TOWARDS BETTER ENGINEERING OF ENTERPRISE RESOURCE PLANNING SYSTEMS

A Dissertation
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Talukdar Sabbir Asgar

In Partial Fulfillment of the Requirements
for the Degree of
DOCTOR OF PHILOSOPHY

Major Program:
Software Engineering

November 2016

Fargo, North Dakota

North Dakota State University
Graduate School

Title

Towards Better Engineering of Enterprise Resource Planning Systems

By

Talukdar S Asgar

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

DOCTOR OF PHILOSOPHY

SUPERVISORY COMMITTEE:

Dr. Kenneth Magel
Chair

Dr. Anne Denton

Dr. Jun Kong

Dr. Julie Pasche

Approved:

April 11, 2017
Date

Dr. Brian Slator
Department Chair

ABSTRACT

In spite of their high implementation failure rate, Enterprise Resource Planning (ERP) software remains a popular choice for most businesses. When it succeeds, ERP software provides effective integration of formerly isolated multiple systems. This integration yields significant business efficiencies. Replacing legacy systems with ERP software requires a great many trade-offs. We found that using bipartite graph can facilitate requirements elicitation in ERP procurement. It can have great impact on different activities in subsequent phases. Such activities include product line engineering, domain analysis, test-driven development and knowledge reuse in system development life cycle (SDLC). Bigraph representation of legacy and ERP requirements also helps determining the enhancement needs. It provides a better control for stakeholders to decide the scope for development and testing.

ERP streamlines operations and flow of information across an organization. Business functions in ERP system are always master-data driven. Migration of data from legacy to ERP is therefore a critical success factor for ERP procurement projects. Correct data conversion is essential for integration and acceptance testing. It is a complex procedure and requires additional effort because of large volume of data. The architecture and design of data conversion process must ensure the referential integrity among different business modules. Our process for data conversion starts with a test-first approach. Next it conducts execution of conversion programs in parallel. Our parallel approach replaces the old sequential style of execution. Through this process we could correct data mapping errors and data anomalies before conversion. Parallelized execution drastically reduces the time needed for execution of conversion programs and provides more time for testing. We verified the feasibility of our approach by multiple industrial projects.

ACKNOWLEDGMENTS

It is not possible to acknowledge everyone in this short note, but I will put my best possible effort. In the beginning, I would like to thank my advisor, Dr. Kenneth Magel, for his continuous support during my long educational career. I would also like to thank Dr. Jun Kong for agreeing to be in my advising committee. I want to thank the external member, Dr. Julie Pasche of Department of Plant Pathology for joining. My special thanks to Dr. Anne Denton for agreeing to fill in as a last minute change. I cannot skip mentioning the help and support I received from Dr. Kendall Nygard. I would also like to mention the contributions of Dr. Dianxiang Xu and Dr. Tariq King.

I want to show my appreciation to the faculty and staff members of Computer Science Department, to my students of CSCI-373 and CSCI-374 of 2004-05 school year, and to all my colleagues and classmates in CS Department and ITS of NDSU. I would also like to thank my project members in Maximus ERP and CherryRoad Technologies and my recent colleagues at Dartmouth-Hitchcock Medical Center. I also want to mention Dr. Daniel Creider, Dr. Sam Saffer and Dr. Sang Suh of Texas A&M University – Commerce. My final appreciation goes to my family and friends who were always with me during this journey.

Sincerely,

Talukdar_Asgar@meei.harvard.edu

DEDICATION

To my mother

PREFACE

“Wonder is the beginning of wisdom.”

—Socrates

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

Companies face many obstacles when they procure Enterprise Resource Planning (ERP) software to replace their existing systems. The principal goal of my research is to minimize such obstacles and improve some of the critical tasks during ERP procurement. My research starts with an extensive literature survey on ERP systems and related topics. The first part of my research focuses mainly on the requirements analysis, or more precisely fit-gap analysis phase. The second part proposes a new approach for data analysis and conversion. Both are pointed out as critical success factors [1] for ERP procurement. Through this work, I was able to improve the data conversion process, one of the major and time-consuming part of the development phase. I will further discuss these problems and obstacles from the viewpoint of a software engineer. It will justify the importance of this work. My research was able to bring accuracy and correctness in defining some parts of ERP procurement process, along with strengthening the overall budget expenditure saving time and money. The upcoming chapters goes like this,

Chapter 2 - discusses background and literature survey for this research.

Chapter 3 - presents the publication based on section 1.4.1.

Chapter 4 - presents the publication based on section 1.4.2.

Chapter 5 - provides the detail research flow.

Chapter 6 - lists the contributions of this work.

1.1. Enterprise Resource Planning (ERP)

Gartner Research Group first coined the term ERP in 1992. It stands for Enterprise Resource Planning. It is a type of Commercial-off-the-Shelf (COTS) software. It provides an integrated optimized solution as a bundle that can take care of entire business functions for a company. ERP is a complex and vast system and procuring ERP involves significant investment

affiliated to implementation and maintenance. Modern day business requires companies to commit to continuous improvement and innovation. To achieve this goal, a better synchronization is essential among stakeholders as well as between different departments of the company [2]. American Production and Inventory Control Society has defined ERP systems as a method for the effective planning and controlling of all the resources needed to take, make, ship and account for customer orders in a manufacturing industry.

Procuring ERP can bring many benefits for a company. The most significant two would be, referential integrity with a robust RDBMS and optimized response to events at any point of operation [3]. Proper integration will create a trustworthy connectivity among business processes for appropriate and adequate communication. It ensures the maintenance of both quality and quantity of required services. Proper configuration management validates the synchronization necessary for different ERP modules. In this way operational errors, such as wrongly charged taxes or item price mismatch, are reduced to zero.

Due to high popularity of ERP, most companies are looking into ways to maximize their system performance and diversify the functionality to help improve services they provide (according to a survey the number is 67%). As well as, companies who have not embraced ERP systems yet are eagerly looking forward to one most suited for their operations (for which the number is 21%).

ERP comes as a bundle of individual modules. Modules interconnect among each other and also can integrate with any third-party products that are available in the market without difficulty. Businesses use ERP for many back office activities. Supply chain management, cost accounting, accounts payables, accounts receivables, asset management, human resources and payroll, time and labor distribution, benefits operations are few examples. Management and

maintenance of work functions happen online with Rich Internet Architecture (RIA). This reduces number of paperwork and physical files. Recent introduction of standard security architecture and e-signature capability adds to this improvement further.

PeopleSoft Enterprise of Oracle Corporation Inc. is a very popular ERP solution in the market. Oracle distributes PeopleSoft through their OPN (Oracle Partner Network) members, who work as vendors for many clients in the industry. The company can be of government, automobile, manufacturing, education or any other type. The main package solutions of PeopleSoft comes in the form of, PeopleSoft Human Capital Management Solution, PeopleSoft Financials and Supply Chain Management, PeopleSoft Customer Relationship Management, PeopleSoft Campus Solutions which include Student Services and PeopleSoft Enterprise Performance Management. PeopleSoft version 1, released in the late 1980s, was the first fully integrated, robust client–server HRMS (Human Resource Management System) application suite. Later PeopleSoft expanded its product range to include a financials module in 1992, followed by the distribution module in 1994, and manufacturing module in 1996 after the acquisition of Red Pepper. [4]

1.2. Procuring ERP

For many years enterprise systems remained as a mix of file-based solutions. The underlying processes used structured programming for deliverables, interfaces and reports. The most common user interface was kiosk-based workstations with command line data definition and data manipulation techniques. The shift to a solution, containing more interactive Graphical User Interface (GUI) with a backend relational database was just a matter of time. The inconsistency of not so user-friendly interface and heavy dependency on data entry kept asking for additional workload for number crunching and error recovery. The amount of resource and

labor spent was continuously cutting into major portion of the expected profit margin. Because of the reliability, integrity of data, consistency and reduction of human errors, ERP slowly but surely turned into the most acceptable resolution for those deficiencies.

Enterprise Resource Planning (ERP) from the very beginning since it came to market ensured agility in business operations. It combined rapid application development (RAD) with GUI of high usability. These qualities were enough to popularize ERP among number crunchers, data entry operators, programmers, analysts, and leadership altogether. At first the system was developed for using in manufacturing industry only. At that time, it was called Manufacturing Resource Planning (MRP). Soon it became clear that some of the modules after few round of modifications can also become useful for non-manufacturing companies. Government and Education were the type of businesses which first embraced this extended version, followed by retail, automobile, utility, etc.

1.3. System Development Life Cycle

The basic phases of an iterative software engineering model are also applicable for an ERP procurement project in the industry. The client, in our case which is a state government, first has to come to an agreement with the ERP vendor. Then the Fit Gap Analysis of the selected ERP software with existing legacy software begins. During this FGA phase, another concurrent phase takes place, namely the Discovery phase. Discovery starts with checking the core business functions in the delivered prototype and comparing the outputs with the legacy outputs for same set of inputs. The client and the vendor comes to an agreement what development needs are going to get addressed and what parts of their analysis can remain outside of project scope.

The primary development areas are data conversion, interfaces, and reports. Any custom enhancement of delivered ERP solution remains under consideration at this stage. The size of

development and testing can grow proportionally with the number of modules to implement according to the agreement. It can grow with the total size of the company as more data means more use case scenarios. Also number of third party transactions can increase the number of interfaces. Recently inclination to data analysis for decision-making happens to increase the reporting needs in such projects. I will discuss all kinds of development needs in section 1.5.2.

The system development life cycle (SDLC) is a combination of iterative method and incremental build of the target domain as project proceeds. More than one iterations of development cycle ongoing at the same time is a typical scenario for such project. The prototype contains an internet architecture and a backend database. Each subsequent environment will have a similar structure. Generally, for development and unit testing, it will have development database with an environment to logging in. It will also contain application development interface (ADI) and database querying tools for the developers and database administrators (DBA). For functional testing it will have a business analysis environment with an underlying database. Similarly for Integration, System and User Acceptance Testing, vendor will provide separate environments with dedicated databases at backend. The properties and core functionality of all these environments should remain same for consistency. There can be a specific environment with a backend database for handling a particular but important implementation associated with essential frequent updates, for example, a tax environment with underlying database to verify and implement tax updates. Tax update happens every quarter in a year.

As data gets converted it is put into the development database first. Data then goes through unit testing. Once assured same conversion takes place into functional testing environment. After passing the functional test, the same source data file from legacy gets

converted into other environments. Same development and testing cycle are applicable for other development pieces, only there is no data conversion involved. So, once data is converted and found to be faulty, it is always expensive to revert. It requires additional effort to clear the environment and get a new set of data to test. There may be interfaces and reports using the part of converted data. Quality of converted data will also impact subsequent development activities for those pieces.

A finalized technical specification document is a must for each piece of requirement which involves a change in the target domain. A technical specification document precisely describes the requirement need along with use case scenarios, the type of development needed and the required deadlines of that development for different phases in the project timeline. The functional requestor in the specification also describes in detail what is the expectation, and where and how in the target domain they plan to accomplish the proposed solution. Based on this specification the assigned developer will analyze and design the development in detail. The developer will also mention which tools and objects will be in use for the development tasks. The technical specification will also have a probable estimation of hours of work necessary for this delivery. The developer next defines all essential unit testing scenarios. Once the review is complete, the document goes through management for approval. Development can start only after all necessary bodies review and approve the document.

Once implementation is complete, the maintenance phase begins. Because of the vast functionality of ERP systems, maintenance phase generally take much longer compared to any other types of software.

1.4. Data Structure

ERP systems typically include two sets of data –

1. Set-up data, and

2. Transactional data.

Set-up data represents system constants and somewhat universal data such as Department Identifier or Position Number. Transactional data is part of data that gets newly inserted, modified, discarded, batched and accessed on a daily basis for business operations such as Employee Benefit Premium or Accounts Receivables Encumbrance Value.

Alwabel, S.A. et al. argue that for a full-fledged discussion of benefits, barriers, and other factors affecting the implementation of ERP to organizations, it needs clarity about the evolution of ERP systems [5]. The clarity also helps figuring out the relationship between ERP and e-business. Recently we see that recognized limitations in ERP have sparked new trends in ERP application development. Web-enabled ERP, inter-enterprise ERP, e-business suites and secure mobile apps are to name a few [6, 7, 8, 9]. For example, SAP contains integrated business solutions for small to medium-sized businesses in three different bundles, SAP Business One, SAP Business ByDesign, and SAP All in One. Each of these is capable of handling small to medium size organization requirements efficiently. The client base is vastly diverse for these solutions. SAP Business ByDesign, a recent addition can be implemented as software-as-a-service with core features selectively enabled, based on the agreement between client and vendor. Additional enhancement and customization are possible as post-implementation tasks.

Modules are designed to focus on specific areas of business. To say that there are a variety of options would be an understatement. Table 1.1 lists four providers. Each has numerous options to cover (order in the chart is by SAP modules respectively with Oracle, PeopleSoft, and JDEdwards equivalents).

1.4.1. What are the common modules?

Financials
Accounts receivable and payable
Asset accounting
Cash management and forecasting
Cost-element and cost-center accounting
Executive information system
Financial consolidation
General ledger
Product-cost accounting
Profitability analysis
Profit-center accounting
Standard and period-related costing

Human Resources
Human-resource time accounting
Payroll
Personnel planning
Travel expenses

Operations and Logistics
Inventory management
Materials management
Plant maintenance
Production planning
Project management
Purchasing
Quality management
Routing management
Shipping
Vendor evaluation

Sales and Marketing
Order management
Pricing
Sales management
Sales planning

Figure 1.1. Common Modules in Popular ERP Software [3]

Table 1.1. Common Business Functions of Different ERP Systems in the Industry

SAP	Oracle	PeopleSoft	JDEdwards
SD (Sales & Distribution)	Sales & Marketing	Supply Chain	Order Management
MM (Materials Management)	Procurement	Supplier Relationship	Inventory & Procurement
PP (Production Planning)	Manufacturing		Manufacturing Management
QM (Quality Management)		Enterprise Perform	Technical Foundation
PM (Plant Maintenance)	Service	Enterprise Service	
HR (Human Relations)	Human Resources	Human Capital Management	Workforce Management
FI (Financial Accounting)	Financials	Financial Management Solutions	Financial Management
CO (Controlling)			Time & Expense Management
AM (Asset Management)	Asset Management		Enterprise Asset Management
PS (Project System)	Projects		Project Management
WF (Workflow)	Order Management		
IS (Industry Solutions)	Contracts		Subcontract & Real Estate

1.5. What are the steps in FGA?

FGA or Fit Gap Analysis is the first phase of ERP procurement. In the project timeline, it occupies the initiation and most part of requirements analysis until iteration starts. Usually, the legacy analysts and experts combined with the Subject Matter Experts of a particular module of the ERP system undertake this task. A module represents a major business function of the entire domain. For example, Accounts is a module, and under accounts, there are two major subdivisions, which are also considered separate modules in ERP, receivables and payables. While evaluating each corresponding area of expertise the analysis team focuses on information and function. In my work, I used the terms, data driven and functionality driven analysis.

It seems relevant to mention the lack of application of formal methods in such requirements analysis since inception of ERP. Requirements analysis in software engineering is

considered the most critical phase where stakeholders work together to define discovery, enhancement, and exclusions. Software engineers generally recommend a formal approach for this phase. Unfortunately for ERP procurement, there was no significant effort in the industry to initiate a formal approach during requirements analysis. The results from many resources also confirm that the high failure rate of ERP procurement projects is due to a faulty or hasty delivery resulting from this phase. Incomplete or burdensome requirements can initiate a quick plummet in the budget while actual progress of the project remains at a standstill. Some inhibitors to a smooth completion of this phase include:

- Resistance to change
- Power struggle among parties
- Not involving all stakeholders
- Attempt to satisfy all stakeholders
- Bureaucratic limitations
- Inefficient change management

Table 1.2 shows the Critical Success Factors (CSFs) with mean and standard deviations. Table 1.3 presents a survey which determines most applicable reason in a procurement failure during different phases of the project, based on the total percentage of stakeholder response.

Table 1.2. Critical Success Factors of an ERP Implementation Project [1]

	Critical Success Factor	Mean	Std. Dev
1.	Top management support	4.29	1.16
2.	Project team competence	4.20	1.07
3.	Interdepartmental cooperation	4.19	1.20
4.	Clear goals and objectives	4.15	1.14
5.	Project management	4.13	0.96
6.	Interdepartmental communication	4.09	1.33
7.	Management of expectations	4.06	1.37
8.	Project champion	4.03	1.58
9.	Vendor support	4.03	1.60
10.	Careful package selection	3.89	1.06
11.	Data analysis & conversion	3.83	1.27
12.	Dedicated resources	3.81	1.25
13.	Use of steering committee	3.79	1.95
14.	User training on software	3.79	1.16
15.	Education on new business process	3.76	1.18
16.	Business process reengineering	3.68	1.26
17.	Minimal customization	3.68	1.45
18.	Architecture choices	3.44	1.19
19.	Change management	3.43	1.34
20.	Partnership with vendor	3.39	1.21
21.	Use of vendor’s tools	3.15	1.57
22.	Use of consultants	2.90	1.20

Table 1.3. Survey Responses in Percentage on Different Critical Success Factors at Particular Stages of ERP Implementation Projects [1]

Stage: Initiation	**Respondents (%)**
Architecture choices	71
Clear goals and objectives	63
Partnership with vendor	61
Top management support	61
Careful selection of packages	60
Stage: Adoption	
Top management support	68
Project team competence	61
Use of steering committee	60
Partnership with vendor	60
Dedicated resources	59
Stage: Adaptation	
Interdepartmental communication	65
Interdepartmental cooperation	63
Project team competence	63
Dedicated resources	60
Use of vendor's tools	60
Stage: Acceptance	
Interdepartmental communication	64
Interdepartmental cooperation	63
Top management support	56
Project team competence	55
Education on new business processes	53
Stage: Routinization	
Interdepartmental communication	51
Top management support	42
Interdepartmental cooperation	41
Vendor support	36
User training on software	36
Stage: Infusion	
Interdepartmental communication	39
Interdepartmental cooperation	35
Top management support	32
Vendor support	28
Partnership with vendor	28

In Table 1.3 we find that the critical issues are not only concentrated in one particular area but widely spread all over the project timeline. The high percentage rate also confirms the underlying complexity that makes achieving a successful implementation challenging. In my work, I emphasize on the following Critical Success Factors,

Table 1.4. How This Research Help Resolve Some Critical Success Factors for an ERP Implementation Project

Number	Factor	What my effort does to it
4	Clear Goals and Objectives	A formalization technique for requirements
11	Data Analysis & Conversion	Data mapping & improved conversion technique for existing legacy data using parallel programming
12	Dedicated Resources	Optimum utilization of acquired hardware and resource
16	Architecture Choices	Choice of a better process design for data conversion which minimizes cost/time and improves quality of deliverables
18	Business Process Reengineering	Major part of ERP procurement involves reengineering of legacy operations into the new environment. This is tied to 4, 11, 12 and 16.
19	Change Management	It involves every phase of 4, 11, 12, 16 and 18. It facilitates an improved and ordered management to change by abandoning the old way of managing things.

It is evident that my work impacts other critical success factors positively. But to achieve that we need better project management and better communication within and in between different teams. Overall the results of my research were able to bring tremendous value in terms of cost and sustenance. Below I will briefly discuss the parts of my research areas. Detail descriptions of the requirement phase and data conversion part of the development phase are available in Chapter 5.

1.5.1. Formalizing ERP Requirements

Clarity of goals and objectives is one of the top five critical success factors for ERP projects. At the atomic level for a software engineer, an objective is just another properly defined requirement which gets discovered during the fit gap analysis. FGA, as discussed, identifies the match and difference in between existing system and the ERP prototype. The prototype is the demo environment initially provided by the vendor for analysis. The feature in demo environment in comparison to what client has in legacy, might be better and advanced or can be equipped with less functionality. Once the gaps are identified then the client has a chance to analyze the demo ERP environment to discover new functionalities. With the help of vendor, now the client can determine whether the feature will be advantageous for their overall operation for that particular module.

1.5.1.1. Fit/Gap Analysis

In ERP implementation we deal with two domains. The **source domain** is the existing system satisfying current business needs of client. The architecture also informs us about the minimum non-functional requirements of the system. We term it as legacy system in business world. Then we have the vendor system for demonstration. We call it the **DEMO environment** (also called Vanilla). The ERP demo environment is our target domain which goes through required configuration management during the early stages of fit-gap analysis phase. Although the data structure do not get fully finalized for setting up or configuring the system, the vendor will place in a high-end workable prototype for analysis. Analysts can choose to work on different modules to test and verify functionalities. It helps them figuring out appropriate business functions, and making decision on discarding obsolete legacy features, suppressing ERP

features those are not essential or prescribing modification needs or customizations for target domain(s). In short, the principal goals of the fit-gap analysis phase is to,

- Create a workable prototype of the target domain(s).
- Assign legacy experts and domain consultants for business areas.
- Involve associated technical experts of both legacy and target domain(s) in analysis.
- Document and specify functional needs for every requirement recognized.
- Iteratively refactor specifications within an agile method of implementation life-cycle.

So far I have participated in 12 different ERP projects in local and global settings with many clients (which include government, automobiles, value chain, healthcare, etc.). From my experience with requirements elicitation and fit-gap analysis, I find that a systematic formalized highlighting of the gaps efficiently but early can minimize risks, system chaos, time to delivery, test mishaps, development flaws and overall maintenance dilemma.

