Stable Information Structure for Inherent Flexibility in Information Systems

Research Dissertation Submitted By

Islam Ali
Reg. No. 270-FBAS/MSSE/F09

Supervisor
Dr. Muhammad Wasif Nisar
Associate Professor, Department of Computer Science,
COMSATS Institute of Information Technology, Wah

Co-Supervisor
Mr. Shahbaz Ahmed Khan
Assistant Professor, Department of Computer Science & Software Engineering, Faculty of Basic and Applied Sciences,
International Islamic University, Islamabad

Department of Computer Science & Software Engineering
Faculty of Basic and Applied Sciences
International Islamic University Islamabad
2012

International Islamic Univ.
CENTRAL LIBRARY
ISLAMABAD

Accession No TH-9171

MSc
004
ISS

Automatic data processing

DATA ENTERED
[illegible] 15/3/13

International Islamic University, Islamabad

Dated: 18 Oct 2012

Final Approval

This is to certify that we have read the thesis submitted by **Islam Ali, Reg # 270-FBAS/MSSE/F09**. It is our judgment that this project is of standard to warrant its acceptance by the International Islamic University, Islamabad, for the Degree of **MS in Software Engineering**.

Thesis Evaluation Committee

External Examiner:
Dr. Aamer Nadeem
Associate Professor,
Department of Computer Science,
MAJU, Express Way, Kahuta Road, Zone V,
Islamabad

Aamer

Internal Examiner:
Mr. Muhammad Usman,
Assistant Professor
Department of Computer Science
International Islamic University
Islamabad

[illegible]

External Supervisor:
Dr. Muhammad Wasif Nisar
Associate Professor
Department of Computer Science,
COMSATS Institute of Information Technology
Wah

[illegible]

Internal Supervisor:
Mr. Shahbaz Ahmed Khan Ghayyur
Assistant Professor
Department of Computer Science and Software Engineering
International Islamic University
Islamabad

[illegible]

DEDICATION

I dedicate this research project to my beloved Prophet

Hazrat Muhammad

(Sallalaho Alaehe wa Aalehi wa Sallam)

Who demonstrated the righteous path to whole mankind and drag it out from the nastiest depths of ignorance to the preeminent level of humanity.

A Dissertation submitted to

Department of Computer Science & Software Engineering,

Faculty of Basic and Applied Sciences,

International Islamic University, Islamabad

As a partial Fulfillment of Requirements for the Award of the Degree of

MS in Software Engineering

DECLARATION

I hereby declare that this thesis **"Stable Information Structure for Inherent Flexibility in Information Systems"** neither as a whole nor as a part has been copied out from any source. It is further declared that I have done this research with the accompanied report entirely on the basis of my personal efforts, under the proficient guidance of my teachers and my friends especially my supervisor Dr. Muhammad Wasif Nisar and Co-Supervisor Shahbaz Ahmed Khan.

If any of the system is proved to be copied out of any source or found to be reproduction of any project from any of the training or educational institutions, I shall stand by the consequences.

[illegible]

Islam Ali

Reg. # 270-FBAS/MSSE/F09

ACKNOWLEDGEMENT

All praises to Almighty ALLAH PAK, the most Gracious and Beneficent, Whose copious blessings enable me to pursue and perceive higher ideals of life, Darood and salaam for His beloved Prophet MUHAMMAD (Sallalaho Alaehe wa Aahlehi wa Sallam) Who demonstrated the righteous path to whole mankind and drag it out from the nastiest depths of ignorance to the preeminent level of humanity.

I am proud to express my deep sense of obligation and special appreciation to my reverend supervisor honorable Dr. Muhammad Wasif Nisar and co-supervisor Shahbaz Ahmed Khan for their dexterous guidance and kind behavior during the project. Their encouragement, moral support and motivation helped me a lot to get through any problem or difficulty during each step.

I am much grateful to all my respected teachers for their guidance and help throughout my life which results in letting me to reach at this stage. I am also thankful to whole administration and management team of International Islamic University especially of Computer Science & Software Engineering Department for their managerial and administrative support at every step. I would like to say thanks to my boss Waqar Ahmed, Director IT for his cooperation and moral support he extended during stress days I was passing through.

Furthermore I must highlight that it was mainly due to my family's moral support during my entire academic career which enabled me to complete my work dedicatedly. Their endless efforts, care and prayers are always around me to keep me safe in every difficulty and help me in attaining every success. I am also thankful to my life partner who encouraged me at those moments when I got exhausted.

Finally, I am whole heartedly thankful to all the fellows who have helped me during achieving this degree.

[illegible]

Islam Ali

Reg. # 270-FBAS/MSSE/F09

ABSTRACT

It's a known fact that change is immortal. The dictum "**change or die**" insinuates that the ability to change or adapt according to situation is the key to survival. Information System (InfoSys) is certainly not an exception. Typical InfoSys are inflexible, rigid and change-averse especially for those, emerging from the changing business requirements. Structural changes in InfoSys can justifiably be referred to as aftershocks, where a business-driven change triggers technical-oriented chain of changes which in turn creates *ripple-effect* and possibly *avalanche-effect.* We propose change-driven *framework for stability of information structure contributing to inherent flexibility in information systems* thereby rendering software really softer able to welcome modifications warmly. The objective is to render modifying InfoSys as easy as the corresponding business systems lend itself to change. This includes how to *break through the Bruce Johnson's Limits to flexible design.*

Behind the approach, Murphy's Law is in action i.e. if something can potentially change, it will certainly change. Most of the potential changes come from the assumptions we make while designing where requisite information is not available by the time. Carry coordinated attack against potential technical sources of change before it attacks our allies (end-users) and ultimately us. So it's a proactive approach to address changes before they come out of the Pandora-box to play havoc. It's not about how to do changes but how not to do i.e. render the InfoSys more user-modifiable and less developer-modifiable. It's not only shifting of modification responsibilities from software-engineers to domain-experts but making developer's job easier doing changes with lesser efforts / time comparatively. As a proof-of-concept, re-design and re-implementation of **FlexInfoSys** has been carried out to demonstrate the viability of the proposed framework for inherent flexibility in InfoSys.

The results of case study show that the proposed framework, comprising of ten pair of enablers, provides a concrete foundation for inherent flexibility in information systems. It contributes to avoid chain of changes (Ripple and Avalanche effects) including how to address the Bruce Johnson's Limits-to-Flexible Design through conceptual inheritance in information structure. In other words it extends, a bit more, the limits of flexible design in information systems marked by Brue Johnson.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF GRAPHS

LIST OF TABLES

LIST OF TEXT BOXES

LIST OF ABBREVIATIONS, ACRONYMS AND DEFINITIONS

S#	Abbreviation/ Acronym / Term	Explanation/Definition
1	SCSs	Set of Change Scenarios
2	InfoSys / IS	Information Systems
3	RigInfoSys	Rigid Information Systems
4	FlexInfoSys	Flexible Information Systems
5	BR	Business Rules
6	Ripple Effect	The problems faced or the series of corrective actions one has to take, within a module itself, to accommodate a requested change is referred to as ripple effect
7	Avalanche Effect	The problems faced or the series of corrective actions one has to take in allied (integrated) modules to accommodate a requested change is referred to as avalanche effect
8	UI	User Interface
9	CoC	Chain of Changes
10	ER	Entity Relationship
11	Information Structure & ER	These are two interchangeable terms
12	ICT	Information & Communication Technology
13	eProfile	Employee Profile
14	UC	University of Cincinnati
15	ERD	Entity Relationship Diagram
16	DWH	Data Warehouse
17	DDPCS	Database-Driven Product Catalog System
18	TSAFE	Tactical Separation Assisted Flight Environment
19	API	Application Program Interface
20	MMT	Management Monitoring Tool
21	OWES	Other Workflow Executive Service
22	WCAP	Workflow Client Application Procedure
23	CAP	Called Application Procedure
24	NADRA	National Database Registration Authority
25	CNIC	Computerized National Identity Cards
26	Corpse Entity	Whose living status is unknown
27	MIS	Management Information System
28	FK	Foreign Key
29	RWBS	Real World Business System
30	AWIS	Automated World Information System
31	SCAMPI	Standard CMMI Assessment Model for Process Improvement

Chapter - 1

INTRODUCTION

1. INTRODUCTION

As it is clear from the title of the proposal, this research activity is about introducing inherent flexibility in information systems by focusing on the change-driven ER-Oriented design strategies for stability of information structure.

Literature acknowledges the predominance of flexibility in software whether it is design or implementation. Plethora of design strategies e.g. structured design, modular design, object-oriented design, software architecture, design patterns, component-based software engineering among others are aimed at to increase flexibility. Flexibility has therefore become a core issue in software engineering generally and software design research particularly.

1.1 Information Systems

Information system [16] is defined as "the effective design, delivery, use and impact of information (and communication) technologies in organizations and society." This definition captures an important part of InfoSys, that is the development of IT applications, and it recognizes that successful applications of ICT require broader attention than just that on the technology. The InfoSys discipline has steadily developed from its initial 'techno-centric' focus to a more integrated technology, management, organizational and social focus. But this definition does not capture the excitement of the discipline.

We are now in a period of great transformation, as organizations change to address their challenges or achieve their goals. It is also a period of structural transformation of the global economy. ICT supports and enables most of these changes, and InfoSys is the only discipline with a primary focus to study the applications of technology by organizations and society. It is therefore particularly relevant during this period of great change.

The following definition suggested by the UK Academy for Information Systems, taken from book "*Information Systems – The State of the Field*" of Wiley Series [16], is somewhat broader than the definition looked at previously:

"The study of information systems and their development is a multidisciplinary subject and addresses the range of strategic, managerial and operational activities involved in the gathering, processing, storing, distributing and use of information, and its associated technologies, in society and organizations". The above definition is, however, somewhat passive about InfoSys as it does not give a sense of the creativity and innovative effort that is part of the potential contribution of InfoSys.

On the technology side of information systems, it is differentiated from computer and IT disciplines by its focus. As illustrated graphically in the Figure 1-1 below, compared with two other IT-related disciplines, computer science and computer systems engineering, InfoSys emphasizes the applications of technology rather than a focus on fundamental technologies and theories [16]. It focuses more on interactions between

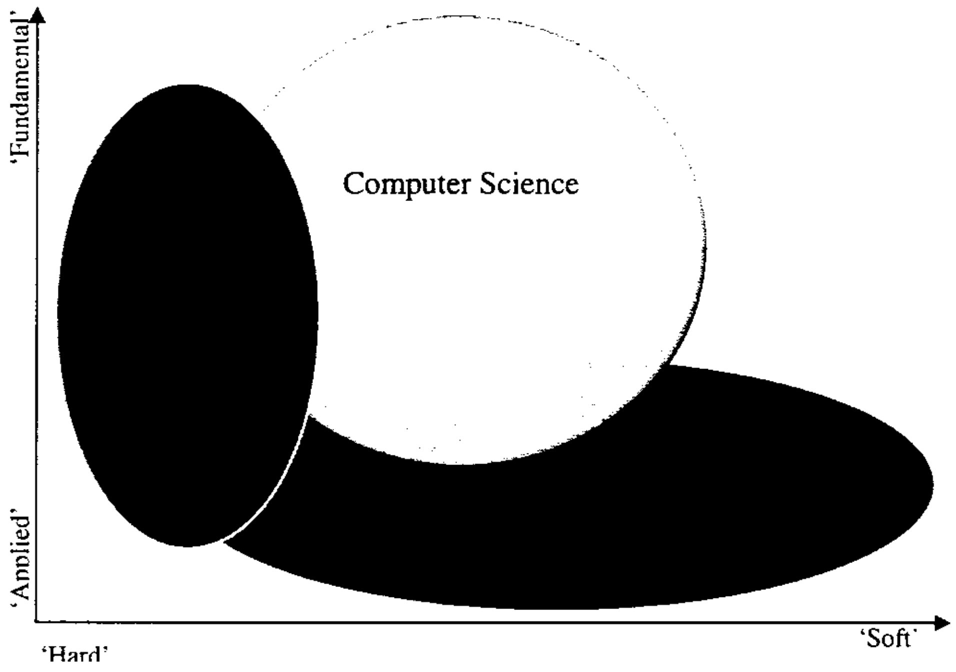


Figure 1-1: Differentiating IS from other IT-related Disciplines

people and organizations (the 'soft' issues) and technology rather than on the technologies (the 'hard' issues) themselves. It should be noted that figure represent the focus of the different disciplines, not the quantum of work conducted in or contributed by any of the disciplines.

"Systems can be built quickly or big or right – but not all three together. They are bound to be small or wrong, if we build them quickly. Building big systems will definitely be either late or wrong and for really big systems; they are guaranteed to be late and wrong" [12]. This fact is illustrated in Figure *1-2* below.

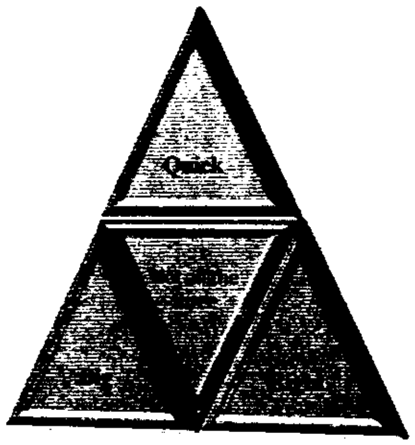


Figure 1-2: System can be built quick or big or right but not all the three

As per Bruce Johnson [12], Successful information system is not the one that usually generate few, if any modification requests but the one that is used more, generally generates continuous demand for modification and having the ability to keep up with demands for modification. This is because as the users get more used to the software, they try to get more out of it by experimenting with new scenarios not included in the scope specified in the first place.

1.1.1 Classification of Information Systems

Information systems used to be classified in 1980s as pyramid of systems resembling organizational hierarchy, starting with transaction processing systems at the lower end of the pyramid, moving up management InfoSys & decision support systems are next levels respectively and executive information systems at the pinnacle as depicted in Figure *1-3*.

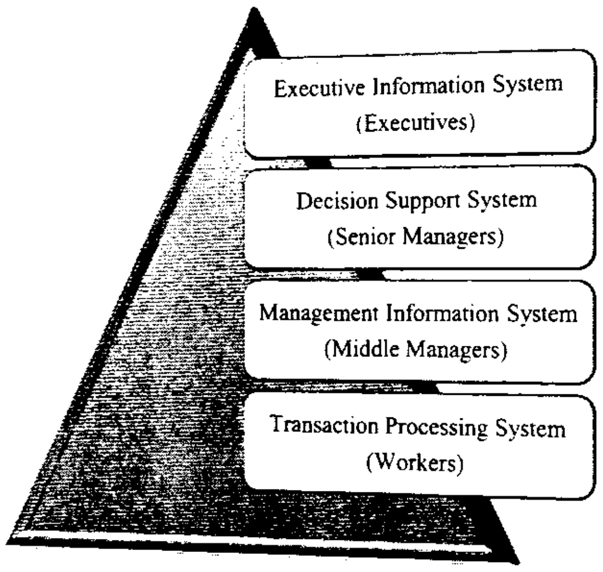

Executive Information System
(Executives)
Decision Support System
(Senior Managers)
Management Information System
(Middle Managers)
Transaction Processing System
(Workers)

Figure 1-3: Pyramid of Systems

However, Information systems can be classified into the following classes as depicted in diagram Figure *1-4* below.

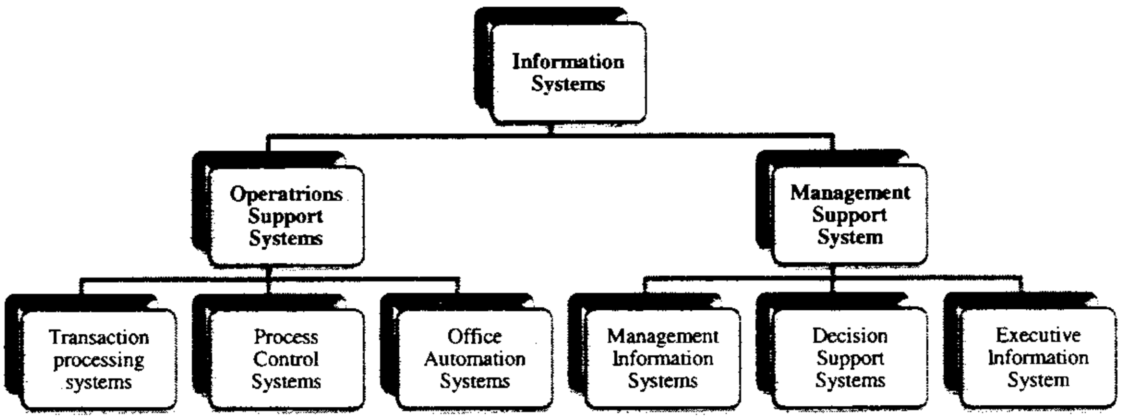

Information Systems
Operatrions Support Systems
Management Support System
Transaction processing systems
Process Control Systems
Office Automation Systems
Management Information Systems
Decision Support Systems
Executive Information System

Figure 1-4: Classification Hierarchy of Information Systems

Some of other information systems are:

- Expert Systems
- End-User Computing Systems
- Integrated Information Systems
- Knowledge Management Systems
- Business Information Systems
- Strategic Information Systems

1.1.2 Real World and Automated World Systems

In order to clearly differentiate, the business system is termed as Real World Systems whereas when the same is computerized, it is known as Automated World System. In fact, Real World Business Systems are changed and Automated World Information Systems are modified [12], [14]. Information system changes/modifications stem from two main sources business and technology, hence referred to as business-driven and technology-oriented changes. When InfoSys faces (structural) changes, it is flexibility and modifiability that keep things in order. Bruce Johnson [12], [23] defines flexibility as: "Flexibility in system means the ability to accommodate a change in business requirement with a minimum of modifications to system components". Delivering functionality and focusing on potential change i.e. flexibility are competing factors. One is visible whereas the other is hidden and intangible. Owing to self-imposed time pressure and other factors the later one is ignored for which all involved have to pay later on in maintenance phase. Business people are not supposed to give such requirements in advanced. Software designers and developers should take on this responsibility to foresee such potential possibilities in advance to avoid manifold efforts as well as downtime of the system.

Generically, information system development cycle has four steps as illustrated in the Figure *1-5* given below. Over the entire life cycle of Infosys, maintenance is more important than development and within maintenance, resynchronization is overwhelming than debugging" [12].

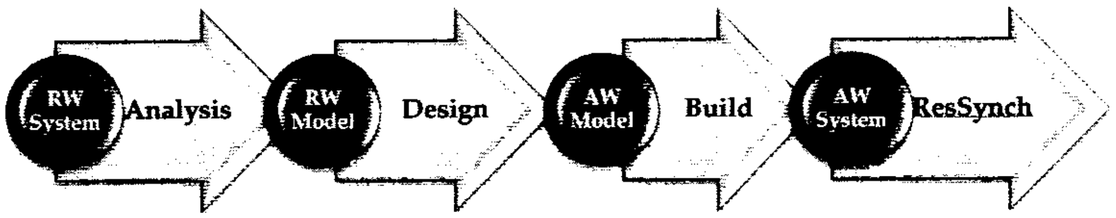


Figure 1-5: Information System Life Cycle

1.2 Software Flexibility

The term flexibility is deemed as vague because of its polymorphous nature and has been found with many different connotations [9] and [11]. Nature of the problem and setup to be investigated are among factors that affects meaning of flexibility [9].
It can be viewed as:

- Inherent property of an entity e.g. IS, Organization etc.
- Response capability to foreseen or unforeseen changes.
- Temporal aspect – How quickly the response should be for an entity to be labelled as "Flexible".
- Capacity to adapt – Adaptive Capabilty

Bruce Johnson [12], [41] defines flexibility as: *"Flexibility in system means the ability to accommodate a change in business requirement with a minimum of modifications to system components"*. Delivering functionality vis-à-vis focusing on potential change i.e. flexibility are two competing factors. One is visible whereas the other is hidden and intangible. Owing to self-imposed time pressure and other factors the later one is ignored for which all involved have to pay later on in maintenance phase

1.3 Types of Flexibility in Information Systems

Two basic approaches [12] have been taken to address flexibility in information systems. Boogard characterizes these as active flexibility and passive flexibility.

- **Active flexibility** is basically fast modifications. It also includes automatic program modification. Improving modification process is the center of attention in Active flexibility.
- **Passive flexibility** is built-in and inherent. The InfoSys is designed to require inherently less modifications. Contrarily, Inherent flexibility focuses on the system itself.

1.4 Characteristics of a Flexible Information System

Bruce Johnson's group [41] and [14] suggested a set of characteristics for a flexible information system. A flexile information system is integrated with other relevant modules in a bigger perspective and ***not functioning in isolation***. It shares common information structure with a super system or community of InfoSys. It has ***stable structure***. The information structure is presented in a manner to provide maximum flexibility model of things, their attributes and relationships should support present as well as requirements of time-to-come. Those characteristics of a business system that are subject to change are embodied in InfoSys in a manner that change in any of it does not warrant modification in data or procedure. The system identifiers represent no system attribute and are meaningless. All hierarchical entities e.g. bill-of-materials, organization structure and chart-of-accounts be recursively structured. It is ***open to extensions*** i.e. new entities / processing feature can be added without changing the old ones. It is ***regulatable*** initially incorporating and then maintaining the business rules / policies through its information structure and able to change system's behavior through changing regulatory values by user instead of modifying the code by developers. It has ***defined threshold to maintenance limits*** i.e. software designers are sure of when code maintenance and changes to information structures will be necessary. A checklist, based on these characteristics, has been developed for flexible information systems placed as Annex 'A'.

Banking upon the hard experiences with rigid systems, we focus on stability of information structure via design strategies to introduce inherent flexibility and to prevent the costly chain of changes and hence its ripple & avalanche effects in InfoSys. Furthermore, we seek to break the ceiling effect being observed for so long.

1.5 Motivation for the Study

Following motivators are active behind this research study.

- The essence of adaptability to change, chiefly in InfoSys, lies in design of information structure. Usually if the requirements change, the information structure needs to be changed as well. Therefore the prime source of inflexibility in InfoSys is the unstable information structure. So attacking core seems to be logical answer to the problem.
- A sensibly designed stable information structure keeping flexibility in view provides a concrete foundation for all other kind of flexibilities. On the other hand, one cannot exploit and reap the full benefits of any flexibility-technique if stable-information-structure is not in place. It is worth mentioning here that modification in information structure [14] that can ruin a traditional Infosys can similarly destabilize an object-oriented system.
- Last but not least, software flexibility help minimize the maintenance cost in terms of time, efforts as well instrumental in hard times.

1.6 Goals and Objectives

The mentionable objectives set-up for the ***FlexInfoSys*** research project, are as follows: -

- Creating change-absorbing capability in ISs to obviate the ongoing adaptive-maintenance - a black hole engulfing precious resources that in turn waste away the development-capacity sharply.
- Forestall the formation of costly chain of changes, resulting into ripple effect triggered by a business-driven change in requirements, making use of "Change-Driven and Structure-Oriented" design strategies rather than indulging in expensive corrective actions.
- Build capacity for change by rendering the InfoSys more user-modifiable and less developer-modifiable and thus rescuing development capacity for more creative undertakings.
- Enabling re-synchronization of the automated world InfoSys with its counterpart real-world business system more effective and efficient i.e. to respond gracefully to changes more easily and with fewer resources.
- Keeping the maintenance-backlog in the manageable threshold.

1.7 Scope of the Research

Generically scope of the research is limited to software designing for changing business requirements and specifically to the following:

- Change-tolerant ER-Oriented design strategies for stability of information structure [1].
- Breaking the Bruce's Limits-to-Design Flexibility [12], [23].

1.8 Problem Statement

The two-fold problem elucidated, with examples from a case study carried out for the purpose, is as under:

1.8.1 Chain of Changes - Ripple and Avalanche Effects

In one of the information systems renamed as RigInfoSys, a table for "Designation" was defined. On arising need, appointments were entered by the user in the same table and being used. This gave a way for designations and appointments to jumble up.

