properly Documentation for the application and comments for all the functions in the code.

8. Concurrency Issues

Whenever more than one person used the application, the first person would get the results of the latest search. This is not acceptable in any software application

Solution: Implemented the concept of threading. Every search query is its own thread and can function independently of the other threads

9. Name Issues

Wrong results would be given for people having the same last name as ExSO was a last name-based search.
Solution: Discovery Tool is a full name based search and the database level changes have been made to remove the ambiguity in names.

Results

Professor/Name Search:

The first scenario discussed is that of Professor Search. The user enters a professor's name and gets a profile page of the expert. The profile page consists of the list of publications of the expert and links to his co-author and friend of friend network. The profile page is as shown in Figure 13.

This search query returns the professor's profile that has his/her list of publications arranged in order of year of publication. Using the networks you can explore the other professor’s connected with this professor. This essentially allows us to check his/her past collaborations.
Figure 14: Author Profile page

a. Co Author Network:

The co-author network shows the co-authors of the expert searched for. The co-authors are arranged as nodes around the expert. The radius of the node increases according to the number of collaborations he had with the expert in the past. The result is as shown in figure 14.
The color depends on the department of the co-author. The department colors are standard across the application. The following algorithm determines these department colors:

1. A dictionary containing all the keywords of a particular department is created. These words are taken from the department description. This dictionary is formed for all departments.

2. Every author has publications that have keywords. These keywords are fuzzy matched with that of the dictionary using a simple linear search.
3. The department that the co-author's keywords matches to the closest, is the department considered.

This method is summarized in figure 15.

![Diagram of KM Data Virtual Repository, author's, department's, word matching algorithms, and improved co-author network with color codes]

**Figure 16: Co-Author network color-coding feature**

b. Friend of Friend network:

The friend of friend network helps to explore two-tier network of collaborations.

The output is as shown in figure 16:
Figure 17: Friend of a Friend

**Department search:**

This search query results in a list of all professors’ profile that is a part of the department. You can enter the professor’s profile from the link provided. It is not necessary that the user enter the full name of the department. The output is as shown in figure 17:
**Authors in computer science whose work appears in the database**

Click on an author to see their profile

<table>
<thead>
<tr>
<th>Author Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agrawal Gagan</td>
</tr>
<tr>
<td>Arnold Mark</td>
</tr>
<tr>
<td>Arcora Anish</td>
</tr>
<tr>
<td>Babic Gojko</td>
</tr>
<tr>
<td>Bair Bettina</td>
</tr>
<tr>
<td>Belkin Mikhail</td>
</tr>
<tr>
<td>Blanas Spiridon</td>
</tr>
<tr>
<td>Boggus Matthew</td>
</tr>
<tr>
<td>Bond Michael</td>
</tr>
<tr>
<td>Buoci Paolo</td>
</tr>
<tr>
<td>Burkhardt Michael</td>
</tr>
<tr>
<td>Catalyurek Umit</td>
</tr>
<tr>
<td>Chaabouni Moez</td>
</tr>
<tr>
<td>Champion Adam</td>
</tr>
<tr>
<td>Chandrasekaran B</td>
</tr>
<tr>
<td>Chou Miao-Chen</td>
</tr>
<tr>
<td>Close Doreen</td>
</tr>
<tr>
<td>Coe Joseph</td>
</tr>
<tr>
<td>Collins Catrena</td>
</tr>
<tr>
<td>Compton Michael</td>
</tr>
<tr>
<td>Cramer Tamera</td>
</tr>
<tr>
<td>Crawfis Roger</td>
</tr>
</tbody>
</table>

Figure 18: Department search results
**Expertise search:**

This search query returns a list of all professors in OSU whose major key area is the search query entered. The result is as shown in figure 18.