We could not find application of any mentionable formal technique for ERP procurement in literature. The complicated situation of failure persists for ERP projects largely because of the inability to identify every fits and gaps during module analysis. It also hinders extracting further advantages from the system.

With the expansion of volume and scope, the coordination among stakeholders becomes more and more challenging. Introduction of Subject Matter Experts in many cases makes the matter worse than improving the situation. Such involvement also stretches the timeline and increases overall cost. Lack of a proper communication method can carry on the ambiguity resulting in process bottleneck. It makes the project management volatile and ultimately causes budget overruns. In two of my projects, we went through such scenarios.

Introduction of formal technique in mapping of functionality and data can be very useful to help improve above situations. Using a bipartite graph technique to identify different features, early in the project bring many advantages. The formal model specifies the matching of functional requirements in the two domains. It establishes the relationships on a requirement-to-requirement basis, using a one-to-one or many-to-one mapping from the legacy system to the ERP system. Rigorous analysis shows that such effort at the beginning of the project facilitates early recognition of all possible features client expects to have in the system. The idea is to prevent legacy features from being omitted or incorrectly mapped, thereby avoiding any associated development bottlenecks.

For data-related aspects of the problem, we propose the use of test-first development in ERP customization. Employing this technique allows the project team to identify data requirements of the features through the formulation of test scenarios. Once functional specification of an enhancement or customization is final, developers start creating small test cases and keep refining the test scenarios until satisfactory results are produced from each specification detail. Test-driven development promptly identifies missing data elements or corrupted legacy data. Another reason to prefer test-first framework is the existence of a high-end prototype for such implementations. It is definitely not the case in many other domain-specific architecture models in the market. The presence of a Demo environment with initially mapped set-up data allow consultants do impact analysis based on different scenarios early. Thus, it can help address myriad development needs to progress effectively and also address configuration issues. The degree of complexity will be proportional to the amount of evolved pieces recommended for the target version. We measure the fits, gaps and enhancements between the two domains by canonical decomposition using the exterior cover properties of

bipartite graphs. We published this work in the international journal, Lecture Notes on Software Engineering, Vol. 4, No. 1, February 2016. I will present the paper in Chapter 3.

1.5.2. Data Conversion using Test First and Parallel Programming

The design areas for development phase in Legacy to ERP transition primarily can be divided into three broad categories.

1. Conversion – Conversion of data from legacy database to new ERP database includes the introduction of the new data structure which drastically alters the way we perform operations of data. Operations can be Data Definition Language (DDL) and Data Manipulation Language (DML) operations. Legacy data also goes through hierarchical and relational changes due to the introduction of new entity-relationship. We measure mapping of data using bipartite matching and cover properties. It identifies any redundancy and also any additional table creation if necessary. Client decides which modules to implement, so many delivered tables in the database can remain unused because functionality on the layer above is not necessary for their system.
2. Interface – Interfaces can be of two types.
    a. Internal to system
    b. Third Party

    A number of internal interfaces can become obsolete and become redundant and can be omitted (as they reside in R2 region of our covering). Because of wider and inter-module connectivity of ERP systems, handling of a lot of legacy interfaces that were using file sharing or Extract-Transact-Load (ETL) type operations, now take place inherently without having to go through multiple step procedures. Third Party

Interfaces, on the other hand, most of the time need a design and development effort. Business interfaces common for any type of company like HIPAA interface for Benefits or check printing interface for Accounts Payables come as one stop solutions in ERP. Due to agreement between client and third party, those delivered solutions still need reviews and in most cases require some tweaking to satisfy client and other stakeholders.

3. Reporting – Company reports are an important development area for all ERP clients. A smooth flow of business functions largely depends on delivering right information to the management at an appropriate time. Data querying and reporting along with recent need of analytics keep this activity in the top tier of design and development.

Other development activities include Workflow and Approval Process, Event Processing and Notification, Data Warehousing, Integration, and Platform Specific development, etc. Part two of my research, models a robust data conversion process which helps mitigate some of the significant drawbacks from the existing practice in the industry.

Next step is to proceed from lessons learnt during part one of the research towards data conversion. Developing an ERP system typically involves porting legacy data from different business functions to a common ERP database, and building applications that access and update the central data store. The primary goal of ERP implementation is to standardize and streamline the business functions and information flow among these various divisions of the organization. A big challenge of ERP procurement is the mapping and conversion of data from different divisions into a shared database [10, 11].

Properly migrating information from existing legacy systems to ERP software modules is, therefore, paramount to the success of a project. However, the large size of system and

volume of data makes this task incredibly daunting. Even the data from a single division can be quite difficult to handle correctly due to complex relationships. Software testing continues to play a significant role in ensuring that software systems meet the needs of customers. Given the complexity of ERP software development, and its tremendous impact in the industry, it is normal to expect a lot of research concentrated in the areas of testing ERP systems. However, this was not the case as we found very little work on ERP testing in the literature. Furthermore, software vendors follow their own methodologies for testing ERP systems. Due to confidentiality agreements, there has been little collaboration and dissemination of knowledge concerning effective software testing for ERP [11].

Incorporation of smoke testing to identify and correct data mapping errors and anomalies before running the conversion process saves time and effort associated with failed conversions. In any process, there exists a void between decision (design developed) and feedback (functionality and performance obtained by implementing that model) [74]. The test-first technique helped reduce this void during our different ERP implementation projects. Incorporation of Parallelization within a processing model was similar to SIMD (Single Instruction Multiple Data Stream) techniques in computer architecture [72]. It significantly reduced time delays that used to occur in a lengthy conversion process. Additionally, we introduced an improved error reporting method. The merit was mainly related to the number of iteration and the processing time against volume of data. In MapReduce by Google [73] one can find many similarities with our approach. We came to know about it much later after we moved on to our next project. In our next project, we implemented the model with a much better graphical user interface for reporting and bug tracking. It was for Financial and Supply Chain suite. We successfully published this work in IEEE Xplore Digital Library as a publication of 9th

Int. Conf. IEEE Information Technology of New Generations 2012. I will present this paper in Chapter 4.

CHAPTER 2. BACKGROUND AND LITERATURE REVIEW

This chapter includes literature reviews and surveys that I did for this research. It mostly discusses any mentionable work found in literature and lessons learned from ERP procurement projects. First I reviewed the evolution pattern of ERP software. I found it relevant to analyze how gradual addition of application functionalities to core functionalities kept increasing the size and scope of the software. This incremental pattern of growing software versions also raises degree of complexity in implementation. The analysis also helps us understand the trend in sales, client satisfaction, and market demand. Next, I looked for possible use of formal methods in industry for ERP. I also examined architectural patterns in use for ERP procurement. Next I reviewed the known testing methods for ERP software. Finally I surveyed the maintenance related aspects of ERP implementation projects.

2.1. Introduction

A survey among Chief Information Officers (CIOs) conducted by Morgan Stanley [12] identifies implementation of ERP systems as one of the top five IT priorities for a company. Another survey by Deloitte & Touché/IDG Research Services Group also yielded similar results [13]. Thus we can say that the popularity of ERP is on the rise. I have discussed the importance of using formal methods especially in requirements engineering of ERP implementation. My research then demonstrates how this model can be combined with a test driven development approach to produce better results in data conversion. It also creates the opportunity for dynamic evolution and software reuse from one project to another.

At this point, I would like to mention a post of a top programmer from a discussion group in linkedin.com.

"The company I work for consists of dozens of other companies that was bought over the years, which put us in the position of having several different ERP systems across the company. For a long time now they have been content to leave it that way, but with the more recent purchase of the company that I joined them from, they are now considering the possibility of consolidating under a single ERP system. We are in the very early stages of considering this, but I have been asked to go forth and gather some information where I can find it. My searching on the web has not provided me with much useful information yet, though I will continue searching there as well. Most of what I find is more in the way of advertisements where the company selling the ERP software lists their own pros and cons with the cons, of course, being the most worthless information in the whole article because *nothing* is very wrong with their own software, of course. So, if anyone here happens to be familiar with any of the following ERP systems, I would like to hear your thoughts on the pros and cons of those that are familiar to you. We are a manufacturing company with around 4,000 employees. Annual earnings are roughly $700 million. The ERP systems we already have in place throughout the company are as follows:

[Microsoft] Dynamics AX, [Microsoft] Dynamics ERP, [Aptean] Made2Manage, [Sage] MAS200, [Sage] ACCPAC, [Epicor] Avante, [QBuild?] Visual Manufacturing

I've been using Made2Manage for six years, so I know the pros and con of that system myself from experience. The rest of them, though, I am unfamiliar with personally and would like to ask the group here for their thoughts. I can tell you that one of our major concerns with M2M is that it does not work well for multi-facility organizations that manufacturing the same products at more than one location. So if any of these other options does not do well in that regard either, then we would not be interested in using it moving forward."

Among all the replies to his question, I chose to mention the one below.

"You need to start with a strategic understanding of the business needs and any existing business plans. Key areas to consider include:

- need to consolidate distribution across several companies,
- strategy for doing e-Business/CRM with the customers, either business by business or across businesses (and are you already using various solutions in this area or not);
- Master Data Management - what is the value to have a consistent classification of products, raw/WIP materials, packaging, customers, vendors, transportation lanes; etc.

I've been down this road before, and this decision returns the best value when it is entirely in tune with the organization's strategic plans (I hope they already have a well thought out one), else that should be the first task before you start into looking at packages. You are a large, mid-size organization, but a big-box, big-$$$ ERP solution should be under scrutiny, as they are not a safer choice than several of the other rated solutions."[14]

A large number of research work in literature is available discussing ways to evaluate appropriate ERP solution for a particular industry. They consider different situations and infrastructure limitations. They cover many important areas, for example, models, formal approaches, financial and cost-effective analysis using Bellman equation, need base suggestions, social networking etc. But what will happen if a company acquires another company and each had different ERP systems? What happens if one of them had an ERP and the other had a legacy system? What steps to follow during an affiliation or a merger? There were three different paths suggested to the problem in [14],

- Go with Sage ERP,
- Go with MS Dynamics or

- Choose a new package.

An evaluation using complexity metrics in this case can determine the right path. But evaluation like this might be overwhelming. The management has to keep the associated cost in mind while doing this. In my discussion of requirement elicitation, I would explain what can be a feasible and shorter formal approach. It enables a client to work with the vendor so client can easily make critical decisions like this which will be best suited for the project. A client may already have their legacy system replaced with multiple COTS like solutions. Maybe they are looking forward to implement a more integrated system compared to what is in place.

2.2. Release Management

Corporations who are medium to large in size commonly contain information systems which are collections of many heterogeneous, distributed software components, large-scale software applications, legacy systems and COTS. Software evolution can be static or dynamic. A newer release of PeopleSoft HCM 9.1 would be a version 9.3. We consider this as a static evolution. Oracle Corporation will be providing the vendors with the official release notes and guidelines for either upgrade or new implementation, whereas a dynamic evolution is a typical scenario in implementation cycle where new patches or bug fixes become available for different modules as the project progresses. Dynamic evolution is unplanned and because of that much harder to handle, for example, an introduction of new tax legislature can force the Payroll module to incorporate that in the entire system. Moreover it will require synchronization of commitment control with Financials. Another example will be, a workflow may vary from State to State, and same purchasing document may need different hierarchical approvals for different State Governments. So a fixed delivered workflow solution will not be compatible with such variance in requirements. Most ERP solutions are capable of handling scenarios for which the

architecture has to evolve dynamically (i.e., on-the-fly evolution). It does not affect other operations of the system. It is tailored to support unpredictable situations like this.

H. Verjus and others argues that Architecture Description Language is insufficient for designing a dynamically evolving system like ERP [15]. They emphasize on architecture-centric development with levels of abstraction. In this model, refinement happens at each level as the project moves forward. Architectural styles and associated languages can then be constructed using a meta-level tower. If we define a style using the nth layer of the language family, we can use the associated syntax to constitute an n+1 layer. By construction, architecture that is defined using the nth layer of the language family has its corresponding description in the n-1 layer. This could be a strong basis for a functional programming language like Scala for certain ERP solutions.

Any popular software which is at a leading position in the market always contain a flexible tool for user-friendly upgrade. The tool should be able to recognize multiple versions of the solution. It is essential that most recent version of any evolutionary software retains desired properties that have been already modeled and validated by multiple high-end customers. Sudden change of a table structure which handles sensitive data may force all the clients to change many customizations. This interdependence between delivered solution and customer enhancement can give rise to many problems and can impact adversely popularity and possible future expansion of the software. It is even more important for critical systems, in which introduction of errors often happen during badly managed evolutionary steps. In a case of heavily customized ERP system, such upgrades can be tricky. For instance, a small bug-fix or updating to a newer patch may become very costly. Although the change is subtle, it may need a repetition of the whole test cycle (unit, functional, QA, integration, system, alpha, beta, UAT, etc.). Any customization in

the application layer, also causes changes in the domain layer. Many modifications like this found to be turning into boiler-plate extensions. This is a hindrance for proper evolution and the software struggles to adapt and deliver new features. Business functions now depend on codes that are distributed in multiple layers. Not only that, those codes are now placed in multiple objects or in objects which are acting only as place holders. Those objects may not be part of the object family that provides the actual feature. The efficiency of system and people involved can largely reduce due to presence of such poorly designed customizations. Many a time it is the cause of frustration and unhealthy segmentation among stakeholders. Change management thus become the least preferred task by a project manager of an ERP implementation project.

The primary reasons of bug-fixes, patches or upgrades are,

- Version control of Database Management System
- Change in legislature, Business Procedures and/or Policy
- Released module was beta- and the supplier want expecting dynamic evolution
- New version of the implemented solution seems more suited with current market trend

Actually number 4 is a very likely scenario in today's economy. Most businesses who already possess complex ERP systems have the urgency to comply with technology advancements and standards. Upgrade specialists of ERP systems are now the most sought after workforce in the industry. Rapid and frequent evolution of the software is the root cause of this.

2.3. Chronology

In this section I will present a brief chronological description of how ERP evolved since 1960. Each sub-section represents a decade.

1960. The focus of Companies was mostly on inventory processing, and their planning and development of software packages essentially targeted to handle just-in-case inventory functions. So it could compensate manual operations like encumbrance, honeycombing, configuration processing, bill of material, etc. [16]. Those legacy systems were predominantly programmed in COBOL, FORTRAN or ALGOL. First, packaged business application in market was Material Requirement Planning (MRP). Joseph Orlicky, who made a proposal to build the solution, later became famous as the father of MRP.

1970. MRP represented a huge step forward in the materials planning process. For the first time, manufacturer had a master production schedule. This schedule was supported by the 'bill of material' files. Those files identified the specific raw materials needed for each item the manufacturer produces. A computer then successfully calculated gross material requirements. Such accuracy of inventory records enabled data availability of on-hand or scheduled-to-arrive materials. This was helping forecasting and decision making process. It then prompted an activity such as placing an order, canceling an existing order, or modifying the timing of existing orders [17]. Besides, tools and technology became imminent to support planning of aggregate sales and production levels (sales and operations planning), master production scheduling, sales planning and on-demand service, and high-level resource analysis. Incorporation of supplier scheduling inside scheduling techniques for the factory floor became very useful. [18]

1980. Manufacturing resources planning (MRP II) systems evolved to include the financial accounting system and the financial management system along with the manufacturing and materials management systems [18]. Such Push Systems were troublesome. Their rigidity, combined with lack of domain knowledge among analysts and lack of managerial interest were few barriers toward making them popular to clients.

1990. ERP evolve to include solutions for many areas, such as product design, information warehousing, materials planning, capacity planning, communication systems, human resources, finance, and project management. Hence, ERP did not have to remain limited to manufacturing companies. Since late 80s enterprise resource planning (ERP) systems for large complex organizations became available. Those who did embrace it instantly experienced significant improvement in their overall operation [18]. These were complex, expensive, powerful but proprietary systems. A new workforce cropped up as system consultants who were specialists dedicated particularly to tailor and implement these systems. Business Process Reengineering (BPR) of business functions resulted radical change in productivity and quality. These systems, unlike the old, traditional company-specific systems, are integrated multi-module commercial packages highly suitable for tailoring [19]. I want to quote the following from literature to illustrate the significance of this rapid advancement of technology, "by accessing enterprise-wide information from databases, IS (Information System) integration is providing numerous opportunities to coordinate organizational activities by facilitating communication and information exchange across departments without the need to go up and down the vertical chain of command. The access to timely, accurate and consistent information is crucial to business process improvement and accounting. Integration through communication networks and database systems, enables organizations to create and sustain process improvement through timely retrieval of consistent and accurate information."

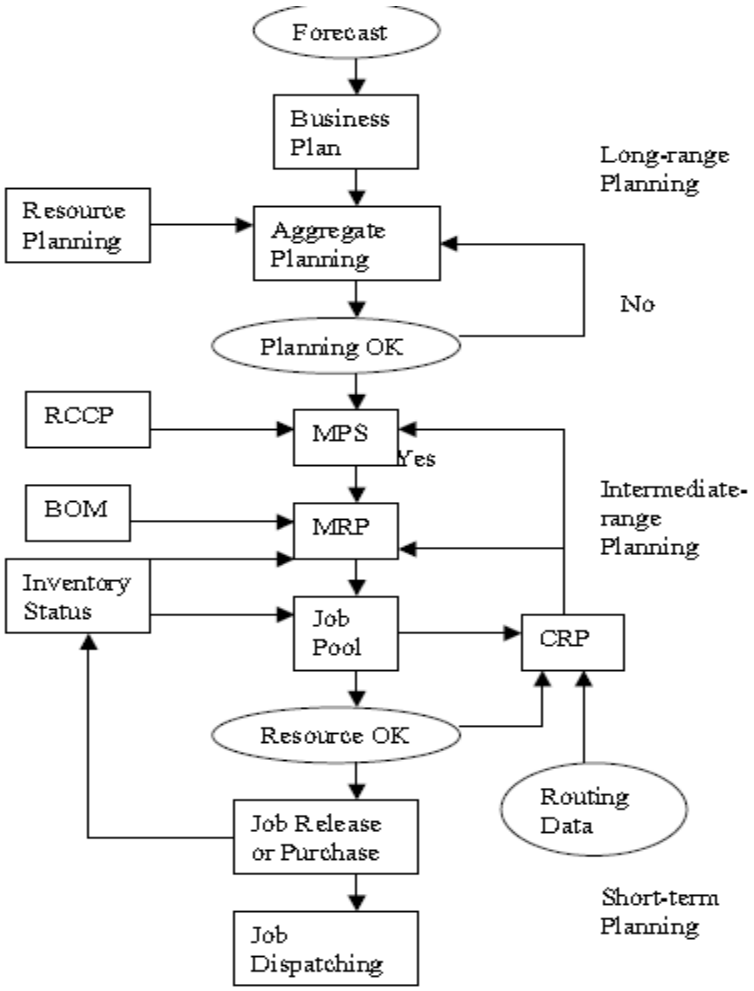

Figure 2.1. MRP II Flow [20]

2000. 21st century experienced significant advent in ERP systems as major businesses from both private and public sector were willing to spend an impressive amount of their investments for a robust synchronized distributed system to ensure smooth operation of the enterprise. Most of this inclination was due to the huge success from early implementations. As companies like Oracle started to buy smaller but successful companies, a shift of technology started to happen. Companies like Oracle were able to combine all the leading products into one specifically defined architectural model. The target is to provide customers a fully integrated

system which can cover and accommodate all parts of their business functions. Hardware advancement, real application cluster (RAC) in database management systems (DBMS), and significant improvements in Operating Systems also had a huge effect in the evolution of ERP.

2010. Oracle Fusion was first introduced in 2008 and it is an attempt to bring in all relevant products under one platform. Reliance of clients on such an integrated firmly packed solution is still facing lot of challenges in the industry. But definitely this will remain to be the focus of the industry as we move on to cloud s-a-a-s (software as a service) and p-a-a-s (platform as a service). Another challenge is to accessing applications through many different types of personal devices. The other revolutionary approach in ERP world is to develop full or semi-open source systems like OpenERP, OpenBravo, etc. Python is currently the most preferred open-source language for ERP developers. Whereas, Oracle introduced JDeveloper into PeopleSoft Interface technology after the acquisition of SUN and made Java the official language for Fusion.

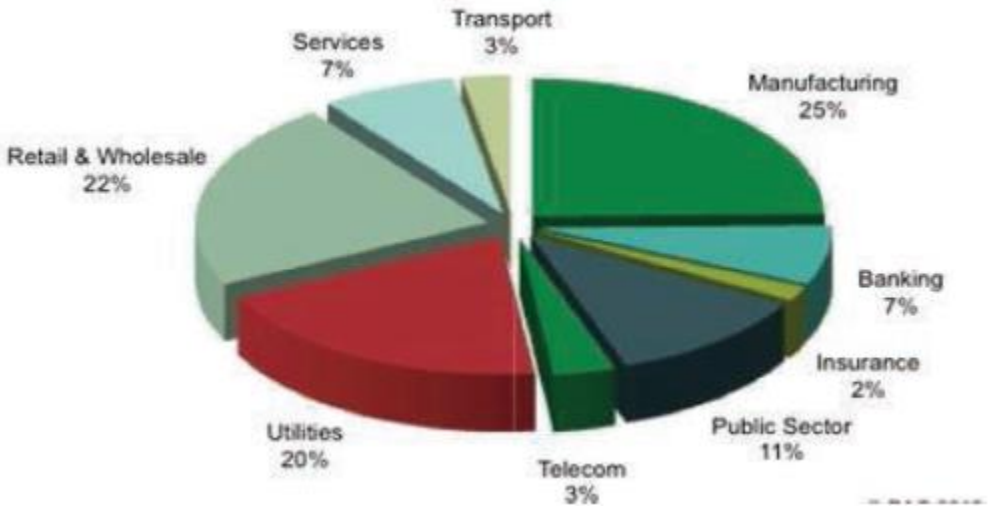

Figure 2.2. Global ERP usage in 2011 [21]

2020 and Beyond. Up until recently functional programming has been finding its way outside the research labs into major industries like banking and trading. To remain competitive with Haskell or Scala, principal programming languages like Java and C# were now delivering libraries and packages like Lambda to enable pattern-matching, higher order functions and lazy evaluation. These are most vital features needed for concurrent programming. As we are seeing rapid progress in this area, an ERP system using functional programming language is very probable in near future.