Action on user request to address this problem leads to change the information-structure as below.

Example 1: *Changes in Designations*

In old rigid InfoSys, a table for "*Designation*" was defined. On arising need, appointments were entered by the user in the same table as shown in Figure *1-6* and its relevant chain of changes is depicted in Table 1-1.

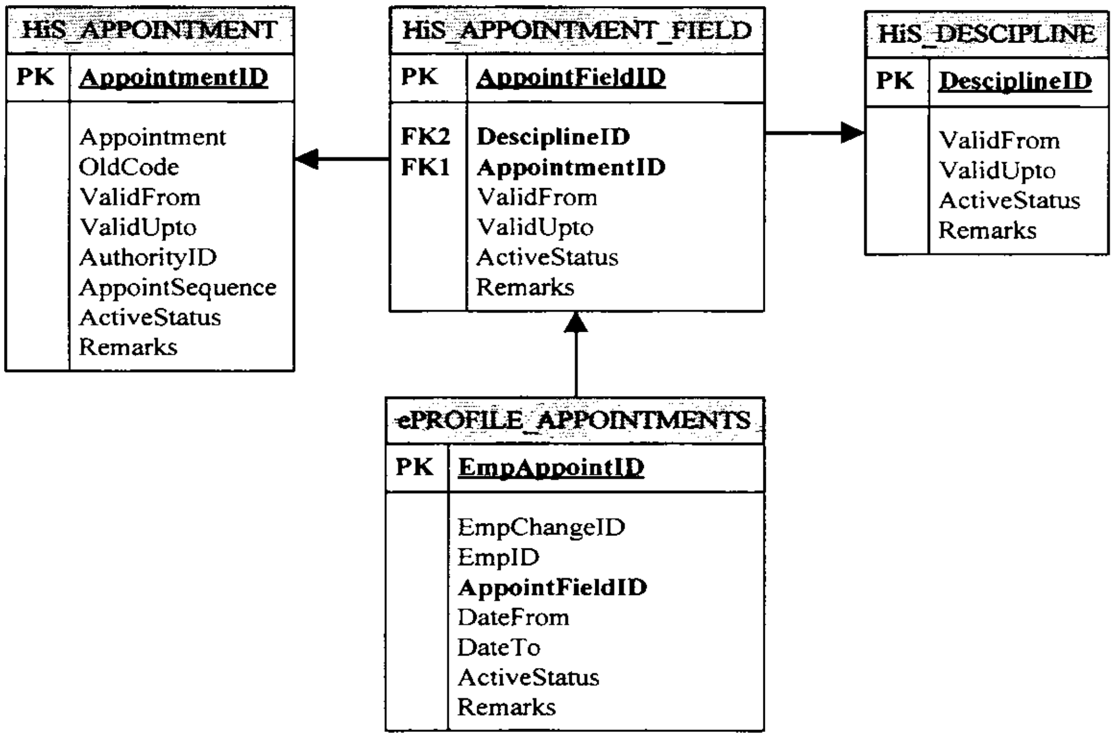


Figure 1-6: Defining appointments and its chain of changes in rigid InfoSys

Table 1-1: Change Impact Analysis for Example - 1

[illegible]	
1	Create HiSAppointment table
2	Create HiSDiscipline table as 1 depends on discipline
3	Create HiSAppointmentField to link the two tables in created in 1 & 2
4	Create PiSAppointment table as appoints of an employee can be more over the period of time.
5	Identify and migrate the appointments already entered into HiSAppointment table along with the old code
6	Map and update the appointments for employees with the help of old code
[illegible]	
All integrated systems making use of designations/ appointments were affected e.g. User Management System and Workflow System and their relevant developer had to take series of corrective actions	

Example 2: *The definition of employee-categories*

The definition of employee-categories was hard coded in a function using if-then-else structure. But when more calculations/reports were demanded regarding the categories e.g. display category II employees, two problems arose. First, one could not have a history of an employee category. Secondly, the function is called hundreds of thousand times and brings down the performance beyond affordable level. The relevant chain of changes is illustrated in Table 1-2. The flexible design for the purpose is illustrated in Figure *1-7* given below.

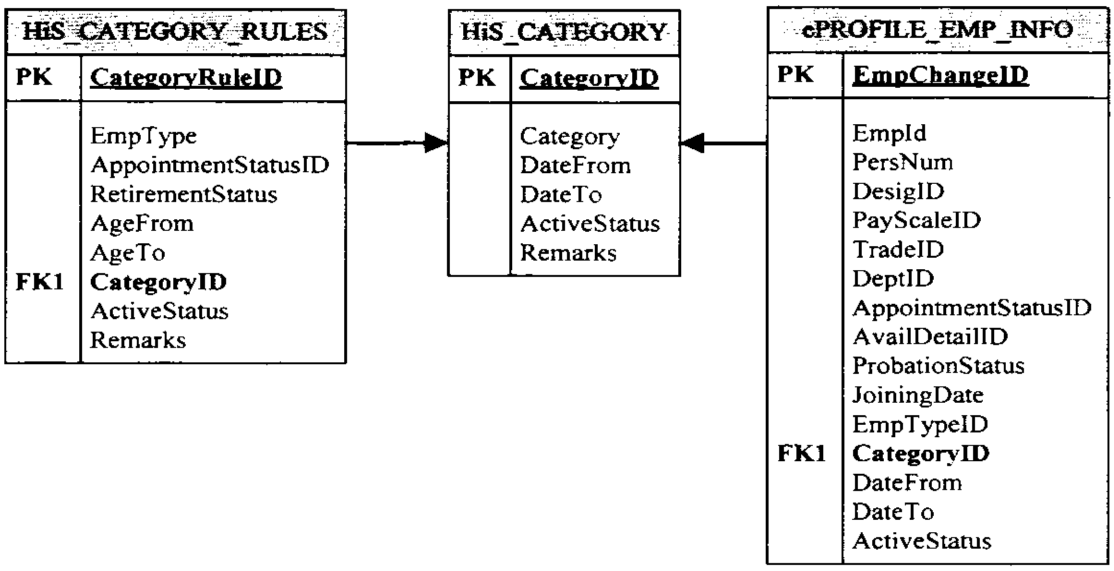


Figure 1-7: Employee Categories and relevant business rules

Table 1-2: Change Impact Analysis for Example - 2

[illegible]	
1	Create HiSCategory table.
2	Analyze the category rules and make patterns of it.
3	Create HiSCategoryRules table and link it with HiSCategory table.
4	Develop interface for the rules to be populated into HiSCategoryRules.
5	Define category as per the rules populated.
6	Process an employee to determine his/her category by matching the employees data and category rules.
7	Save an employee category in PiSEmpInfoTable for future processing.
[illegible]	
In this case, there was no avalanche effect as employee category table was not created and hence not used by any other module(s).	

Example 3: *Employee Courses*

Making assumptions is the root cause of potential changes. Here in this example, it was assumed that Trade can provide the possible values for qualification field and was linked with transactional course table. Later on it transpired that it is not the case and results into the following ripple effect. This is illustrated in Figure *1-8*. Its ripple and avalanche effects are tabulated in Table 1-3.

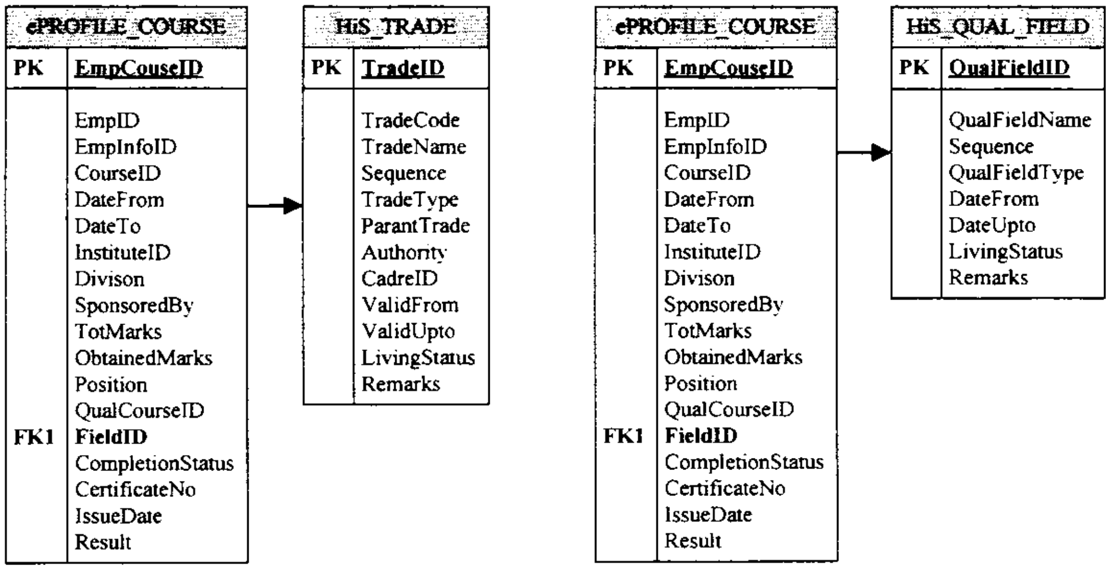


Figure 1-8: Trade and Qualifications of an employee a change

Table 1-3: Change Impact Analysis for Example -3

[illegible]	
1	**Create HiS_QUAL_FIELD configuration table.**
2	Transfer the already existing trades in HiSTRADE table into HiS_QUAL_FIELD table along with the old IDs.
3	**Map the new relevant IDs with help of old IDs**
4	Update the PiS_COURSE.FieldID values by relevant new HiS_QUALFIELD.QualFieldID values.
5	**Link the HiS_QUAL_FIELD table with PiS_COURSE & de-link Trade.**
6	Do 3 & 4 for Qualification transactional tables as well.
[illegible]	
All integrated systems making use of Trade as qualification/course field were affected e.g. Training Module and their relevant developer had to take series of corrective actions.	

TH 9171

Furthermore, change-resistant information-structure does not allow breaking this chain once formed and proves more stubborn & costly. It's because you can build a new building but can't modify the past. Secondly this chain viciously erodes the development capacity as precious resources are drowned in fire-fighting maintenance-backlog. The challenge is to prevent this chain-formation process instead of fighting against its resulting ***ripple*** and ***avalanche effects***. Designing stable information structure, enabling flexibility in IS, will help render the adaptive maintenance more cost-effective.

1.8.2 Bruce Johnson's Limits to Design Flexibility

Bruce Johnson [12], [23] highlights that there is limits-to-flexible design. For example, in a human resource system, there may be two types of employees hourly and salaried. Certain information would be recorded for each type. Let us say that a new type, contractor, was introduced. Simply adding a new value to the list of employee types does not work. It is likely that new field(s) specific to contractors would need to be added. Most of the emerging changes necessitates information structure and programs to be modified. This is termed as Bruce Johnson's limits-to-flexible-design. This is diagrammatically depicted in 4.5.10. So we have to:

- Prevent/avoid aforementioned chain-formation process instead of fighting against its resulting ripple and avalanche effects.
- Addressing Bruce's Limits-to-Flexible-Design.

1.9 Research Questions

This work include delving into and digging out the following:

- What are the ER-Oriented **Rigidity Enablers** responsible for fostering rigidity in InfoSys?
- How to avoid the chain of changes (**ripple & Avalanche effects**) caused by rigidity enablers by making use of ER-Oriented **flexibility enablers**?
- How to break the limits-to-flexible design (**ceiling-effect**) marked by Bruce Johnson, while dealing with inherent flexibility in InfoSys?

1.10 Contribution of the thesis

The contributions of the thesis are given briefly as follows:

- Brought the ER Oriented rigidity / flexibility enablers into a coherent framework for stability of information structure.
- Figured out generic chain of changes each rigidity enabler causes.
- Extended the limits of inherent flexibility by adding 8th, 9th and 10th pair of flexibility enablers in the proposed framework including devising FlexEnab-10

to address Ceiling Effect (RigEnab-10) marked by Bruce Johnson as limits to design flexibility.

- Defined "How to Use Guidelines" for each flexibility enabler.

1.11 Organization of Thesis

This thesis is organized as per the guidelines of institution. Introduction provides definition of InfoSys, its types, software flexibility. Introduction concludes with characteristics of flexible infosys. A problem statement, to be addressed, is elucidated with examples. Literature review throws light on various approaches adopted by different authors, critical summary of the publications of other researcher, who worked in the same domain. A framework has been proposed for stability of information structure to infuse flexibility in information systems subsequently. A case study "*FlexInfoSys*" conducted is narrated. The viability of the framework is demonstrated based on criteria set out and change scenarios provided in case study. The conclusion drawn, lessons learned and potential future work is presented at the end.

1.12 Summary

In this introductory chapter, information system is defined. Information system can be developed big or right or quick but not all the three simultaneously. Increasing number of development professional is not the answer. The bigger the team, the more coordination and monitoring overhead get involved. Classification hierarchy of InfoSys is depicted graphically. The difference between the "Real World" and "Automated World System" is elucidated. Information system lifecycle is sketched with the help of a diagram. Flexibility is defined briefly along with its basic type. Bruce Johnson's characteristics of a flexible information system are numerated subsequently. Motivation, goals and objective and scope of the study is narrated respectively. The two-fold problem is elucidated with the examples from case study carried out for the purpose. Research questions and contribution of the study presented. The chapter concludes with elaboration of thesis organization.

Chapter - 2

RELATED WORK

2. RELATED WORK

In today's ruthless and cut-throat competition and changing business environments, requirements never remain unchanged. Managing this change in requirements poses a real challenge.

Unfortunately, it transpires from closely-related literature review that research devoted to this particular issue is rare and insufficient. There is a pressing need to look into it in concrete and comprehensive way. In the area of software flexibility, my co-authored paper [15], elaborating and comparing deployment strategies to figure out the optimal one for a re-engineered flexible InfoSys in context of legacy system is published in research journal of applied sciences, engineering and technology. The relevant research work carried out so far is being highlighted here in chronological order and organized as abstract, contribution, methodology used as well as limitations.

2.1 Laura Jacome's Approach (2011)

2.1.1 Abstract

Laura Jacome [11] explores the potential flexibility in the use of information technology. Laura tried to reconcile varying standpoints of flexibility by presenting a flexibility model for information systems after in-depth analysis of those standpoints and definitions of flexibility. The model focuses only *efficient versatility* and *robust responsiveness* accommodating *foreseen and unforeseen* changes respectively. The author further proposes flexibility enablers for both types of flexibility. However Laura evaluates the viability of robust responsiveness flexibility only by conducting a case study on IS meant for competitive industry to accommodate unforeseen changes.

2.1.2 Contribution

Lara Jacome's and co-authors proposed following twenty one flexibility enablers to accommodate foreseen and unforeseen changes as tabulated in **Table 2-1**.

Table 2-1: Flexibility enablers to address foreseen and unforeseen changes

Type of Flexibility	Business Process	Information Technology	Data
Efficient Versatility Expansion Flexibility to address Foreseen Changes	**1. Broad knowledge**	**1. Large number of application feature**	**1. Data scalability**
	2. Deep Knowledge	2. In-depth embedded knowledge on selected feature	2. Expansive data Dictionary
	3. Strategy and then Technology	**3. Well integrated structure among features**	**3. Location free access**
	4. Good Documentation	4. Application parameterization	
Robust Responsiveness to address Unforeseen Changes	**1. Meta Knowledge**	**1. Modularity among industry level standards**	**1. Data independence from applications**
	2. Loose Coupling	2. Industry level standardized process and content interfaces	2. Inter-organizational structured data connectivity
	3. Decoupling	**3. Relatively speedy integration of new software module**	**3. Industry level data standardization**
	4. Technology and then Strategy		

2.1.3 Methodology Used

As elucidated, the presented model has two major parts, flexibility enablers for accommodating foreseen and flexibility enablers for addressing unforeseen changes. A ***case study*** was carried out on a data warehouse information system to examine the later part i.e. flexibility enablers for unforeseen changes. The study was conducted in a telecom industry of Latin American. As DWH of the firm served decision makers, the changes were supposed to respond swiftly. The changes were not only limited to interfaces, databases, DWH services, business rules, regulations but similar information available in the system. Because of its nature, the changes were not simply predictable well before, hence, the data warehouse software was supposed to be capable of accommodating these unforeseen changes in a robust manner.

2.1.4 Limitations

The study focus is quite broad focusing on two types of changes i.e. foreseen and unforeseen changes. Though highlighting some of the relevant flexibility enablers such as well-integrated structures, application parameterization and data scalability but not delved deeply on how to provide flexible ER structures. Furthermore, it's not addressing the Bruce's limits to design flexibility. Encircled is the focus of the research.

2.2 Asma Alkalbani and Kinh Nguyen Approach (2010)

2.2.1 Abstract

Asma Alkalbani and Kinh Nguyen [1] present some techniques to implant the built-in flexibility in information system. The real challenge is not to meet the demanded requirements but to accommodate a handful of potential ones in the time to come. The root cause behind the rigidity lies in setting traditional objective of system design i.e. functional accuracy only. Sacrificing long term objectives i.e. flexibility and maintainability for the short term objectives i.e. current requirements, result into fragile information systems. System's flexibility does not come for free; it takes deliberate effort to enable system capable of accommodating potential changes. The essence of adaptability to change lies in information system data structure. Usually if the requirements change, the data structure design needs to be changed as well. Therefore, the source of problems lies on data structure design.

2.2.2 Contribution

Asma Alkalbani and Kinh Nguyen studied the problem of inflexible information systems carefully to clarify:

- What exactly is meant by inflexibility, and
- To what extent can inflexibility be avoided, and how.

They identified some of the main sources [1] of the problem that lead to inflexibility in information systems: These are weaknesses in:

- Designing identifiers
- Designing entities
- Design relationships
- Business rules support

Secondly, they carried out a comprehensive review of past research work, especially researches that try to deal with the problem by designing information structure with built-in flexibility. They examined in detail the existing flexible information system design approach by Bruce, Walter, Robert and Cindy (Bruce Johnson's Group) and suggested a flexible design solution for the product line of Product Catalogue that requires multi-language support.

Finally, they presented a technique to further enhance the flexibility. The technique allows the user to enter a relationship type by entering some appropriate data. To define a new relationship type, one needs to create two tables; one with attributes (Relation-ID, Relation_Degree) and another with attributes (Relation_ID, Role, Entity_Type, Min_Bound, Max_Bound). To store the relationship instance, create a third table with three attributes (Rlation_Instance_ID, Relation_ID, Entity_ID). This technique, one can observe, is the extension of the technique to design generic structure of an organization capable of absorbing changes to its organogram.

2.2.3 Methodology Used

The Case Study was carried out to validate the proposed techniques. A flexible Product Catalog Database known as Database-Driven Product Catalog System (DDPCS) was designed using the proposed technique to provide multilingual support.

2.2.4 Limitations of the approach

The following limitations were observed in his approach.

- Sources/mistakes identified are generic & limited
- Not addressing the Bruce Johnson's Limits-to-Flexibility
- No framework for InfoSys flexibility

2.3 Christopher Ackermann, Mikael Lindall & Greg Dennis Approach (2009)

2.3.1 Abstract

The objective of the software maintenance is to modify it without harming its integrity. Structures that let you do easy and quick modifications are referred to as flexible. Contrarily, rigidity thwarts an effort to synchronize a system to modified requirements or extensions. Christopher Ackermann and co-authors [2] studied architecture of such rigid system inherited from another party. They brought well-known principles of design into play to figure out what to change, improving flexibility, in order to accomplish planned extensions. They shared the lessons learned about flexibility issues experienced along with how to cope with the issues by redesign and reimplementation. The study shows that the same well-established design principles are equally effective to redesign software.

2.3.2 Contribution

Basic & well-established design concepts can be used to guide the design and re-design of software. The detailed description of the changes and reasoning based on basic design principles can be useful when applying redesign to other software systems lacking flexibility. The modifications can also serve as examples of how to prepare software systems for adoption of future changes.

2.3.3 Methodology Used

In this paper, the authors describe how they analyzed a working software prototype referred to as "Tactical Separation Assisted Flight Environment (TSAFE)" which was used as a basis for a software test bed. The rigid system was analyzed with respect to conceptual view, structural view and program flow. Some problems were found to be hurdles in the way of implementing proposed changes. The TSAFE-I was redesigned and re-implemented named as TSAFE-II. The specifications for which was carried out by NASA AMES Research Center where as the development was accomplished by Greg Dennis at MIT. It was a main component of a bigger Automated Airspace Computing system aimed at shifting load from person to computers. The prototype ensures that flights adhere to flight plans. Moreover, predicting "To be trajectories" as well as displaying output on a map as its functions. The two prototypes though were different structurally but enjoyed same behavior and matching graphical user interface.

2.3.4 Limitations of the approach

- Not addressing the Limits-to-Flexibility in InfoSys
- No framework for InfoSys flexibility

2.4 Xiaoping Qiu, Li Tan and Jianbin Chen Approach (2008)

2.4.1 Abstract

Xiaoping Qiu, Li Tan and Jianbin Chen [27] used workflow technology to render the procurement process flexible, enabling today's enterprise to cope with its rapid changes in requirements. A stable data structure, supporting flexible workflow based on reference workflow model, was presented. In order to testify the fact, the author executed an experiment, that workflow technology can unequivocally be used as flexibility enabler in infosys.

2.4.2 Contribution

The author proposed a flexible information structure for the workflow reference model as depicted in *Figure 2-1*. Experiment was carried out to validate the viability of advanced technical idea (stable information structure) for workflow. The workflow was used to render the procurement process flexible enabling it to accommodate emerging process changes in a today's enterprise. When the logic of application and workflow was separated, it allowed user to simply modify a process model rather than to rework the whole system.

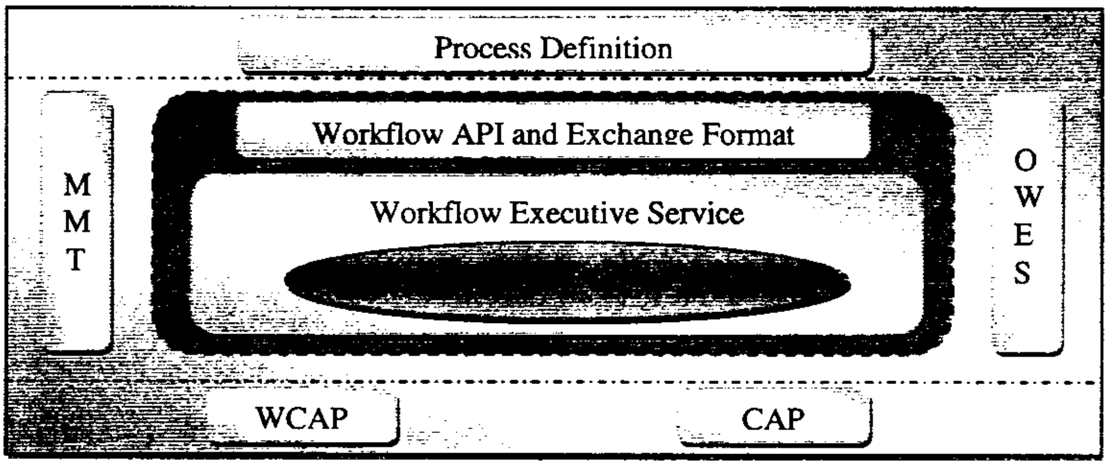


Figure 2-1: A Reference Workflow Model used by Xiaoping Qiu

A stable data structure has been proposed for the reference workflow model illustrated above.