**Experts in the field of wireless:**

Click on an author to see their publications, then, click on an area of expertise to see just the relevant papers

<table>
<thead>
<tr>
<th>Author Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arora Anish</td>
</tr>
<tr>
<td>Eryilmaz Atilla</td>
</tr>
<tr>
<td>Xuan Dong</td>
</tr>
<tr>
<td>Koksal Can</td>
</tr>
<tr>
<td>Sinha Prasun</td>
</tr>
<tr>
<td>Erkin Emre</td>
</tr>
<tr>
<td>Zhao Lingying</td>
</tr>
<tr>
<td>Khalil Walid</td>
</tr>
<tr>
<td>Sertel Kubilay</td>
</tr>
<tr>
<td>Abraham William</td>
</tr>
<tr>
<td>Lai Ten-Hwang</td>
</tr>
<tr>
<td>Ling Peter</td>
</tr>
<tr>
<td>Moses Randolph</td>
</tr>
<tr>
<td>Ramanath Rajiv</td>
</tr>
<tr>
<td>Rojas-Teran Roberto</td>
</tr>
<tr>
<td>Zheng Zhizhan</td>
</tr>
<tr>
<td>Srinivasan Kannan</td>
</tr>
<tr>
<td>Bahrani David</td>
</tr>
<tr>
<td>Chen Qian</td>
</tr>
<tr>
<td>Hasan Ayesha</td>
</tr>
<tr>
<td>Chen Chi-Chih</td>
</tr>
<tr>
<td>Davor Steven</td>
</tr>
</tbody>
</table>

Figure 19: Expert Search results
Feedback:

Since the discovery tool is still in the development phase, users can enter their feedback to help develop the tool even better. The feedback form looks as in figure 19.

Figure 20: Feedback form
Chapter 6: FUTURE ENHANCEMENTS

Once the discovery tool is functioning well and is deployed, the next question is its evolution. Any software application needs to leave space for growth within its growth. As user feedback is received, as more visualizations are required, as more data is acquired the need to make enhancements to the tool also becomes more important. This section is used to show prototype features that could be added to develop the discovery tool further. These case studies are merely a way to show the future visualizations that can be done using the modified database and modified code structure. It is very easy to modify and add new features to the Discovery Tool since all the steps of software development life cycle are followed.

Case Study: A Prototype to explore the functionality of Discovery Tool

Word Clouds

Every broad area of study be it computer science, medical, mechanical engineering, the arts, geography etc., are all expanding. New discoveries, new inventions, new cutting edge research is done everyday. Some sub areas become more important than the others and major funds are pushed into that area. In turn it also happens that in the academic world, more papers are
published in a particular sub area than the other. Many universities are known for research in these sub areas. Students often wish to work in these research areas. But how would they know which areas are those? Hence, we can use a visualization called word clouds that help to identify these “hot” topics of research. We essentially search through all the words, in this case author keywords, and create word clouds basing on how often they repeated.

Figure 21: Word Cloud result for CSE department
In the current example, using KMData and Discovery Tool, we create one search word cloud for the computer science department. The result is as shown in figure 20.

We can see that some of the “hot” areas are wireless, networks, image, sensors, security, data, MPI and analysis.

**Inter Department Collaborations:**

This prototype is to compare the various departments a particular author has collaborated with. This helps us to see the inter department collaboration of the author. This can be extended to any author. A potential collaborator can now approach the author by seeing his inter departmental collaborations. In this prototype:

- We consider a random author from the database
- His co-author network shows us the authors he has collaborated with
- Using that information we can find the departments.

In this case, the author considered is “Chandan Sen”

We first check who is co-authors are and then compare and graphically represent all the departments that his co-authors are in, in a pie chart as shown in figure 21.
Figure 22: Pie chart representing the inter department collaborations of Dr. Sen

Hence, combining visualization techniques like pie charts, word clouds, networks etc:

- We can form a profile of an author
- Find out his/her current areas of research
- Find out if he’s actively collaborating

Hence we can increase micro collaborations with the university and equip the discovery tool to better function as the source of information for all expert related information in OSU.
Other future work that can be included in the Discovery Tool to enhance the functionality of the Discovery Tool is:

1. Predictive searching:
   A method to monitor a user’s most frequent searches can be included. This will help us find his interests and build a profile. The user can then receive updates based on his interests.

2. Citation Information
   A method to use citation information to determine the author’s key areas, rank his publications etc.

3. Grant information
   A method to use grant information to determine the current area of research of an author should be introduced in the Discovery Tool. Very often it happens that researchers concentrate on one sub area within their area of interest according to the grants they receive. This will help potential students as well as industry collaborators to get funding as well as a chance to work in the current research area.

4. Map of science visualization
   Enhance the HCI component by introducing visualizations that show the academic networks in OSU
References:
3. www.research.osu.edu
6. Non-Functional Requirements Practices and Recommendations-A brief Synopsis, L Chung, University of Texas, Dallas
12. https://kmdata.osu.edu/