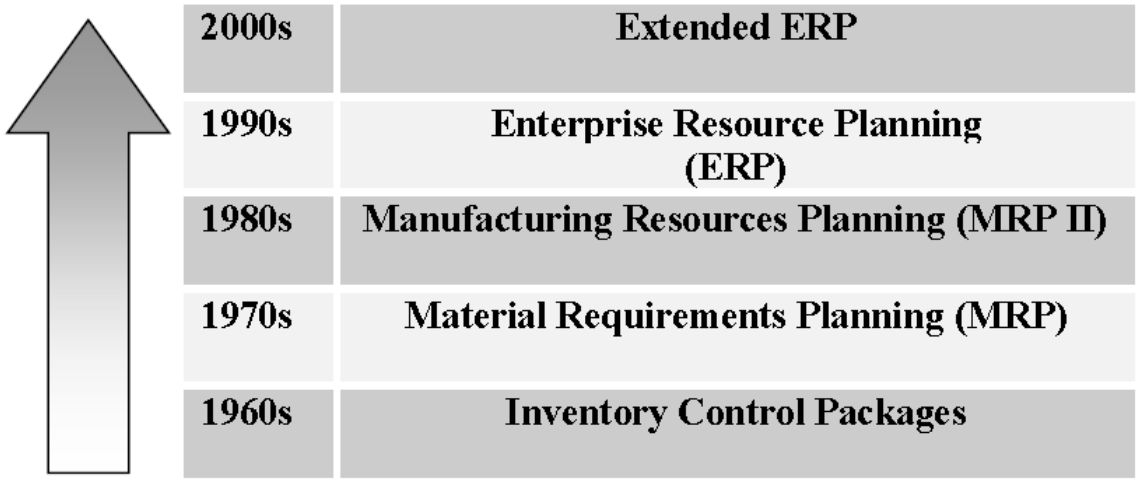

Figure 2.3. ERP Evolution [19]

2.4. Literature Survey

The Project Management Institute's Guide to the Project Management Body of Knowledge (Project Management 2000) defines risk management as the systematic process of identifying, analyzing and responding to project risks with an aim to minimize probability and consequences of adverse events to project objectives. Agrawal, A., Finnie, G. and Krishnan, P. describes the Organizational Risk Evaluation (ORE) framework developed based on the PMI guidelines with a hierarchical organizational decision-making process. ORE provides support for

all three levels of decision-making (operation, tactic, and strategy) based on the principle of 'divide and conquer' and fixed time-sampling, during iterative development phases. The ORE methodology of risk computation involves identifying a set of factors F with members $F_1 ... F_n$, assuming fixed weights w_1 ..... w_n for each factor. For each factor F_i it identifies a set of metrics M_i, consisting of metrics M_i^1 ...M_i^n. At any given time *t*, only a subset of *F* may be relevant, say F^t. Further, the contribution to a factor $F_i \in F^t$. at time *t*, may only be a subset of M_i, say M_i^t. The risk value due to factor F_i, at time *t* can be denoted as $\sum_{Mi \in Mi^t} M_i(t)$. Thus organizational risk value at time *t* is $\sum_{Fi \in Fi^t} w_i * F_i(t)$. Two organizational risk measures at times t_1 and t_2 are comparable if $F^t_1 = F^t_2$, and for every $F_i \in F^t_1$, $M_i^t_1 = M_i^t_2$, (that is the set of factors and metrics are same) [22]. Risk reduction is a sensible thing to practice for complex software projects. Opposite is a smorgasbord approach which can be detrimental to project outcome.

Ward and Bennett describe a legacy system evolution process by formally defining a wide spectrum language (WSL) [23]. The approach is first to translate a given source program P into an equivalent form P` expressed in WSL language, in which all subsequent operations are performed. This broad spectrum language must facilitate the representation of both low-level imperative constructs (e.g. goto's, aliased memory) and also non-executable specifications in first order logic. Although the paper is comparatively old, surprisingly ERP providers still follow a same generic model for evolution engineering. One of the key criteria of their approach was ability to select a sub-component of a large existing system and a guaranteed preservation of the interactions of the sub-component with the rest of the system. This creates a maintenance `hot spots,' and thus proven to be a piecemeal approach to evolution.

Mens, T., Jeff M., and Bernhard R. describe the importance of ADL and their viable use in current industry [24]. Depending on the software artifacts' type and granularity, the impact and

rate of change may differ. Source-code artifacts need to be frequently changed—for example, to fix bugs—but often have a local impact only. Changes to the architecture occur less often but have a global impact.

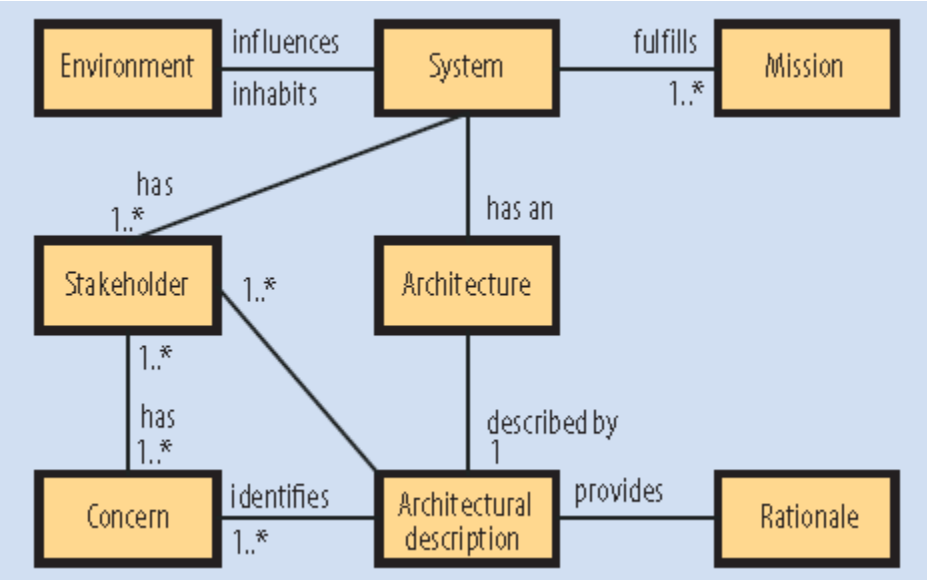

Figure 2.4. Fragment of IEEE Std. 1471 conceptual model of architectural description

One of the earliest research works in formal software evolution is an IBM funded work by Ward, M.P. and Bennett, K.H. [25]. The authors discussed evolution of Assemblers and provided a guideline for software engineers towards a methodical approach to evolution planning and decision making. According to Rashid, M. et.al. [19] an ERP system is required to have the following characteristics:

- Modular design comprising many distinct business modules such as financial, manufacturing, accounting, distribution, etc.
- Use centralized common database management system (DBMS)
- The modules are integrated and maintain seamless data flow among them,
- Ensure operational transparency through standard interfaces

- Complex systems involving high cost
- Highly flexible and offer best business practices
- May need time-consuming tailoring and configuration setups for specific business
- Modules work in real time with online and batch processing capabilities
- If not, will be Internet-enabled soon.

The authors also suggest, areas like aerospace, logistics, etc. provide opportunities for ERP to evolve. Through their work they tried to emphasize the need of downsizing ERP solutions to meet budget constraint of small to medium enterprises.

In fact, it was not easy to find an evolution research solely based on ERP software. Instead valuable works exist and the number is growing in the area of choosing the right ERP solution, for a particular industry or problem domain. Another good number of research exists in the area of cost-effective evaluation. But rarely one or two published work can be seen mentioning or forecasting on areas where evolution engineering is the original discussion. Although in a minor importance to extend their original research [19] or Mens, T. and Michel W. in their workshop report [26] discuss this topic. In later, authors describe many criteria to formalize the evolution process of software one of which is highly significant and has been observed as a common trend, that is co-evolution. ERP seems to be coevolving mostly with underlying tools - DBMS, middleware, Server OS etc. Knapp, A. et. al. discuss the significance of people in the process of evolution [27]. In software engineering people, process and product are interwoven to achieve success in a project. Gurba et.al. suggest a versioning repository technique to evaluate the knowledge measure of people involved [28]. The management keeps track of all the changes proposed by a certain individual or consultant and after a fixed period measures the value of each of those changes. This evaluation also reflects the resource capacity

and their utilization. But many researchers question the method of using a versioning repository for evolution practice. The reason is historically a repository was always in use for configuration management, out of which evolution merely *evolved* as a side-effect.

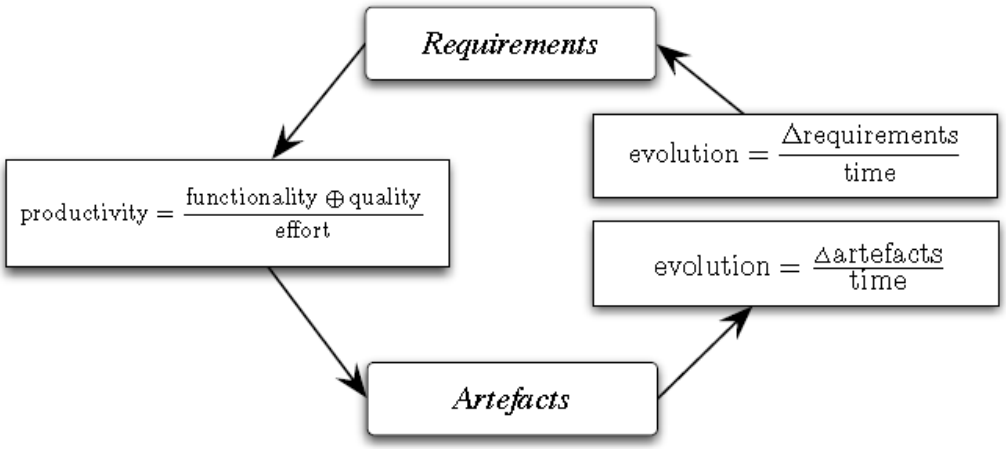

Figure 2.5. Software evolution and productivity [27]

In figure 2.5 we can see that degradation of quality causes productivity to decrease. The authors agree to three phenomenal observations, evolution is inevitable and also backbone of productivity, dynamic abstraction is the key to handling evolution and agile methodology which emphasize people over process is a suitable manifesto for initiating change.

Most ERP systems support object oriented programming (except SAP at core, which follows procedural programming) with new generation tools and technology. Some of the works mentioned here seem to be adjustable for better enhancement of ERP to reach optimization goal. Some authors state that non-functional requirements simply cannot be formalized [27]. But, I was able to find research evidence that this is not true. One such notable research work uses UML-RT meta-model with hypergraph to handle evolution needs dynamically [29].

Gorschek, T. et.al. show how a model-driven technique of requirements elicitation can help gather vulnerable gaps profoundly and efficiently for domain specific systems [30]. Sometimes following a rigid iterative method can be too inflexible and in such cases, out of the box thinking might help. Gathering various requirements precisely will need a better management in place as a prerequisite. With growing domain knowledge of previous legacy system and ERP, validations of certain facts need prioritization. One significant advantage would be reuse of artifacts. Similar to basic product line engineering, reusable artifacts can be defined as leaves of a core. It can cut down cost, create a window of opportunity to evolve, save additional time for development and utilize available resources without asking for more. The other factor is the delivery date. In most cases ERP implementations fall short of time due to possible dynamic evolution combined with prolonged customization. We find that applying reusable developed pieces from one project to another can be a successful venture for ERP procurement.

Productivity improvement is essential for adopting evolution. It frees up people and resources which eventually minimizes effort and overhead [30] for similar results. Recently such approach has become common in the industry with the successful application of market-driven requirements engineering process model (MDREPM). The primary driver of the model is exhaustive communication and graphical assessment of survey results to identify areas of improvement. Such questionnaire based attempt can be a good feed for the domain analyst to determine problem areas. It can also help recognize and forecast the pattern for future evolution trend. The main drawback of this approach is the lack of totality in the knowledge of the domain. Also consumers of same group can narrate their needs differently. If a qualitative language can

be documented and the participants can get information about the grammar beforehand, it could be a much better approach to determine software evolution criteria.

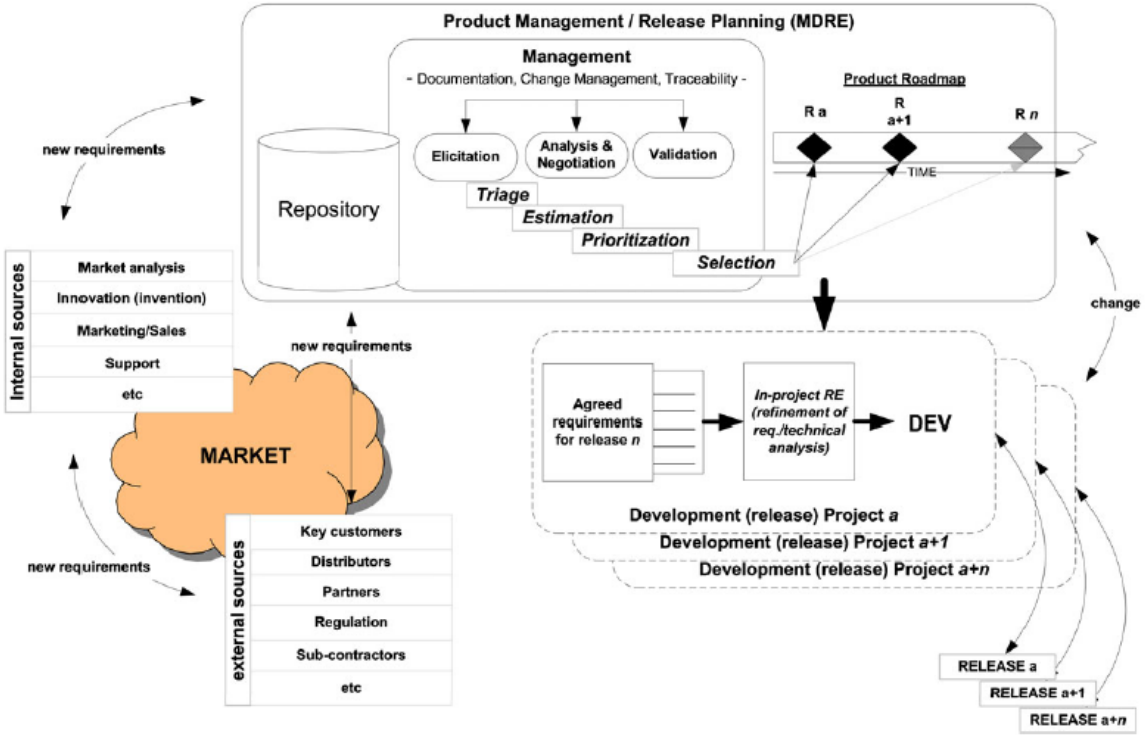

Figure 2.6. Model Driven Requirements Engineering [30]

Table 2.1. A model to record future requirements [31]

Attribute	Description
ID	Unique identifier, e.g., auto number
Title	Title for the requirement
Description	Free description of the requirement
Rationale	A description of the rationale/benefit of the requirement from the req. source's perspective
State	What state the req. is in at present, e.g., new, dismissed, specified, planned for released, released, etc.
Version	Version of the requirement (maybe with the possibility to view different versions and differences btw. versions
Source	The source of the requirement, that is the one who champions it
Estimation	Cost/time for implementation
Dependency	Dependency and type of dependency
Priority	The priority of the requirement on a scale of 1-5 where 5 is more
Test	Links to test cases
etc.	etc.

In short, to avoid chaos and mitigating future threats evolution engineering should be well optimized. A rigorous testing procedure for evolving pieces is necessary for successful implementation. The failure rate of ERP is alarming, and the most common reason is unplanned customization. New vendor features continuously disrupts the past developments and the client eagerness to upgrade to newer releases continues to blow up the associated cost.

McQuaid, P.A. presents how mission critical projects can create risks to an extent as it can sometimes affect human lives [32]. Management inadequacies, lack of appropriate follow through, overconfidence in risk assessments and limited incorporation of software engineering phenomenon were the principal reasons for this. The three vivid case studies of space travel, warfare and medical treatment respectively shows the flaws in evolution and it is understandable a better domain analysis during initiation phase could minimize such loss. Only fixing the bug whenever it shows up is not going to resolve all the issues. Assessing Risk should be in the context of the domain. Audit trails armed with incident report database could help understand better the system and that way mitigate future mishaps [33]. It seems integration of event reporting techniques within domain analysis can be a better and cost-effective approach towards robust evolution.

2.5. Related Works

Software Development Life Cycle for agile implementation of ERP software goes through an iterative process with fair amount of platform maintenance. The vendor works with client to take care of dynamic evolution. Figure 2.7 depicts the typical flow of a developed piece of functionality, both off-shore and on-site. It can be a customization, report, interface or a small change in delivered portal.

Figure 2.7. SDLC for ERP implementation (On-shore and Off-shore combined) [34]

2.5.1. Elicitation

Requirement engineering seeks to develop tools and techniques that support the definition of correct, complete, consistent, and unambiguous software specifications [35, 36]. Requirements engineering has attention for ‘‘soft issues" such as politics and people's values. Dealing with delicate issues can be problematic as there is little guidance on how to do so. Resistance to change has so far been the greatest inhibitor for achieving successful implementation. Requirements Engineering involves requirements elicitation, analysis, specification, and validation. The specification process may be formal or informal. During Business Analysis interviews with users at various phases of a project is a necessity. At *Initiation*, stakeholders are interviewed to determine scope, rationale (for conversion, report,

interface, security, portal, workflow, administration, etc.) and core gaps in requirements. During *Analysis*, users meet to discover and document requirements for enhancement, customization and re-engineering. Sessions with stakeholders while in testing phases help verify and validate those gaps. The expected outcome from those sessions is either mitigation already in effect or they are negligible or can be skipped totally. It is worth mentioning that these approaches are iterative and agile. From chapter 1 we also know that they are parts of critical success factors as well.

Elaboration is a strategy in which individuals verbally expand, or embellish, on a concept, domain, or knowledge that is new to them. To ensure that these discussions create genuine learning, learners will

- apply their initial understanding of the domain or concept with concrete examples (or analogies) with which they have had experience,
- explain their assumptions in regards to their view of the target concept,
- identify alternative assumptions that could apply, and
- account for the concept using multiple interpretations, different formats, and varied perspectives.

Collaborative elaboration (CE) extends this notion of discussion to suggest that a learner is unlikely to find these inconsistencies by herself. Collaborators who are surrounding the learners first will encourage the learner to answer the "why" questions. It is a reminder for the learner to look for previous experiences and convey that knowledge to make the current concept more concrete. In general, collaborators guide the learner to use the CE techniques. Formulating her explanations can also help unwind the early resistance. It is this guidance and gentle probing by others that makes the learning process a collaborative one. Actor learning is limited when contextual practices remain unquestioned and cognitive frames remain unexplored. The use of

CE strategies may provide a natural means by considering those factors during the design process. As such, CE may provide a means to "break powerful imageries and institutional bonds at a deeper level," which may become highly useful for complex implementations like ERP [37].

2.5.2. Domain Analysis

Reuse of application domain knowledge is becoming more and more important in software engineering. It is because software is now considered more of an asset rather than just a mere service. It is now well established in the industry that domain analysis through proper modeling, a flexible but well-defined architecture and a dictionary defining the language, can be the principal ingredient to channel a software implementation project towards success. During early days of reuse the central reusable piece was a piece of the code. But with expansion of scope of IT, the various myriad deliverables of early phases of development are now considered as reusable assets by software engineers. In this discussion, I will indicate the areas to improve in Enterprise Resource Planning implementation that will provide a better insight for future projects. During domain analysis, we end up with three valuable outputs,

- domain model, which describes the standard functions and relationships between data and artifacts
- a dictionary, which defines the terminologies used in the domain
- an architecture, which describes the packages, control, and interfaces.