- *Process definition*

  PD (NoID, Name, Description, Version, CreatedTime, ModifiedTime, ParentNo, ValidFrom, ValidTo, StartUpTime, CancelTime,ExceptionalTime)

- *Activity definition*

AD (ProcID, NoID, Name, Description, Type, MaxTime, MinTime, Join, Split, MaxWait, ValidFlag)

- *The activities and roles*

  AR (ProcCode, ActCode, RoleCode, Description)

- *The activity and workflow data*

  AW (ProcCode, ActCode, DataCode, Name, Description, TableName, FieldName, Type, Length, ValidRelation, DefaultValue)

- *The Activity and Application*

  AP (ProcCode, ActCode, AppID, Name, Type, Description, Path)

- *The activity and transition information*

  AT(ProcCode, ActCode1,ActCode2, InforID, Name, Description, IfStatement, Loop)

2.4.3 Methodology Used

Two experiments were carried out. First experiment was carried out for workflow relevant data structure in coordination with a series of workflow-oriented applications duly controlled by workflow engine. The results received were as expected. The workflow engine, supported by workflow relevant data structures, allocates the tasks automatically. User, acting as different role, can assume duty of varying nature on different times. The notions that appropriate user assume a suitable role and can retrieve the relevant tasks any point in time depends largely on how exactly the process logic works.

Second experiment was conducted to fine-tune the procurement process making use of workflow technology. Two procurement processes having varying sequence of activities were taken. It was assumed that activities in the both the processes have the same operation requirements. The successful demonstration implies that applications can be reused.

2.4.4 Limitations of the approach

Although the author has provided a flexible information structure for workflow oriented procurement information system and is good contribution of its own kind, but it hasn't addressed the limits to flexible design marked by Bruce Johnson.

2.5 Bruce Johnson's Group Approach (2002, 2005)

2.5.1 Abstract

An initial set of flexibility characteristics [41] were proposed by this group. They worked on [39] information free identifiers in 2001 and consider it a key to flexible information systems but only talk of making the identifier free of meaning. In fact, keys should be auto-generated, unmarried and numeric one. The aforementioned authors also worked on Generic Entity Clouds [13] in 2001. They claim that GEC [13] is a stable information structure for flexible computer systems. Stable information structure lays down the core and most critical part of the flexible information system. Typical information systems are inflexible, not easily accommodating changes in business requirements. As per Bruce [23], "the ideal system would be one in which changes in system behavior result entirely from business staff modifying data values rather than from IT staff modifying file definitions or program code. In those cases requiring coding changes, they are restricted to local modifications that do not result in a chain of reaction of compensating modifications".

In 2005, Bruce Johnson's group came up with a book on "Flexible Software Design – System Development for Changing Requirements" published by Auerbach Publications [12]. This book includes the author's new as well as old research work.

2.5.2 Contribution

Bruce Johnson's group suggested specific steps that developers can take to achieve system flexibility. Guidelines suggested are as under:

- Conceptualize the system as a whole dynamic entity, not as a set of connected pieces.
- Define and enforce consistent design standards for both technical requirements and the user interface.
- Separate the user interface, business rules, and data (n-tiered architecture)
- Identify logical data entities and entity types, and maintain them as individual objects.
- Eliminate meaning from record keys

- Store business rules as data.
- Code reusable logic in callable routines.
- Develop general purpose, reusable business processes whenever possible.

He furthermore examined the impact of flexible design on business and technical staff. The benefits achieved through the case study by following principle of flexible design were increased business control over system behavior, reuse of system solution and enabling business people to manage change in requirements without or little involvement of IT people.

2.5.3 Methodology Used

A case study was conducted in the university of Cincinnati (UC) needing to replace its old rigid SIS (Student Information System) comprising of un-integrated modules being used by specific business functions for specific processes. The system was redesigned keeping principles of flexibility in view. UniverSIS was implemented. As expected, flexible design of UniverSIS let enhancements to be accommodated in the days to come entirely by business people or with minimum help from development experts.

2.5.4 Limitations of the approach

This work, though uncovers many of the relevant issues, is silent about the flexible information-structure support for InfoSys integration, workflow and life status/history of an entity etc. It also highlights that there is limits-to-flexible design explained in 4.5.10.

2.6 Conclusion of Literature Review

It has been observed, after an extensive survey and contemporary literature review, that although different efforts have been made to render information system flexible by making information structures more stable and change-absorbent. Some offered new techniques to make information structure more dynamic and some focused on a particular aspect like dynamic information-structure support for flexible workflow. Bruce Johnson earmarked the limits to (information-structure oriented) flexible design and no one has tried to break this ceiling effect. As, today, most of the information systems are developed in relational databases, there is a pressing need to overcome this ceiling effect and maximize the area under the curve of information system flexibility.

2.7 Summary

Literature review throws light on the contemporary work about the topic and provides a sort of context to the thesis work. A total of more than seventy research papers were downloaded from various digital research libraries, among which about forty have been referred in the thesis. The more closely related work [1],[2],[11],[12],[14],[23],[27] and [41][40] has been thrown light on - highlighting the author's contribution, methodology used and limitations in the context of the proposed framework. It may be highlighted that references are alphabetically ordered by first character of the author's second name. The chapter is concluded with the pressing need for the proposed work.

FRAMEWORK For STABILITY OF INFORMATION STRUCTURE

3. FRAMEWORK FOR STABILITY OF INFORMATION STRUCTURE

In order to devise proposed framework, like CMMI, architecture of the framework illustrated in Figure 3-1 was established in first place.

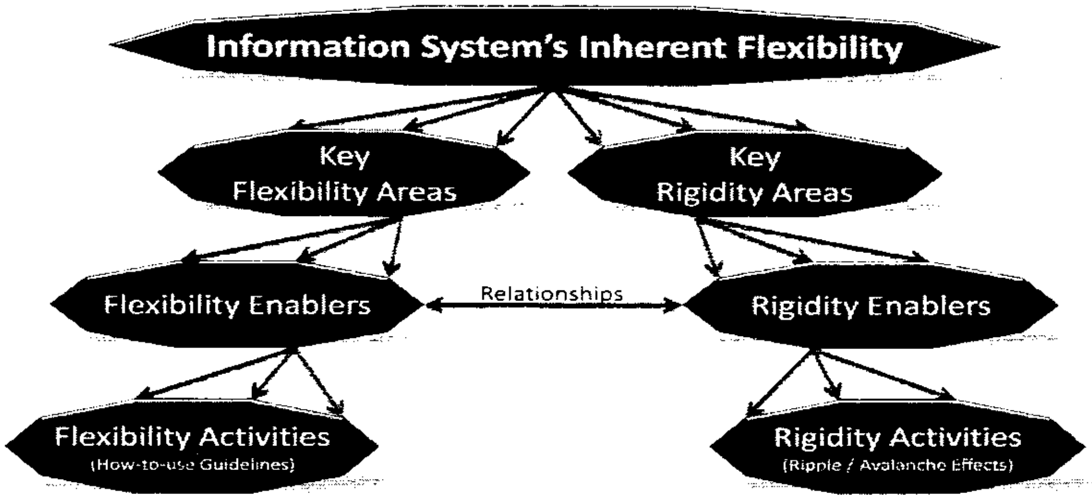


Figure 3-1: Architecture of the Proposed Framework

Based on the literature review and experience, a framework for stability of information structure is presented in Figure 3-2 below:

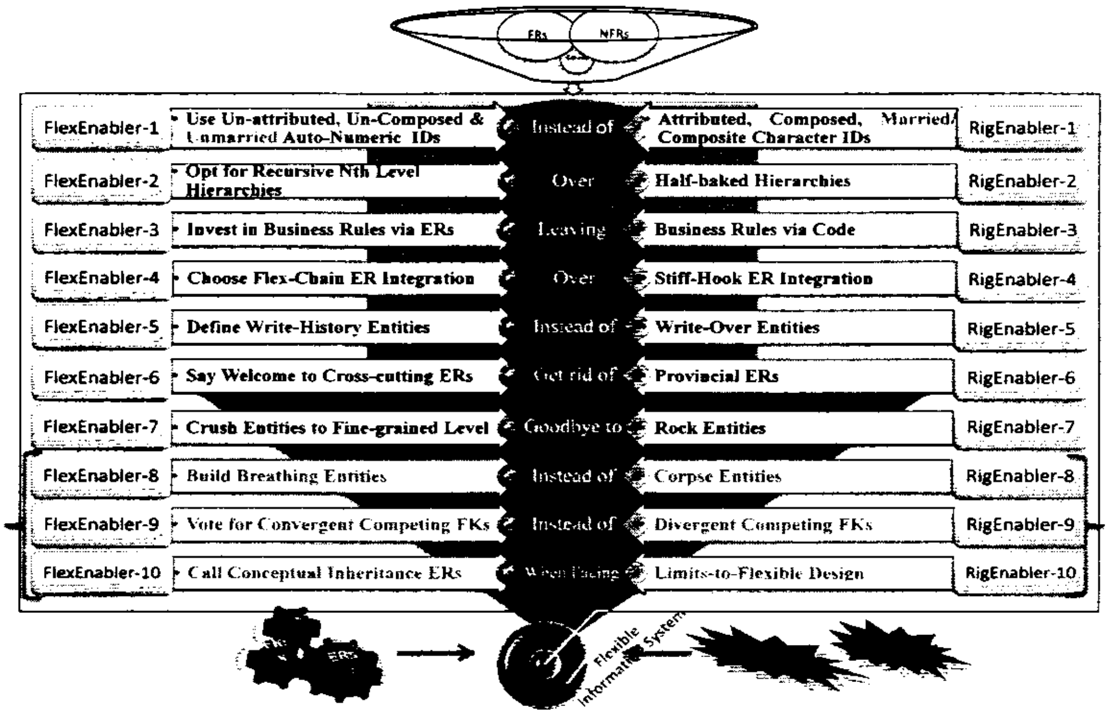


Figure 3-2: Framework for Stability of Information Structure

The framework adheres to this architecture illustrated in Figure 3-1 systemically. The framework comprises of ten pairs of rigidity / flexibility enabling practices and there is one-to-one relationship between them. Rigidity enabling practices are known as ***Rigidity Enablers*** lead you to rigid InfoSys whereas flexibility enabling practices as ***Flexibility Enablers*** that guide you to flexible InfoSys.

ΣRigEnablers = Rigid Information Structure => Fragile Information System.......(1)
ΣFlexEnablers = Flexible Information Structure=>Flexible Information System.....(2)

The provided rigid and flexible enablers resemble the patterns and anti-patterns. Avoiding the rigid one or following the flexible once will lead you to stable information structure and ultimately to flexible information systems. If for any reason someone needs a fragile / rigid information system, the Rigid ERs is the right track to follow.

It also suggests when to use which flexible enabler. In case more than one rigidity enablers are involved, as many relevant flexibility enablers will apply. The framework can equally be used for both engineering as well as re-engineering information system projects. The framework is extensible as and when more rigid and their redress flexible enablers are identified.

3.1 First Pair of Rig-Flex Enablers

3.1.1 RigEnabler–1 (Attributed, Composed, Married Character Identifiers)

Using *attributed identifiers* is rigidity enabler. Moreover this exacerbates the situation when identifiers are defined as character and let users feed-in the identifiers. Modifying identifier is the most costly activity in maintenance because it is used time and again as foreign-keys by other tables of the same as well as other modules. Married key is the combined effect of more than one key enabling someone to locate a record. Married keys do nothing but increase joins.

3.1.2 Generic Chain of Changes RigEnabler–1 Causes

The activities tabulated in Table 3-1 are bound to be carried out, if set of change scenarios - 1 or similar others are applied that proves the existing key to be changeable.

Table 3-1: Generic Chain of Changes caused by RigEnabler–1

Generic Chain of Changes (Attributed, Composed, Married Character Identifiers)	
Ripple Effect	
Following activities will have to carry out, if the existing key proves to be changeable.	
1	Create another field, you deem to be suitable as primary key. Make this a primary key whereas render the old primary key as an ordinary attribute
2	Create additional FK in all tables of module
3	Update all FKs in other tables of the module with new primary key with the help of old one (mapping process).
4	Test the data for integrity /quality.
5	Amend all primary key related functions / routines accordingly.
Avalanche Effect	
Amend all allied systems making use of existing primary key accordingly.	

3.1.3 FlexEnabler–1 (Un-attributed, Un-composed, Un-married and Auto Generated Identifiers)

Oppositely, *un-attributed, un-composed, un-married and auto-generated identifiers* enable flexibility.

3.1.4 How to Use Guidelines

Unattributed /Meaning-free Identifiers

- Make sure that no identifier remains attribute of an entity or in any sense have some business meaning/ value. Tomorrow if not today, attribute(s) of an entity will change, so don't keep an attribute as an identifier no matter how much it is suitable to be an identifier. A rule of thumb is - don't even look for business entity as an identifier.
- A simple series should be used for numbering employees. No matter what the label of key is it must be permanent an unchangeable.

Un-Composed Identifiers

- Make sure that your key is single and not composed of characters from various attributed e.g. tacking first two characters from Personal Number and last two digits of a year.

Unmarried / Un-composite Identifiers

- Find out the composite / married Keys
- Add third numeric field and define it a primary key.
- Treat the composite keys as attributes

- Use the primary key as FK whenever required

Auto-Generated and Numeric

- Auto-generated Keys does not give chance to user to play havoc with keys – thus churning out maintenance work for programmer.
- Promote only numeric keys. Character keys can also be auto-generated but go for numeric only.

3.2 Second Pair of Rig-Flex Enablers

3.2.1 RigEnabler-2 (Half-baked Hierarchies)

The notion to address only current requirements drives the software engineers to define entities e.g. organizational structure like division and departments as fields of table. Later on, when organizational matures or expands as a result new level(s) in org: hierarchy is introduced e.g. subsections or groups. Especially when company undergoes a process of reorganization, some departments/divisions are merged, others are created as new and some of old ones no longer exist. Later on, when organizational matures or expands as a result new level(s) in org: hierarchy is introduced e.g. subsections or groups. Especially when company undergoes a process of reorganization, some departments/divisions are merged, others are created as new and some of old ones no longer exist. As organizations change, so the system's requirements change. As significant parts of these requirements lie in the time to come hence obtainable by the time the InfoSys is designed [1].

Usually, in information systems, no due care is taken in appreciating the hierarchical business entities and designed horizontally as attributes of an entity. This becomes a potential earth quack center and shakes everything creating ripple effect within and avalanche effect across modules. Another example is approval chain. This chain may expand or contract as per the criticality or importance of the approval process or entities involved. This rigid structure may be referred to as *Half-baked Hierarchies*.

3.2.2 Generic Chain of Changes RigEnabler–2 Causes

Each time new layer is added in hierarchical structure, the chain of activities bound to be carried out in case rigid half-baked hierarchies are in place is tabulated in Table 3-2.

Table 3-2: Generic Chain of Changes caused by RigEnabler–2

Generic Chain of Changes (Half-baked Hierarchies)	
Ripple Effect	
Each time new layer is added, the following activities will have to carry out.	
1	Create table(s) for each new layer of the hierarchy and user interface for the purpose.
2	Create a new field in the transactional table(s) for accommodating foreign key. Even then no link in various layers will exist e.g. Which section works under which department.
3	Write new queries / triggers / routines for new features involving the new type
4	Amend the existing queries / triggers / routines etc involving new type especially the summary reports [Adding table/attributes names].
5	Quality of data adversely affects because of no link between various layers.
Avalanche Effect	
All allied systems making use of existing hierarchy will have to amend the integration routines by adding new tables(s) / attributes if the new layer is involved	

3.2.3 FlexEnabler -2 (Nth Level Recursive Hierarchies)

This, one of the potential sources of future changes, can be eliminated by structuring hierarchy recursively i.e. interlinking different levels with its parent by putting a “ParentID” field in the same table. They may be referred to as *Recursive Nth Level Hierarchies*.

3.2.4 How to Use Guidelines

- Define all business attribute of an entity in first place.
- Define a specific attribute of ParentID pointing to its parent instance of an entity. Remember don’t define child attribute.
- Include Type/Nature field to differentiate parent-child.
- Add the following fields to make it breathing.
    - ValidFrom, ValidUpto, Living Status, Sequence.

3.3 Third Pair of Rig-Flex Enablers

3.3.1 RigEnabler -3 (Business Rules via Code)

It’s the business rules that drives the system whether manual or automated. InfoSys remains alive as long as it is synchronized with the business rules. Business rules are susceptible and more prone to change. Business rules go unchanged for a limited duration. They get changed or new rules replace it or vanished without replacement. The notion to code everything is one of the major sources of rigidity. If these rules are totally encapsulated in code, such scenarios are indicators of inflexibility and contribute to generate major portion of the maintenance and are termed as *Business Rules via Code.*

3.3.2 Generic Chain of Changes RigEnabler–3 Causes

Each time a change in rules happen, following problems transpires, in turn some activities will have to carry out as shown in Table 3-3.

Table 3-3: Generic Chain of Changes caused by RigEnabler–3

Generic Chain of Changes (Business Rules via Code)	
Ripple Effect	
Each time a change in rules happen, following problems emerge, in turn some activities will have to carry out.	
1	**Access code and amend the code to reflect the change in rules.**
2	Still one does not know which rule was applied for which duration.
3	**Calling the routines supposed for business rules processing, instead of storing its results have its own ripple effect. Sometimes brings the performance beyond the unaffordable levels.**
4	Emergence of new rules even of similar patterns means amending code that requires programmer's involvement.
5	**Test Data for Integrity / Quality.**
Avalanche Effect	
In this case an avalanche effect of little magnitude is involved as no, rigid, ERs is involved	

3.3.3 FlexEnabler-3 (Business Rules via ER)

Contrarily, well-designed information structure for business rules provides end-user much flexibility and may be called as *Business Rules via ERs.*

3.3.4 How to Use Guidelines

- Understand business rules.
- Make pattern of business rules.
- Identify parameters to fine-grained level.
- Make relationships keeping business rules in view.

3.4 Fourth Pair of Rig-Flex Enablers

3.4.1 RigEnabler - 4 (Stiff-Hook ER Integration)

No business function can operate in space. Output of one becomes input for other which provides a reason for system/module integration. In MIS setup, where each developer is responsible for development/maintenance of his assigned module, each developer is concerned with the smooth operation of his/her module and less careful about others. They use the data of each other by just picking the desired data directly and similarly drop data by directly into the relevant tables/fields of other modules. In case an information structure changes later on while other modules are hooked. The result will be dysfunctional systems. This may be termed as *Stiff-Hook integration.*

3.4.2 Generic Chain of Changes RigEnabler–4 Causes

A change in information structure causes the allied modules including self go down. What's in the way is tabulated in Table 3-4.

Table 3-4: Generic Chain of Changes caused by RigEnabler–4

Generic Chain of Changes (Stiff-Hook ER Integration)	
Ripple Effect	
A change in Shared Information Structure (SIS) causes the allied modules to go down.	
1	Aforementioned changes take developer of the allied even of the same module by surprise. The developer has to find out where the problem lies.
2	Amend the integration code to synchronize it with the amended information structure.
3	Test the code for integration.
4	Bring the module(s) including the allied one up.
Avalanche Effect	
The frequency of change in common information structure and the number & nature of integration defines the magnitude of the ripple or avalanche effect.	

3.4.3 FlexEnabler - 4 (Flex-Chain ER Integration)

In fact, the module itself should be responsible to provide the desired data requested by other module and also for keeping the data providing objects synchronized when any of its information structure changes. This may be termed as *Flex-Chain Integration.*

3.4.4 How to Use Guidelines

- As a rule of thumb, never permit others to select, update, save etc in tables of your module(s). Make sure, these operations be performed by the module itself. Allow other only to access module's views, interfaces, functions etc for the purpose.
- Analyze, design, implement and make it available for others to use.
- Make sure don't update, save, and access information structure of other module directly.

3.5 Fifth Pair of Rig-Flex Enablers

3.5.1 RigEnabler-5 (Write-Over ERs)

When an entity is structured in a way that important old information are updated whether configurational or transactional is a source of changes, later or sooner user will demand for historic information not available. This leaves the user with no option to know the history. This may be referred to as *Write-Over ER.*

3.5.2 Generic Chain of Changes RigEnabler - 5 Causes

Table 3-5: Generic Chain of Changes caused by RigEnabler–5

Generic Chain of Changes (Write-Over ERs)	
Ripple Effect	
Each time user updates the old information with new one, substantial irreversible information get lost.	
1	Even if ER is changed by adding additional tables to retain history, old lost information can't be retrieved.
2	Write-Over ER does not lend itself to prepare "When happened What" reports.
Avalanche Effect	
•	Other allied modules starts saving relevant historic info with them. This is can be referred to as 'scattered pieces of history' not linked to make sense.
•	Moreover, in case ER is changed to accommodate historic info, allied modules will have to be changed accordingly.

3.5.3 FlexEnabler-5 (Write-History ERs)

For keeping such information this point onward you have to do major structural changes. Such structures may be referred to as *Write-Over* whereas the opposite as *Write-History* ERs.

3.5.4 How to Use Guidelines

- As a rule of thumb, use Write-History ERs when change in data is expected or visible even if it's not the user's requirements.
- One can maintain history in many forms. Select one of them depending on the scenario or criticality of the historic information.

3.6 Sixth Pair of Rig-Flex Enablers

3.6.1 RigEnabler-6 (Provincial ERs)

In "Get things done on the fly" environment, a little attention is given to what could be reused to avoid rework and so waste of energy and efforts. In the absence of centralized technical control, developers oblivious of what is going on around starts reinvent what has already been accomplished by another colleague. This may be termed as "*Provincial ERs*".

3.6.2 Generic Chain of Changes RigEnabler - 6 Causes

Each time a new entity with similar nature is needed, the activities bound to carry out is tabulated in Table 3-6.

Table 3-6: Generic Chain of Changes caused by RigEnabler–6

Generic Chain of Changes (Provincial ERs)	
Ripple Effect	
In Rigid System, each time a new entity with similar nature is needed, the following activities will have to carry out.	
1	Create /reinvent similar ER for the purpose.
2	Develop user-interface for the reinvented ER.
3	Scattered pieces of reinvented ERs do not lend itself to prepare cross-cutting summary reports.
4	Especial additional efforts are required each time new reinvented ER is added for amending the summary routines, triggers, procedures etc will have to be changed.
5	For a generic change introduced later on, all reinvented ERs will have to be amended.
6	More than n x time's efforts will be required to transform all reinvented ERs into uniform one.
Avalanche Effect	
In this case an avalanche effect of little magnitude is involved as each module has its own reinvented ERs. However in case of a generic change, nec amendments will have to carry out in each module.	

3.6.3 FlexEnabler-6 (Cross-Cutting ERs)

Watch out for recurring and cross-cutting ERs which can be shared across MIS, design it thoroughly once and for all. These can be called as *Cross-cutting ERs.*

3.6.4 How to Use Guidelines

- Establish a centralized organizational control over design activity as well as incorporation of change in the same.
- Monitor the overall information structure and separate cross-cutting ERs.
- Design, develop generic modules and enforce all to use the same.

3.7 Seventh Pair of Rig-Flex Enablers

3.7.1 RigEnabler-7 (Rock ERs)

Defining *rock entities* is another source of potential changes. For example putting complete address of an employee into one field of database provides reason for clients to raise change requests in the days to come.

3.7.2 Generic Chain of Changes RigEnabler-7 Causes

Table 3-7: Generic Chain of Changes caused by RigEnabler–7

Generic Chain of Changes (Rock ERs)	
Ripple Effect	
Transforming rock entity into fine-grained one takes the following:	
1	Split the rock entity / attribute into fine-grained entity / attributes.
2	Create new fields to accommodate the fine-grained attributes of an entity instance.
3	Configure the fine-grained attributes and link it with main entity instance by FKs.
4	Make hierarchies, if nature of entity suggests so, and link it with transactional table.
5	Populate the existing data by mapping process i.e. updating relevant FKs.
6	Re-engineer the rock entity by concatenating the constituent attributes.
7	Validate the data for its integrity / quality.
8	Amend the existing routines / views / procedures / triggers / reports accordingly
Avalanche Effect	
Amend all the allied modules to be compatible with the fine-grained entity instance.	

3.7.3 FlexEnabler-7 (Fine-grained ERs)

Generally a whole entity can be fabricated by concatenating its constituent pieces but not the vice versa. So *crush the rock entities into more fine-grained attributes* provide you more flexibility in searching and reporting and also storing it in concatenated form not let performance go down.

3.7.4 How to Use Guidelines

- Don't rush for coarse-grained entity design. Test every entity you design for atomicity.
- Crush an entity to fine-grain level if it lends itself for subdivision.
- Concatenate all constituent fine-grained attributes and store it as a whole again in the same table as rock field.
- Use flexibility enabler - 2 in case it forms bill-of-material like structure.

3.8 Eighth Pair of Rig-Flex Enablers

3.8.1 RigEnabler-8 (Corpse ER)

Definition of entities does not remain valid for ever. They live their life. They born and die like living things. They get changed. Entities with unknown living status may be termed as *Corpse Entities*. Moreover, this is a redefinition of what entity is also not traceable. They are referenced in transactional tables and must not be deleted or changed.

3.8.2 Generic Chain of Changes RigEnabler-8 Causes

Table 3-8: Generic Chain of Changes caused by RigEnabler–8

Generic Chain of Changes (Corpse ERs)	
Ripple Effect	
Spraying out the smell of corpse and transforming it into breathing entity takes:	
1	**Find out what old definition of an entity is being displayed with the new one.**
2	Preserve the old entity definition by reverting back the old configuration.
3	**Redefine the entity instance (revised entity) by creating new master record.**
4	Find out & amend the FKs of all transactional tables where the entity instance is referred. Nested Ripple Effect – as separating affected old & new records is difficult.
5	**Also add DateFrom / DateTo and LivingStatus attributes for traceability.**
6	Link the revised entity with old entity definition.
Avalanche Effect	
All allied modules integrated will have to amend the FKs wherever referred.	

3.8.3 FlexEnabler-8 (Breathing ERs)

Oppositely, the entities with its birth, death dates as well as living status may be termed as *Breathing Entities*.

3.8.4 How to Use Guidelines

- Add the following fields in all entities generally and in configurational files especially.
  - Sequence, ValidFrom, ValidUpto, LivingStatus, Linkedwith
  - Fill / update the ValidFrom, ValidUpto fileds automatically with system dates.

3.9 Ninth Pair of Rig-Flex Enablers

3.9.1 RigEnabler-9 (Diverging Competing FKs)

In case when more competing foreign keys are coming from different tables, usually they are kept separate. This may be termed as *Divergent Competing FKs*.

3.9.2 Generic Chain of Changes RigEnabler-9 Causes

Each time new FK is added because of adding new entity type i.e. table, the following activities will have to carry out.

Table 3-9: Generic Chain of Changes caused by RigEnabler-9

Generic Chain of Changes (Divergent Competing FKs)	
Ripple Effect	
Each time new FK is added because of adding new entity type i.e. table, the following activities will have to carry out.	
1	Create table(s) for the new type of entity.
2	Develop User-interface for the newly added entity.
3	Add a FK for the newly added entity into the relevant transactional table.
4	Amend the user interface to accommodate the newly added entity.
5	Amend the existing queries / triggers / routines etc involving newly added entity.
6	Write new queries / triggers etc for new features involving the newly added entity
Avalanche Effect	
All allied modules integrated with this module, will have to amend the integration routines by considering tables(s) / attributes of new entity wherever needed.	

3.9.3 FlexEnabler-9 (Converging Competing FKs)

Table name as identifier of the foreign key may minimize he the number of FKs. This may be termed as *Convergent Competing FKs*.

3.9.4 How to Use Guidelines

- Look for other competing FKs whenever you making a relationship via FK.
- Keep all tables in schema with necessary info in a separate table
- Reduce the number of FKs by combining competing FKs.
- Add a filed named "TableID" to locate the origin of each FK.

3.10 Tenth Pair of Rig-Flex Enablers

3.10.1 RigEnabler-10 (Bruce Limit's to Flexible Design)

When emerging types of entities have varying information structure, designing it necessitates adding more tables, fields as well as new user interfaces. This is, as mentioned in problem statement, is called *Bruce Johnson's Limits to Design Flexibility*. Actually, a concept drift occurs when a new type emerges for an entity.

3.10.2 Generic Chain of Changes RigEnabler-10 Causes

Each time new type emerges with different attributes, the activities to carry out is tabulated in Table 3-10.

Table 3-10: Generic Chain of Changes caused by RigEnabler–10

Generic Chain of Changes (Bruce Limits to Flexible Design i.e. Ceiling Effect)	
Ripple Effect	
Each time new type emerges, the following activities will have to carry out.	
1	**Create table(s) for the new type of the existing entity.**
2	Develop User-interface for the new type.
3	**Write new queries / triggers / routines for new features involving the new type**
4	Amend the existing queries / triggers / routines etc involving new type especially the summary reports [Adding table/attributes names].
5	**Scattered pieces of ER doe not lend itself to prepare Cross-cutting Summary reports.**
Avalanche Effect	
All allied modules making use of existing type entity will have to amend the integration routines by considering new tables(s) and their attributes where newly introduced type is involved	

3.10.3 RigEnabler-10 (Conceptual Inheritance in ER)

The Johnson's limits-to-flexible design can be overcome by adopting a coordinated hybrid approach by InfoSys & business domain experts of the organization and employing *Conceptual Inheritance*. Provision of linked tables by developers for additional fields and dynamically specifying on the base of type selected where to save additional fields of a particular entity of new type. Invoking relevant table and interface dynamically at runtime necessitates keeping record of all tables in schema and interface artifacts i.e. forms. This definitely increases the complexity and calls for expert programming skills. However, as it is one-time effort but provides flexibility at user-end as well as makes developer's life easier in the time to come.

3.10.4 How to Use Guidelines

- You may go for conceptual inheritance when you are dealing with types, subtypes that form hierarchical shape.
- Evaluate the efforts involved in realizing the conceptual inheritance against the flexibility required.
- Keep all tables of schema with necessary information in a table.
- Keep all interface artifacts i.e. forms, reports etc of all modules in with necessary information in a separate table.
- Put two fields SubtypeID and TableID in each table of hierarchy to locate where further information about a particular type are stored.
- Use the same linking path for retrieval of relevant information.

3.11 Limitations of the Proposed Framework

The framework has the following limitations:

- Limited to InfoSys only
- Addresses stability of information-structure only
- More ER-based flexibility results in more program complexity. There is a pressing need to strike a delicate balance between the two.
- Not tested for other than MIS Setup.
- Not meant for web-based Information Systems.

3.12 Applicability of Flexibility Enablers

There is one-to-one relationship between rigidity enablers and flexibility enablers. This implies that choose flexibility enabler when relevant rigidity enabler is involved. In case more than one rigidity enablers are in action, as many relevant flexibility enablers will apply e.g. solving the Bruce Johnson's limits to design flexibility (ceiling effect) involves, by design, the Nth level hierarchy enabler besides conceptual inheritance itself.

Though the framework may partially be used as a guide while engineering IS, it applies specifically *how to re-engineer (transform) rigid 'IS' into a flexible one.* Furthermore, the name of the flexibility enablers also suggest the situation / scenarios when to apply one or more flexibility enabler(s). However, based on experience, when a need to call another enabler arises is illustrated below in Table 3-11:

Table 3-11: Applicability Relationship among Enablers

Enablers	RigEnab-1	RigEnab-2	RigEnab-3	RigEnab-4	RigEnab-5	RigEnab-6	RigEnab-7	RigEnab-8	RigEnab-9	RigEnab-10
FlexEnab-1	★	★	★	★	★	★	★	★	★	★
FlexEnab-2		★	○		○	○	○	○		★
FlexEnab-3			★			○	○	○		
FlexEnab-4			○	★	○	○	○	○		
FlexEnab-5		○	★		★	○	★	○		
FlexEnab-6			○			★	○	○		
FlexEnab-7		○	○		○	○	★	○		
FlexEnab-8		○	★		★	○	★	★		
FlexEnab-9									★	
FlexEnab-10										★

★ **Shall Call relevant FlexEnabler** ○ **Might Call the relevant FlexEnabler**

3.13 Impact of Flexibility Enablers on Quality of Data

Data populated in the re-engineered stable information structure, which was designed keeping flexibility enablers in view, is appraised against the following quality parameters by experts and drawn in Table 3-12. The purpose of evaluation is to determine the quality (positivity / negativity) of the impact and not the quantity of the impact. The table shows that flexibility enablers have overall positive impact on data quality.

Table 3-12: Impact of Flexibility Enablers on Quality of Data

Enablers/ Data Quality	FlexEnab-1	FlexEnab-2	FlexEnab-3	FlexEnab-4	FlexEnab-5	FlexEnab-6	FlexEnab-7	FlexEnab-8	FlexEnab-9	FlexEnab-10
Data Standardization	*	+	+	+	+	+	++	+	+	+
Data Integrity	+	+	*	+	+	+	+	+	-	*
Resilience to Wrong Data Entry	+	+	+	+	+	*	+	++	*	+
Amenability to Cross-Cutting Reports	*	++	*	+	+	++	*	*	+	++
Data Relationships	+	+	+	*	+	+	+	+	+	+
Supportive to Data Warehousing	+	-	+	*	++	+	-	++	*	*
Accessibility of Data	++	+	+	+	+	+	++	+	+	-
Completeness of Data	+	++	+	*	++	*	*	*	*	*
Amenability to Analysis	*	+	+	+	+	+	+	+	+	-

Legend:-> + Positive Impact, ++ Double Positive Impact, - Negative Impact, * No Impact

3.14 Impact of Flexibility Enablers on Program Complexity

It is highlighted that program complexity has been analyzed from software implementation point of view. The nature of parameters evaluated is indicative of the same. The results, drawn in Table 3-13, are based on the case study data thoroughly judged by experts. The overwhelming negatives in the table show that inherent flexibility increases the program complexity. However, the experience demonstrates that this complexity decreases gradually as the programmer's programming skills sharpens. The objective here is to determine the quality (nature) of impact and not to measure the quantity of impact i.e. program complexity.

Table 3-13: Impact of Flexibility Enablers on Program Complexity

Enablers/ Data Quality	FlexEnab-1	FlexEnab-2	FlexEnab-3	FlexEnab-4	FlexEnab-5	FlexEnab-6	FlexEnab-7	FlexEnab-8	FlexEnab-9	FlexEnab-10
Keys Handling	+	-	*	*	-	+	-	*	-	--
Nested Structure	+	--	-	*	-	-	-	-	-	--
Depth of Queries	+	--	-	*	-	-	-	-	--	--
Report Generation	*	-	*	-	-	+	+	+	-	--
Entity Structures	+	-	-	*	-	-	-	*	-	--
Code Size	+	+	-	*	-	+	-	-	-	*
Algos	+	-	-	*	-	-	-	-	-	-

Legend:-> + Simplicity, - Complexity, -- More Complexity * Normality

3.15 Summary

This chapter presents the proposed framework, comprising of ten pairs of rigidity & flexibility enablers, for stability of information structure. The framework depicts one-to-one relationship between rigid and its relevant flexibility enabler. Generic chain of changes is tabulated for rigidity enabler concerned whereas "How-to-Use" guidelines are provided with each flexibility enabler. Rigidity and flexibility enablers are explained with examples. The chapter concludes with Limitations of the framework, applicability of the flexibility enablers and impact of the same on both data quality & program complexity.

Chapter - 4

FlexInfoSys

A CASE STUDY

4. FLEXINFOSYS – A CASE STUDY

In this chapter the implementation details of our research work has been discussed. Case study has been carried out to appraise the effectiveness of the proposed framework. Two change-averse legacy InfoSys have been chosen for study of rigidity. We rename these systems as RigInfoSys, as non-disclosing measure. One belongs to Govt. organization whereas the other taken from Private Ltd company. Name of the organization/company is also not being made public owing to non-disclosure agreement. One of the rigid systems, covering the scope, has been re-engineered, named as FlexInfoSys, making use of the proposed change-driven information-structure-oriented design strategies. The reason for selection of RigInfoSys is that all its modules are integrated within as well as with other business modules. Moreover, the Employee Profile InfoSys abbreviated as eProfile is the core module acting as a foundation for most of the modules comprising the bespoke small ERP.

4.1 What is Case Study?

As per Robert K. Yin "The case study research method is an empirical inquiry that investigates a phenomenon under scrutiny, within its real-life context; when the boundaries between phenomenon and context are not clearly drawn; and in which multiple sources of evidence are used". It brings to light actual story what led to the results or outcome. It is useful for testing whether scientific theories and models really work in the real world environment. Suppose a scientist or engineer formulate a wonderful computer model explain how the ecosystem of a rock pool works. It is case study which is used to try it out on a real life pool to see its effectiveness. Case study is preferred to be use for its flexibility. Sometimes, while trying to testify a hypothesis, it brings about unusual results thereby leading to new directions or open new dimensions.

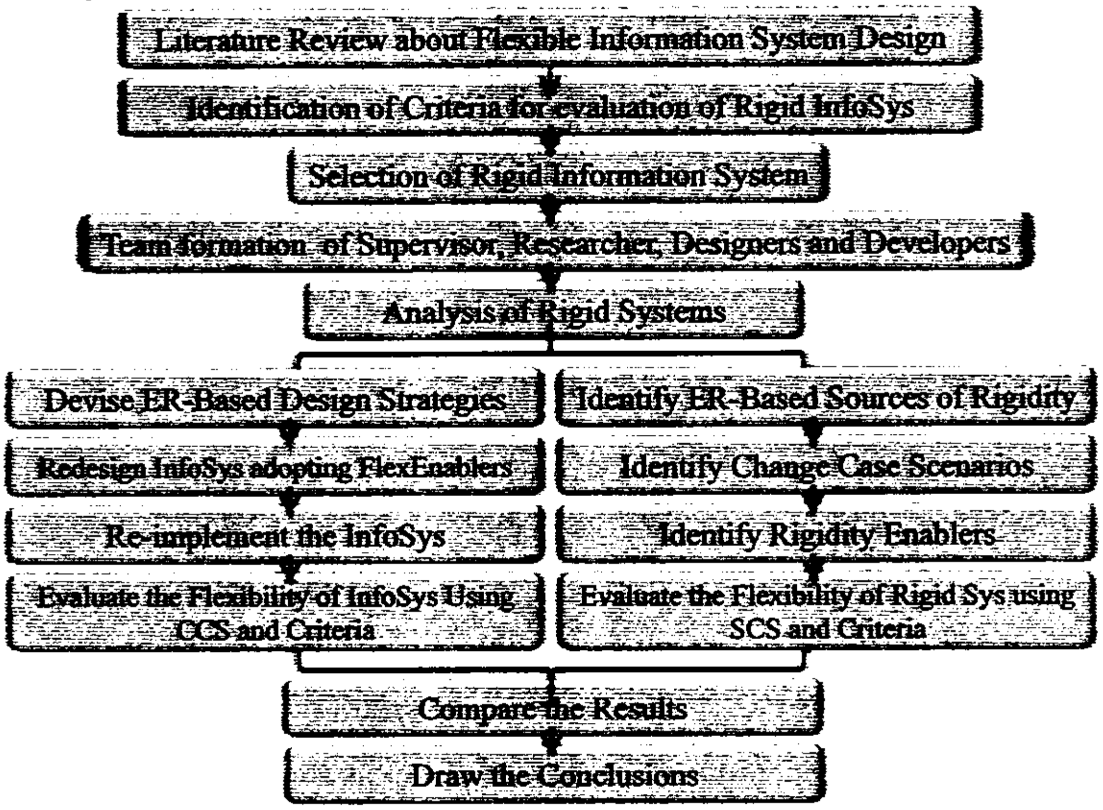
4.2 Rationale for Selection of Case Study as a Research Methodology

Case study [30], [33] is appropriate methodology for research in software engineering as it provides an opportunity to investigate phenomena in its real life context. Basically, this study is aimed at analyzing the common Rigid ER Practices and proposes a framework for stability of information structure to infuse flexibility in information systems. Generally, software engineering and especially design of information systems takes place for a given business or industry having customer's / sponsor's requirements or expectations including certain assumptions and constraints on software / information system design. It was also not appropriate to establish a lab environment, so experiment research methodology was not considered for this research work.

The other strong candidate research methodology was to conduct a survey. Because the survey is a non-experimental, descriptive research method, conducting survey can be useful when data needs to be collected on phenomena that cannot be directly observed. Last but not least, the most of the closely related research work has been carried out as case studies. The literature review in Chapter-2 testifies the same.

4.3 Main Activities of Case Study

The major activities are illustrated in the Figure 4-1 as under:



Literature Review about Flexible Information System Design
Identification of Criteria for evaluation of Rigid InfoSys
Selection of Rigid Information System
Team formation of Supervisor, Researcher, Designers and Developers
Analysis of Rigid Systems
Devise ER-Based Design Strategies
Identify ER-Based Sources of Rigidity
Redesign InfoSys adopting FlexEnablers
Identify Change Case Scenarios
Re-implement the InfoSys
Identify Rigidity Enablers
Evaluate the Flexibility of InfoSys Using CCS and Criteria
Evaluate the Flexibility of Rigid Sys using SCS and Criteria
Compare the Results
Draw the Conclusions

Figure 4-1: Case Methodology Used

4.4 Change Scenarios

Change scenarios are generally used to test the rigidity or flexibility of a system. Following change scenarios have been designed based on a decade experience and in-depth analysis of change requests raised by end-users in real business environment. These scenarios relate to problems faced by business people and/or developers over a decade period. Careful selection ensures that most of InfoSys changes fall in one of the scenarios. These scenarios are used in conjunction with the Bruce Johnson's criteria to judge the effectiveness of the proposed framework. These change scenarios are interspersed in the case study with relevant rigidity / flexibility enablers.

4.5 Change-Driven ER-Oriented Rigidity Enablers and their Redress Flexibility Enablers.

The case study was carried out taking various steps in its logical order as depicted in Figure 4-1. However the two enablers i.e. the rigidity enablers identified while analyzing the RigInfoSys and the Flexibility Enablers employed in re-engineering FlexInfoSys are given under one heading for increased understandability and ease of comparison. The chains of changes i.e. ripple & avalanche effects have been shown with relevant rigidity enabler.

4.5.1 Use Non-Attributed, Un-Married Auto-Generated Numeric Keys instead of Attributed, Composed Character Keys

Usually IDs are composed to form IDs of entities. In eProfile, personal number was used as primary key in first place. Later on, a new business rule that a new personal number will be allotted to an employee on re-recruitment and also once staff employees goes into officer category. This led to change the ID. Next time, a national identity card was selected as a primary key with the assumption that it is not going to change. This is depicted in Figure 4-2. Later on, National Database Registration Authority (NADRA) introduced a new computerized Cards referred to as CNIC leading to the same situation. This suggests that no information/attribute is exception to change.

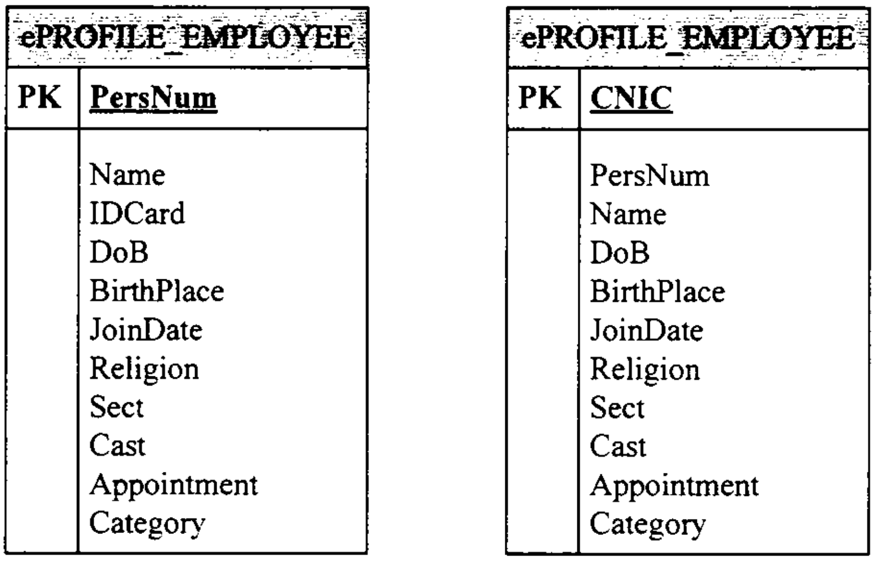


Figure 4-2: Use of Attributed, Composed Character keys

If the change scenarios tabulated in Table 4-1 or similar others are applied on aforementioned ERs, the generic chains of changes i.e. ripple and avalanche effects given in Table 3-1 unravel. The more specific effects are certainly more deep and adverse.

Table 4-1: Set of Change Scenarios - 1

	Change Scenarios for Evaluating First Pair of Rig-Flex Enablers (Attributed & Composite VS Non-Attributed & Non-Composite Keys)
SCS-1(A)	**A business rule that two different series of service-ids are maintained. When an employee promotes from working to management position, its service-id is changed and next number of the series is allotted to an employee as service-id**
SCS-1(B)	When an employee is re-recruited in the same company e.g. a regular officer is re-recruited as contract employee; he/she will be allotted a new service-id whereas service-id is being used as primary key in the RigInfoSys.
SCS1-(C)	**Govt. decides to Computerize National Identity Cards. Govt. asks Cards providing authority to provide computer generated new IDs for its citizens whereas old NIC is currently being used as primary key.**

Bruce Johnson group [12], [23], [39] and Asma Alkalbani [1] have worked a lot on this issue. It is further revealed from experience that single un-composed auto-generated numeric keys should be encouraged because of performance issues. Moreover, composite keys should be discouraged. Here EmpID is a numeric key other than employee personal number shown in Figure *4-3*.

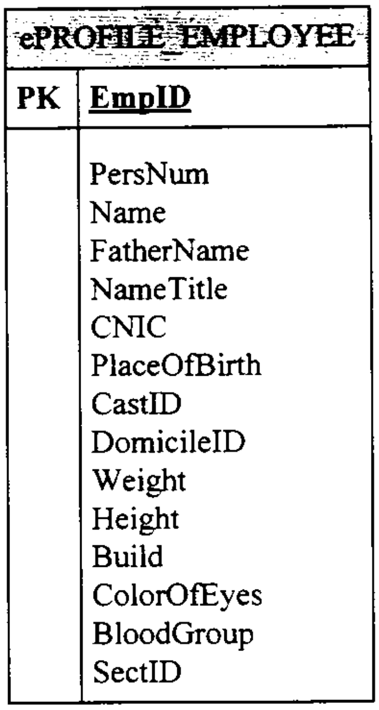


Figure 4-3: Non-Attributed, Un-Married and Auto-generated Numeric keys

```
FUNCTION NonAttributedAutoNumericKey(vTargetTable VARCHAR2, vTargetColum,
VARCHAR2) RETURN NUMBER IS
    KeyMaxID NUMBER:=1;

    Query_Str VARCHAR2(2000);
  BEGIN
    Query_Str := 'select nvl(max('||vTargetColum||').0) +1 from '||vTargetTable;
    execute immediate Query_Str into KeyMaxID;
    RETURN(KeyMaxID);

  END;
```

Text Box 4-1: A generic function for auto-generation of next non-attributed numeric key. Use of database provided identifier has been found even better than the above mentioned function.