Reuse can cover family of systems in a domain. It produces a domain model with parameters to accommodate the differences. It defines a standard architecture based on which IT can develop and integrate software. That way by domain analysis we can achieve the following advantages,

- reuse of gathered knowledge for future project

- determining a context platform or framework by which we can ascertain reusability of candidate components,
- classification, storing, refactoring of components marked reusable
- a framework for tools and systems synthesis or meta-system

It usually involves a reuse analyst and a domain specialist. With the assistance of domain expert the reuse analyst decides on three levels of reuse:

- system
- sub-systems
- functions in requirements and components in design

By such analysis, we may conclude on few answers to help finalize a reuse plan,

- is the component functions already obsolete for future implementations
- what degree of commonality does the component carry in the domain
- is there any duplication in application
- what are the hardware compatibility matrix
- -how the software responds in different platform
- is it optimized
- how feasible is porting of the component from project to project
- is it suited for change management
- can the component be an originator for newer components

Table 2.2. Different interviewing methods for stakeholders [38]

Format	What	When	Benefits	Disadvantages
One-on-one interviews		During Initiation and Analysis steps	Is easy to organize	Reconciling discrepancies is time consuming
Brainstorming	Group interview for enlisting new ideas	During Initiation step and whenever the project is "stuck"	Breaks old ways of thinking	Does not yield detailed requirements
Joint Application Development (JAD)	Group interview to gather requirements	During Analysis step	Simplifies reconciling of discrepancies, decreasing analysis time. Can be used to create various deliverables, including BRD Proof of concept Strategy Screens Decision tables	Difficult to get all interviewees in one room at same time Group-think
Structured Walkthrough	Group interview to verify requirements	During Analysis step, after early draft of requirements is available	Moves testing forward, reducing the impact of mistakes	

We successfully reused a lot of our implementation knowledge from previous project to a newer one. Like other projects, we used finite state machines and data flow diagrams to better visualize the relationship between different functions in the domain. This reuse of feature modeling is successfully carried over from one project to another. It also gives assurance of managing variability in ERP arena. Here the potential trade-off is data integrity and sustenance with financial advantage.

So, what is the scope of reuse or what deliverables we can choose as reusable pieces of a domain? Aside from code; designs, specifications, test cases, planning and approach documents

and problem definitions are the fair candidates for reuse. For example, studies on spillover effects of knowledge between related research programs, product generations, and manufacturing organizations, as well as studies of best practice transfers, focus on how a "recipient organization" acquires and applies the knowledge of the "source organization" in an effort to replicate the essential elements, etc. The early literature on reuse discusses very large scale reuse (VLSR), encouraging reuse of code components at an enormous scale with a lazy evaluation or call-by-need for small, relatively unimportant details. The effort to reuse of project management and engineering method mostly compensate at what level of abstraction newer components can link to the existing implementation. It defines the degree of generalization of the implementation project itself. So to maximize possible reuse culture among engineers, opportunities must be sought at different phases of system development. Our approach is to enable identification of components to reuse early during elicitation. We find that it can maximize the overall productivity in subsequent steps. The greatest occurrence of VLSR is reusing an experienced project team for similar projects to apply their retained insights, which is termed as “personnel reuse” [38]. The overall guideline for reuse may contain,

- domain analysis,
- assessment,
- reuse of all deliverables from different phases for chosen component,
- generalization,
- ensuring flexibility towards change,
- practice of reuse as a non-functional requirement etc.

Simultaneously with their radical redefinitions, reuse specialists in the more innovative cases can develop conceptual approaches that are not tied to the past, for example, they can use analogies

and extensions to anchor the concepts [37]. In our re-implementation of test-first and parallel processing techniques in data conversion from legacy to ERP as we continue applying the technique in different projects, a similar reuse was necessary to make the porting happen.

2.5.3. Formal Modelling

Gradual increase of complexity in information technology proportionally increases the likelihood of error. The critical success factors mentioned in section 1.4 indicate the potential loss of time and money and of productivity and resource if the highly volatile nature of such projects were not undertaken seriously. How can we reach the goal of delivering a reliable system? Formal methods have been proven to significantly increase our understanding of a system by revealing inconsistencies, ambiguities, and incompleteness that might otherwise go undetected [69]. Our case study at section 3.9 and later a data mapping between two systems for conversion in Chapter 4 mainly follow these criteria regarding project success, reduced time and cost-effective solution.

Requirements analysis necessarily deals with customers who often have an imprecise idea of what they want; formal methods can help customers nail down their system requirements more precisely [69]. ERP implementation is no difference, and through my work in Chapter 3 I will show how it can identify specifications and group them accurately in FGA phase.

Formal methods research in software engineering has produced several classes of analysis that can prove useful in software development. In section 2.6.1.3, in literature review we found that there is still no relevant research involving application of formal methods, either in industry or in academia, for ERP related projects. I also mentioned in Table 1.4 how application of formal technique in my research could relate to critical success factors.

A formal analysis technique may also be classified as either "light-weight" or "heavy-duty." The user of a light-weight method does not require advanced mathematical training and theorem proving skills [70]. The application of covering properties of a bipartite graph to specify the requirements of a project replacing legacy with ERP system is a light-weight approach.

Formal methods have been proven to be useful in debugging and assessing precision of algorithms and protocols. The other class where it is commonly useful is where it demonstrates that a specification of a system (or system part) satisfies certain properties. A specification is a process of describing a system and its desired properties. It includes some preconditions and some post-conditions. The kinds of system properties might include functional behavior, performance characteristics, or internal mechanism and structure. In my case, I am dealing with two systems and following a graphical method to precisely identify the specifications for recognized business functions and data pattern.

The process of specification is the act of representation with precision. The analysis of such specifications does not usually require deep reasoning. Such light-weight method as experienced in my project seems more cost-effective than heavy-duty theorem proving. It can also help produce feedback which is less cryptic and more satisfactory to all stakeholders. In section 1.4 I mentioned some of the inhibitors of such project. Those can easily be suppressed using clarity in feedback received from this method which can also act as catalyst to advance to a deeper understanding of both systems in question. So to uncover design flaws, inconsistencies, ambiguities or (as in my example project) incompleteness we apply covering properties of bipartite matching. The model can also be a useful communication tool between customer and implementer, and between implementer and tester.

Initially, early introduction of formal methods did not receive much applause due to many inscrutable limitations such as recherché notations, lack of scalability and inadequate tool support. The expertise and available resource were also a big concern. It is now agreed in the software community that formal methods must be convenient and easy to use. If the learning curve is too steep or effort required to apply is too cumbersome it may not be the approach of choice. ERP implementation projects deal with many modules (Figure 1.1). It is an extensive and time-consuming process. So a preliminary look at the problem can make the software engineers reluctant to take the approach. The initial notion would be it will require a heavy-duty formal approach and so can be ignored. But, through our case study we found that a light-weight approach to certain modules or chosen problem areas can be the way to handle some of the critical factors. This will also help reduce the state explosion problem which emphasizes on the reduction of size of the state space. An effective way to reduce state explosion is to apply abstraction. It will remain a challenge to establish abstraction as ERP systems get more and more distributed and diversified. But, in some of the core business functions abstraction is applicable and can help reduce the state space.

Another road block is the language. It should be "natural" with syntax and semantics within the grasp of the stakeholders. The language must also have an explicitly defined formal semantics, and it should scale. In my case, I am using simple graphical notation which is independent of any specific formal language. Because of simplicity and light-weight, this approach can easily be communicated and presented with well-known mathematical techniques.

Formal techniques with their ever growing complexity become costlier to automate. My approach while able to cover the whole gamut of ERP implementation can easily be incorporated and represented using existing scientific software without any transformation.

After first time application of the technique I found that it also helps the workforce to become more adept in design analysis, code optimization, test case generation, etc. as we carry over the knowledge from one project to another. The main activities included in formal methods are: writing a formal specification, proving properties about the specification, constructing a program by mathematically manipulating the specification, and verifying a program by mathematical argument. Program verification is only one aspect of formal methods [71]. In my research, the properties analyzed included both application-independent properties and application-dependent properties.

Following the model to establish specification through functional analysis (presented in Chapter 3), I was able to answer some of the complex questions that are typical for a software project such as,

- What exactly is the required system behavior?
- What are the critical system properties?
- How can I make the specification easy for others to understand?
- What are the likely changes to the specification and how can I organize the specification to facilitate those changes?
- What should the missing behavior be? How can this non-determinism be eliminated? How do I change the specification to avoid this property violation?

Another challenging aspect of the recent software industry is to accommodate reusability to prevent reinventing the wheel. When a model is established based on core functionality, it can also iteratively evolve within the product line. More features get added to core level from application level. It will ensure reusability along with reliability as the retesting of those features gets eliminated.

Formal methods have been useful in the different stages of software development, especially in requirements specification, analysis, and design [39]. Using formalisms force the software engineer to ask questions that may have otherwise got postponed until implementation [40]. It can lead to early fault detection in the software documentation and may be the distinguishing factor between a successful or failed project. Horowitz and Munson first stressed the need for formal approaches toward reuse [41] basing the content, concept, and context of a domain. The framework of their initial analysis used Data Flow Diagrams (DFDs) considering tight-coupling to the input and output parameters. The idea was that developing based on data is easier than based on functionality. This idea prevailed like a myth during the early days of reuse. Programming paradigm like functional programming which is purely based on lambda calculus and inherently follows formal specifications provides the opportunity to look at issues like reuse in a different perspective. In this paradigm, functionality gets the priority in development regardless of quantity or value of data to handle. A lot of work in the literature stretches the need of formal methods, more precisely graph theorems, for proper software evolution practices. My research finds it important for ERP procurement as well. With the high complexity of the ERP software engineering problem, one would expect many approaches that focus on quality construction and validation. However, areas of ERP development with formal modeling and testing continue to be underdeveloped in the research community [10, 11, 42]. My research seeks to define the co-relationships between the requirements of an existing legacy system and an ERP software during migration. Formalizing these relationships seeks to help ERP requirements engineers identify and remove ambiguity at an early stage. To address data migration, we apply a test-first approach that utilizes real data from the legacy system to guide the ERP customization process.

Through formalization, both the client and vendor can gain a better understanding of the two systems, and focus their effort on correct construction as opposed to post-hoc analysis [39]. General questions that should be asked when formalizing the requirements model of a new ERP system are: 1) Does the proposed functionality already exist in whole or as a part of the legacy system or is the feature entirely absent? 2) If the feature exists in the legacy system, can it be extended for what is required, and suppressed to disable unnecessary or burdensome features? Answering these questions can identify the essential elements and relationships inside the model. Graphing the two sets of features establishes a bipartite relationship between the two systems. We can, therefore, model the problem as a bigraph, so we can apply graph theory in the context of ERP requirements analysis.

2.5.4. Product Line Engineering

The SEI Framework for Software Product Line Practice, Version 4.2 serves a number of purposes by identifying and defining

- fundamental concepts which forms a platform for software product lines and the essential activities to consider before developing a product line,
- practice areas that an organization developing software product lines must master, and
- needed guidance to an organization about how to move to a product line approach for software.

From our discussion in chapter one and also in this section, it is evident from column one of table 2.3 that product line variability models represent the "knowledge base" containing contextual information. Our initial emphasis for domain analysis was also an inspiration from the inclusion of that in column one of the table. It includes architectural aspects of service based

solutions as well as information on their inherent variability. Most major ERP solutions in the market are currently loosely-coupled and have moved on to accommodate service-oriented architecture for handling asynchronous communication in intra- and inter-system data exchange operations. Integrating tools such as service bus with SOAP or REST-ful API's and XML data transfer along with AJAX interactive pages and dashboards has been the preferred evolution trend to ERP software for last five years or so [44]. We find it evident to claim that involving end-users as early as in the elicitation phase can mainly help planning service-oriented ERP features, for example those interfaces which involve significant amount of transactional data (both inbound and outbound). A service provider usually knows how a service is adapted to

Table 2.3. SEI Practice Areas for Software Product Line [43]

Software Engineering Practice Areas	**Technical Management Practice Areas**	**Organizational Management Practice Areas**
Architecture Definition Architecture Evaluation Component Development COTS Utilization Mining Existing Assets Requirements Engineering Software Systems Integration Testing Understanding Relevant Domains	Configuration Management Data Collection, Metrics and Tracking Make/buy/Mine/Commission Analysis Process Definition Scoping Technical Planning Technical Risk Management Tool Support	Building a Business Case Customer Interface Management Developing an Acquisition Strategy Funding Launching and Institutionalizing Market Analysis Operations Organizational Planning Organizational Risk Management Structuring the Organization Technology Forecasting Training

different contextual situations, (1) in our approach this knowledge is codified in the form of product line variability models, (2) which is used as input for an end-user requirements elicitation tool. (3) The tool attempts to configure the required product based on the answers of the user and (4) updates the model with new contextual information if there is any (5) need to generate or to modify a service based prototype application. Previously there were no approaches to formalize the requirements associated with porting an existing legacy system to an ERP

solution. In Chapter 3 as part of the publication I will further mention some of the background works in this regard.

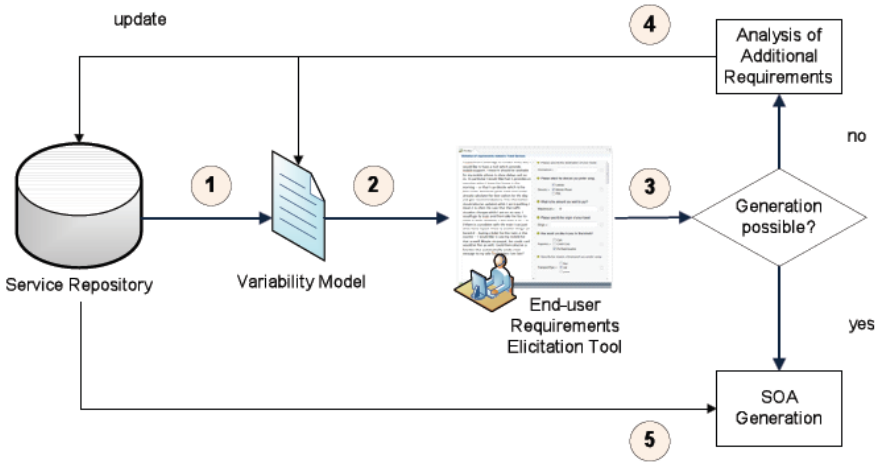

Figure 2.8. Requirements elicitation for services [44]

2.5.5. Data Conversion

Implementation of an ERP system as a replacement of a legacy software typically involves porting legacy data from accounting, human capital management, customer relationship management, services, sales and other divisions to a common ERP database, and building applications that access and update the central data store. The primary goal of ERP implementation is to standardize and streamline the business functions and information flow among these various divisions of the organization. ERP has, therefore, become a prime contender of software implementation for large organizations such as state governments, multi-site and multinational corporations. A grand software engineering challenge of ERP system development is the mapping and conversion of data from different divisions into a shared database. Properly migrating information from existing legacy systems to ERP software modules is, therefore,

paramount to the success of a project. However, the large size of system and volume of data makes this an incredibly daunting task. Even the data from a single division can be quite difficult to manage correctly due to complex relationships. Software testing continues to play a significant role in ensuring that software systems meet the needs of customers. Given the complexity of ERP software development problem, and its tremendous impact on the industry, one would expect that there would be a lot of research devoted to testing ERP systems. However, this is not the case as only a few works can be found in the academic literature [10, 11]. Furthermore, although software vendors have their methodologies for testing ERP systems, there has been little collaboration and dissemination of knowledge concerning effective software testing strategies for ERP systems.

In general, there has been little research on testing ERP systems, and the topic continues to be neglected in academic research. To the best of our knowledge, the approach presented in this paper is the first to describe how we can apply a test-first method to ERP data conversion. However, several researchers have recognized the need for developing new testing methods, tools, and data-driven testing techniques for ERP systems [10, 11, 48, 50]. We now summarize related work found in the research literature. A more exhaustive literature survey is available in Chapter 4 as part of the publication along with the description of the conventional approach and the limitation involved which were mitigated by my work.

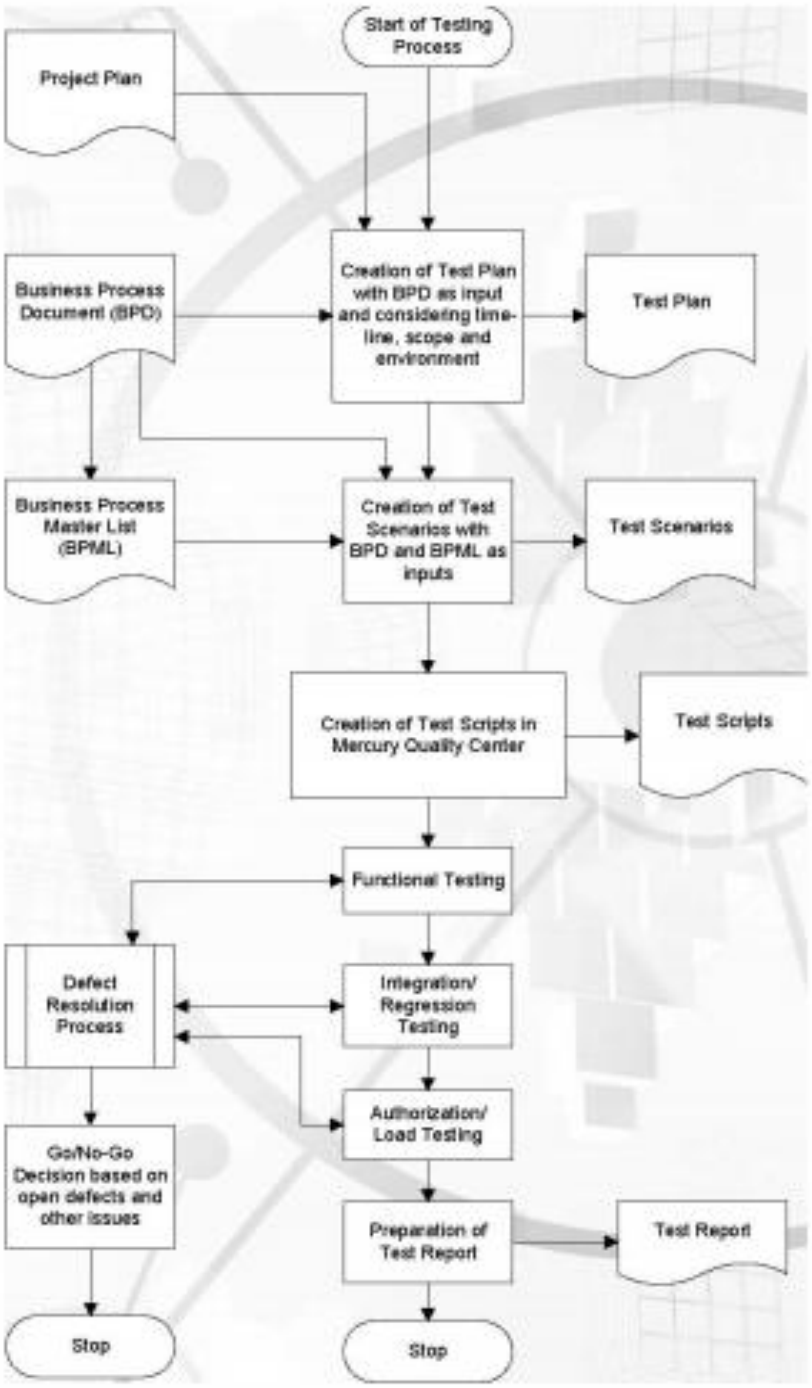

Figure 2.9. Process Flow of SAP Testing [49]

2.5.6. Test Driven Development

O'Leary, Pádraig, et al. define APLE, a software product line engineering process intermingled with agile manifesto to enable better response to change. It involves more inputs from people, and reduces the rigorous documentation. It combines XP (eXtreme Programming) practices with Scrum backlogging [52]. These identified common goals of agile methods and software product line include enhanced customer satisfaction, increased team productivity, reduced time to market and reduced product expense. An Agile-SPL approach may have the potential of supporting the effective and efficient development of much larger families of products than a traditional SPL approach. It can accommodate the use of test-driven development for developing platform and/or product components. The authors suggest two phases, namely derivation and configuration, before going for TDD. The goals of derivation are,

- define scope and requirements of product,
- define guidance for decision makers,
- allocate each requirement in certain iteration.

The goals of configuration are,

- make a partial build (initially a prototype),
- iteratively optimize the use of platform artefacts.

The principal goal of the construction phase would be to take care of the dynamic evolution portion that is out of scope of configuration. Although the authors mentioned TDD, but nowhere in their discussion of the approach was it evident how they were applying the framework. Test-driven development (TDD) uses quick and brief development iterations based on test cases. Tests are written prior to the coding process, which then drive enhancements and new code features [53]. TDD has gained a lot of momentum in the software community because of its potential to

improve the state of software construction [54]. In test driven approaches, developers combine their understanding of the technical system requirements with a broad view of the business needs.

2.5.7. Maintenance

Hooper J.W. and Chester R.O. argue that maintenance is itself a collection of similar activities as same as the development phases [38]. Not much to deny about that especially for ERP software as upgrades and enhancements are common for ERP clients. As we have seen, in general, maintenance cost surpasses the actual cost of implementation in many ERP projects, this difficult truth has taught consultants the hard lesson of initiating reuse of knowledge and customizations as much possible to minimize such expenses. This can reduce client discomfort with added satisfaction from service.

CHAPTER 3. FORMALIZING REQUIREMENTS IN ERP SOFTWARE IMPLEMENTATIONS[1]

3.1. Introduction

Research surveys have identified enterprise resource planning (ERP) implementations and upgrades as top IT priorities [12, 13]. These studies suggest that commercial off-the-shelf ERP systems are likely to survive within the recent trend towards cloud and app-based software. For example, PeopleSoft introduced FluidUI for handheld devices. Porting existing legacy software to a new system has been an important factor in the success of ERP implementations. Proper migration of both functionality and data is necessary to satisfy key stakeholders, which include clients, vendors, and third-parties. However, the high complexity of ERP makes developing these types of systems non-trivial.