4.5.2 Invest in Nth Level Hierarchies and Leave Half-baked Hierarchies

In RigInfoFlex, the Organizational structure i.e. the Directorates and Branches and regions as City, District. Province and Country as in *Figure 4-4*:

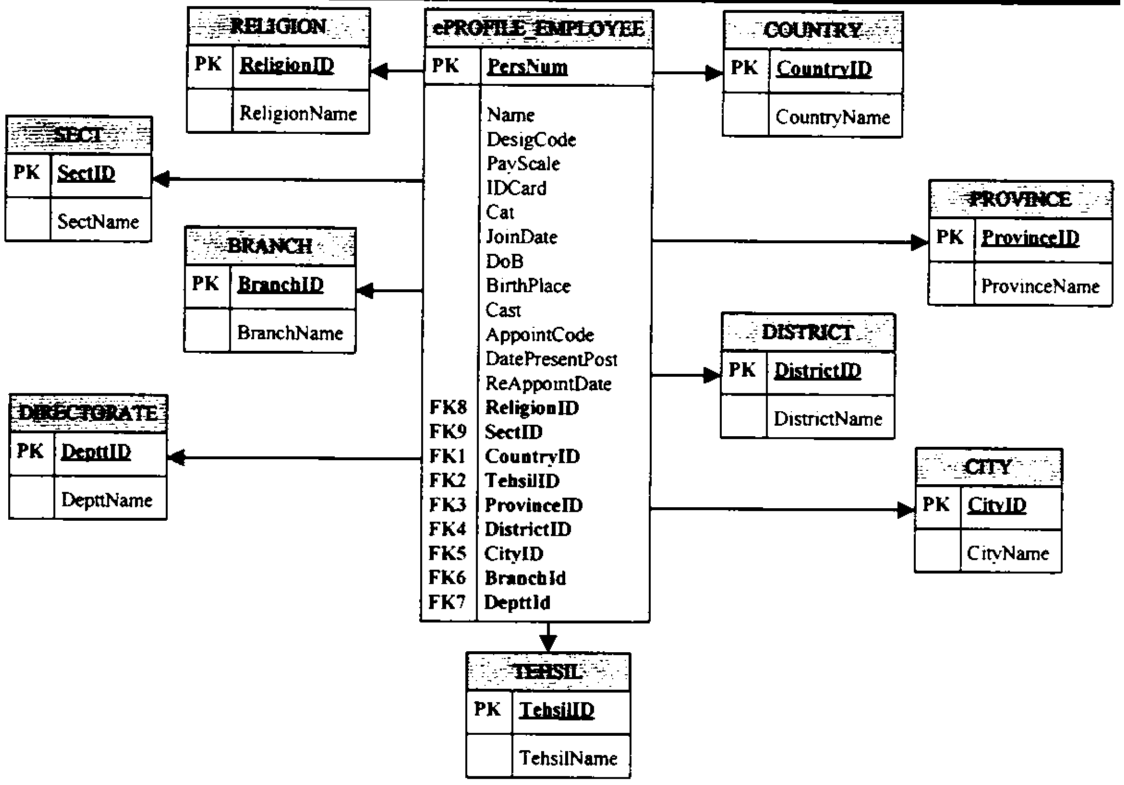


Figure 4-4: Defining Half-baked Hierarchies

These hierarchies were used by other modules. Other operations were critically dependent on it hence got disrupted and chain of unrequested amendments started unraveling.

If Set of Change Scenarios – 2 tabulated in Table 4-2 or similar other scenarios are applied, the chain of changes illustrated in Table 3-2 unravel.

Table 4-2: Set of Change Scenarios - 2

Change Scenarios for Evaluating Second Pair of Rig-Flex Enablers (Half-baked VS Recursive Hierarchies)	
SCS-2(A)	**Country "X" is divided into Provinces, Districts, Tehsils and Cities. District 'D' originally in one province is remapped with another province of the country.** **[Note: -The system should be able to provide the current as well as old organizational structure. A particular employee was working in which department or section in a specific period even if the department no more exists]**
SCS-2(B)	Company "Y" is organizationally structured as Divisions and Departments. As new CEO takes over, the company undergoes reorganization. New layer of Sections and/or Groups are added. Some departments are split and others merged. New departments are introduced and some old ones are dissolved . [Note:- Other business modules integrated with this core module should not be disrupted e.g. a user of an inventory module accesses employee of a section]
SCS-2(C)	**Annual Confidential report writing involved initiation by OIC and countersigned by relevant director. New policy dictates that CEO will review the outstanding reports as a second counter-signee.**

In FlexInfoSys, the organizational structure, regions and religions given in Figure 4-5 were designed as depicted in following diagram. Workflows and chart of accounts are among other examples.

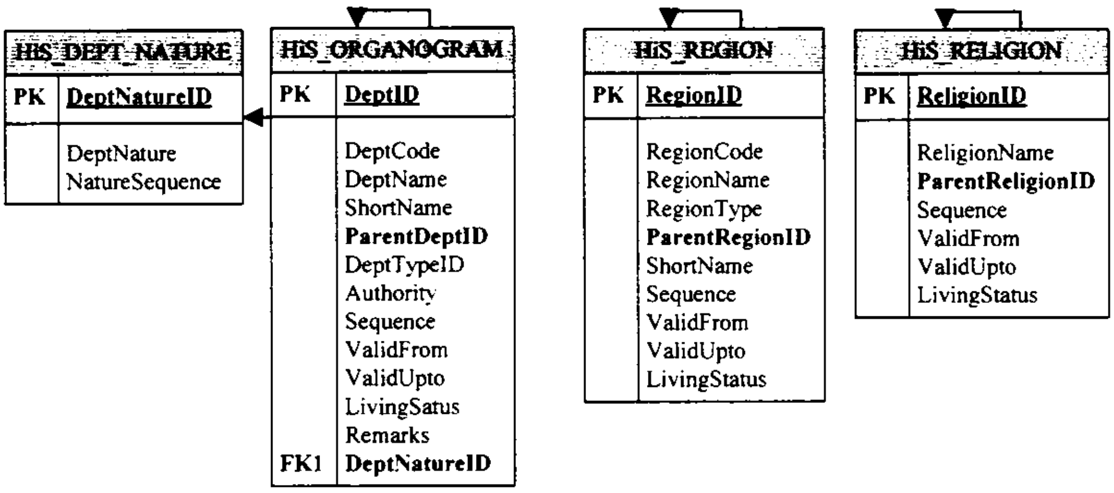


Figure 4-5: Recursive definition of Hierarchies – Organogram, Religion, Region

4.5.3 Leave the Rigid Way of Achieving Business Rules via Code for more Flexible Business Rules via ERs.

In RigInfoFlex, the Leave Rules were hardcoded. Some are shown here in **Text Box 4-2**. Promotion policy is among other representative examples.

```
IF v_old_EmpID is null AND v_old_jdate is not null THEN
    LFPs:=round(months_between(last_day(sysdate-30),v_old_jdate))*4;

ELSIF v_old_EmpID is not null AND v_old_jdate is null THEN
    LFPs:=round(months_between(last_day(sysdate-30),rec.j_date))*4;

ELSIF (v_old_EmpID is not null AND v_old_jdate is not null) OR (v_old_EmpID is null AND v_old_jdate is null) THEN
    LFPs:=round(months_between(last_day(sysdate),rec.j_date))*4;
```

Text Box 4-2: A business rules provides privilege that an employee can avail 48 leaves with full pay per year. Now form a particular date onward this privilege is reduced to 40 per year.

```
IF :leave_detail.leave_type = 'MTN' and upper(:global.ap_stat) not like 'SCHOOL STAFF%' THEN
   :leave_detail.to_date:=:leave_detail.from_date+89;
   :leave_detail.no_of_days:=90;

ELSIF :leave_detail.leave_type = 'MTN' and upper(:global.ap_stat) like 'SCHOOL STAFF%' THEN
   :leave_detail.to_date:=:leave_detail.from_date+44;
   :leave_detail.no_of_days:=45;

END IF;
```

Text Box 4-3: A business rules to cater for varying number of leaves for different employee type e.g. if maternity leaves for school staff is less than other employees in the same enterprise

If Set of Change Scenarios – 3 given in Table 4-3 or similar other scenarios are applied, the chain of changes illustrated in Table 3-3 unleash.

Table 4-3: Set of change Scenarios - 3

	Change Scenarios for Evaluating Third Pair of Rig-Flex Enablers (Business Rules via Code VS BR via ER Design)
SCS-3(A)	A business rules provides privilege that an employee can avail 48 leaves with full pay per year. Now from a particular date onward this privilege is reduced to 40 days per year
SCS-3(B)	Employee categories are hard coded in RigInfoSys. Definition of a category is changed. In another case category of an employee is changed because of changing parameters the category depends on.
SCS-3(C)	New type of leave say "Opportunity" leaves is introduced which a particular employee category can avail. Officer of that category can avail up to 35 whereas staff up to 50 per year and these will not be deducted from casual leave.
SCS-3(D)	Contract female teachers of schools run by the company can avail the maternity leave of two months twice a life whereas company regular female employees can avail three months thrice a life.

In FlexInfoSys, the business rules for leave policy and employee categories were designed as depicted *Figure 4-6* and *Figure 4-7* respectively. Rules are not being narrated but only depicted in diagrams for brevity.

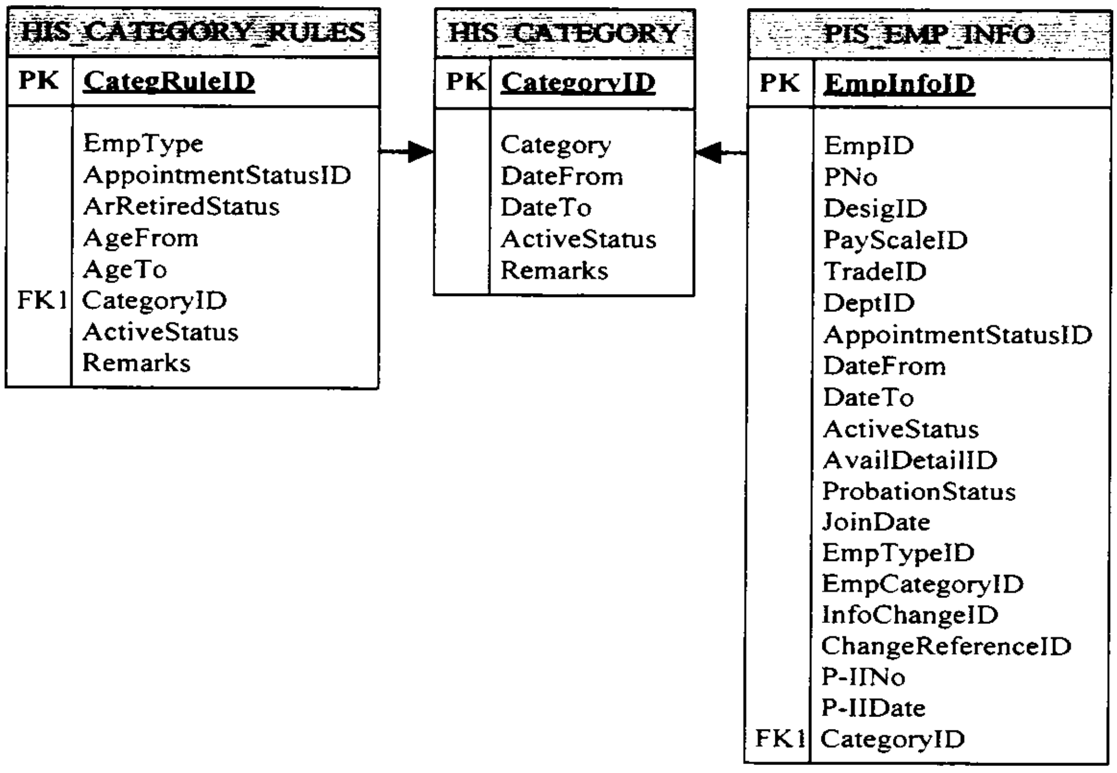


Figure 4-6: Employee categories and its design for its business rules

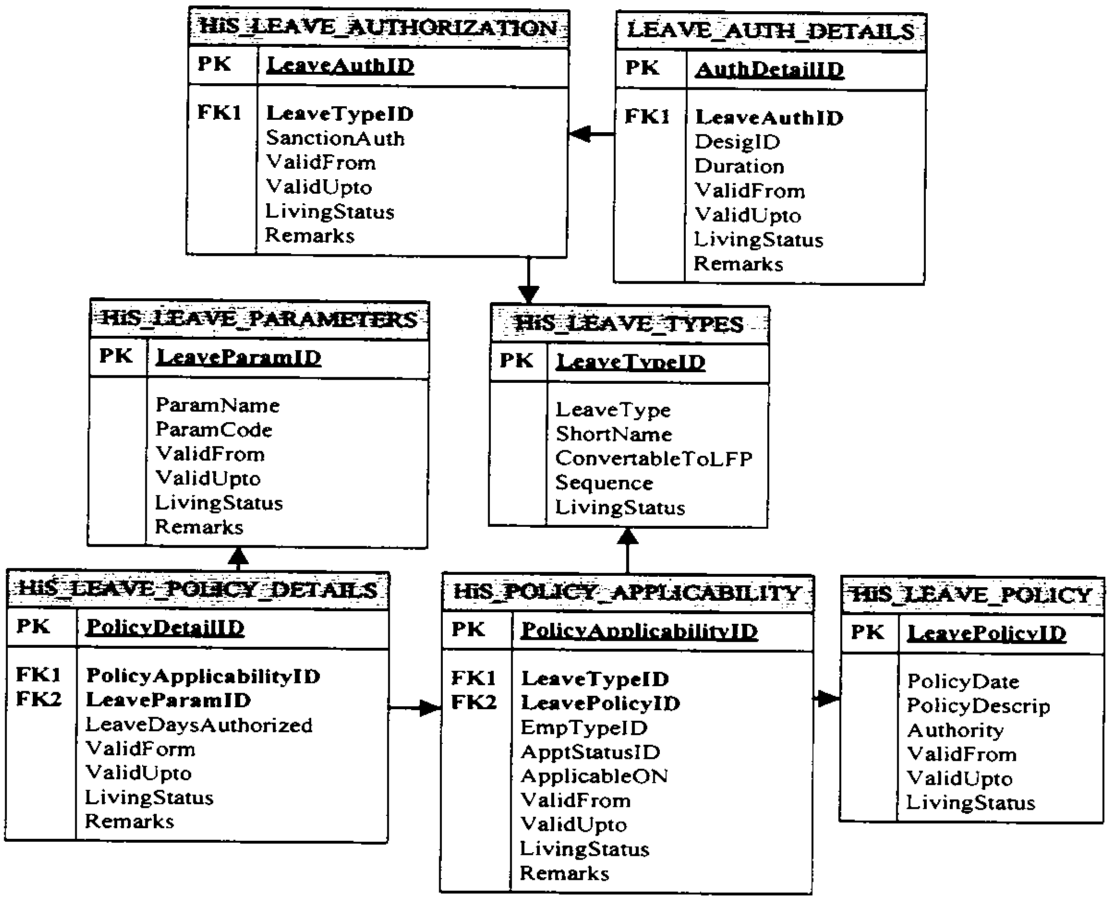


Figure 4-7: Business rules via ER Design e.g. Busines rules for Leaves

4.5.4 Go for Flex Chain Instead of Stiff Hook Integration between Modules

In RigInfoFlex, the integration was carried out as depicted by the Figure *4-8* below and a piece of code how developer of one module hooked to another module's information structure picking or dropping data directly is shown as Text Box 4-4 subsequently.

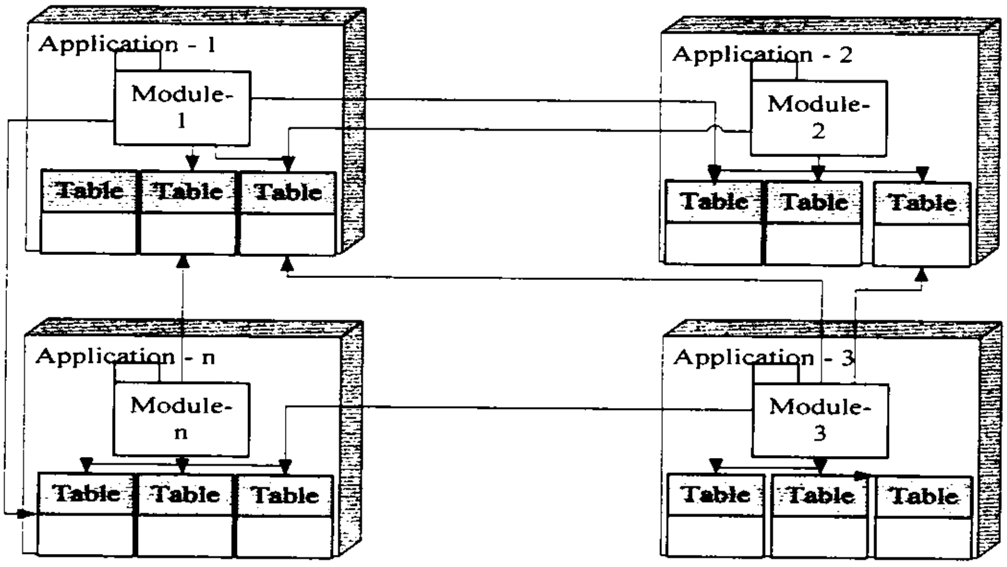


Figure 4-8: Stiff-Hook Integration – Just pick & drop data directly

```
PROCEDURE EVENT_PROBATION_TERMINATION(EMSID IN NUMBER) IS

-- Selecting data from Events Management System
SELECT   emp_info_id, emp_id,with_effect_from. date_upto,probation_status
FROM HRM.EMS_Prob_Terminate
 WHERE Evnt_id = EMSID

-- Updating PIS
IF i.probation_status ='Termination' THEN
   UPDATE HiS.ePROFILE_EMPLOYEE.Probation
   SET date_to = i.with_effect_from, probation_status='Completed'
   WHERE emp_id = i.emp_id and emp_info_id = i.emp_info_id
   AND probation_status='Active';
 END IF;
 END
```

Text Box 4-4: Integration of two Modules e.g. Integration of Event Management and eProfile - Probation Termination Process in Event Management System showing post-effects on eProfile.

Pick & Drop integration increases the maintenance and rigidity by the rate "how many other modules are hooked. If a developer changes his/her table structure and even if he/she communicate these changes to all relevant, the concerned developer will have to change his/her code/query to adjust his new structure. Otherwise efforts involved become twofold, the relevant artifact/object responsible for integration with a particular module becomes invalid and the concerned developer will have to trace out where the problem lies.

If Set of Change Scenarios – 4 tabulated in Table 4-4 or similar other scenarios are applied, the chain of changes illustrated in Table 3-4 unleashes.

Table 4-4: Set of Change Scenarios - 4

Change Scenarios for Evaluating Fourth Pair of Rig-Flex Enablers (Stiff-Hook VS Flex-Chain Integration)	
SCS-4(A)	**Health services gets stopped or other relevant policy applies when an employee is discharged, gets resign, dies or re-appointed as contract employee. Health Services module accesses eProfile for updated status and category of an employee – then relevant policy is applied in response. Structural change to this hampers provision of vital services**
SCS-4(B)	Organizational hierarchy includes 'Reports-To' field. This field is accessed by workflow, budget allocation, ACR writing and similar other modules that need to use hierarchy. In case of reorganization, change in this field leaves all other modules dysfunctional.
SCS-4(C)	**Promotion module updates directly designation of an employee in his/her profile whereas the same field is also updated in case of re-appointment. Allied modules goes down if change in this infor-struct happens.**
Note	Change in any of above mentioned field will render the integrated modules not functioning as

	expected rather error will prompt.

In FlexInfoSys, Integration between modules was rendered flexible by:

- A module provides data to the requester instead of the requester fetch the requisite data directly.
- A module takes the data provided by others and manages to save it by self instead of allowing other modules to drop data directly.

In this case, any change in information structure did not disrupt the other module.

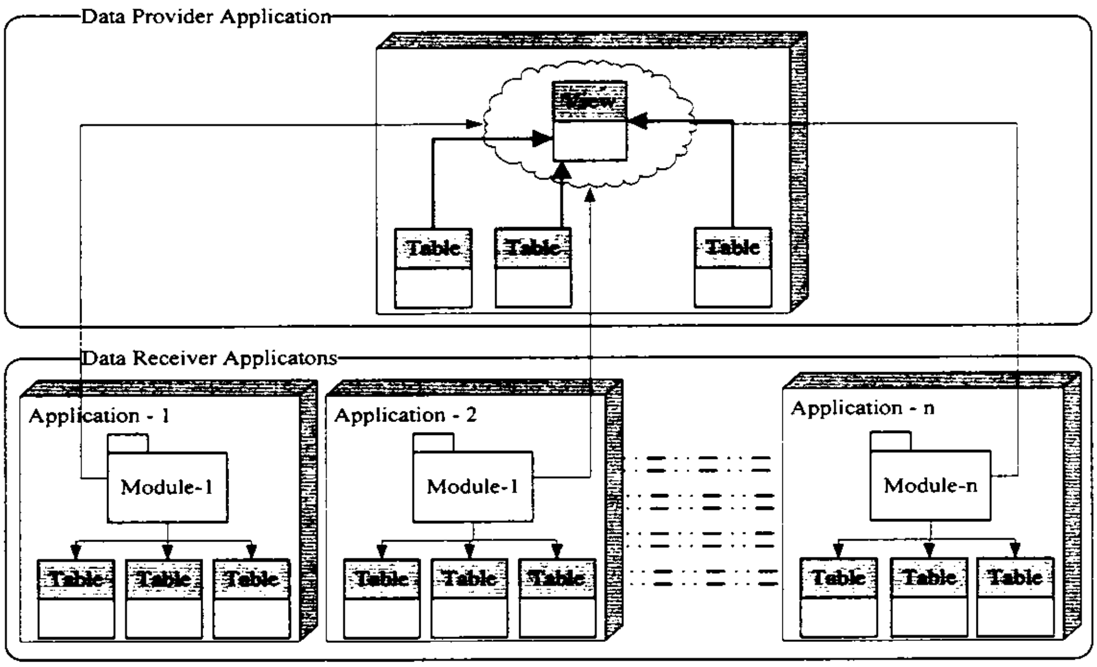


Figure 4-9: Flex Chain Integration

4.5.5 Opt for Write-History over Write-Over Entities

An appointment of an employee was changed from Manager to Senior Manager; the user remained unable to know what the employee's earlier appointment was.

ePROFILE_EMPLOYEE	
PK	PersNum
	Name IDCard DoB BirthPlace JoinDate Religion Sect Cast Appointment PayScale DeptCode SubDeptCode

Figure 4-10: Use of Write-Over Entities

```
PROCEDURE DISSIMINATE_CHANGE_IMPACTS(vOLD_VALUE NUMBER,
vNEW_VALUE NUMBER, vTARGET_COLUMN VARCHAR2) IS

Query_Str VARCHAR2(2000);

BEGIN
   Query_Str := 'UPDATE HRM.ePROFILE_EMPLOYEE
   SET '||vTARGET_COLUMN||'='||vNEW_VALUE||
   WHERE
ACTIVE_STATUS='||'''Active'''||'AND'||vTARGET_COLUMN||'='||vOLD_VALUE;
   EXECUTE IMMEDIATE Query_Str;

END DISSIMINATE_CHANGE_EMPACTS;
```

Text Box 4-5: Generic procedure of replacing employee information e.g. appointment on promotion of an employee

If set of change scenarios – 5 given below or similar other scenarios are applied, the chain of changes illustrated in Table 3-5 unravel.

Table 4-5: Set of Change Scenarios-5

Change Scenarios for Evaluating Fifth Pair of Rig-Flex Enablers (Write-Over VS Write-History ERs)	
SCS-5(A)	**An employee changes his religion or sect e.g. from Hinduism to Islam or form Shiite to Sunni or vice versa.**
SCS-5(B)	An employee is promoted from Junior Manager to Senior Manager. Pay-scale is also changed accordingly. The system should maintain and be accessible an employee's current as well as old historic information.
SCS-5(C)	**"Line of Work" of an employee is changed e.g. from Admin to IT. Similarly sub-category in the same line of work is changed.**
SCS-5(D)	Address of an employee is changed.

In FlexInfoSys, replacing old information was avoided by maintaining double-keys in transactional tables, one permanent (EmpID) having attributes of permanent nature but not unique whereas the other temporal (EmpChangeID) with changing information as and when change happened. Both keys were interlinked for traceability.