Requirement engineering (RE) seeks to develop tools and techniques that support the definition of correct, complete, consistent, and unambiguous software specifications [35, 36]. RE is recognized as a vital aspect of software projects since the quality of requirement specifications impacts all phases of software development. Formalizing RE activities can help to identify requirement problems early, and avoid long delays, missed delivery deadlines, and budget overruns.

With the high complexity of ERP systems and the potential benefits of formal methods, one would expect that there would be many formal approaches to ERP requirements

[1] The material in this chapter was co-authored by Talukdar Asgar and Dr. Tariq King (during his tenure in NDSU). Talukdar had primary responsibility for collecting samples in the field and for interviewing users of the test system. Talukdar was the primary developer of the conclusions that are advanced here. Talukdar also drafted and revised all versions of this chapter. Dr. King served as proofreader and checked the math in the statistical analysis conducted by Talukdar.

specification, development, and validation. However, formal modeling and validation of ERP systems continue to be underdeveloped in the research community [10, 11, 42].

Furthermore, there is a little adoption of formal methods for ERP software development in industry. In this paper, we present a formal approach to product-line engineering for ERP systems to facilitate dynamic evolution and reuse across software projects. Our approach aims to guide requirement engineers in the specification, analysis, and estimation of the ERP customization effort. The major contributions of the paper are:

- Describes an approach that uses bi-graphs to assist software engineers in mapping legacy functionality to ERP features,
- Presents a small case study on an event bidding system to demonstrate the feasibility of the formal specification and analysis approach
- Discusses our experiences using the approach in an agile product-line development cycle in the industry.

The rest of this paper is organized as follows: the next section provides background material necessary for understanding the paper. Section 3.3 defines the research problem under investigation. Section 3.4 presents the formal modeling approach to ERP requirements specification. Section 3.5 describes how the approach can be used to support decision-making during requirements analysis. Section 3.6 contains the case study. Section 3.7 provides related work, and Section VIII concludes the paper and discusses future work.

3.2. Background

This section contains background material on enterprise resource planning, fit-gap analysis, and bipartite matching as it related to requirements engineering.

3.2.1. Enterprise Resource Planning

ERP, a term coined by Gartner Research Group in 1992, is a type of Commercial-off-the-Shelf (COTS) software that seeks to provide an integrated, optimized solution as a bundle for businesses to take care of functions including all internal and external operations necessary. The breadth of this definition encompasses the complexity of large-scale integrated systems and the cost associated with implementation and maintenance. ERP systems allow management through integration of all aspects of business including production planning, purchasing, manufacturing, sales, distribution, and accounting. ERP solutions allow the operations mentioned above to be performed cohesively [55, 56].

The ERP implementation process consists of the following phases: a fit and gap analysis (FGA) to identify business processes and customization requirements, which act as the principal decision-making phase for customization needs; system design and development; data conversions from legacy system; testing of the new system; training of the super-users and end users; and deployment. These activities may vary with the nature of the project based on if it is an upgrade or merger. ERP is distributed in two forms [57]: Generic or Preconfigured. Generic ERP software targets a variety of businesses and industries, and therefore must be configured early. Preconfigured ERP software is tailored to a specific industrial sector. However, both forms need customizations.

3.2.2. Fit Gap Analysis

During requirements elicitation, interviews with users during various phases of a project are a necessity. At project initiation, stakeholders are interviewed to determine scope, rationale and core gaps in requirements. During analysis, meeting users help discover and document requirements for enhancements, customizations, and reengineering needs. Sessions with

stakeholders throughout the testing phase helps to validate and verify that gaps have been closed. In agile development, these activities are iterative and incremental, and therefore fit-gap analysis (FGA) represents a plausible solution.

In FGA, we deal with two domains. The first is the domain that already exists, which is the source domain of the business needs and basic non-functional requirements. It is commonly termed as the legacy system. The second domain is the demonstrative ERP solution. The ERP demo environment is the target domain which will be customized and adjusted with required configuration. In the beginning, a high-end workable prototype is placed for analysis. Analysts can, therefore, choose to work on different modules of functionalities to identify the fitting business functions, discard obsolete legacy features, suppress additional ERP features, and/or prescribe necessary customizations.

The principal goals of FGA include: creating a workable prototype of the target domain; assigning legacy experts and domain consultants for business areas; engaging associated technical experts of both legacy and target domain in the analysis; documenting and specifying functional needs for each requirement; and iteratively refactoring specifications according to an agile development life-cycle. Non-functional requirements such as the minimum hardware and platform vary based on the vendor software and release version.

3.2.3. Bipartite Matching

Formal specifications use rigorous mathematical notations and techniques to precisely describe the properties a system should have, without unduly constraining how to achieve them [40]. In requirements specification, analysis, and design, formal methods can play a significant role towards success [39]. Using formalisms force the software engineer to ask questions that may have otherwise been postponed until implementation [40]. It allows early fault detection in

documentation and may be the distinguishing factor between a successful or failed project. Horowitz and Munson first stressed the need for formal approaches towards reuse [41] basing the content, concept, and context of a domain.

In graph theory, a bipartite graph or bi-graph may be defined as a graph G = (U, V, E) where: U and V are disjoint sets whose elements make up the vertices of G, and every edge e ∈ E connects a vertex in U to a vertex in V [58]. In other words, U and V are independent sets that partitions G into two. Applying this model to ERP requirements engineering involves matching functional requirements into the two domains as mentioned above as relationships on a requirement-to-requirement basis. A one-to-one or many-to-one mapping may be used to associate legacy to ERP. One-to-many relationships can be ignored since it will be contradictory to both reusability and portability. In other words, the probable transfer will be shifted to a more robust and cohesive product than the one proposed if the latter is prominent. Thus legacy features are ensured of not being omitted or incorrectly mapped.

3.3. Research Problem

Portability is a special type of reuse, which can be determined from rigorous domain analysis [38]. More specifically, it is the ability to reuse some aspect of the software within different operating environments and contexts. For ERP implementation we redefine portability considering the product line engineering approach, combined with the agile manifesto [52]. Replacement of a legacy system with ERP software requires that at a minimum the desirable qualities of the existing system be ported to the new implementation, both for system features and underlying data. If there are no significant improvements, the acceptability of the new ERP environment would be questionable. In the worst case scenario, the client could retract to the

previous system in spite of heavy losses. Here we apply the two systems using bipartite graphs to determine matching functionalities through analysis.

3.3.1. Functional Challenges

Developers of ERP product line software are faced with the task of analyzing the portability of existing legacy functionalities, which have already met the users' satisfaction criteria, and mapping them to corresponding ERP features that should also meet client needs and expectations. If the approach to tackling this problem is not exercised during initiation, monetary commitments may fall short, or the project life cycle may cross the originally planned timeline.

Some aspects of the legacy functionality that may make attaining the same (or greater) level of satisfaction in ERP difficult can be if the desired feature is: operating smoothly; heavily cross-linked and distributed; flexible and extensible to cheaper COTS; 100% compatible and requires no/minimal support and maintenance; capable of achieving a very high level of user satisfaction. In the research problem under investigation, the targeted functionality is performing satisfactorily, and we address the question of how requirements engineering can be used to support porting this functionality to a new product line implementation while satisfying all business and data constraints.

3.3.2. Data Challenges

Another aspect of the research problem is determining how the variations of data in the current system will be represented in ERP. ERP implementations already have their underlying database, which is well-defined for most of the known modules of the target business function(s). Beside interconnection and embedded data operations fulfilling general requirements, almost all implementation projects will end up specifying a handful of customizations that must be

synchronized among the systems implemented. It increases complexity during version control, upgrades, new acquisitions and significant change management operations.

ERP systems contain two sets of data: 1) setup data, and 2) transactional data. Set-up data represents system constants and universal data, e.g., department identifiers or position numbers. Transactional data is the data that gets newly inserted, modified, discarded, batched and accessed on a daily basis for business operations, e.g., employee benefit premium, accounts receivables encumbrance value. Experience indicates that communication alone is insufficient for a precise gap estimate in the data requirements if the representative or actual data is not utilized. Lastly, integrity constraints must be defined to ensure that there is no data loss or corruption, thereby allowing every piece of required information to be accurately transferred to the new system. For instance, if human resources already passed big bang adoption and financials is in transition, to receive a valid ledger for the business we will still need payroll and related information from the new system. Furthermore, the legacy system needs to be simultaneously operating to avoid errors in the budget year.

3.4. ERP Requirements as Bigraph

By mapping existing legacy functionality to ERP features, we can help make a decision upon the degree of customization with a better understanding of the two systems, and focus project efforts on correct construction as opposed to posthoc analysis [39]. Questions that should be asked during modeling the graph representation are: 1) Does the proposed functionality already exist in whole or as a part of the legacy system or is the functionality completely absent? and 2) If it exists as part of the legacy system, can it be extended for what is required and/or suppressed to disable unnecessary or unworthy features? Answering these questions can help to

identify the basic elements and relationships that should be defined in the model. Similar questions can be enumerated for other scenarios such as upgrades.

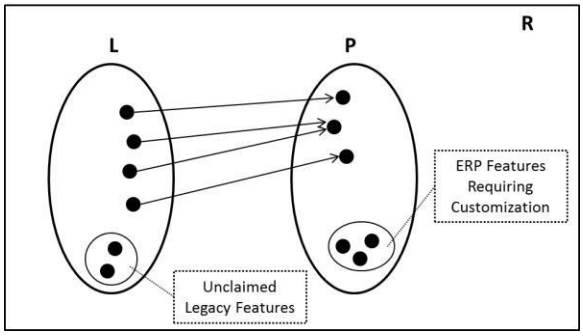

Figure 3.1. Bipartite matching of legacy features to ERP features

Let L represent the set of legacy features, and P be a set of ERP features (disjoint from L) available prior customization. It is possible to construct a bi-graph G = (L P, E) where the elements of L and P make up the vertices of G, and every edge e Є E connects a vertex in L to a vertex in P. Our formalization provides a means of analyzing the similarities and dissimilarities between the functionalities of the two systems (Fig. 3.1). Some elements of L will have related elements in P; others will not. If the relation R describes the mapping between L and P, then L-{l Є L | R(l) ≠ Φ} yields the features that need to be developed for P.

Initially, the relationships between elements of L and those in P can be established at a high-level of abstraction, i.e., mapping high-level features from legacy to ERP. Next, we refine to analyze sub-feature relationships with use-cases. If requirements for both systems have been specified as use cases, we can first identify relationships between each legacy use case and that of ERP. Analysis of relationships between use case flows (discrete interactions) can help decide whether the feature is fully or only partially covered. Once the commonalities are precisely

determined in between the two domains the non-cliquing vertices needs to be addressed which can be termed as non-portable functionalities.

The ideal situation would be to have a balanced bi-graph where $|L| = |P|$. This would mean that the existing features could be completely ported to the new environment without having to perform any customizations. However, in most cases, the client would be requesting more features, and thus we expect to have $|L| > |P|$. It will help the client determine the cost-benefit ratio of the project, and select among a large number of functionalities typically provided as add-on modules. Our problem is now reduced to listing the bi-cliques by associating features with left and right-hand side vertices, respectively, and then enumerating all bi-cliques with all legacy vertices and at least α_m (the minimum set of) right-side vertices. A vertex cover of G is a subset of vertices $C \subseteq V$ such that for each $(u, v) \in E$, $u \in C$ or $v \in C$ holds. The complement set V\C is an independent set in G. The graph obtained by removing a vertex v from G, and all its adjacent edges are denoted as G\{v} [59]. For a decision to suppress feature or reduce the scope for monetary or security grounds, we use the covering with vertex removal.

The exterior covering further allows decomposition of a bipartite graph into a core part and an inadmissible part [60]. The core of the decomposition consists of some irreducible sub-graphs and at most two tails. Each of the tails is the union of some minimal semi-irreducible sub-graphs. The decomposition has no tails if and only if G is an $n \times n$ bipartite graph and has a transversal of order n [58]. The core part can then be further decomposed into irreducible parts to produce a canonical reduction of the graph. Applying exterior covering to the matching problem, we introduce edge labels that associate the relationships from L to P with values. An edge e mapping a legacy feature l to an ERP feature p (denoted $e_{l,p}$) has the value 1 if the functionality of l matches or exceeds p, otherwise $e_{l,p} = 0$. The exterior cover then is defined using a pair of

sets [A, B], where A includes all the features of L, and B includes all the features of P, at any point of operation. In graph theory, [A, B] are referred to as the minimal exterior pair (MEP) [58].

An exterior cover during ERP project initiation can be useful for certain critical decisions which may impact the overall success:

- identifying ambiguity in requirements which can hinder the establishment of co-relationships;
- assessing the degree of similarity or dissimilarity between the two systems;
- evaluating additional properties of the problem using bipartite theories of canonical decomposition,
- help domain analysts to make decision on non-enhancing possible portability features to reduce dynamic evolution of variable artifacts,
- allow exact customization information being available at any point of implementation life cycle.

3.5. Decision on Enhancement

In our discussion let K be a subset of L ×P, representing the Cartesian product of the related functionalities (edges) from each system, corresponding bi-graph representation thus would be of finite exterior dimension. Since the legacy system exists and is functioning, we know that some requirements for it have already been specified. Also, the practical norm of software development is that customers are delivered a contracted number of features per release, and evolution of the product is controlled through versioning. We can, therefore, conclude that K is a matrix (which is also a forest for very large scenarios) where both L and P have a finite or countable number of elements ordered l_1, l_2, l_3,… and p_1, p_2, p_3,… and can be represented $a_{i,j} = 1$

if the pair (l_i, p_j) is an element of K, otherwise $a_{i,j} = 0$. An element in K is said to be inadmissible if it is not an edge of any disjoint sub-graph K*, such that E(K*) = E(K); other-wise the edge is admissible. For our problem, K is of finite exterior dimension; the subset K_c consists of all admissible edges of K, which is referred to as the core. K can be decomposed into 3 regions, R_1, R_2, R_3 by canonical decomposition [60].

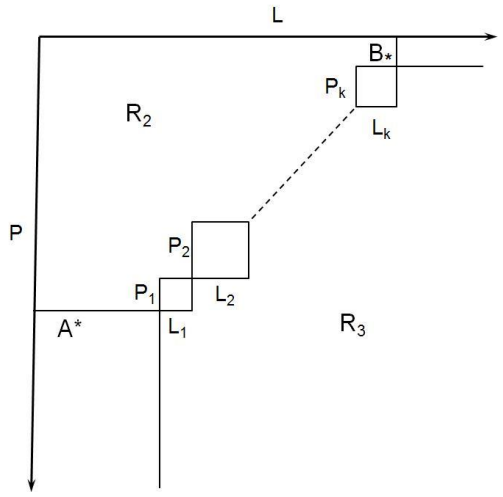

Figure 3.2. Canonical Decomposition of L × P as modified from [60]

In Figure 3.2, R2 is shown in the upper left-hand corner and R3 in the lower right. R1 is the region separating R2 and R3 and will be discussed with cover properties. Alterations to K, and the role played by the core Kc is our tool for identifying mismatching functionalities (inadmissible edges), and help forecast probable success or failure of our project. Following cover properties were identified:

1. If K is altered by the addition or removal of edges from R2 the resulting graph has the same core as does K. Each of R1, R2, and R3 remain unaltered indicating features in legacy that have equivalent features in ERP. A scenario with such a property contributes to a fit. However, all of these features may not be needed by the client so

should be limited to the core requirements. As mentioned earlier removal from R_2 will hold the property G\{v}.

2. If edges in the region R_1 are added to K, the resulting graph will have the same decomposition of L × P as does K, and hence each added element is admissible. The addition of edges from this region will continue to reduce the gap between the two systems, without affecting project parameters such as development, testing, and maintenance. More edges from this region mean increased portability and flexibility to change. In our case study, this corresponds to moving a feature (variable artifact) from application domain to core during product-line evolution. For example, Strategic Sourcing was not a core module before PeopleSoft Financials, and Supply Chain Solutions v9.1 came out and was handled by domain specific customizations as application artifacts with reuse capability.
3. Edges can be removed from K interjection R_1 without decomposing L × P. Some legacy features may have replacements in ERP that are functionally equivalent but are more streamlined and efficient. Such paths within the non-disjoint graph can be eliminated without affecting the graph as a whole [17], [18].
4. If K is altered by adding edges from the region R_3, the resulting graph will have an entirely different decomposition of L × P. This represents a widening of the requirements gap and should be controlled as the project expands with increased client familiarity with the new product. Thus an agreement should be put in place to limit the rate of expansion proportionally to cost, time, and resources. Our case study identifies the bid confirmation number as a custom feature to be added to an already core version of Strategic Sourcing module. As we proceed through our system

development life cycle, we continue to remove edges from R_3 and keep adding more edges in R_1. Some of these features can then become part of the core during the next implementation as a preconfigured portion of the demo solution, and reside safely in R_2.

3.6. Case Study

As ERP implementation consultants for five state projects along with many others, we experienced phases where even a fine-grained piece of functionality can offset the allocated time, resources, and budget of an entire project. To better manage ERP engineering challenges in these projects, methodology toolkits and solution portals for managing the system development life cycle have been developed.

3.6.1. Event Bidding Process

Our experience indicates that one of the primary reasons for failures along with negligence in Critical Success Factors as described in [1], the inability to define a disciplined, quantifiable customization plan with future release features and upcoming patches kept in mind. A formal research method can be the answer to reducing this enormous risk. A case study of an event-bidding, such as undertaken in one of the projects mentioned in Chapter 4, can illustrate the practical application of our formal approach.

Legacy: The state asks bidders to propose a price for a specific purchase event. The legacy system achieves this function via newspapers, circulars, message boards, internet advertisements, and regular mail. Bidders respond with their proposals, which can include multiple files, design, and contracts. This information is mailed to the specified response address, and once received a confirmation receipt is sent to the bidder. The event management office selects a winner and notifies all the bid participants through mail.

The method above has limitations which include: a potential bidder can miss the circulation and fail to meet the deadline; mailed responses may be delayed, lost or damaged; and after an offer is placed, the bidder has little means to modify or improve it as even if a new bid package is sent it may not meet the deadline. The integrated ERP solution of the event-bidding system is described as follows:

ERP: The state asks bidders to propose a price for a specific purchase event. The ERP system creates the event within the integrated system. Suppliers can instantly view events where a bid can be placed. Bids are made online, attaching the necessary documentation and confirmation with a unique code is generated. The event manager logs into the system and can see all the bids placed and information on each of the bidders. Once the event deadline passes, the event manager can analyze the information and select a winner. All the bidders get instant notification of acceptance or rejection.

3.6.2. Analysis

Whereas the legacy system only provided basic messaging and tracking of purchases through data entry, ERP provides a completely integrated online solution with Rich Internet Architecture (RIA), in addition to keeping referential integrity zeroing in the chances of human error. The implementation used is PeopleSoft Enterprise Strategic Sourcing [61], which does not provide a mechanism for generating confirmation numbers. The corresponding ERP functionality, therefore, had to be developed through customization and then was integrated into the system so that event managers and bidders could use it to view confirmed bids. This discrepancy was well-judged and the correctly recognized as a gap during FGA.

In our formal model the bid confirmation number is one of the features that exists in L, but not contained in the ERP feature set P. In the legacy system, this number provides

information on the bidder’s identity, whereas the pre-customized ERP system can only generate incremental integers per bid submission. Bid confirmation number that conglomerates the bidder identity was a missing edge in the bi-graph of the event-bidding problem. An additional edge was then added in the R_3 region of its canonical decomposition, which enforced further testing and enhancement of the system as it changed the decomposition of L × P. The identified gap resulted in major changes to the delivered data repository, underlying record, and code related to the panels, fields, and other components. It also changed the way querying and message call was intended by the original client.

3.6.3. Agile Product-Line Implementation

Early formal representation of matching and exterior covering thus can help analysts decide on the quality of such design and based on the volume; change management can help practitioners choose between options. Furthermore, in other planned changes for related modules or their delivered regular business functions also need consideration. Proper mitigation effort is a necessity to trace back down to all related components and objects in the system. Matured solutions in the market whether ERP or any other packaged system provide a flexible tool for the user-friendly upgrade with backward compatibility. The mainstream reasons of bug-fixes, patches or upgrades can be: version control of database management systems; change in the legislature, business procedures and/or policy; dynamic evolution of a beta released module; and a new version of the implemented solution which is more suited with the current business trend.

As customizations happen in the application layer, change in domain layer of a system can prove that such modifications are now becoming boiler-plate extensions adversely affecting the operation both qualitatively and quantitatively. The efficiency of system and people involved can largely reduce as frustration and unhealthy segmentation among stakeholder's increases out

of this. Classifying customization as leaves of a core, similar to basic product line engineering helps the constraints on market size, window, and penetration, along with time, budget and resources available. The other factor is the delivery date, as most cases, ERP implementations fall short of their project duration due to possible dynamic evolution. Using the model as represented in Fig. 3 we were able to achieve faster elicitation within an agile implementation cycle.

3.6.4. Results

For proprietary reasons, we are going to apply code names for our clients. Projects **N**, **F**, and **FP** will respectively represent a project which used common RE (N), formal FGA and RE as described (F) and formal FGA and RE with Product-Line Engineering as shown in Fig. 3.3 (FP).

In project **N** we had state oriented data for approximately 3300 individuals, and the total duration of RE including FGA before entering the development phase took 15 weeks, overlapping our 12 weeks estimation. In project **F** we first time implemented our model. Although project **F** contained state-oriented information for total 55000 individuals, the FGA process being formalized was ahead of initial timeline projected. The estimated 24 weeks were not necessary before we could enter the development phase, as it was around 21.5 weeks when we started construction. The further benefits achieved along the system development life cycle comparing these two projects can be found in Chapter 4.