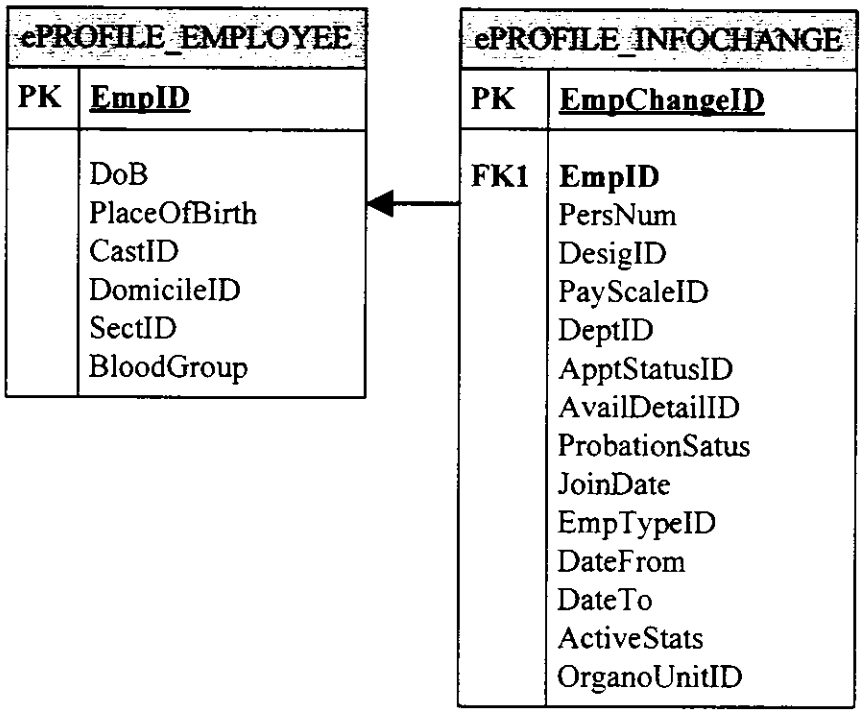


Figure 4-11: Use of Write – History ERs

4.5.6 Say Well come to Cross-Cutting and Goodbye to Provincial Entities

In this particular case study, built-in workflow, limited to 2 or 3 levels, has been developed by each developer for his own module. Condemnation board is another representative example.

BOARD_MAIN	
PK	Condem_Board_No
	BoardPlace BoardAuth BoardProceedingDate PresidID PresidName PresidDeptt Member1ID Member1Name Member1Desig Member1Deptt Member2ID Member2Name Member2Desig Member2Deptt Member3ID Member3Name Member3Desig Member3Deptt Member4ID Member4Name Member4Desig Member4Deptt Member5ID Member5Name Member5Desig Member5Deptt Member6ID Member6Name Member6Desig Member6Deptt

Figure 4-12: Rigid Structure for Condemnation Board Proceedings

If set of change scenarios – 6 tabulated in Table 4-6 or similar other scenarios are applied, the chain of changes illustrated in Table 3-6 unravels.

Table 4-6: Set of Change Scenarios - 6

Change Scenarios for Evaluating Sixth Pair of Rig-Flex Enablers (Provincial ERs VS Separate Cross-cutting ERs)	
SCS-6(A)	**Localized static workflow is defined in purchase module. Now similar workflow is needed in writing annual employee performance report writing and budget approval process.**
SCS-6(B)	Static "Condemnation Board" information structure is designed in inventory module for writing-off inventory items. Now it is also required in recruitment, inquiry, promotion, purchasing etc.
Note:- The no of times ER is re-used decide the effectiveness of ER	

In FlexInfoFlex, workflow between modules was built once illustrated in Figure *4-13* and used by all instead of creating built-in workflow in each module. Similarly constituting boards by different departments of the organization for different purposes should be designed once and used for all as illustrated in subsequent figure.

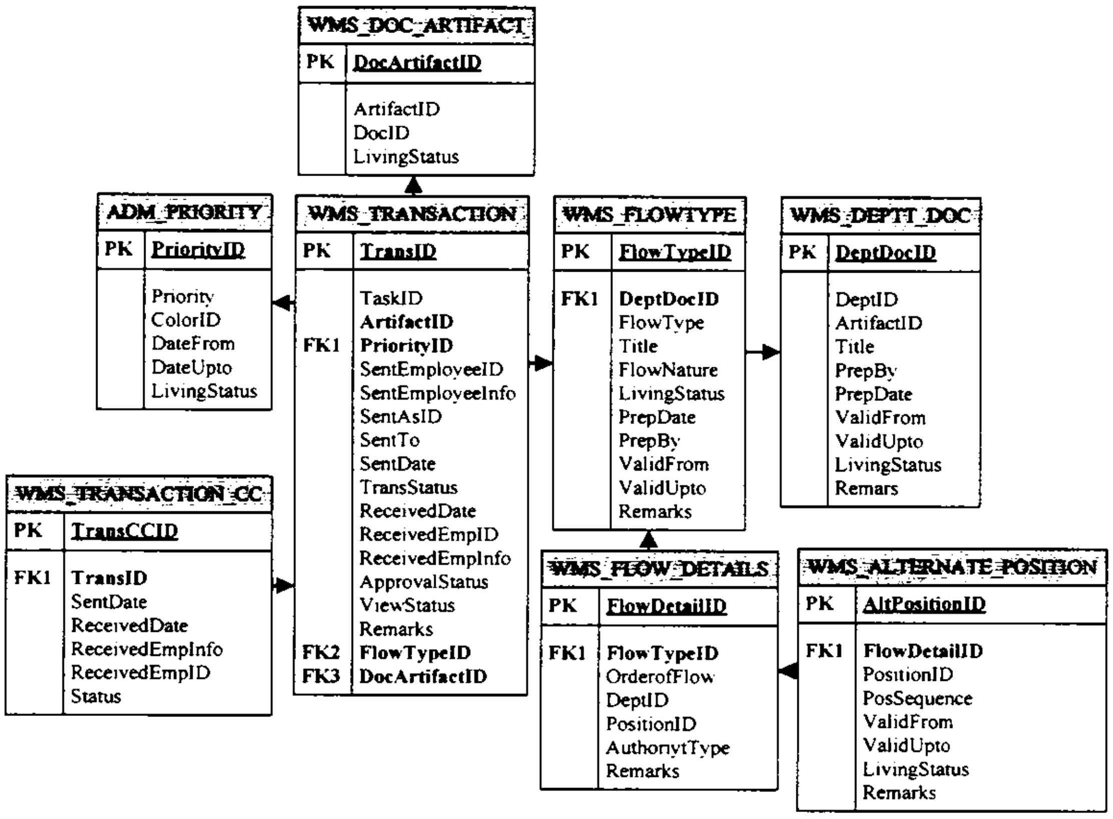


Figure 4-13: Separation of Crosscutting Concerns – Workflow Management

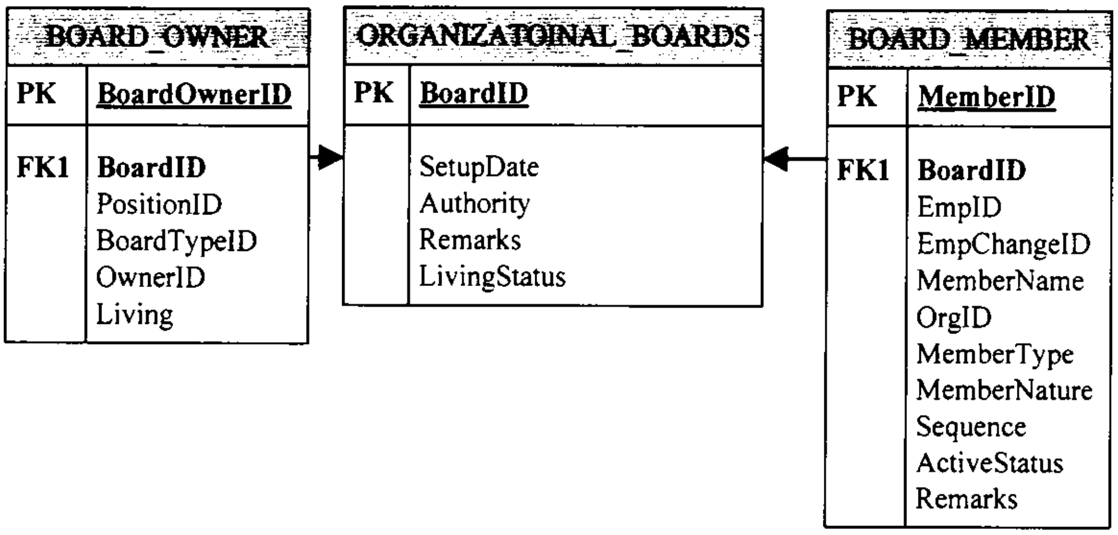


Figure 4-14: Separation of cross-cutting concerns e.g. Board Proceedings, Workflows, Org: Calendars, Org: Duty Rosters, Board Proceedings

4.5.7 Crush Entities to Fine-grained Level – Not leaving it as Rock.

In RigInfoSys, the address was defined as under:

ePROFILE_EMPLOYEE	
PK	<u>PersNum</u>
	Name IDCard DoB BirthPlace JoinDate Religion Sect Cast **CompleteAddress**

Figure 4-15: Use of Rock Entities e.g. Address as one attribute

If Set of Change Scenarios – 7 tabulated in Table 4-7 or similar other scenarios are applied, the chain of changes illustrated in Table 3-7 unleashes.

Table 4-7: Set of Change Scenarios - 7

Change Scenarios for Evaluating Seventh Pair of Rig-Flex Enablers (Rock VS Fine-grained Entities)	
SCS-7(A)	Make a report to find those employees who belong to city "AA" whereas one database-field is specified for complete address in RigInfoSys.
SCS-7(B	Name of a Province has changed from Province-'X' to Province- 'Z'. Rename and update address of all those employees who belong to Province-'X' accordingly. City / Country name has been fed in various formats e.g. USA, United States, US, America.
SCS-7(C)	City "CC" belonging to Province-I has been remapped with Province-II. Changes be made accordingly.

In FlexInfoSys, the *Rock Entity* (address) was crushed to fine-grained level as illustrated in Figure *4-16* and linked to more generalize hierarchical region structure.

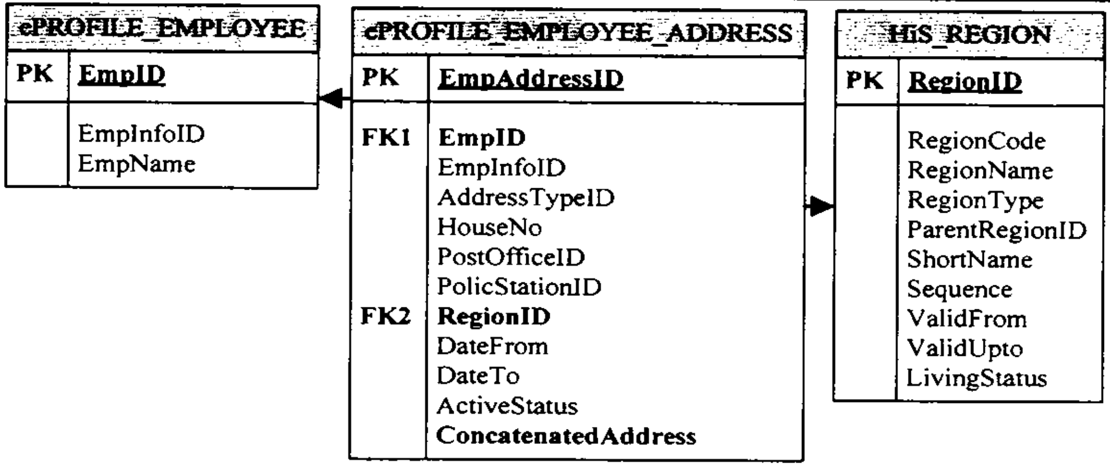


Figure 4-16: Fine-grained entities e.g. split address into fine-grained pieces

4.5.8 Choose to Define Breathing over Corpse Entities

In RigInfoSys, an appointment "Senior Manager" for an employee in a company existed and become obsolete after quite some time. It was referenced in transactional tables and cannot be deleted as per referential integrity rule, nor could it be changed /updated. No one was able to determine for which duration this appointment was under use or what appointment in the past was actually equivalent to "Principle Manager" now. Appointment entity was defined with no birth /death dates or living status as under.

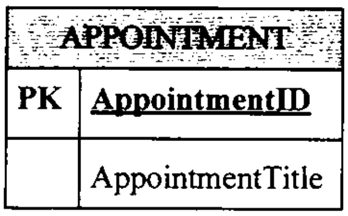


Figure 4-17: Use of Corpse Entities i.e. entities with unknown living status

If Set of Change Scenarios - 8 tabulated in Table 4-8 or similar other scenarios are applied, the chain of changes illustrated in Table 3-8 unleashes.

Table 4-8: Set of Change Scenarios - 8

Change Scenarios for Evaluating Eight Pair of Rig-Flex Enablers (Corpse VS Breathing Entities)	
SCS-8(A)	An appointment ***"Assistant Project Manager"*** existed in the company. It becomes obsolete and no more used. In other words an ***entity instance is died.*** Change accordingly.
SCS-8(B)	Junior Manager will be termed as ***"Deputy Manager"*** and Senior Managers as ***"Chief Manager"*** from this time onward. An entity changes its status.
Note:- [The system should be able to show a particular employee has been what in a specified period of time]	

In FlexInfoSys, the Corpse entity was transformed into Breathing by adding following highlighted features illustrated in Figure *4-18*. It tells when an entity took birth, is it currently alive? and when it became inactive or dead e.g. appointments.

HIS_APPOINTMENT	
PK	**AppointID**
	AppointID Appointment **Sequence** **ValidFrom** **ValidUpto** **LivingStatus** **LinkedWith** Remarks

Figure 4-18: Defining Breathing Entities e.g. Appointment

It's worth mentioning that if definition of a configurable entity changes, create a new one with new definition and render the old one inactive instead of changing its definition and link the two entities, otherwise the transactional entities featured with this configurable entity will always be displayed with the new one even if the transactional entity was defined before the definition of configurable entity.

4.5.9 Vote for Convergent FKs instead of Divergent FK by adding TableID as Attribute

Defining multiple competing foreign keys only limits the dynamic nature of program. The less dynamic programming defines the more rigid InfoSys e.g.

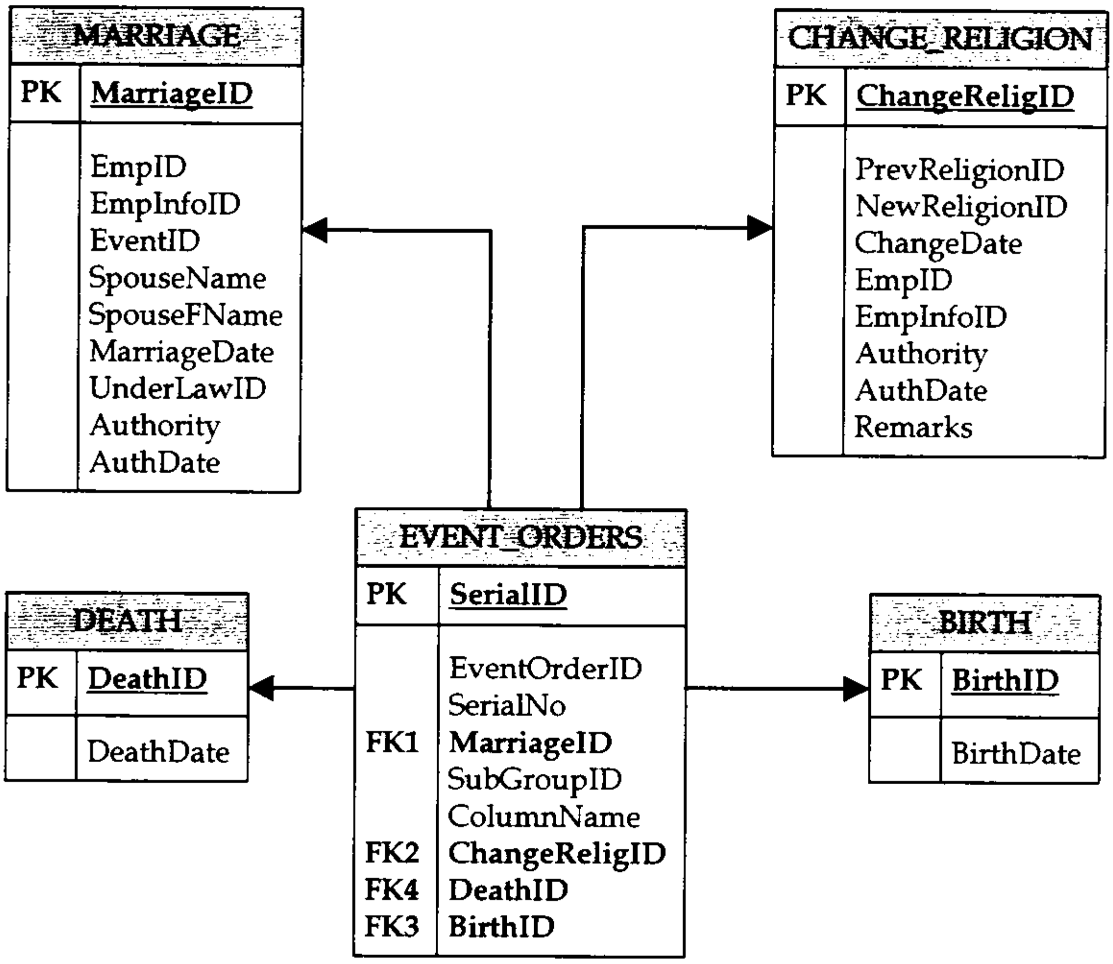


Figure 4-19: Use of Foreign Keys – Various entities of Events Management System

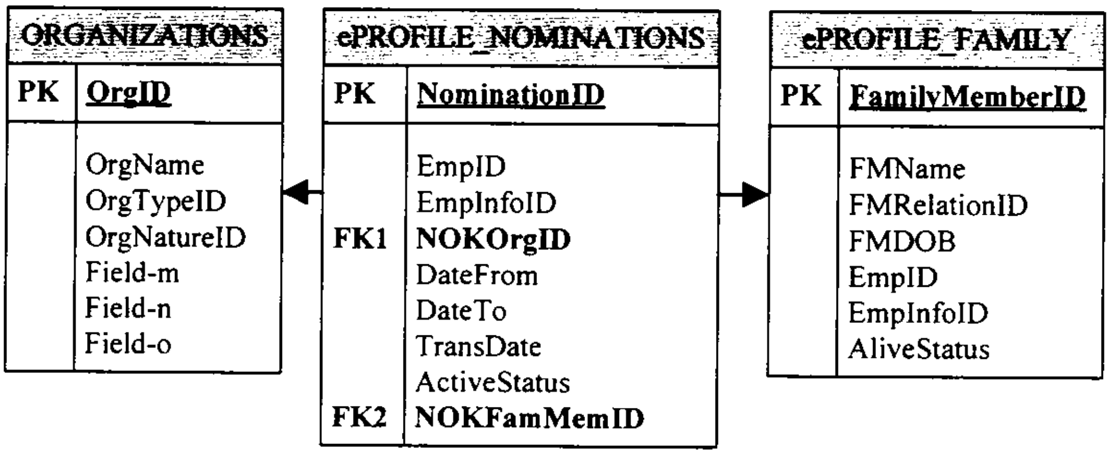


Figure 4-20: Use of Multiple FKs - Family Nominations for various types of employee's Funds

If Set of Change Scenarios - 9 tabulated in Table 4-9 or similar others are applied, the chain of changes illustrated in Table 3-9 unleashes.

Table 4-9: Set of Change Scenarios - 9

	Change Scenarios for Evaluating Ninth Pair of Rig-Flex Enablers (Diverging VS Convergent FKs)
SCS-9(A)	**Currently an employee can only nominate family member as next-of-kin. Now he/she can nominate a charity, govt. sector, private or any other organization as well. Info-Structure shows that family members and different organizations are maintained in different database tables.**
SCS-9(B)	Employment / Experience and Education record of an employee is being maintained. Now the company wants to record the same for employee's dependants and in-laws as well for security and ease of recruiting suitable individuals for a position.
SCS-9(C)	**New event 'Blood Donation' is added into Event Order System. Change accordingly please.**

In FlexInfoSys, this was designed using the table-name i.e. Marriage, Birth etc as a database field to trace foreign key as illustrated in *Figure 4-21* and *Figure 4-22*. The result is to minimize FKs and increase program dynamicity simultaneously.

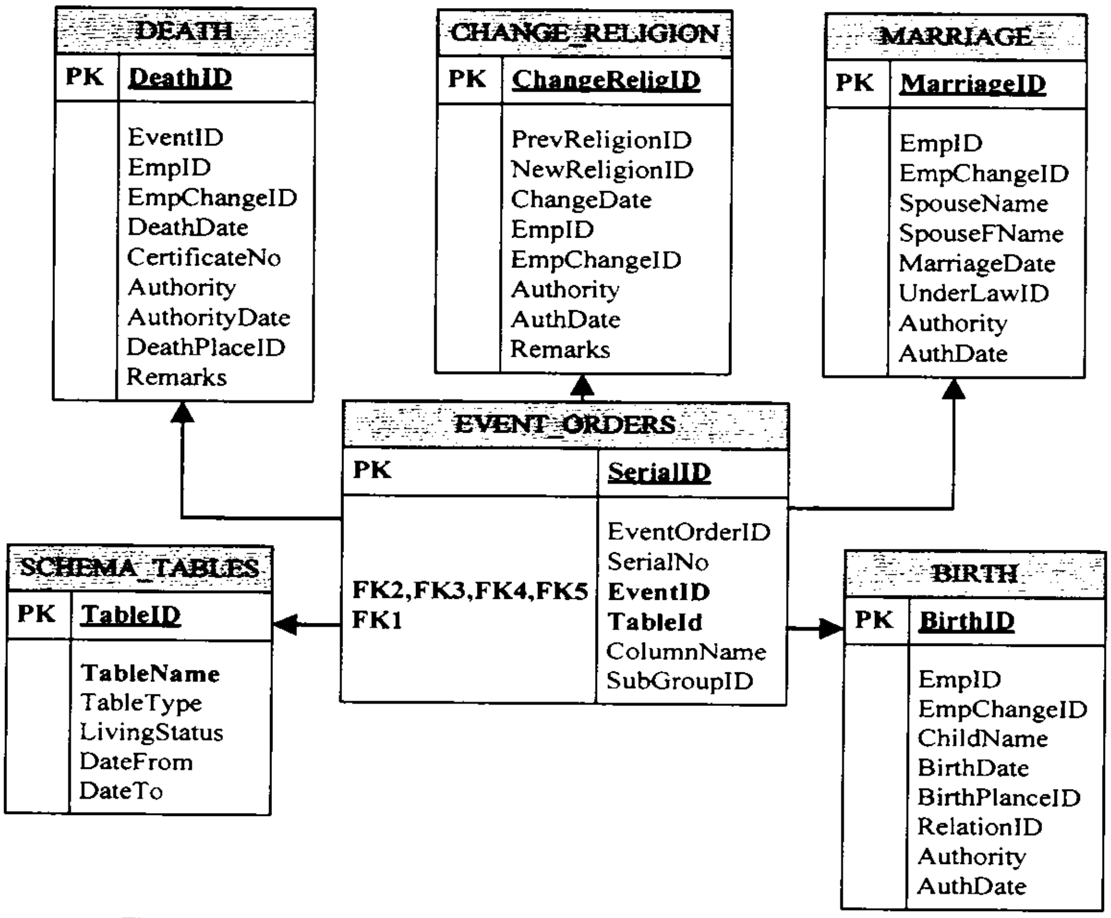


Figure 4-21: Convergent Competing Foreign Keys – Event Management System

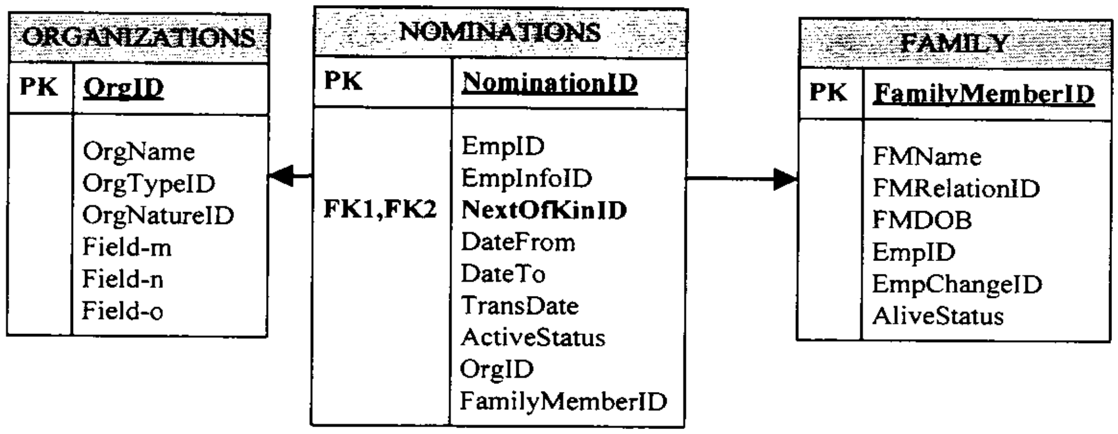


Figure 4-22: Converge multiple foreign keys into one by introducing table name as a database field

4.5.10 Use Conceptual Inheritance When Facing Bruce Limits-to-Flexibility

As explained in the problem statement, here it is diagrammatically illustrated by two examples emergence of new employee types and keeping record of more types of organizations e.g. vendors, institutes, banks etc.

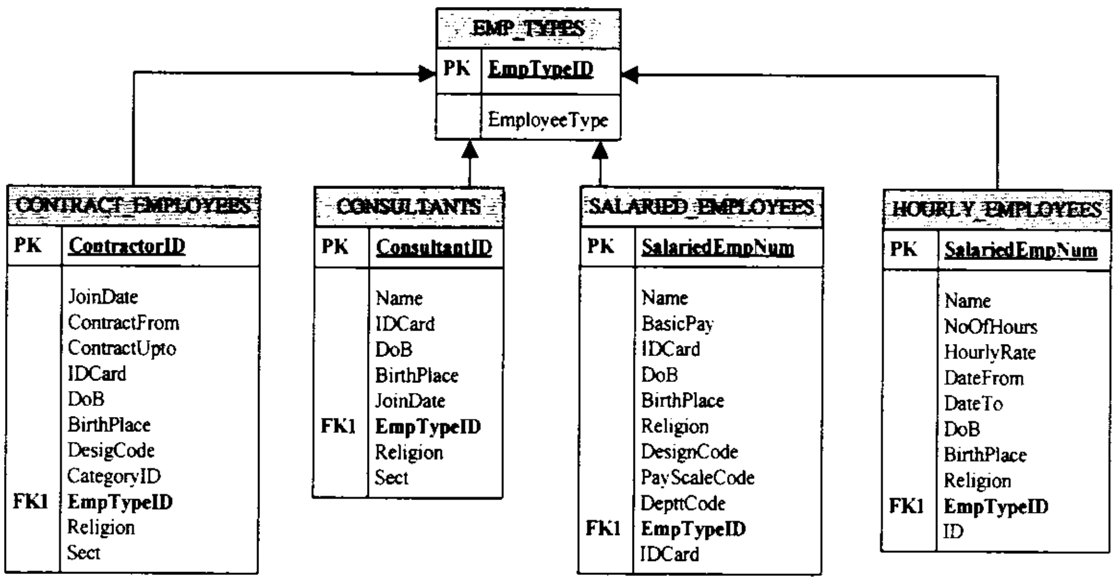


Figure 4-23: Bruce Johnson's Limits to Design Flexibility - Ceiling Effect for Employee Types

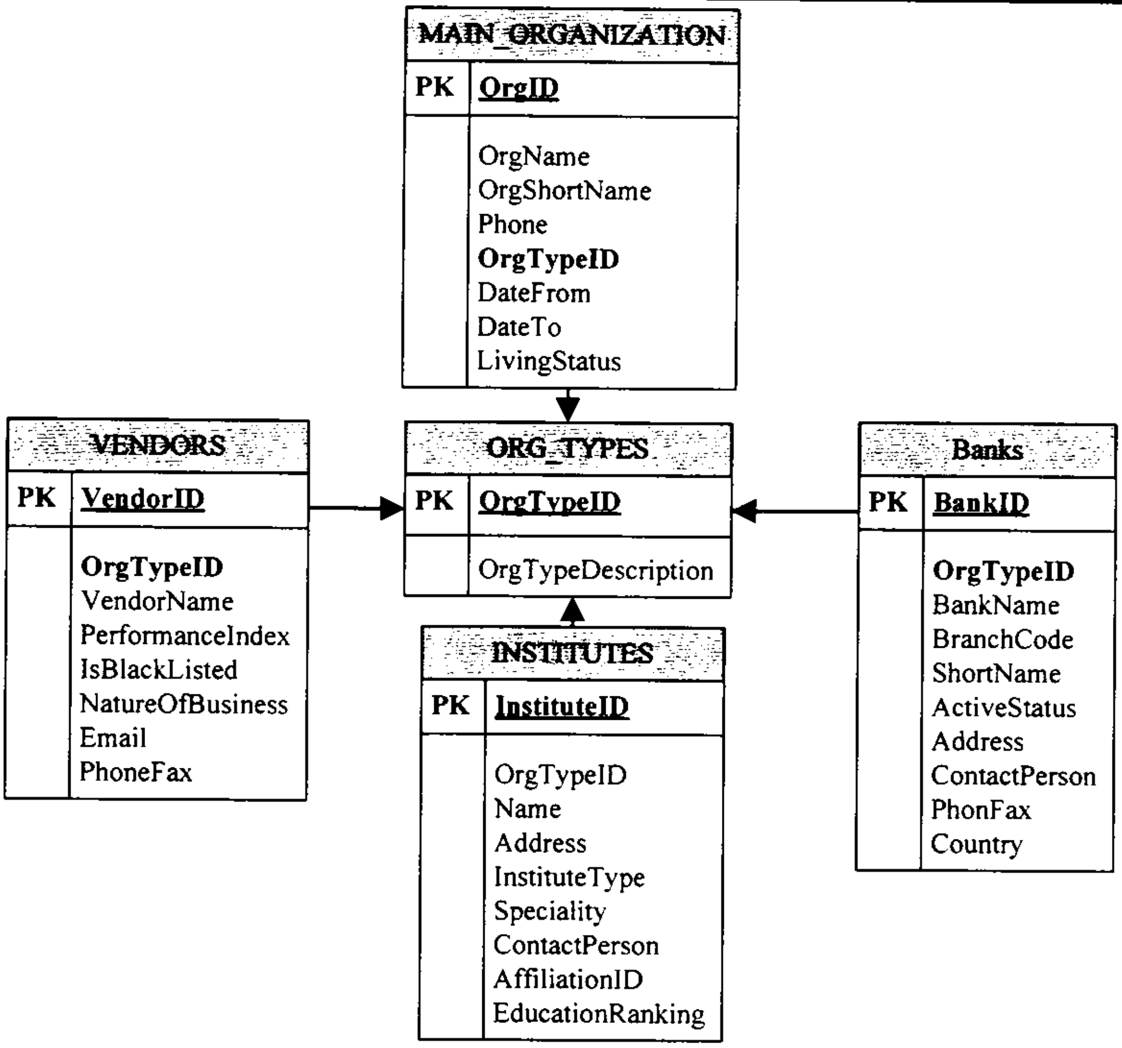


Figure 4-24: Bruce Johnson's Limits to Design Flexibility - Ceiling Effect for Organization Types

When we applied Set of Change Scenarios - 9 tabulated in Table 4-10, the chain of changes illustrated in Table 3-10 unleashes. Same is the case for other similar scenarios.

Table 4-10: Set of Change Scenarios - 10

Change Scenarios for Evaluating Tenth Pair of Rig-Flex Enablers (Ceiling-Effect VS Inheritance in ERs)	
SCS-10(A)	The company has up till now recruited regular employees only. It now decides to recruit some contract employees as school staff and hourly employees for some horticultural tasks.
SCS-10(B)	The organization is currently keeping record of its organizational hierarchy. It now wants to keep record of some commercial, private, Govt. Organizations, NGOs and Educational institutions each having different attributes

In FlexInfoSys, the ceiling effect was dispelled by employing the conceptual inheritance by putting the common/generic attributes in initial level of hierarchy and as the concept drift occurs, put the more specific attributes deep down the hierarchy. The employee and organization has been designed as illustrated in Figure 4-25:

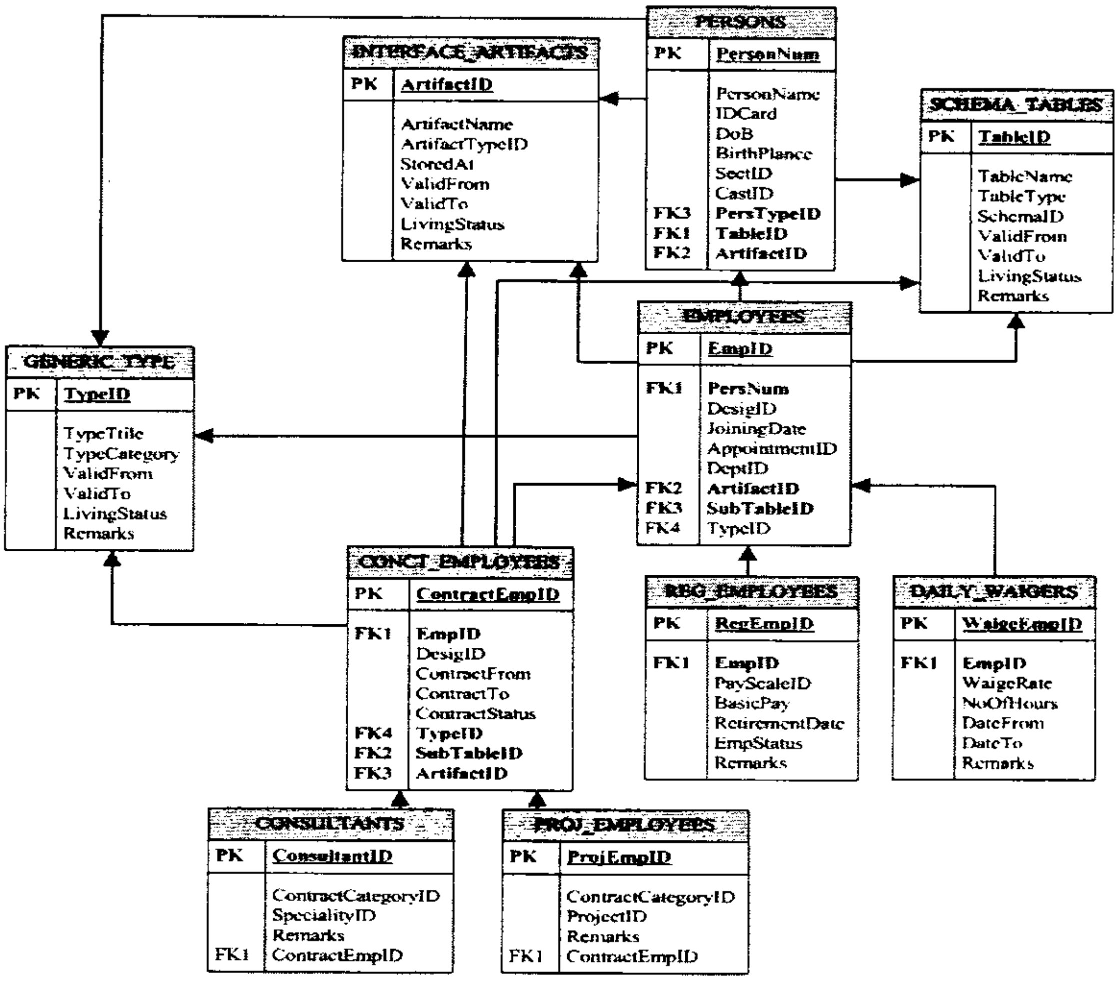


Figure 4-25: Solving Bruce Johnson's Limits to Flexible Design via Inheritance in ERs

4.6 Summary

In this chapter, the case study, carried out to validate the proposed framework, is spelt out. Rationale for selection of case study as methodology is provided. Main activities of the case study are elaborated diagrammatically. Change case scenarios, extracted from the analysis of the two rigid systems, are narrated. All rigidity and flexibility enablers are explained with examples from case study carried out. One of the two rigid systems was redesigned and redeveloped using the proposed framework. The chapter is concluded with the case study design keeping flexibility enablers in view.

Chapter - 5

VIABILITY OF THE PROPOSED FRAMEWORK

5. VIABILITY OF THE PROPOSED FRAMEWORK

When amount of efforts required to modify a system is out of proportions in comparison to the change requested, it's a first sign of rigidity in InfoSys. In order to examine the viability of the framework presented, we have used the aforementioned change-scenarios. Secondly as we address the Bruce Johnson's limits to flexible design, we prefer to use his criteria for the purpose as given in 5.1. The case study was designed in a manner to keep the results impartial and nullify or at least minimize the impact of factors affecting it. The following major potential factors were considered and kept constant for the said study both for rigid as well as re-engineered flexible InfoSys:

- Development Environment
- Information System Development Life Cycle
- Skills of Information System Professionals i.e. Architects, Designers, Analysts and Developers etc.
- Development Tools
- Information System Users
- Network Infrastructure
- Hardware and Computing Machines Used

5.1 Bruce Johnson's Criteria for evaluating Rigidity/Flexibility of an InfoSys

Flexibility Taxonomy	Cost in Terms of Efforts	Changes in Information Structure	Changes in Program Code	Changes in Data Values
Weak Flexibility	Most Costly	✓	✓	✓
Medium Flexibility	Next Most Costly		✓	✓
Strong Flexibility	Least Most Costly			✓

We have even further refined his criteria for validating the effectiveness of proposed strategies as under:

- Watch for ripple and avalanche effects involved in change scenarios.
- Count the steps and heaviness of the steps involved
- Generally, steps involved in avalanche are heavier than ripple as many times as the number of allied/integrated systems

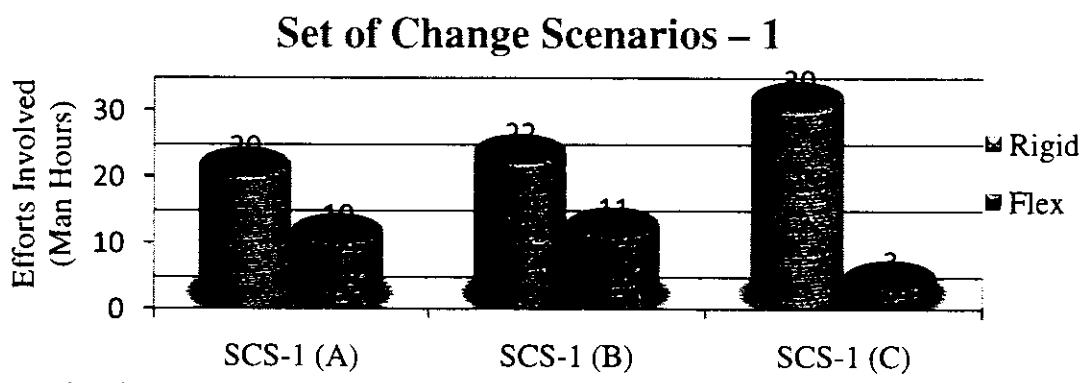
5.2 Graphs showing comparative analysis of Rigid / Flexible Practices

The graphs shown depend on the data collected against change-scenarios before and after applying the proposed framework i.e. for flexible ER practices against each rigid ER and also taking expert judgment.

Note: - In following comparative graphs, the x-axis shows the sets of change-scenarios for both rigid and flexible InfoSys(s) whereas the y-axis depicts the efforts involved in terms of man-hours for both rigidity and flexibility enablers.

5.2.1 Evaluating Attributed VS Non-Attributed Keys:

The set of changes shown in Table 4-1 can be incorporated by various ways. The fact is that by either way attributed keys takes more efforts to change. This is depicted in Graph 5-1.

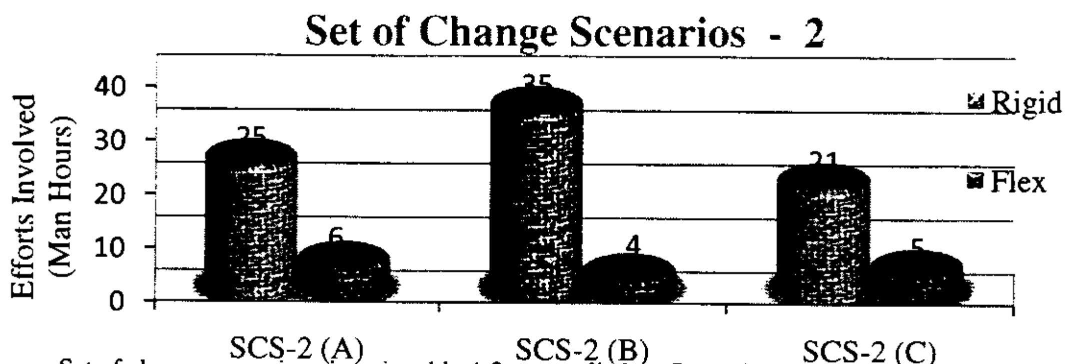


Set of change scenarions givne in table 4-1 are applied on First Pair of Rig-FlexEnablers

Graph 5-1: Attributed/Composed Character VS Non-Attributed, Auto-generated Numeric Keys

5.2.2 Evaluating Half-baked VS Recursive Nth Level Hierarchies

The efforts consumed both in rigid and flexible enablers while incorporating the changes shown in Table 4-2 is displayed in Graph **5-2**.

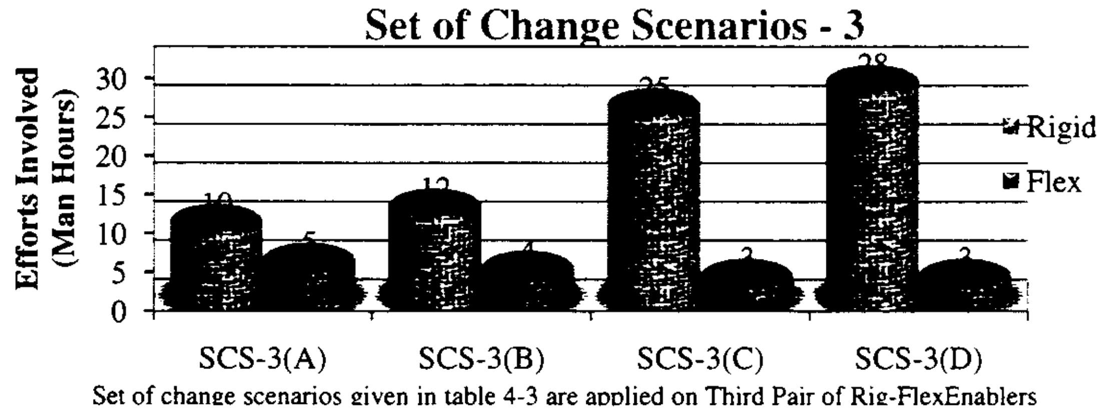


Set of change scenarios given in table 4-2 are applied on Second Pair of Rig-Flex Enablers

Graph 5-2: Half-baked VS Recursive Nth Level Hierarchies

5.2.3 Evaluating BR via Code VS BR via ER Design

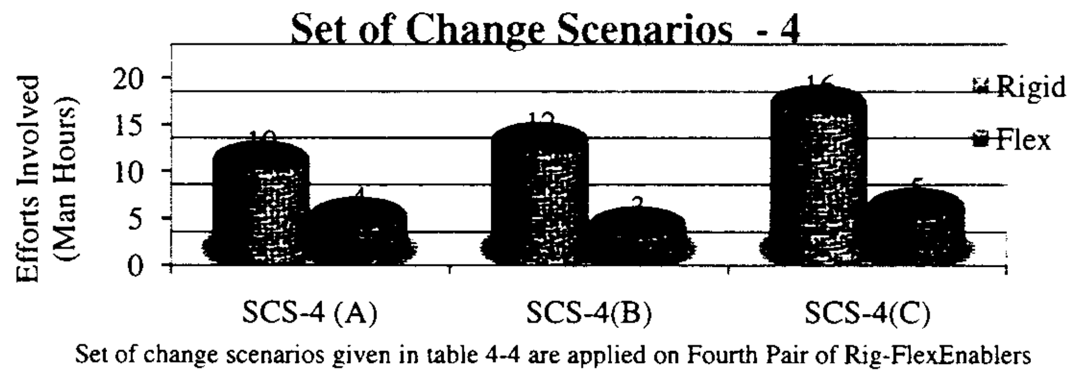
The efforts involved in incorporating the changes shown in Table 4-3 is displayed in Graph 5-3.



Graph 5-3: Business rules via Code VS Business rules via ERs Design

5.2.4 Evaluating Stiff-Hook VS Flex-Chain Integration

The efforts consumed in incorporating the changes shown in Table 4-4 is displayed in Graph 5-4.

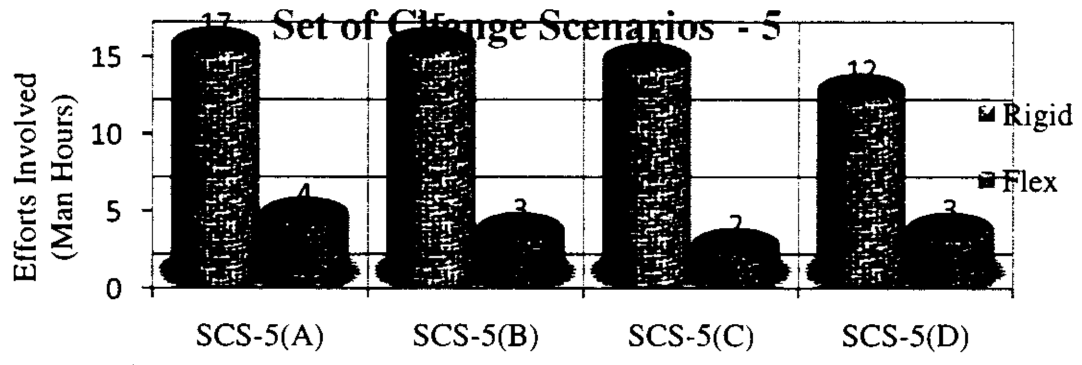


Graph 5-4: Stiff-Hook VS Flex-Chain Integration

Note: The efforts involved depend on the number of modules integrated with the module where the changes have occurred. However, the maintenance generated by rigid system if ER is changed is always greater than flexibly designed system.

5.2.5 Evaluating Write-Over VS Write-History ERs

The efforts consumed both in rigid and flexible enablers while incorporating the changes shown in Table 4-5 is displayed in Graph 5-5.

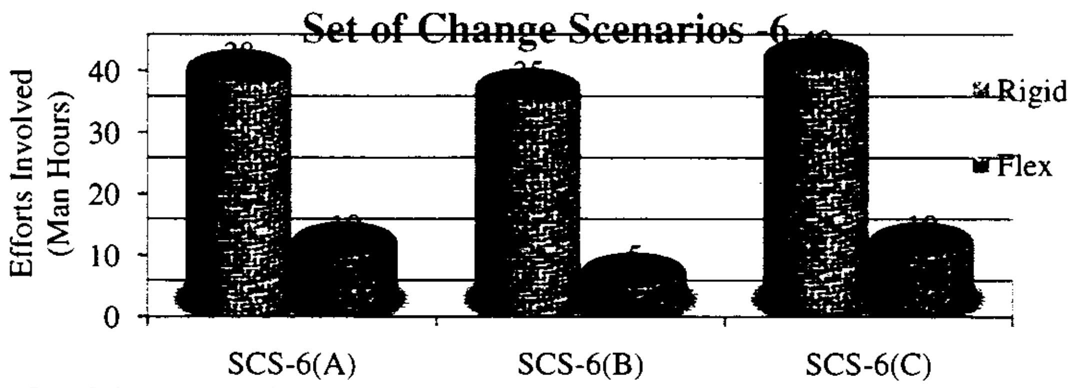


Set of change scenarios given in table 4-5 are applied on Fifth Pair of Rig-FlexEnablers

Graph 5-5: Write-Over VS Write-History Entities

5.2.6 Evaluating Provincial ERs VS Separation of Cross-cutting ERs

The efforts consumed both in rigid and flexible enablers while incorporating the changes shown in Table 4-6 is displayed in Graph 5-6.