Our final project **FP** was a reuse enabled implementation of an upgraded version where we initially developed a methodology tool kit besides configuring the domain artifact of **F** to enable it to be used for **FP**. Above case study was a part of **FP** and as being a financial implementation comparing with **F** provides much-improved results as the entire FGA were downsized to less than a half of the duration required for **F** regardless of the total size which is

closely equivalent in the quantity of data and resource. We need to keep in mind that we used all our prior knowledge and reusable artifacts of **F** in **FP** which provided a quicker response in elicitation, but this streamlining was possible because of the bipartite matching that we started using since implementation in **F**.

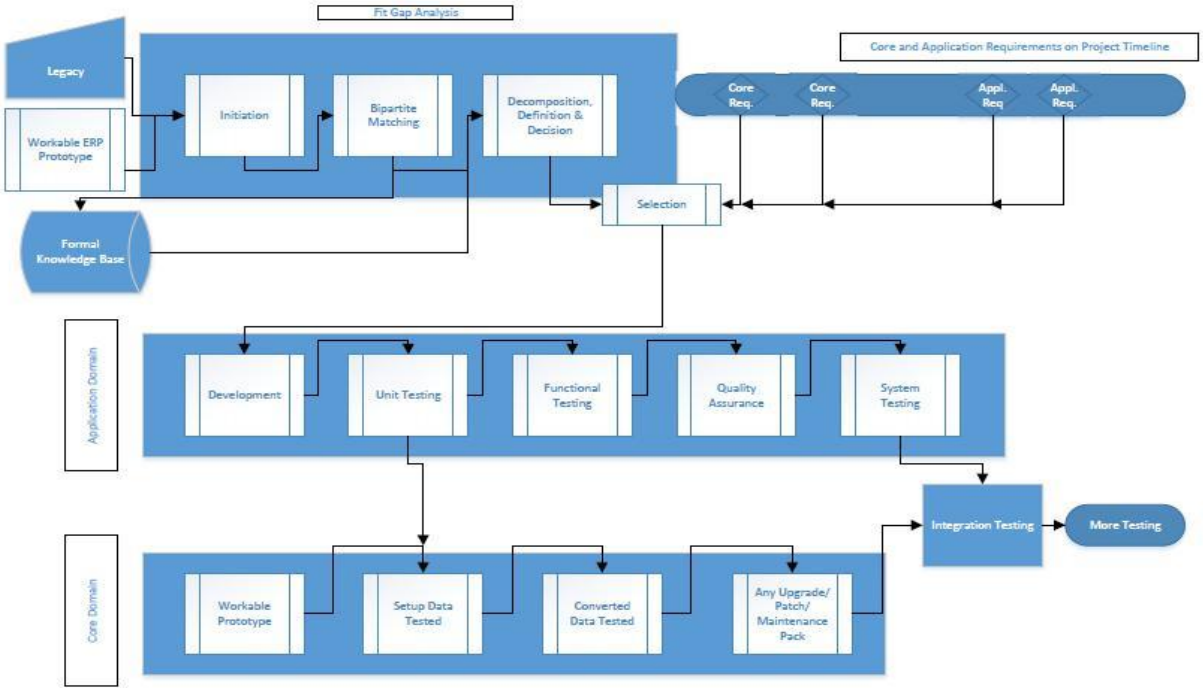

Figure 3.3. Agile product-line engineering with formal FGA in ERP implementation

3.7. Related Work

In our research of using a test-first parallel processing technique for assuring appropriate migration of data from legacy to ERP, we mentioned about data mapping during FGA in Chapter 4. Those mappings were a product of the bi-graph approach discussed in this paper. To the best

of our knowledge, our research is the first seeking to formalize the requirements associated with porting an existing legacy system to ERP.

The work by Illa *et al*. [42] is most closely related to this research. It describes an approach for selecting ERP products using a formal description of their relevant characteristics. The authors extend previous work on the Systematic Help for ERP Acquisitions (SHERPA) methodology [45]. However, the new version of SHERPA is based on a formal language for translating user needs into requirements over the ERP products. In this paper, we do not address the issue of selecting an ERP product from a candidate pool. However, utilizing such a methodology before applying our approach could help identify the best software candidate to be used for matching up functionalities. It would help to minimize the number of customizations needed from the beginning, thereby reducing the problem complexity and improving the overall approach.

Soffer *et al*. [46] describe the process of modeling ERP system. They consider a whole range of process alternatives supported by the ERP system and the interdependencies between them. In addition to defining a comprehensive set of generic modeling steps for ERP, they provide desirable characteristics of ERP modeling languages. While they take a more general approach to ERP modeling, we focus in on formal modeling of the requirements involved in migrating from legacy to ERP.

3.8. Conclusion

Our research work provided several benefits in various facets of the ERP software engineering process. Though implementation scope grows exponentially, putting all of the domains involved in one single bi-graph definition was not possible, so we used a collection of sub-graphs defined as part of a forest [59]. A bipartite graph G can be factored as a product $G \sim =$

$H \times K_2$ of a graph H with the corresponding complete graph K_2 similar to factoring an even integer g into a product $g = h \times 2$, except that for graphs the factorization need not be unique [62]. This trade-off was handy in running iterative enhancement processes.

Future work involves developing a tool able to define multiple domains, input in the features formally, and then generate the intended cover property of customizations. Any views, opinions, findings, and recommendations presented in this paper are those of the authors and do not necessarily reflect those of the Ultimate Software Group, Inc.

CHAPTER 4. TEST-FIRST AND PARALLEL PROCESSING TECHNIQUES IN ERP[2]

4.1. Introduction

Enterprise Resource Planning (ERP) continues to experience strong market growth, currently standing at approximately $22 billion and projected to reach $30 billion within the next year [49]. ERP systems are highly complex, distributed systems that are almost always customized to meet the specific requirements of an organization. Also, ERP often requires the migration of functionality and data from multiple legacy or mainframe-oriented systems into a single integrated solution [10].

Developing an ERP system typically involves porting legacy data from accounting, human capital management, customer relationship management, services, sales and other divisions to a common ERP database, and building applications that access and update the central data store. The primary goal of ERP implementation is to standardize and streamline the business functions and information flow among these various divisions of the organization [10, 11]. ERP has, therefore, become a prime contender of software implementation for large organizations such as state governments, multi-site and multinational corporations. Two of the engineering challenges associated with ERP system development include mapping and

[2] The material in this chapter was co-authored by Talukdar Asgar, Mohammed Akour and Dr. Tariq King (during his tenure in NDSU). Talukdar had primary responsibility for collecting samples in the field and for interviewing users of the test system. Talukdar was the primary developer of the conclusions that are advanced here. Talukdar also drafted and revised all versions of this chapter. Mohammed and Dr. King served as proofreader and checked the math in the statistical analysis conducted by Talukdar.

conversion of data from different divisions into a common database [11]. The importance of properly migrating information from existing legacy systems to ERP software modules is, therefore, paramount to the success of a project. However, the large size of system and volume of data makes this an extremely daunting task. Even the data from a single division can be quite difficult to manage and handle correctly due to complex relationships.

Software testing continues to play a major role in ensuring that software systems meet the needs of customers [10]. Given the complexity of ERP software development problem, and its tremendous impact on the industry, one would expect that there would be a lot of research devoted to testing ERP systems. However, this is not the case as only a few works can be found in the academic literature [11]. Furthermore, although software vendors have their methodologies for testing ERP systems, there has been little collaboration and dissemination of knowledge concerning effective software testing strategies for ERP systems. In this chapter, we discuss and resolve the problem of Legacy-to-ERP data migration using a test-first approach, followed by parallelized execution of instances of the data conversion program. Our approach seeks to identify and correct data mapping errors and anomalies before running the conversion process, thereby saving time and effort associated with failed conversions. Parallelization has been incorporated in the approach to reduce further time delays that can occur due to a lengthy conversion process. The major contributions of this work are as follows:

- Presents a test-first approach to ERP data conversion, including a method for incorporating the approach into commonly used ERP implementation activities.
- Elaborates on the results, experiences, and lessons learned from applying the approach to an industrial case study.

The rest of this paper is organized as follows: the next section contains background material. Section 4.3 defines the research problem under investigation. Section 4.4 presents the approach and detailed steps of our method. Section 4.5 describes the case study. Section 4.6 is the related work, and in Section 4.7 we conclude and outline future directions.

4.2. Background

In this section, we provide background material on enterprise resource planning and test-first approaches.

4.2.1. Enterprise Resource Planning

Enterprise Resource Planning (ERP) is defined as an information system that manages, through integration, all aspects of business including production planning, purchasing, manufacturing, sales, distribution and accounting [63]. These software platforms combine several aspects of the company, allowing operations to be performed more cohesively [55]. A standard ERP development process consists of the following phases [55]: a fit and gap analysis to identify business processes and customization requirements; system design and development; data conversions and migration from the legacy system; testing of the new system; training of the super-users and end users; and deployment. Examples of ERP software packages include SAP [56], and PeopleSoft [64], both of which target seamless integration of information flowing through various divisions of an organization.

4.2.2. Test First Approaches

In a general sense, a test-first approach is any strategy that seeks to evaluate a system, component or activity before its development and/or execution. Test-first methods have been applied to a variety of hardware and software problems with much success [65, 66]. As pertains to software systems, a smoke test is a preliminary test that seeks to determine if a program works

as expected at the basic level. Smoke testing is intended to reveal failures that are simple yet severe enough reject a software release or avoid wasting time doing further testing [66]. In this paper, we apply the concept of smoke testing in the context of a software process activity, i.e., performing preliminary tests to determine whether or not we should move on to a subsequent process activity.

4.3. Research Problem

The research problem being addressed in this paper investigates the mapping and conversion of data from a legacy system to an ERP implementation. Figure 4.1 provides an activity diagram of a typical approach used in industry to convert legacy data to ERP. As a preliminary step to data conversion, business analysts and technical specialists, from the client and vendor sides, meet to conduct the data mapping activity. During the data mapping activity, field-by-field relationships between multiple tables in the databases are established. Such relationships indicate how legacy data map to ERP modules. The output of this activity is an informal (or semi-formal) document specifying the data mappings between legacy and ERP. With the data mapping document finalized, the client and vendor formalize its representation so that the data conversion process can be fully automated. This involves a client-side activity and a vendor-side activity, which can be performed in parallel once information exchange formats have been agreed upon. On the client-side, technical specialists create a specification, typically in the form of a flat file, containing a list of the source fields from the legacy system. Meanwhile, the vendor designs a component interface to facilitate migration of data to the ERP module. The component interface should represent the complete set of ERP database fields targeted in the migration. Once these formal representations have been created, developers from the vendor-side design and write a program that reads the field specification file, and directly inserts data into the

ERP module via its component interface. In the next activity, the data conversion program is executed by technical specialists on the vendor-side. If errors occur during the automated conversion process, the program throws and logs these errors as runtime exceptions. After the program completes, the vendor scans the log file and notifies the client-side business analysts that conversion was unsuccessful. Meetings between the two parties are then scheduled to fix any data anomalies that caused the conversion failures. This event sequence of running the program, checking the log, and holding meetings to address conversion issues is repeated until all the required data has been ported to ERP. Acceptance testing of the converted data is then performed. The major drawbacks of the aforementioned approach to ERP data conversion are twofold:

1. lack of formalized testing of legacy data before using it as input to the program means that data anomalies are not discovered until relatively late in the conversion process; and
2. development of the module is kept at a standstill until the conversion program, which needs to run to completion multiple times, produces no errors or exceptions.

These drawbacks can translate into the project being delayed or going over budget. Our research investigation is specifically targeted at alleviating some of the problems associated with these two drawbacks, which are commonly experienced in the industry.

4.4. Approach

To address the major challenges associated with the research problem, we propose a test-first approach to ERP data conversion. It is accepted in software engineering practice that testing in the preliminary stages of development helps to avoid costly time delays and failed projects, through early error detection and bug removal. To the best of our knowledge, there are no

approaches or experience reports in the literature related to test-first or test-driven the development of ERP systems. Furthermore, to further reduce the overall time spent conducting data conversion activities, our approach coordinates multiple instances of the conversion program and generates an error report that makes debugging conversion problems easier. This section describes our approach, which includes a set of proposed enhancements to traditional ERP data conversion approaches. As part of our description, we provide details on how the new approach can be realized using ERP technologies.

4.4.1. Overview

Figure 4.3 provides an overview of our proposed approach to ERP data conversion. The shaded nodes correspond to new activities and decisions that have been incorporated to enhance the traditional approach to ERP data conversion. These enhancements, as relates to the labels in Figure 4.3, are as follows:

1. development of infrastructure to support data staging, including tracking the acceptance status of fields to be migrated by the conversion program;
2. the performance of smoke testing to detect data mapping problems early;
3. process scheduling of multiple instances of the data conversion program to reduce the overall conversion time, and
4. generation of an error report that conveniently summarizes any problems encountered during conversion.

4.4.2. Data Staging

To facilitate a test-first approach to ERP data conversion, we propose the introduction of a data staging phase. The idea is to develop a small infrastructure that would temporarily hold records from the legacy database for processing, and allow the data to be tested before inputting

it to the ERP module. Having mechanisms for record staging would facilitate querying specific sets of data so that they could be inspected to identify missing links and anomalies before actual conversion takes place. It would require that a staging table is created by developers on the vendor-side.

The staging table should include a status field for tracking acceptance or rejection of the legacy data. This field would be used later by the data conversion program to determine whether or not specific rows of data should be included in the conversion process. In other words, the conversion program will only process rows if they have been accepted after a smoke test.

4.4.3. Smoke Testing

Smoke testing provides an opportunity for the business analysts to verify whether the mapping fields and legacy data are correct, complete, and consistent. During this phase, the client and the vendor work together to identify data mapping errors and anomalies through systematic querying of the legacy database. If problems are detected with specific sets of data, their status field values are set to Rejected. After the smoke tests, the team may decide to fix the problems identified and change the corresponding status field values or defer corrections until later. Testing at this stage also helps developers to better understand the functionality required by the conversion program and ERP module.

4.4.4. Process Scheduling

Many ERP software packages contain process scheduling features for running batch applications. Multiple schedulers are usually supported, with each scheduler being capable of running a finite number of processes concurrently. Under our approach, ERP process scheduling under different operating environments is harnessed for concurrent execution of multiple instances the conversion program. It must be noted that each program instance would need to

have access to temporary table definitions for staging data. Furthermore, the total number of program instances that are run in parallel should be considered together with the power of the underlying hardware.

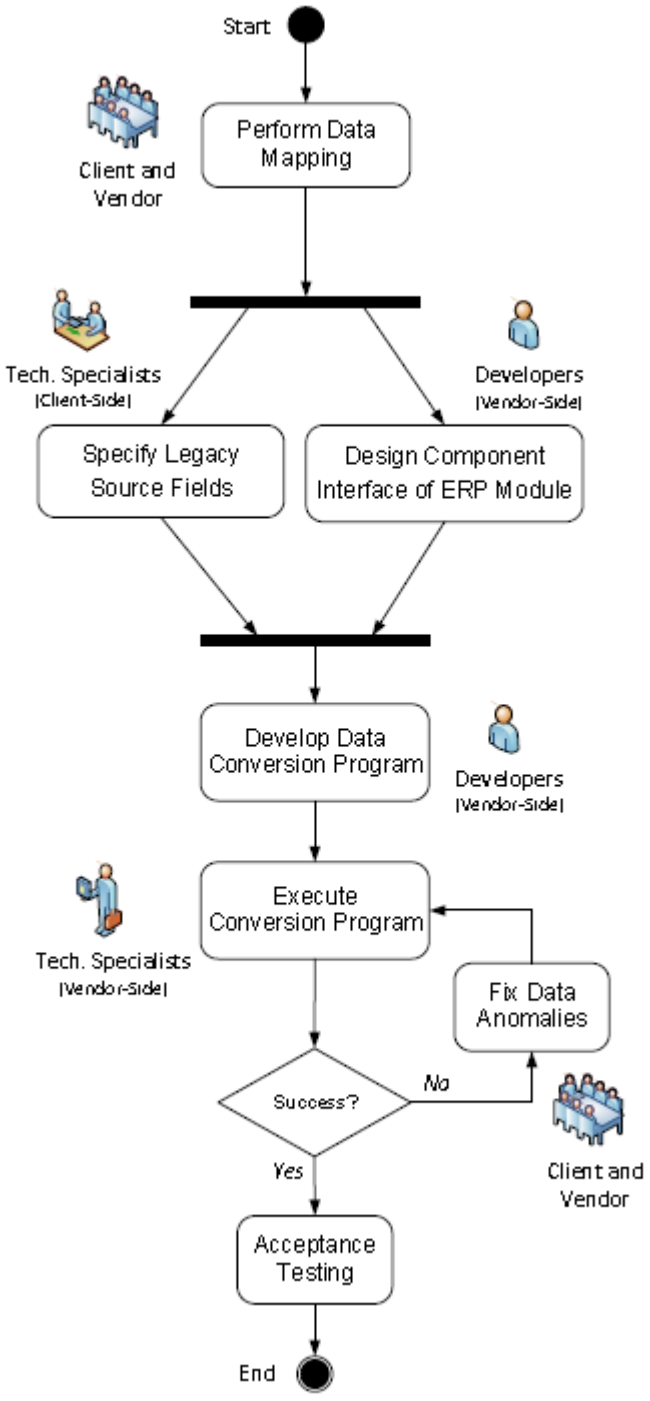

Figure 4.1. Common Approach to ERP Data Conversion

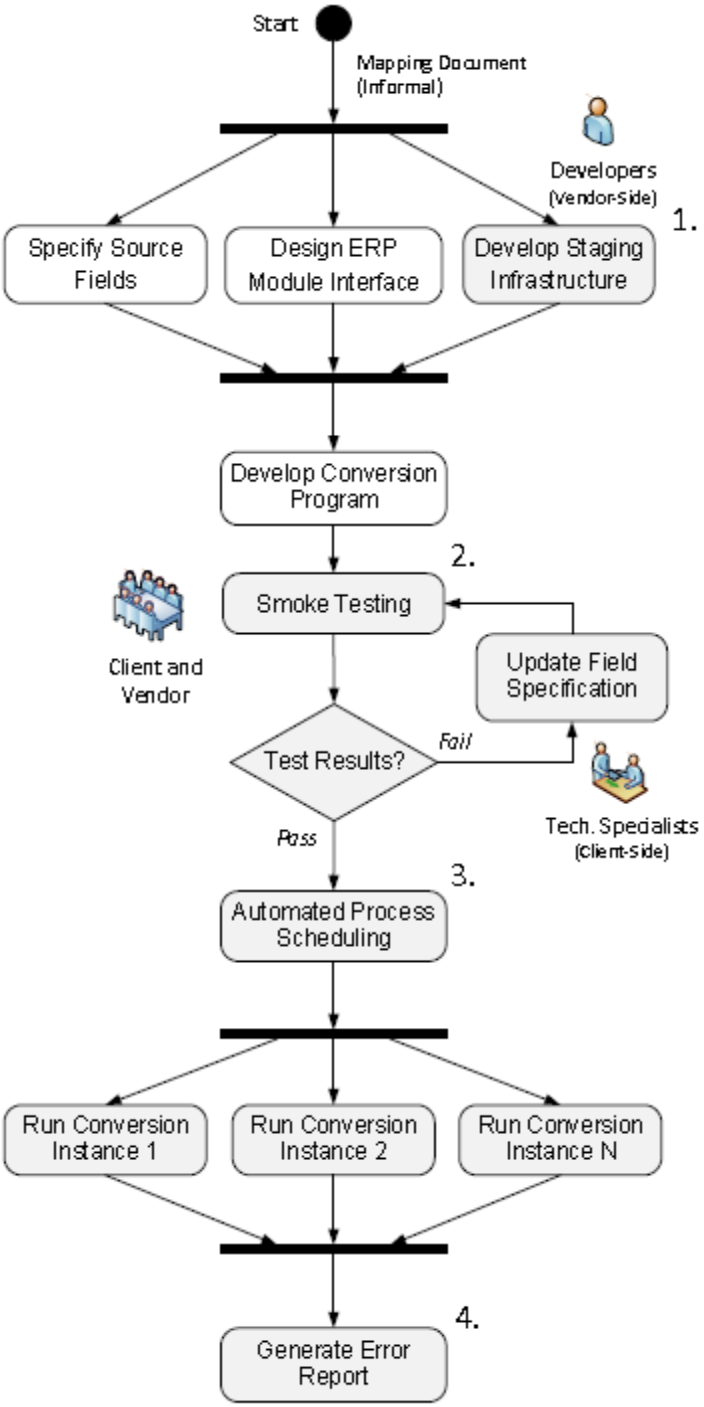

Figure 4.2. A test-first parallelized approach to ERP Data Conversion

4.4.5. Error Reporting

Another status field can be included in the staging record to indicate whether or not a row of data migrated to ERP without problems. It will improve the post-conversion error reporting capabilities further. For each row that failed to migrate to the ERP module, the corresponding log message that describes the failure can also be included as a field in the staging table. This would provide the team with quick and easy access to conversion failure information, rather than having to scan through separate log files (i.e., one for each instance of the conversion program).

4.5. Case Study

To evaluate the usefulness of our approach in practice, we conducted an industrial case study which included two ERP software projects. Both projects involved the conversion of data from a legacy system to a unified ERP solution. In this section, we describe the two applications used in the study, present our findings, and discuss our experiences from conducting the study.