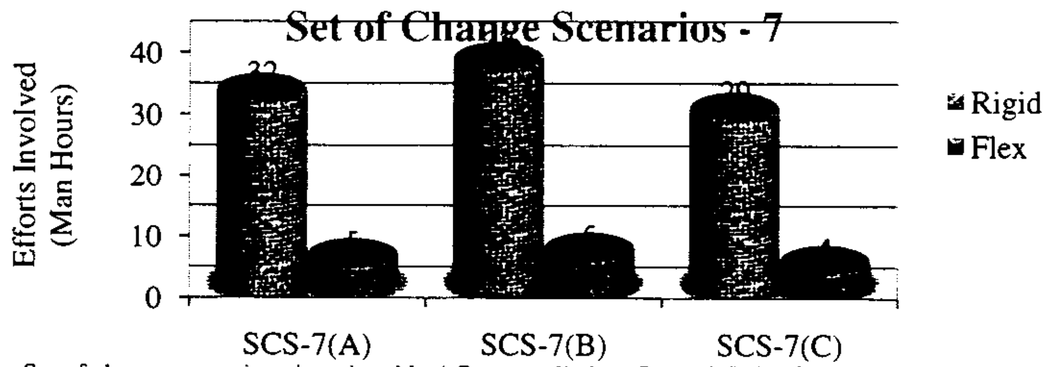


Set of change scenarios given in table 4-6 are applied on Sixth Pair of Rig-FlexEnablers

Graph 5-6: Provincial ERs VS Separation of Cross-cutting ERs

5.2.7 Evaluating Rock VS Fine-grained Entities

The efforts consumed both in rigid and flexible enablers while incorporating the changes shown in Table 4-7 is displayed in Graph 5-7.

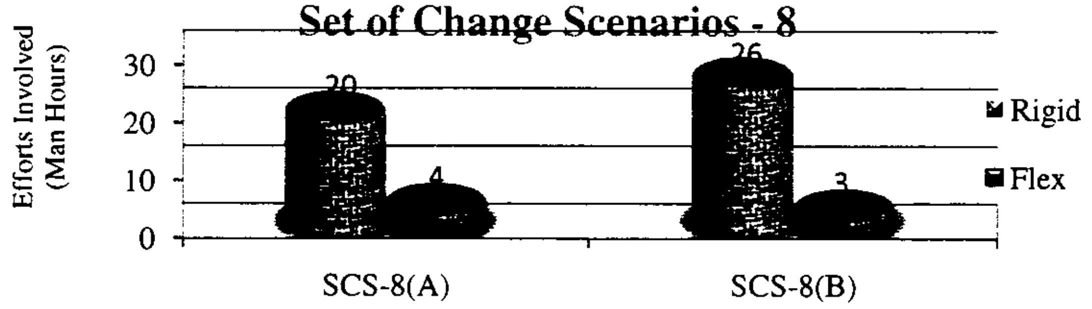


Set of change scenarios given in table 4-7 are applied on Seventh Pair of Rig-FlexEnablers

Graph 5-7: Rock Entities VS Fine-grained Entities

5.2.8 Evaluating Corpse VS Breathing Entities

The efforts consumed both in rigid and flexible enablers while incorporating the changes shown in Table 4-8 is displayed in Graph 5-8.

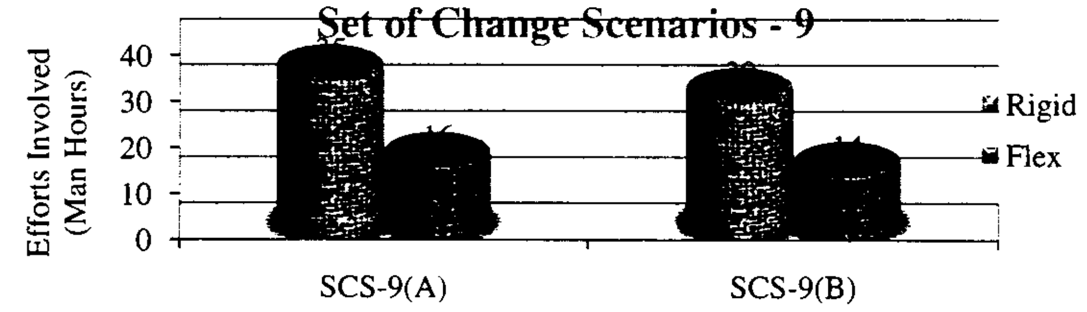


Set of change scenarios given in table 4-8 are applied on Eight Pair of Rig-FlexEnablers

Graph 5-8: Corpse VS Breathing Entities

5.2.9 Evaluating Divergent VS Convergent Entities

The efforts consumed both in rigid and flexible enablers while incorporating the changes shown in Table 4-9 is displayed in Graph 5-9.

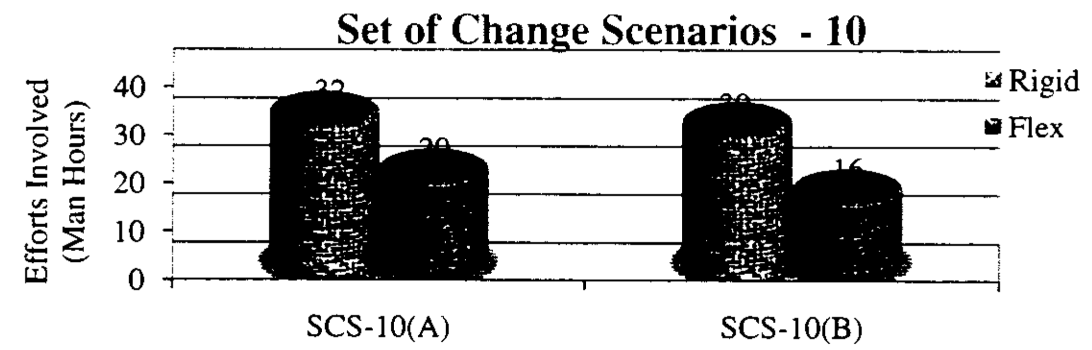


Set of change scenarios given in table 4-9 are applied on Ninth Pair of Rig-FlexEnablers

Graph 5-9: Divergent VS Convergent FKs

5.2.10 Evaluating Bruce Limits to Flexible Design VS Inheritance in ERs

The efforts consumed both in rigid and flexible enablers while incorporating the changes shown in Table 4-10 is displayed in Graph 5-10.

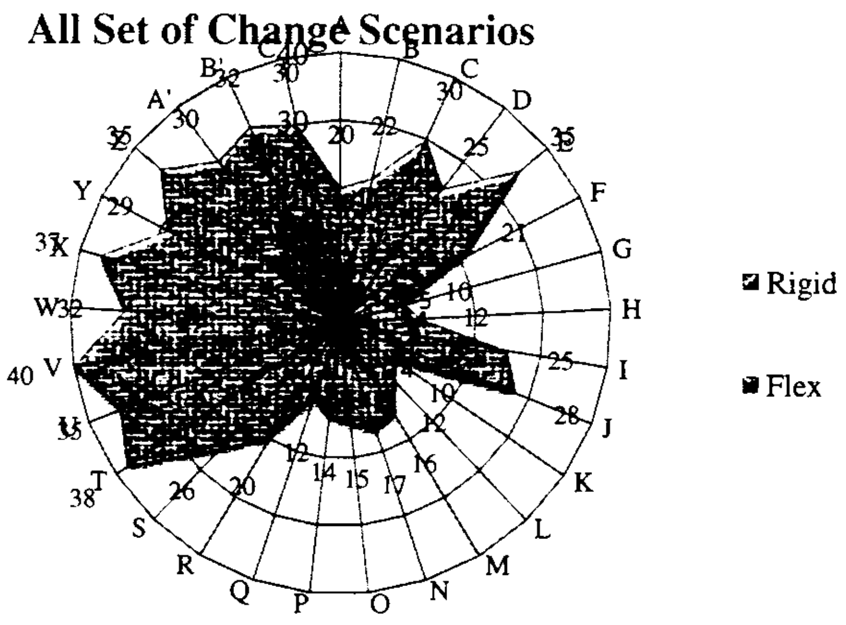


Set of change scenarios given in table 4-10 are applied on Tenth Pair of Rig-FlexEnablers

Graph 5-10: Limits to Flexible Design VS Inheritance in ERs

5.2.11 Overall Comparison of Rigidity and Flexibility Enablers

This graph shows the combined efforts consumed by all rigidity enablers for all scenarios in brown color in comparison to combined efforts consumed by all flexibility enablers for the same scenarios in blue color.



Graph 5-11: Overall comparison of Rigid and Flexible ER Enablers

This shows the viability of the proposed framework comprising of rigidity enablers and flexibility enablers in information systems.

5.3 Summary

In this chapter, viability of the proposed framework is validated by using Bruce Johnson's criteria and sets of change scenarios for each flexibility / rigidity enabler. Firstly, Bruce Johnson's criterion is articulated. Secondly, the same criterion is further refined by taking the steps of ripple / avalanche effect as well as heaviness of the steps involved into account. The criterion is applied in conjunction with the change case scenarios narrated in the case study (chapter - 4). The data collected against the criteria while accommodating the change case scenarios for each rigidity / flexibility enablers duly authenticated by two teams – each comprising of fifteen professionals. This data is shown graphically to compare the efforts consumed against each rigidity enabler and the relevant flexibility enabler. Finally, an overall comparison of rigid and flexible enablers is depicted through a graph. The graph exhibits that flexibility enablers takes less efforts when implemented than rigidity enablers. This demonstrated the viability of the framework.

CONCLUSION AND FUTURE WORK

6. CONCLUSION AND FUTURE WORK

6.1 Conclusion

In this thesis, we propose a framework for stability of information structure contributes to inherent flexibility in information systems. It shows that this framework provides a concrete foundation to stop chain of changes i.e. ripple and avalanche effects including how to address the Bruce Johnson's Limits-to-Flexible Design i.e. Ceiling Effect via introducing inheritance in information structure. This assisted to maximize the area under the curve of inherent-flexibility in InfoSys and hence minimizing the area under the curve of inherent rigidity in InfoSys. In other words extended, a bit more, the limits of flexible design in information systems marked by Brue Johnson.

However, the framework is limited to InfoSys and furthermore addresses only stability of information-structure.

6.2 Lessons Learned

Besides the framework for stability of information structure and ultimately flexibility in information systems, the following lessons were learnt, as a bi-product, through the case study.

- Always think bigger and generic.
- Don't forget future while designing and anticipate potential changes.
- Develop a common integrated information structure for entire system firstly and may opt for "one-at-a-time" incremental development afterwards.
- Leave running after "finished requirements" - they are never finished as change is immortal instead try to render InfoSys flexible enough to accommodate changes gracefully.
- Dry run your information structure for potential changes.
- Always opt for quality of information structure over deadlines whenever you have option to choose among the two.
- Use m: n association between entities where association information is distant to know as it provides one-to-one, one-to-many as well as many-to-many association.

- Revisit an entity and revise it to fit the continuing changing requirements.
- Always accord priority, while designing, to flexibility over functionality, as focusing functional requirements only work against flexibility.
- Flexible ER design, renders programming complex, don't be afraid as it is one-time activity and let you enjoy all the times to come.
- Like functional requirements are the user's business, infusing flexibility, most of the times, is software designer's concern.
- Transforming a legacy rigid InfoSys into flexible one is manifold harder than engineering flexible system for the first time.
- Testing phase should necessarily include ensuring the flexibility characteristics along with testing business features.
- In MIS setup, creation of information structures i.e. ERs must be centrally controlled instead of letting developer play havoc with information system foundation.

6.3 Future Work

TOP is always vacant, it's very rare when there comes an end in research of any field, unless some new or more attractive filed is introduced which grasp the features of parent or child field. The future work includes:

- Balancing flexibility in information systems with other quality attributes e.g. performance, usability etc
- CMMI like framework, **FMMI** (Flexibility Maturity Model for Information Systems) –Defining Levels, Key Flexibility Areas,– Generic Goals / Generic Practices, Specific Goals / Specific Practices, SCAMPI like flexibility assessment methodology for information systems.
- Impact of flexibility on quality of data in IS.

6.4 Summary

This chapter concludes overall thesis highlighting that the proposed framework provides a concrete foundation for stable information structure and in turn contributes to flexibility of information system, on one hand, future work is highlighted subsequently for researchers. On the other hand, lessons learned while carrying out the case study, have been shared for practitioner's community.

REFERENCES

Note: The references are arranged alphabetically in order of the first character of last name of first author and then by year of publication.

[1] A. AlKalbani and K. Niguyen, "Designing flexible business information system for modern-day business requirement changes," in *Proc. of 2nd International Conference on Software Technology and Engineering*, vol. 2, Oct 2010, pp. 112-118,

[2] C. Ackermann, M. Lindvall and G. Dennis, "Redesign for flexibility and maintainability: A case study", in *Proc. of the European Conference on Software Maintenance and Reengineering*, IEEE Computer Society, Mar 2009, pp. 259-262.

[3] K. Andresen and N. Gronau, "Adjustment Strategies for Managing Unanticipated Changes in Software Development", in *Proc. of Wirtshaftinformatik*, Berlin, Germany, Jan 2009, no. 12, pp. 717-726.

[4] E. Byrne, "Using action research in information systems design to address change - A south African health information system case study", in *Proc. of SAICSIT*, pp. 131-141, 2005.

[5] C. Coronel, S. Morris and P. Rob, *Database Systems – Design, Implementation and Management*, 9th ed, Cengage Learning, 20 Channel Center Street Boston, MA 02210 USA, 2010.

[6] C. Chen and P.C. Chen, "A holistic approach to managing software change impact", *Journal of Systems and Software,* vol. 82, no. 12, pp.2051–2067, Jun 2009.

[7] A. Eden and T. Mens, "Measuring Software Flexibility", *IEEE Software,* vol. 153, no. 3, pp. 113–126, Jun 2006.

[8] J. Gebauer and F. Schober, "Information System Flexibility and the Cost Efficiency of Business Processes", *Journal of the Association for Information Systems*, vol. 7, no. 3, pp. 122-147, Mar 2006.

[9] W. Golden and P. Powell, "Towards a Definition of Flexibility: in Search of the Holy Grail?" *The International Journal of Management Science*, vol. 28, no. 4, pp. 373-384, Aug 2000.

[10] O. Hollschke, J. Rake, P. Offermann and U. Bub, "Improving Software Flexibility for Business Process Changes", *Business & Information System Engineering,* vol. 2, no. 1, pp. 3-13, Oct 2010.

[11] L. Jacome, T. A. Byrd and L. W. Byrd, "An Examination of Information Systems Flexibility", *International Journal of Information Processing and Management,* vol. 2, no. 2, pp. 69-77, Apr 2011.

[12] B. Johnson, W. Woolfolk, R. Miller and C. Johnson, *Flexible Software Design – System Development for Changing Requirements,* 1st ed, Boca Raton, Auerbach Publications, 2005.

[13] B. Johnson and W. Woolfolk, "Generic Entity Clouds: A stable Information Structure for Flexible Computer Systems", *System Development Management,* Oct 2001.

[14] B. Johnson, W. Woolfolk, and P. Ligezinski, "Counterintuitive Management of Information Systems Technology", *Magazine Business Horizon,* pp. 29-36, Apr 1999.

[15] M. Khan, W. Nisar, E. Munir, W. Anwar and I. Ali, "Deployment Strategies for a Reengineered Information System in Context of Legacy System", *Research Journal of Applied Sciences, Engineering and Technology,* vol. 4, no. 3, pp. 178-185,2012.

[16] L. King and K. Lyytinen, "*Information Systems – The State of the Field*", John Wiley and Sons Ltd, The Atrium Southern Gate Chichester, West Sussex PO19 8SQ, England, 2006, ch. 1, pp. 1-15.

[17] W. Kadir and P. Loucopoulos, "Relating evolving business rules to software design", *Journal of System Architecture*, Manchester UK, vol. 50, no. 7 pp.367-382, 2004.

[18] E. Kirda, M. Jazayeri, C. Kerer and M. Schranz, "Experiences in Engineering Flexible Web Services", *IEEE Computer Society*, vol. 8, no. 1, pp.58-65, 2002.

[19] F. Lee and J. Gebauer, "The Role of IS-Flexibility for the Management of an E-Procurement System: A Case Study", in *Proc. of Twelfth Americas Conference on Information Systems (AMCIS),* Acapulco Mexico, Aug 2006, pp. 1895-1901.

[20] R. Molero, M. Barry, H.A. Hunter and T. Shunnar, "Flexible Database Structures for Land Records", in *Proc. of the FIG Congress – Facing the Challenges – Building the Capacity,* Sydney Australia, , Apr 2010, pp. 1-18.

[21] H. Mubarak, "Developing Flexible Software Using Agent-Oriented Software Engineering", *IEEE Software*, vol. 25, no. 5, pp. 12-15, 2008.

[22] S. Mary, "Writing Good Software Engineering Research Papers", IEEE Computer Society, in *Proc of the 25th International Conference on Software Engineering, IEEE Computer Society*, 2003, pp. 726-736.

[23] R. Miller, B. Johnson and W. Woolfolk, "Flexible information system, easy to change", *Educause Quarterly*, vol. 25, no. 3, pp. 44-51, 2002.

[24] S. Oliver., Helge K. and Volker W., "How to Make Software Softer - Designing Tailorable Applications", in *Proc of the 2nd conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques*, ACM New York, NY, USA, 1997.

[25] T. Pusalti and M. Sanjay, "A discussion on IS and Software Measurement Terminology – Flexibility as an example", in *Proc. of the International conference on Computational Science and Its Applications, IEEE Computer Society*, Mar 2010, pp.250-254.

[26] S. Peng, L. Shen, H. Liu and F. Li, "User-oriented Measurement of Software Flexibility", in *Proc. of the World Congress on Computer Science and Information Engineering, IEEE Computer Society*, , 2009, pp.629-634.

[27] X. Qiu, Li. Tang, Z. He and J. Chen, "The development of procurement management information System based on workflow technology", in *Proc. of the World Congress on Computer Science and Information Engineering*, Los Angeles, CA, vol. 3, 2009, pp 470-474.

[28] A. Rashid, W. Y. C. Wang and F. B. Tan, "Information Systems Maintenance: A key driver of Business Process Innovation", in *Proc. of the Sixteenth Americas Conference on Information Systems*, Lima Peru, Aug 2010, pp. 1-9.

[29] D. Robert and L. Devin, "Weighing the Benefits and Costs of Flexibility in Making Software: Towards a Contingency Theory of Determinants of Development Process Design", *Information Systems Research*, vol. 20, no. 3, pp.462-477, Sep 2009.

[30] P. Runsen, M. Host, Guidelines for conducting and reporting case study research in software engineering, *Empirical Software Engineering*, vol. 14, no. 2, pp. 131-164, 2008.

[31] F. Schober and J. Gebauer, "How Much to Spend on Flexibility? Determining the Value of Information System Flexibility", *Decision Support Systems*, vol. 51, no. 3, pp. 638-647, Jun 2011.

[32] S. Truren, *Improving software flexibility in a smart business network*, M.S. thesis, Faculty of Technology, Policy and Management Section Information and communication Technology, Delft University of Technology, Netherland, 2010.

[33] W. Tellis, , "Introduction to case study", *The Qualitative Report* [On-line serial], vol. *3*, no. 2, Jul 1997. (http://www.nova.edu/ssss/QR/QR3-2/tellis1.html)

[34] Wikipedia, The Free Encyclopedia (Online, Mar 2012), (http://en.wikipedia.org/wiki/Main_Page).

[35] M. Werner, C. Loebbecke and R. Baskerville, "Moderating Effects of Requirements Uncertainty on Flexible Software Development Techniques," in *Proc. of the 5th International Research Workshop on IT Project Management (IRWITPM)*, St. Louis, Missouri, Dec 2010, pp. 91-106.

[36] T. Wang, J. J. Pei-Hung and G. Klein," The effects of change control and management reviews on software flexibility and project performance", *Information and Management*, no. 45, pp. 438-443, 2008.

[37] B. Weber, M. Reichert and S. Rinderle-Ma, "Change patterns and change support features – Enhancing flexibility in process-aware information systems," *Data & Knowledge Engineering*, vol. 66, no. 3, pp.438-466, 2008.

[38] V. Wulf, V. Pipek, M. Von, "Component-based tailorability: Enabling highly flexible software applications", *International Journal of Human Computer Interface*, vol. 66, no. 1, pp. 1-22, Aug 2007.

[39] W. Woolfolk and B. Johnson, "Information Free Identifiers – A key to flexible information systems" *Data Base Management*, Aug 2001.

[40] F. Walter, "Should Computer Scientists Experiment More?" *IEEE Computer Society*, vol. 31, no. 5, pp. 32-40, May 1998.

[41] W. Woolfolk, P. Ligezinski and B. Johnson, "The Problem of the Dynamic Organization and the Static System: Principles and Techniques for Achieving Flexibility," in *Proc. of the 29th Annual Hawaii International Conference on System Sciences*, Wailea, HI, USA, Jan 1996, vol. 3, pp. 482-491.

[42] D. Zeng and L. Zhao, "Achieving Software Flexibility via Intelligent Workflow Techniques" in *Proc. of the 35th Hawaii International Conference on System Sciences, IEEE Computer Society*, Jan 2002, pp. 606 – 615.

Checklist for Valuating Stability of Information-Structure.

S#	Parameters to be Checked	Status	Remarks
1	Are the identifiers used as an attribute of an entity?	Y/N	
2	Are they of character type or non-numeric?	Y/N	
3	Are they generated automatically or any user is responsible to feed it in?	Y/N	
4	Is there any composite / married identifier used?	Y/N	
5	Do configuration tables / control data allow updating / change information?	Y/N	
6	Do transactional tables allow updating the attributes without keeping the old values as historic information?	Y/N	
7	Do transactional tables allow updating the FKs without keeping the old values as historic information?	Y/N	
8	Is any information of hierarchical nature is not designed as recursive?	Y/N	
9	Does a change in hierarchical structures e.g. change in organizational structure is automatically reflected in other structures like inventory – does it disrupt everything that is dependent on it?	Y/N	
10	Can the basic association rules of business be changed by end-users? If the system enforces the rule e.g. one person/position or one supervisor can be changed to multiple positions, supervisors etc without programming.	Y/N	
11	Do entities whether configurable or transactional have its birth, living status and death defined?	Y/N	
12	Are the business-rules are designed as information structure and business rules data stored as records in information structure to the fine-grained level.	Y/N	
13	Does the system use common information structure?	Y/N	
14	The design of information structure depicts top-down or bottom up approach i.e. each information structure is in its proper place?	Y/N	
15	Does a module access the information structure of other modules directly?	Y/N	
16	Do any data need to be entered more than once?	Y/N	
17	Can you find the same type of information structure (provincial ERs) meant for one entity e.g. Built-in workflow in more than one module?	Y/N	
18	Do you see any entity that seems to be rock and can be crushed into more fine-grained one.	Y/N	
19	Have the more fine-grained attributes concatenated and stored as one-whole.	Y/N	
20	Are there more than one FKs which can be minimized to one by adding table name as attribute?	Y/N	