4.5.1. Project Description

The applications used in the study were in the domain of Human Capital Management (HCM). The legacy systems from both projects, therefore, contained a wealth of employee data including general information, labor history, benefits, time, tax records, among others. However, for the study, we only focused on migrating general employee information. Successful conversion of general employee information is typically a mid-level requirement to initiating dependent conversion processes (such as labor, benefits, etc.), thereby emphasizing the need for quick and accurate migration. In both cases, the ERP software package used was PeopleSoft HCM 9.0 [64].

For proprietary reasons, we refer to the two projects as HCM-Common and HCM-Enhanced. In the HCM-Common project, one conversion program instance was created and run

in a single Windows process scheduler, and read a text file containing the legacy field specifications line-by-line. Approximately 3,000 records were set to be converted to ERP by inserting them directly into the Job component of PeopleSoft. Neither record staging nor smoke testing was performed, and all errors were stored in the program log file. This project was, therefore, representative of the approach commonly used in industry to tackle legacy to ERP data conversion.

As per the approach presented in this paper, we incorporated test-first and parallel processing techniques into the HCM-Enhanced project. The conversion program was implemented with data staging to facilitate smoke testing. The entire legacy field specification file was read once to initialize the staging infrastructure, and smoke tests were administered on a set of 55,000 records using SQL query executions and peer review. Whenever failures were produced during smoke testing, these faults were resolved by updating the source field specifications or fixing anomalies in the legacy data before conversion. A total of 60 conversion programs were instantiated in parallel and were equally distributed across 4 UNIX process schedulers (i.e., 15 instances per scheduler).

4.5.2. Results

Table 4.1 provides a summary of the results recorded during the case study. One measure used to compare the two approaches was the number of project timeline stretches that were experienced due to having a lengthy conversion process. For HCMCommon there were multiple timeline stretches (more than 10), in contrast with HCM-Enhanced which had no timeline stretches. It took approximately 72 hours to convert the 3,000 records in HCM-Common, while the 55,000 records in HCM-Enhanced were converted in less than 12 hours. In fact, the entire conversion hierarchy of HCM-Enhanced was processed within 1 week.

Table 4.1. Results of ERP Data Conversion in two HCM Projects

Project Name	# Records	Conversion Time (Hours)	# Timeline Stretches
HCM-Common	3000	72	>10
HCM-Enhanced	55000	12	0

A comparison of the relative effort of performing iterations of Convert-Fix-Reconvert in HCM-Common, as opposed to Test-Fix-Retest in HCM-Enhanced, revealed significantly lower costs at the staging-level. This was measured in terms of time spent tracking down the source of conversion failures and fixing them. It must also be noted that since HCM-Common was time-stretched multiple times, only one iteration of acceptance testing was performed to avoid further delays. Our testing experiences have been included as part of the discussion section.

4.5.3. Discussion

The main purpose of conducting the case study was to provide preliminary evidence that the approach presented in this paper is feasible and beneficial in practice. Feedback from applying the test-first approach using staging records was positive. Spending time identifying and fixing data anomalies before conversion proved to be more efficient than iterating through multiple failed data conversions, due to the lengthiness of the actual conversion process. Not having timeline stretches early in the project meant that we were well ahead of time when it came to integration and user acceptance testing.

Incorporating post-conversion error reporting mechanisms into the staging infrastructure was also instrumental in the success of the project. The team was able to access all of such errors that occurred during conversion quickly, and then query the database to start analyzing possible root causes of those mistakes. The previous method of having to search through the program log

file for thrown exceptions, and then go back to the database tended to break the rhythm of focusing on the data.

Although our use of parallel conversion processes yielded notable time reductions, the results were not as promising as we had anticipated. It was attributed to performance lag due to a strain on the hardware resources. Our plan was to use the maximum 20 instances per scheduler that are allowed by the PeopleSoft ERP software package. However, after performing initial benchmarks, we opted to use 15 per scheduler since throughput was better using fewer instances per scheduler. It revealed the importance of tweaking these parameters and conducting performance benchmarks before running the conversion programs.

Our case study has several limitations and possible threats to validity. Firstly, since we are at the preliminary stages of this research, the goal was to investigate the feasibility of the proposed ideas. Proper evaluation of the approach requires controlled experimentation using appropriate metrics for comparison, treating each enhancement in isolation. For example, the results as presented do not allow us to differentiate between time savings from gained from parallelism vs. early error detection and removal. Furthermore, a formal evaluation of applying the test-first approach for quality improvements is needed. Threats to validity include its small size (i.e., only two projects from a single domain), and the lack of similar studies in the literature for comparison of results. However, we have already begun applying our approach to Financial and Supply Chain Management projects with favorable results.

4.6. Related Research

In general, there has been little research on testing ERP systems, and the topic continues to be a highly neglected in the literature. To the best of our knowledge, the approach presented in this paper is the first to describe how a test-first method can be applied to ERP data conversion.

However, several researchers have recognized the need for developing new testing methods, tools, and data-driven testing techniques for ERP systems [10, 11, 48, 67]. We now summarize related work found in the research literature.

Gerrard [11] reports lessons learned in ERP projects that indicate that failures could be reduced significantly by applying benefits-, risk-, and coverage-based testing. Their report acknowledges that most of the functionality in ERP systems is master-data driven, and classifies data migration to be a highly challenging aspect of ERP projects. Gerrard [11] emphasizes that existing migration approaches "need evaluation, reconciliation, and testing using the integrated application." Our approach seeks to address this need through the description of how test-first development and parallel processing can be used to improve the ERP data migration process. Wieczorek et al. [10] describe the special characteristics and challenges associated with test data provision for ERP systems. They describe two general views of ERP data: (1) Business View – master data that represents long-term use case scenarios, and transactional data that is short-lived due to daily operations; and (2) Technical View – system data for the applications internal data set, and input data that will be provided by outside users and external components. Their focus is on the technical ERP data view and describes properties of system data and input data to support black-box testing. Previously, Wieczorek and Stefanescu [48] described the state of practice in ERP testing as motivation for their research activities in this area.

Schliesser [51] provides an approach to tackling the ERP testing problem using Service-Oriented Architecture (SOA). Their approach focuses on improving automated testing techniques for ERP and handling the complexity aspects of ERP testing. Schliesser [51] describes an architecture that uses SOA components to enhance ERP testing, which claims benefits such as: better utilization of business matter experts; quick react to change; and

qualitative and improved automated testing. Similar to the approach presented in this paper, they present their approach as an enhancement to the existing ERP testing processes used in industry today.

4.7. Conclusion and Future Work

This paper described how test-first and parallel processing techniques could be integrated into the current state-of-practice in ERP data conversion. Our approach introduced data staging, smoke testing, and parallelized conversion activities into the workflow of ERP system development. An industrial case study that applied the approach in the context of Human Capital Management was also presented. The results of the study, although limited due to its small size, provided some evidence that the proposed approach is feasible and beneficial in practice. Future work calls for a more comprehensive case study that uses controlled experimentation to evaluate the pros and cons of the approach. We also plan to investigate the migration of legacy functions to ERP and move towards test-driven development of ERP systems.

4.8. Acknowledgment

The authors would like to acknowledge Iyad Alazzam for his contribution to this work. We would also like to thank our industry partners for their cooperation on this research project. This work was supported in part by the National Institutes of Health (NIH) under grant 2R44RR024779-02A1. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NIH.

CHAPTER 5. DETAIL DESCRIPTION OF RESEARCH STEPS

In this chapter I describe the process which I followed for this research work in detail. Some of the discussions are already available in Chapter 3 and Chapter 4. This chapter provides a stepwise description of my research with additional information.

5.1. Bipartite Graph

My research starts with defining the bipartite graph and matching it with our two domains,

1. Source Domain: Legacy
2. Target Domain: PeopleSoft Demo Environment

Section 1.5.1.1 describes in detail the process we go through for fit/gap analysis. Drawing a bipartite graph is the initial step of doing a formalized grouping of these requirements.

Let L represent the set of legacy features, and P be a set of ERP features (disjoint from L) available before customization. It is possible to construct a bigraph

G = (L, P, E)

where the elements of L and P make up the vertices of G, and every edge

e € E

connects a vertex in L to a vertex in P.

Figure 3.1 is a simple bigraph. The interconnection between a point in L and a point in P is drawn as an edge leading from L to P. There may be multiple points in L that will end up to one point in P. There can be points in both L and P which may not have any interconnection. So we have,

1. A one-to-one mapping: One Feature in L directly available as a feature in P

2. A many-to-one mapping: Multiple features in L available as a single feature in P
3. No edge from L: Legacy features those are not available in P
4. No edge to P: ERP features that did not exist in L to match

The ideal situation would be to have a balanced bigraph where

$$|L| = |P|$$

However, in most cases, the client would be requesting more features and thus we expect to have

$$|L| > |P|$$

Let D be a subgraph of G and let D have at least one edge. The subgraph D has a unique subgraph D' which has the same edges as D and is such that every vertex of D' has at least one edge. This sub- graph D' is called the essential subgraph of D. For which defines everything we have mapped by 1 and 2 between L and P in G.

If the relation R describes the mapping between L and P, then,

$$L - \{l \in L \mid R(l) \neq \emptyset\}$$

represents the features that need development for P through customization. Initially, the relationships between elements of L and those in P are established at a high-level of abstraction and then get refined to analyze sub-feature relationships. After specification of requirements for both systems as use cases, we first identified relationships between each legacy use case with corresponding ERP use case. We then analyzed the relationships between use case flows (discrete interactions) to determine if the feature is wholly or only partially covered.

5.2. Exterior Cover

Based on above discussion on Bigraph next, I define exterior cover for these two domains. A vertex cover of G is a subset of vertices

$$C \subseteq V$$

such that for each

$$(u, v) \in E,$$

$$u \in C \text{ or } v \in C \text{ holds.}$$

The complement set V \C is an independent set in G. The graph obtained by removing a vertex v from G and all its adjacent edges is denoted as

$$G\backslash\{v\}.$$

Exterior covering provides a natural way to decompose bipartite graphs. It allows decomposition of a bipartite graph into a core part and an inadmissible part which eventually helps to differentiate core requirements and application requirements. There is no hard and fast rule to define the scope of core requirements, and it is an agreement between the vendor and client to come up to where they will draw the line between, a) things system should have and b) things that would be nice to have. The core part is then further decomposed into irreducible parts to produce a canonical reduction of the graph. Edge labels that associate the relationships from L to P are introduced with values 1 or 0. An edge *e* mapping a legacy feature *l* to an ERP feature *p* denoted

$$e_{l,p}$$

has the value 1 if the functionality of *l* matches or exceeds *p*, otherwise

$$e_{l,p} = 0.$$

The exterior cover can then be defined using a pair of sets

$$[A, B]$$

where

A includes all the features of L, and

B all the features of P,

at any point of operation. In bipartite graph theory,

[A, B] are referred to as the minimal exterior pair (MEP)

Finding out MEP helped us identify ambiguity in a specification and assess similarity and dissimilarity between legacy and ERP.

5.3. Canonical Decomposition

Let K be a subset of

$$L \times P$$

the Cartesian product of the related functionalities. It is of finite exterior dimension. Since the legacy system exists and is functioning, we know that many requirements for it have already been specified. Customers are delivered a contracted number of features per release, and evolution of the product is controlled through versioning.

We conclude that K is a matrix where both L and P have a finite or countable number of elements ordered

$$l_1, l_2, l_3, \ldots \text{ and } p_1, p_2, p_3, \ldots$$

and can be represented

$$a_{i,j} = 1$$

if the pair (l_i, p_j) is an element of K, otherwise

$$a_{i,j} = 0.$$

An element in K is said to be inadmissible if it is not an edge of any disjoint subgraph K, such that,

$$E(K^*) = E(K)$$

otherwise, the edge is admissible.

Since K is of finite exterior dimension, the subset K_c consists of all admissible edges of K, which is our core. K has elements (*l,p*) which are edges. A_*, A, A_i, L_i, A* are subsets of L and B_*, B, B_i, P_i, B* are subsets of P.

$A_i \cup A_j$, $A_i \cap A_j$, and $A`_i$ represent union, intersection and complement with respect to A. Similarly they are defined with respect to B.

Theory of minimal exterior pair defines,

A_* is a proper subset of A, or null

A is a proper subset of A*

B_* is a proper subset of B, or null

B is a proper subset of B*

Canonical decomposition of K gives us,

$L_i \cap A_* = \text{null}$ for $i = 1, 2, \ldots, k$

$A_i \cap A_j = \text{null}$ for all $i, j, i \neq j$

$P_i \cap B_* = \text{null}$ for $i = 1, 2, \ldots, k$

$B_i \cap B_j = \text{null}$ for all $i, j, i \neq j$

K can be decomposed into 3 regions, R_1, R_2 and R_3 from above enumerations.

Let,

$$R_1 = (A_* \times B`^*) \cup (L_1 \times P_1) \cup (L_2 \times P_2) \ldots\ldots \cup (L_k \times P_k) \cup (A`^* \times B_*)$$

$$R_2 = (A_* \times B^*) \cup (A^* \times B_*) \cup (L_i \times P_j) \text{ when } i < j$$

$$R_3 = (A`_* \times B`^*) \cup (A`^* \times B`_*) \cup (L_i \times P_j) \text{ when } i > j$$

In Figure 3.2, R_2 is shown in the upper left-hand corner, and R_3 in the lower right. R_1 is the region separating R_2 and R_3 and

$$R_1 \times R_2 \times R_3 = L \times P$$

These regions are shown in Figure 3.2. The regions represent,

R_1 – region which corresponds to all straightforward matching (Fits)

R_2 – region which shows features that have improved features in the new domain.

R_3 – region which shows mismatches that require custom development (Gaps)

5.4. Cover Properties

After I had grouped the requirements based on regions obtained through canonical decomposition of a bigraph, I analyzed the following cover properties for our implementation.

1. An edge added to R_2 – No change in core, additional feature in ERP, something nice to have. During implementation projects several times it happened that a newer version of a module was released and available for use. In one project somewhere in development phase decision was made to apply those new features in the existing test environment.
2. An edge added to R_1 – That means we have found another fit, admissible to core.
3. An edge removed from R_1 – Same as adding to R_1. In this case, it reduces the scope of must have requirements. (rare scenario)
4. An edge added in R_3 – This is a gap and we may need customization.(very common)

5.5. Examples

Here are two examples of customization:

1. One example in strategic sourcing was where submission of bid by a state vendor would be to provide the bidder a **bid confirmation number**. The functionality in ERP was completely based on online transaction eliminating any paper mailing and data entry by another operator (in ERP the vendor does their own entry through

bidding portal). The only limitation was it did not generate a confirmation number which was a tracking feature in the existing legacy domain.

2. Another example was **approval process** for a Purchase Order (PO). The legacy approval process required three level approvals, at the line level it could be one of the team members who could approve the PO and forward the approved paper document to the next level. Existing approval process in ERP did not have this required feature - allowing one of the team members to approve through online, it was defaulting to one person and only residing under that person's queue to get approved. It was not able to send notification and let other team members learn about that queued PO, neither had they had the ability to approve and push it forward. The customization asked for putting the PO into all the team members queue for approval and get pushed on a first come first serve basis if one of them approved it. Similarly, if someone disapproved proper customization had to be in place to push it backward.

There were many more customization examples for region R_3.

5.6. Data Mapping (Technical Specification)

In Chapter 1.3 I discussed the process of developing Technical Specification. A Technical Specification Document generally consists of following sections,

1. Front page (the title and hierarchically sign-off areas for designated project personnel)
2. Problem Introduction
3. Functional Analysis
4. Functional Test Plan
5. Technical Analysis

6. Design Detail
7. Unit Test Plan
8. Hours Estimate
9. Dependencies
10. Special Needs (Hardware, Software, Files)

For each required conversion program, we produced a technical specification. The technical specification contained along with above sections, previously completed necessary data mapping for the conversion. Domain anlysts and vendor consultants need to sit together and finalize the mapping beforehand. Based on this documentation the developer can decide what objects needed to be devleoped during each stage of the conversion process.

5.7. Conversion Process

Figure 4.1 present the conversion steps which were in practice in the industry. We were following this model in projects before we introduced our test-first parallel processing model for data conversion.

1. Through fit-gap analysis and data mapping, client and vendor decide which data fields for a particular ERP component will be generated row-by-row from legacy.
2. Right after finalizing this mapping the legacy expert then produced the source file for the component. A component can cover one particular business function within a module. For example, Health Benefits Plan, Flexible Spending Accounts, COBRA Retiree Plan, etc. each is a component within Benefits module of Human Capital Management system.
3. ERP developers in my team then developed the corresponding **component interface** for the component and generated the delivered code from the component interface in

Application Designer. Application Designer is the Integrated Development Tool for PeopleSoft ERP systems. The proprietary language for programming in Application Designer is PeopleCode. The batch processes developed using SQL and PeopleCode inside Application Designer are called Application Engine programs.

4. Next, we tweaked and modified this code for reading the file so it can populate desired fields in the component hierarchically using source data directly from the file. The component can have multiple tables at different levels. Obviously, a higher level table has all the keys of a lower level table and few additional keys for preserving consistency and integrity. One source file is used to populate all necessary fields of all the tables in a component. Some fields if not key and not needed can be kept empty or can contain default values if null values are not accepted.
5. The conversion program developed thus did following things,
    a. Opened up the source file which resides on a shared network location
    b. Loaded and initiated the Component Interface to populate the underlying Component
    c. Reads the first row in the file (the file is usually Tab delimited, and also length of fields are maintained compatible with ERP table field lengths so it can be consistent)
    d. Starts populating the fields using the application engine code hierarchically to all component field variable assignments.
    e. Commits the inserted data and then moves the file pointer to the next row.
6. While doing that if there is a unique constraint (because if same key values were repeated in the file), file read error for unknown characters or data value sent did not

match the field type (a date field format is incorrect or a string is found for a numeric field) or any other issue, the application engine program stops with an error.

7. We can recognize by querying the component tables how much data was already processed and looking at error log can find out very little information about what is causing the error.
8. Most commonly the file is requested again with the error information to the legacy expert and due to risk involved we would clear out the tables to start afresh once a new file is developed.
9. This model was very time consuming and left no opportunity for testing while running the conversion. Whole conversion process along with testing was set for 3 months in a project for 3000 employees and we had to stretch that planned timeline.
10. It also requires a constant monitoring as we are unable to leave it running overnight because of the inflexibility in error handling. If there was an error then we cannot gain any benefit running it off-hours which was a common scenario.

Other significant aspect is we cannot convert at will, we have to follow certain order of conversion module by module, for example, before conversion of benefits module, employee job information needs to be converted which could be done only after personal information conversion and some other configuration data conversions such as Job Code, Position Number etc. In my next project the volume of data increased from total of 3000 employees to 55000 employees. For 55000 employees now health benefit data was 1100000+ rows. We cannot follow the above procedure to make this conversion a success. We needed a rapid, robust model for quicker results but also with better testability, less defect rate and dynamic error reporting solution. So we incorporated few techniques to improve above model. Figure 4.2 shows the

improved process flow. Following are the improvements which we applied to the existing model. We were able to gain a much better output from our effort.

5.8. Decoupling file I/O

The old model of data conversion consisted of reading the file directly inside the program. Any file I/O error or any other data anomaly caused the process to stop and start all over with rolling back the database. Also, the error report gave us little information about the exact reason for the error. My enhanced model decoupled the file reading and actual conversion into two step process by the introduction of intermediate staging table.

5.9. Staging Table

Staging table is a custom database table in PeopleSoft which I introduced for reading the file data into the database before actual conversion process happens. So, in our new model instead of reading from the file and relying on the file pointer to control the process to proceed, the process now can read data from this table, which gave us more flexibility to presort the data or mark a subset for processing while ignoring other sets for testing purposes. The flexibility of querying source data before processing or sorting in a desired order could not be achieved in a file. A staging table contains the fields with actual data type that should be coming from legacy source in each row exactly by design. This design was final during the data mapping activity. Now if there was any I/O error or the file was corrupt we could inform the client before even actual conversion for feeding the data into component started. Reading the file into the staging table was achieved by a file reading class written using PeopleCode API with a file layout definition. This application engine process needed two parameters, the corresponding staging table name and the source file name. Since the reading a file into a table was much faster,

recognizing the file problems early help mitigate any loss of time during actual conversion which was common earlier due to file related errors or I/O errors.

5.10. Smoke Testing

Having data in the staging table before conversion processing gave us the opportunity to apply a test-first model. A smoke test is a basic test which does a sanity check to quickly evaluate whether a claim or the result of a calculation can possibly be true. It helps us rule out certain types of obviously false results. We applied delivered table edit feature in PeopleSoft Application Designer through PeopleCode to do smoke testing on the legacy data to validate their integrity. As mentioned in section 5.9 above, the expected data type was strictly applied in our staging table. Then we created a separate batch program to run the following method prior running conversion.

The *ExecuteEdits* method executed the standard system edits on every field in the record. The types of edits performed depend on the *edit level*. If no *edit level* is specified, it executes all system edits. All *edit levels* are already defined either for the record definition or for the field definition, that is:

- Reasonable Date Range (Is the date contained within the specified reasonable date range?)
- 1/0 (Do all 1/0 fields contain only a 1 or 0?)
- Prompt Table (Is field data contained in the specified prompt table?)
- Required Field (Do all required fields contain data? For numeric or signed fields, it checks that they do not contain NULL or 0 values.)
- Translate Table (Is field data contained in the specified translate table?)
- Yes/No (Do all yes/no fields only contain only yes or no data?)

We did not specify any *edit levels* in our program so all edit checks were performed.

The second phase of smoke testing involved the business analysts. Using the delivered query tools PSQuery and other SQL query tools (such as TOAD, SQL Developer) for Oracle database the subject matter experts for each module queried those tables and followed the data mapping document to verify and validate the source data consistency prior actual conversion. Sometimes by looking at query results, they could meet up with the legacy experts and further discuss their findings to come up to a satisfactory answer to any question regarding source file data.

5.11. Error Flag 1

Using staging table gave us the flexibility to introduce a flag field for each row of data. Our application engine program now could update this field based on the result from *ExcuteEdits* method. As soon as all the fields of one row of data passed the edit check, the program will put an ‘A' into this flag field for acceptance. If any of the checks finds unexpected value, then this flag field gets an ‘R' for reject. It also gave the functional analyst similar flexibility. We pass table data to them to verify prior conversion. Using query tools if they could determine any problem for any row a request was sent to development team to update the flag for that particular row as ‘R'. There were cases where the number of rows flagged as ‘R' were too high which also forced us to request the legacy experts to generate a cleaner file. Communication with legacy personnel became much easier as staging table helped us to show them the actual values we received from source file, by using simple reports created from the table. None of this could be achieved if data was still read from file and directly inserted into components.

5.12. Process Scheduler

Based on our test-first model, we end up getting a much cleaner source data file. After we load the file into designated staging table and mark down each row with proper flag value, it is now ready for actual conversion. Batch processes are processes that contain DML (Data Manipulation Language) instructions to process a large amount of data. Application Engine is a type of batch program we developed in Application Designer for data conversion. To run batch processes, PeopleSoft utilizes Process Scheduler Servers. A process scheduler server can reside either on a UNIX server or on a Windows NT server. We found UNIX platform more feasible for our purpose. To implement parallel processing, we decided to use multiple process scheduler servers. We found scheduling our target conversion program in 4 process schedulers all operated on UNIX platform was the most optimum architecture for this project. Multiple process schedulers also provided the flexibility to continue processing on individual process schedulers independently if, for some external reason such as network failure, OS issues one of them goes down.

5.13. Scheduler Program

To achieve parallel processing and evenly distribute conversion program instances in the 4 different process scheduler servers, we needed a master program which will take care of the following,

1. Invoke the right component interface program
2. Assign the right staging table to start processing
3. Schedule the program instance to desired numbers for each process scheduler
4. Combine and generate a richer error report from all the instances

Parameters for this master scheduler programs were,

1. Staging table name
2. Component Interface program name
3. Number of process schedulers (Optimum number was 4)
4. Number of process instances (Optimum number was 60)

5.14. Component Interface Programs

The Component Interface in PeopleSoft is an interface to retrieve, insert or update data from or to a component. Each component interface is built upon one component. Components are collection of objects (fields, tables, panels, menu, registry, permissions) which take care of a particular business function for a module inside PeopleSoft system using transactional data. Component interface for conversion process helps us populate the component with relevant data. We can generate delivered template PeopleCode for an Application Engine program by using the component interface. Then we can modify and enhance this code to decide which fields in the hierarchy of tables we want to populate with the accepted source data (marked as 'A' in the flag field) in the corresponding staging table.

5.15. Parallel Processing

Parallel processing is used when considerable amounts of data must be updated or processed within a limited amount of time. In most cases, parallel processing is more efficient in environments containing multiple CPUs and partitioned data. The master program while given the right parameter, schedules the component interface program instances evenly into the 4 process schedulers. For parallel processing, I used the Single Instruction Multithreaded Design (SIMD) for the scheduled instances. We were using UNIX platform and 4 process schedulers. After much permutation and combination, the optimum number of instances that gave us the fastest processing output was 60 instances. Each scheduler had 15 instances of the conversion

process running concurrently. Table 4.1 shows the results of this whole processing at part B of section 4.5. The hardware platform also had some effect, but we used the standard available hardware at client site without any new recommendations. Parallel processing gives us the flexibility to continue running the instances independently, without worrying about unexpected situation when any of the instances for some reason did not go to completion. As data is properly marked and distributed, we can quickly identify associated rows for a failed instance and separate them for reprocessing.

5.16. Temporary Table

To use parallel processing, we partitioned the data between multiple concurrent runs of a program, each with its own dedicated version of a temporary table. Temporary table is Oracle Database delivered tool which we implemented for our processing. It required following tasks,

1. Define and save temporary table records in Application Designer.
2. In Application Engine, assign temporary tables to Application Engine programs, and set the maximum instance counts (for our processes it was 60) dedicated to each program.
3. Build temporary table records in Application Designer to assign tablespace.

For each component, we divided total rows in source data in staging table by 60. If we had smaller sized data for a component, we could also choose to run a lower number of instances using 1 or 2 process schedulers rather than 4, based on the fact that resource consumption might be more efficient by lowering those. For, smaller data set for components such as Flexible Spending Plan we chose 20 instances divided into 2 process schedulers each running 10 instances.

The Application Engine program invokes logic to pick one of the available instances. Run control parameters passed to each instance enable it to identify which input rows belong to it, and each program instance inserts the rows marked as ‘A’ from the source staging table into its assigned temporary table instance.

5.17. Error Flag 2

Our initial project did not give us the flexibility to choose between subsets of data while running the conversion process. Every time there was a processing error we had to ask for a new file and start afresh. In our new approach, by using error flag 1 during smoke testing, we were able to mark some of the error rows early and restrict them to get into processing during conversion. Now there will be more errors during the actual conversion process. The component interface can find a field of data inside a row which it is unable to insert into the component. Instead of shutting down the whole process this will go through the delivered error handling and update another flag field accordingly. I introduced this flag for this part of the processing, using the values ‘P’ for processed or ‘E’ for errored. So it differentiates from our earlier flag values and strictly tells us the fail happened during this process.

5.18. Iteration

The iterative process is that we mark delivered source data in staging table using two sets of flags, one during smoke testing and one during actual processing. We communicate to rectify those errors once after test-first phase and again after conversion. This way client is notified exactly which data rows were marked as unacceptable to the system. Client can work on this subset of data. While we read the file next time we can compare with existing accepted data in the system and mark our staging table again with proper flags. So we can separate out the small subsets which were newly delivered after correction. Then we can reprocess those rows and

iteratively lower down the size of the error sets as we go through this cycle unless all required rows are entered and converted into the ERP system.

5.19. Post Conversion

The business analysts go through testing the core functionalities for each business areas within a module and carefully work with legacy experts to verify all required transactions once all data for a module is converted.

5.20. Test Environments

Generally, a separate database is maintained for development and initial processing of conversion programs. Once data is fully converted and all post conversion checks are finalized this database can be used to refresh other development and test environments such as, development, functional test, quality assurance, system test and user acceptance test environments.

5.21. Other Development Works

Other development works such as Interface and Report development and unit testing get started while a full conversion cycle is underway. The developers can go through very first-hand testing with the target prototype domain using the delivered sample data that comes by default with the ERP software. Once conversion was done, they were able to go through full-fledged testing of what they developed then. In many cases, a lot of the developments had to wait for complete conversion of the corresponding module. So a quick, robust processing of data conversion helped to design, build and test other development works start early.

CHAPTER 6. LIST OF CONTRIBUTIONS

The research was done altogether in the span of three different ERP implementation projects. I have already highlighted the advantage gained from applying the research findings in real world scenarios. Here I list the major contributions of this research,

- Takes into account some of the major critical success factors of an ERP implementation project by improving Requirements Engineering and Data Analysis and Conversion areas.
- Uses bi-graphs to assist software engineers in mapping legacy functionality to ERP features.
- Applies derived regions from canonical decomposition of the bi-graph to group the findings during Fit Gap Analysis and applies exterior cover properties for domain analysis.
- Presents implementation examples from industry to demonstrate the feasibility of this formalization and analysis.
- Discusses our experiences using the approach in an agile product-line development cycle in industry
- Establishes a test-first approach to ERP data conversion, including a method for incorporating the approach into commonly used ERP implementation activities.
- Introduces staging table to read in legacy source file data as a pre conversion process to do testing using built in ERP features and also sanity checking by business analysts before actual conversion program gets initiated.
- Implements parallel processing technique using SIMD architecture concept in the large amount of data conversion by adjusting the number of necessary batch servers,

dynamically scheduling the number of instances by choice and by applying temporary table feature of relational database management system (RDBMS).

- Introduces accept/reject and processed/errored pairs of flags for each row of source data to better inform the stakeholders about any error row in the source file and reducing the overhead of cleaning up the converted data. It also allows gradual handling smaller subsets of data for each iteration until all rows get into the system.

Thanks!

REFERENCES

1. T. M. Somers and K. Nelson, “The impact of critical success factors across the stages of enterprise resource planning implementations,” in Proc. of the 34th Annu. Hawaii Int. Conf. on Syst. Sci., IEEE Comp. Soc., 2001, pp. 3-9
2. C. Dillon, Stretching toward enterprise flexibility with ERP, APICS—The Performance Advantage, pp 38–43, 1999.
3. J. U. Elisabeth, R. H. Ronald, and M. M. Umble, "Enterprise resource planning: Implementation procedures and critical success factors,” European J. of Operational Research, vol 146, no. 2 pp. 241-257, 2003.
4. L. Anderson, Understanding PeopleSoft 8, Sybex ISBN 0-7821-2930-7, p. 22, 2001.
5. S. A. Alwabel, A. M. Ahmed, & M. Zairi, “The Evolution of ERP and its Relationship with E-Business. Bradford University School of Management,” Int. J. of Enterprise Information Systems, vol. 2, no. 4, 2005.
6. T. C. Loh, & S. C. L. Koh, "Critical elements for a successful enterprise resource planning implementation in small-and medium-sized enterprises," Int. J. of Production Research, vol. 42, no.17, pp 3433-3455, 2004.
7. K. Walsh, The ERP Security Challenge, CSOonline, CXO Media Inc., 2009.
8. J. O'Brien, Management Information Systems (MIS), McGraw-Hill, Irwin, NY, 2011, p. 324.
9. S. Alok & M. Jyothirmayee, “Predicting the behavioral intention to use ERP: an empirical study of the manufacturing industry,” IUP J. of Operations Management, vol. 15, no. 1, pp 7-24, 2016.

10. S. Wieczorek, A. Stefanescu, and I. Schieferdecker, “Test data provision for erp systems,” in Proceedings of the 2008 International Conference on Software Testing, Verification, and Validation. Washington, DC, USA: IEEE Computer Society, 2008, pp. 396–403.
11. Gerrard, Paul. "Test methods and tools for ERP implementations." Testing: Academic and Industrial Conference Practice and Research Techniques-MUTATION, 2007. TAICPART-MUTATION 2007. IEEE, 2007.
12. Togut, D.M. and Bloomberg, E. Morgan Stanley CIO Survey Series: Release 4.5. Morgan Stanley Research Report, 2003.
13. Achieving, Measuring, and Communicating IT Value. Deloitte Touche and IDG Research Services Group Report, 2002.
14. J. Griffith, “Consolidating ERP Systems from 7 down to 1: (M2M, MS Dynamics, MAS200, Avante, ACCPAC, Visual Manf.). Pros & Cons?” in question posted to “ERP Community”, a professional discussion group of, http://www.LinkedIn.com.
15. H. Verjus, S. Cimpan, & I. Alloui, An Architecture-Centric Approach for Information System Architecture Modeling, Enactement and Evolution. InTech, 2012, pp 15-46.
16. C. Ptak, & E. Schragenheim, ERP: Tools, Techniques, and Applications for Integrating the Supply Chain, St. Lucie Press, Boca Raton, FL, 2000.
17. S. Shankarnarayanan, ERP systems—using IT to gain a competitive advantage, March 23, 2000.
18. H. Oden, G. Langenwalter, & R. Lucier, Handbook of Material and Capacity Requirements Planning, McGraw-Hill, New York, 1993.

19. M. A. Rashid, L. Hossain, & J. D. Patrick, The evolution of ERP Systems: A historical perspective. 2002, pp 1-16.
20. J. W. Toomey, MRP II: planning for manufacturing excellence. Springer Science & Business Media, 1996.
21. Source: Press Release by Pierre Audoin Consultants (https://www.pac-online.com), 2012
22. A. Agrawal, G. Finnie, & P. Krishnan, "ORE: A framework to measure organizational risk during information systems evolution," J. Information Systems Development, Springer, US, 2009, pp 675-686.
23. M.P. Ward & K. H. Bennett, “Formal methods for legacy systems,” J. of Software Maintenance: Research and Practice, vol. 7, no. 3, pp 203-219, 1995.
24. T. Mens, J. Magee, & B. Rumpee, "Evolving software architecture descriptions of critical systems," Concern 1, p 1, 2010.
25. M. P. Ward, & K. H. Bennett, "Formal methods to aid the evolution of software," Int. J. of Software Engineering and Knowledge Engineering, vol. 5, no. 1, pp. 25-47, 1995.
26. T. Mens, & M. Wermelinger. "Formal foundations of software evolution: Workshop report" in ACM SIGSOFT Software Engineering Notes, vol 26, no. 4, pp. 27-29, 2001.
27. M. Wirsing, A. Knapp, & S. Balsamo, eds. Radical Innovations of Software and Systems Engineering in the Future: 9th International Workshop, RISSEF 2002, Venice, Italy,. Vol. 9. Springer, 2004.

28. T. Girba, et al. "How developers drive software evolution" in 8th Int. Workshop on Principles of Software Evolution, IEEE, 2005.
29. L. Gunske, "Using a Graph Transformation System to Improve the Quality of Characteristics of UML-RT Specifications," Adv. In UML and XML-based Software Evolution, pp. 21-48, 2005.
30. T. Gorschek, A. Gomes, A. Patterson, & R. Torkar, ”Introduction of a process maturity model for market-driven product management and requirements engineering,” J. of Software Evolution and Process, vol. 24, pp. 83-113, 2012.
31. B. Regnell, P. Beremark, & O. Eklundh, “A market-driven requirements engineering process: results from an industrial process improvement programme,” Requirements Engineering, vol. 3, no. 2, pp. 121-129, 1998.
32. P. A. McQuaid, "Software disasters—understanding the past, to improve the future," J. of Software: Evolution and Process, vol. 24, no. 5, pp. 459-470, 2012.
33. N. G. Leveson, & C. S. Turner, “An investigation of the Therac-25 accidents,” Computer, vol. 26, no. 7, pp. 18–41, 1993.
34. [Online] http://www.vbsoftindia.com/files/images/software-development-pro.jpg
35. von der Beeck, Michael, Tiziana Margaria, and Bernhard Steffen. "A formal requirements engineering method for specification, synthesis, and verification."Software Engineering Environments, Eighth Conference on. IEEE, 1997.
36. Bray, Ian. An introduction to requirements engineering. Pearson Education, 2002.
37. Majchrzak, Ann, et al. "Managing client dialogues during information systems design to facilitate client learning." Mis Quarterly (2005): 653-672.

38. Hooper, James W., and Rowena O. Chester. Software reuse: guidelines and methods. Springer, 1991.

39. Hall, Anthony. "Realising the Benefits of Formal Methods." J. UCS 13.5 (2007): 669-678.

40. Wing, Jeannette M. "A specifier's introduction to formal methods." Computer23.9 (1990): 8-22.

41. Horowitz, Ellis, and John B. Munson. "An expansive view of reusable software."Software Engineering, IEEE Transactions on 5 (1984): 477-487.

42. Illa, Xavier Burgués, Xavier Franch, and Joan Antoni Pastor. "Formalising ERP selection criteria." Software Specification and Design, 2000. Tenth International Workshop on. IEEE, 2000.

43. Clements, Paul C., John D. McGregor, and Sholom G. Cohen. The structured intuitive model for product line economics (SIMPLE). No. CMU/SEI-2005-TR-003. CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST, 2005.

44. Dhungana, Deepak, Norbert Seyff, and Florian Graf. "Research preview: supporting end-user requirements elicitation using product line variability models." Requirements Engineering: Foundation for Software Quality. Springer Berlin Heidelberg, 2011. 66-71.

45. Sistach, F., and J. Pastor. "Methodological acquisition of ERP solutions with SHERPA." First World Class IT Service Management Guide (Ed. J. van Bon), tenHagenStam (2000).

46. Soffer, Pnina, Boaz Golany, and Dov Dori. "ERP modeling: a comprehensive approach." Information systems 28.6 (2003): 673-690.
47. Bueno, Salvador, and Jose L. Salmeron. "TAM-based success modeling in ERP." Interacting with Computers 20.6 (2008): 515-523.
48. Wieczorek, Sebastian, and Alin Stefanescu. "Improving testing of enterprise systems by model-based testing on graphical user interfaces." Engineering of Computer Based Systems (ECBS), 2010 17th IEEE International Conference and Workshops on. IEEE, 2010.
49. ITC InfoTech, "ERP testing (whitepaper)," ITC InfoTech Ltd., New Jersey, NJ, Tech. Rep., Nov. 2010. [Online]. Available: http://www.itcinfotech.com/erp/ERP-Testing.aspx
50. C. Kaner, J. Bach, and B. Pettichord, Lessons Learned in Software Testing. New York, NY, USA: John Wiley & Sons, Inc., 2001.
51. S. Schliesser, "An approach to erp testing using services," Software Science, Technology and Engineering, IEEE International Conference on, vol. 0, pp. 14–21, 2007.
52. O'Leary, Pádraig, et al. "An agile process model for product derivation in software product line engineering." Journal of Software: Evolution and Process24.5 (2012): 561-571.
53. Kollanus, Sami, and Ville Isomöttönen. "Test-driven development in education: experiences with critical viewpoints." ACM SIGCSE Bulletin. Vol. 40. No. 3. ACM, 2008.

54. Janzen, David S. "Software architecture improvement through test-driven development." Companion to the 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications. ACM, 2005.

55. K. Sheikh, Manufacturing Resource Planning (MRP II) with Introduction to ERP, SCM, and CRM. McGraw-Hill Professional, 2002.

56. E. Hau and M. Apar´ıcio, “Software internationalization and localization in web based erp,” in Proceedings of the 26th annual ACM international conference on Design of communication, ser. SIGDOC ’08. New York, NY, USA: ACM, 2008, pp. 175–180.

57. H. Klaus, M. Rosemann and G. G. Gable, “What is erp?” Inf. Sys. Frontiers, vol. 2, pp. 141-162, Aug. 2000.

58. Dulmage, A. L., and N. S. Mendelsohn. "Two algorithms for bipartite graphs."Journal of the Society for Industrial & Applied Mathematics 11.1 (1963): 183-194.

59. K. Kutzov, “An exact exponential time algorithm for counting bipartite cliques,” Inform. Process. Lett., vol. 112, no. 13, pp. 535-539, 2012.

60. Dulmage, Andrew L., and Nathan S. Mendelsohn. "Coverings of bipartite graphs." Canadian Journal of Mathematics 10.4 (1958): 516-534.

61. Oracle Corporation. (2007). PeopleSoft Enterprise Strategic Sourcing. [Online]. Available: http://www.oracle.com/us/products/applications/peoplesoft-enterprise /srm/053957.html

62. Abay-Asmerom, Ghidewon, et al. "Direct product factorization of bipartite graphs with bipartition-reversing involutions." SIAM Journal on Discrete Mathematics 23.4 (2010): 2042-2052.

63. B. Beizer, Software Testing Techniques, 2nd ed. New York: Van Nostrand Reinhold, 1990.

64. SAP AG, “SAP Enterprise Resource Planning,” 2007, http://www.sap.com/solutions/business-suite/erp/index.epx (October 2011).

65. Oracle Corporation, “PeopleSoft Enterprise Human Capital Management,” 2006, http://www.oracle.com/us/products/applications/peoplesoft-enterprise/hcm/index.html (October 2011).

66. D. S. Janzen, “Software architecture improvement through test-driven development,” in Companion to the 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications, ser. OOPSLA ’05. New York, NY, USA: ACM, 2005, pp. 240–241. [Online]. Available: http://doi.acm.org/10.1145/1094855.1094954

67. C. Kaner, J. Bach, and B. Pettichord, Lessons Learned in Software Testing. New York, NY, USA: John Wiley & Sons, Inc., 2001.

68. Oracle Corporation, “PeopleSoft Enterprise Human Capital Management,” 2006, http://www.oracle.com/us/products/applications/peoplesoft-enterprise/hcm/index.html (October 2011).

69. E. M. Clarke & J. M. Wing, “Formal methods: State of the art and future directions” in ACM Computing Surveys (CSUR), vol. 28, no. 4, pp. 626-643, 1996.

70. C. Heitmeyer, "On the need for practical formal methods" in International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems, Springer, 1998, pp. 18-26.

71. A. Hall, “Seven myths of formal methods” in IEEE software, vol. 7, no. 5, pp. 11-19, 1990.
72. S. H. Roosta, Parallel processing and parallel algorithms: theory and computation. Springer Science & Business Media, Dec 6, 2012.
73. J. Dean & S. Ghemawat, “MapReduce: simplified data processing on large clusters” in Communications of the ACM, vol. 51, no. 1, pp. 107-113, Jan 1, 2008.
74. E. M. Maximilien & L. Williams, “Assessing test-driven development at IBM” in IEEE Proceedings of 25th Int. Conf. in Software Engineering, May 3, 2003, pp. 564-569.